# BOFormer: Learning to Solve Multi-Objective Bayesian Optimization via Non-Markovian RL

**Yu-Heng Hung**[†] **Kai-Jie Lin**[†] **Yu-Heng Lin**[†] **Chien-Yi Wang**[*‡] **Ping-Chun Hsieh**[*†]
[†]Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan
[‡]NVIDIA Research
{hungyh.cs08,pinghsieh}@nycu.edu.tw, chienyiw@nvidia.com

## Abstract

Bayesian optimization (BO) offers an efficient pipeline for optimizing black-box functions with the help of a Gaussian process prior and an acquisition function (AF). Recently, in the context of single-objective BO, learning-based AFs witnessed promising empirical results given its favorable non-myopic nature. Despite this, the direct extension of these approaches to multi-objective Bayesian optimization (MOBO) suffer from the *hypervolume identifiability issue*, which results from the non-Markovian nature of MOBO problems. To tackle this, inspired by the non-Markovian RL literature and the success of Transformers in language modeling, we present a generalized deep Q-learning framework and propose *BOFormer*, which substantiates this framework for MOBO via sequence modeling. Through extensive evaluation, we demonstrate that BOFormer constantly achieves better performance than the benchmark rule-based and learning-based algorithms in various synthetic MOBO and real-world multi-objective hyperparameter optimization problems.

## 1 Introduction

Bayesian optimization (BO) offers a sample-efficient pipeline for optimizing black-box functions in various practical applications, such as hyperparameter optimization [1–3], analog circuit design [4, 5], and automated scientific discovery [6, 7]. Notably, these real-world engineering tasks usually involve multiple objective functions, which are potentially conflicting. To search for the set of candidate solutions under a sampling budget, *multi-objective BO* (MOBO) integrates the following two components: (i) MOBO utilizes Gaussian processes (GP) as a surrogate function prior for capturing the underlying structure of each objective function and thereby offering posterior predictive distributions in a compact manner; (ii) MOBO then iteratively determines the samples through an acquisition function (AF), which induces an index-type strategy based on the posterior distributions. The existing AFs are built on various design principles, such as maximizing one-step expected improvement [8, 9] and maximizing one-step information gain [10–12]. However, the existing AFs for MOBO are mostly handcrafted and *myopic*, i.e., greedily optimize a one-step surrogate and lack long-term planning capability. With that said, one important and yet under-explored challenge of MOBO lies in the design of *non-myopic* AFs.

In single-objective BO (SOBO), one promising non-myopic approach is to recast BO as a reinforcement learning (RL) problem, and several RL-based algorithms [13–15] have recently witnessed competitive empirical results. Specifically, AFs could be parameterized by neural networks and learned by either actor-critic [13] or valued-based RL [14], where the state-action representation consists of the posterior mean and variance of a candidate point as well as the best function value observed so far, and the reward is defined as a function of negative simple regret, as shown in Figure

*Equal advising

ICML Workshop on Automated Reinforcement Learning: Exploring Meta-Learning, AutoML, and LLMs.

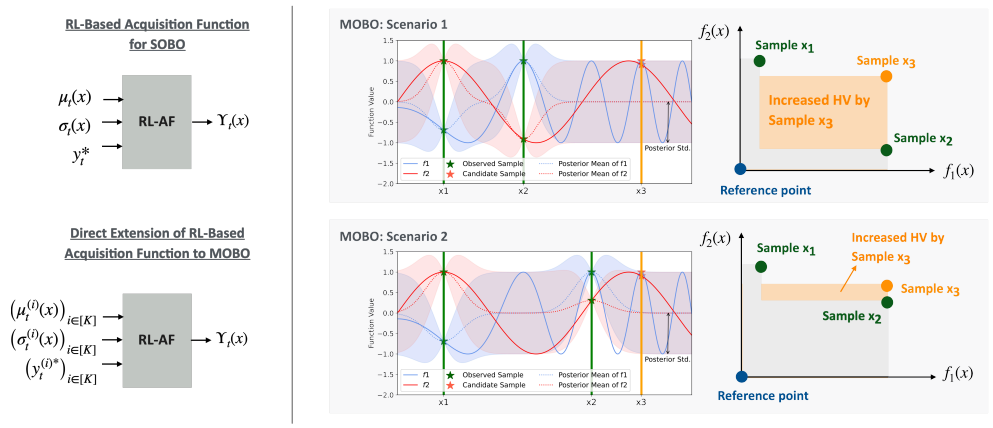



Figure 1: *Left*: In SOBO, an RL-based AF (e.g., FSAF [14]) takes the posterior mean and standard deviation $(\mu_t(x), \sigma_t(x))$ and the best function value observed so far $y_t^*$ as input and then outputs the AF value $\Upsilon_t(x)$. An direct extension to MOBO simply takes into account the same set of information about all the $K$ objective functions. *Right*: The hypervolume identifiability issue can be illustrated by comparing the hypervolume improvement incurred by the sample $x_3$ in the two different scenarios above. Clearly, despite that the AF inputs at $x_3$ are the same in both scenarios, the increases in hypervolume upon sampling $x_3$ are rather different. Hence, the increase in hypervolume is not identifiable solely based on the AF input $(\mu_t^{(i)}(x), \sigma_t^{(i)}(x)), y_t^{(i)*})_{i\in[K]}$ of the existing RL-based AFs.

1. However, a direct extension of these single-objective RL-based AFs to MOBO could suffer from the fundamental *hypervolume identifiability issue*. To better illustrate this, we provide a motivating example in Figure 1, which shows that one can construct a pair of scenarios that cannot be distinguished based solely on the current posterior distributions and the current best values. This example also highlights that the identifiability issue actually results from the inherent non-Markovianty in MOBO as the improvement in hypervolume is *history-dependent*. Note that this identifiability issue is much milder in the SOBO setting since a good candidate point (i.e., with a high function value) remains good regardless of the history. As a result, there remains one important open challenge in MOBO:

*How to learn a non-myopic AF for MOBO without suffering from the above identifiability issue?*

To tackle the above challenge, we propose to rethink MOBO from the perspective of non-Markovian RL via sequence modeling. Specifically, motivated by the optimality equations in the general non-Markovian RL [16], we tackle the hypervolume identifiability issue by presenting the non-Markovian version of deep Q-network termed *Generalized DQN*, which extends the standard Markovian DQN [17] by learning the generalized optimal Q-function defined on the history of observations and actions. To implement Generalized DQN, inspired by the significant success of the Transformers in language modeling, we propose BOFormer, which leverages the sequence modeling capability of the Transformer architecture and thereby minimizes the generalized temporal difference loss. As a general-purpose multi-objective optimization solver, the proposed BOFormer is trained solely on synthetic GP functions and can be deployed to optimize unseen testing functions. Moreover, to facilitate the training process, we present several useful and practical enhancements for BOFormer: (i) *$Q$-augmented observation representation*: Regarding the representation of per-step observation, we propose to use the posterior of the candidate point augmented with its $Q$-value, which serves as an informative indicator of the prospective improvement in hypervolume. Under this design, the representation is completely domain-agnostic and memory-efficient in the sense that it does not increase with the domain size. (ii) *Prioritized trajectory replay buffer and off-policy learning*: To improve convergence and data efficiency during training, we utilize a prioritized trajectory replay buffer, which can be viewed as a generalization of the typical replay buffer of vanilla DQN. Through this buffer, BOFormer naturally supports off-policy learning and more flexible reuse of training data. (iii) *Demo-policy-guided exploration*: While randomized exploration (e.g., epsilon-greedy) remains popular in many RL algorithms, this exploration scheme can be very inefficient as the sampling budget in BO is usually much smaller than the domain size (i.e., the number of actions). To achieve

efficient exploration, we propose to collect part of the training trajectories through a helper demo policy. In practice, one can resort to a policy induced by any off-the-shelf rule-based AFs, such as Expected Hypervolume Improvement [8].

Notably, as a general-purpose multi-objective optimization solver, BOFormer enjoys the following salient features: (i) *No hypervolume identifiability issue*: Built on the proposed Generalized DQN framework, BOFormer systematically addresses the identifiability issue such that it can approximately recover the Pareto front under a small sampling budget. (ii) *Zero-shot transfer*: BOFormer is trained solely on synthetic GP functions and can achieve zero-shot transfer to other unseen testing functions. With that said, it does not require any fine-tuning or any metadata during inference at deployment. (iii) *Cross-domain transfer capability*: BOFormer nicely supports cross-domain transfer in the sense that the domain size and dimensionality of the black-box functions for training can be different from those of the testing functions. (iv) *No Monte-Carlo estimation needed during inference*: As a learning-based AF, BOFormer completely obviates the need for the computationally heavy Monte-Carlo estimation required by many rule-based AFs and thereby enjoys efficient inference during deployment. The main contributions could be summarized as follows:

- We identify the critical hypervolume identifiability issue due to the inherent non-Markovianity in learning the AFs for MOBO. To resolve this, inspired by the literature of general RL, we present the Generalized DQN framework for non-Markovian environments.
- To substantiate the Generalized DQN framework, we propose BOFormer, which leverages the Transformer architecture and reinterprets MOBO as a sequence modeling problem. To the best of our knowledge, BOFormer serves as the first RL-based AF for MOBO. Moreover, several practical enhancements are proposed to facilitate the training of BOFormer.
- We evaluate the proposed BOFormer on a variety of black-box functions, including both synthetic optimization functions and real-world hyperparameter optimization problems. We demonstrate that the proposed AF significantly outperforms both the existing rule-based AFs and other Transformer-based RL benchmark methods.

## 2 Related Work

### 2.1 Multi-Objective Bayesian Optimization

**Random Scalarization**: To leverage the plethora of AFs for SOBO in the MOBO setting, random scalarization addresses MOBO via iteratively solving single-objective BO subproblems under a scalarization function, such as a direct weighted sum or the Tchebycheff scalarization function [18]. Notably, random scalarization was originally developed for recovering the Pareto front under evolutionary methods, such as the celebrated ParEGO [19], MOEA/D [20], and RVEA [21], and has been subsequently adapted to solving MOBO [22, 23]. Despite its simplicity, as random scalarization enforces exploration mainly by the random sampling of the scalarization parameters, this approach is known to be sensitive to the scale of the different objective functions and could suffer under high-dimensional search spaces [24].

**Improvement Maximization**: Another popular class of AFs is built on the maximization of improvement-based metrics, such as the expected one-step improvement in hypervolume (EHVI) in [8, 25, 26, 9] (also known as the $\mathcal{S}$-metric in [27, 28]), sequential uncertainty reduction [29], and the hypervolume knowledge gradient [30]. However, evaluating the one-step expected improvement typically involves a multi-dimensional integral, which is difficult to derive directly and hence needs to be approximated by the costly Monte Carlo estimation. Accordingly, to tackle the above computational complexity issue, differentiable methods have recently been developed to enable fast parallel evaluations of these AFs, such as $q$EHVI [31, 32] and $q$NEHVI [33] in the BoTorch framework [34].

**Information-Theoretic Search Methods**: Various information-theoretic criteria have been utilized in the context of MOBO. For example, Hernández-Lobato et al. [10] proposes Predictive Entropy Search for MOBO (PESMO), which selects the candidate point with maximal reduction in the entropy of the posterior distribution over the Pareto-optimal input set, and is subsequently extended to the constrained setting [35]. Subsequently, Belakaria et al. [11] propose MESMO, which extends the Max-value Entropy Search approach [36] to the principle of output space entropy search for MOBO, i.e., utilizes the information gain about the Pareto-optimal output set as a more computationally tractable sampling criterion [37]. Suzuki et al. [38] propose Pareto-Frontier Entropy Search, which utilizes information gain of the Pareto front in the AF design. Moreover, Joint Entropy Search (JES)

further takes into account the joint information gain of the Pareto-optimal set of inputs and outputs [12, 39]. On the other hand, USeMO [40] uses the volume of the uncertainty hyper-rectangle as an alternative uncertainty measure for sampling.

Despite the plethora of AFs developed for MOBO, most of them are built on optimizing one-step information-theoretic metrics and do not explore the possibility of multi-step look-ahead policies. By contrast, the proposed BOFormer takes the overall long-term effect of each sample into account through non-Markovian RL and sequence modeling.

### 2.2 Single-Objective Black-Box Optimization via Learning

Several recent attempts have tackled SOBO problems from the perspective of RL-based AFs. Volpp et al. [13] propose MetaBO, which leverages actor-critic RL to learn AFs from GP functions for transfer learning. Subsequently, Hsieh et al. [14] proposes a meta-RL framework termed Few-Shot Acquisition Function (FSAF), which learns a Bayesian deep Q-network as a differentiable AF and adapts the Bayesian model-agnostic meta-learning [41] in order to enable few-shot fast adaptation to various black-box functions based on metadata. [15] proposes to solve SOBO through a combination of transformer-based deep kernels and RL-based acquisition functions. Despite the above, the existing solutions all focus on SOBO under the standard RL formulation and therefore cannot be directly applied to the non-Markovian problem of MOBO. Moreover, as optimization of single-objective black-box functions is essentially a sequential decision making problem, several recent attempts manage to learn sequence models in an end-to-end manner. For example, Chen et al. [42] propose OptFormer, which focuses on hyperparameter optimization (HPO) and leverages Transformers through fine-tuning on an offline dataset to enable adaptation to the HPO tasks. More recently, Maraval et al. [43] propose Neural Acquisition Process (NAP), which is a multi-task variant of Neural Process (NP) simultaneously learning an acquisition function and the predictive distributions, without using the surrogate GP model. To the best of our knowledge, our paper offers the first learning-based solution to MOBO.

**Remarks on Application Scope and Objectives**: Notably, there are two salient differences between BOFormer and the above two works: (i) *Application scope*: BOFormer is positioned as a general-purpose multi-objective black-box optimization solver with superior cross-domain capability (i.e., the size and the dimensionality of the input domains can be different between the training and deployment phases). In contrast, OptFormer is designed specifically for HPO, and NAP is built on the idea of transfer learning in BO and has limited cross-domain transferability. (ii) *Multiple objectives*: Both OptFormer and NAP focus on single-objective problems and are not readily applicable to recovering the Pareto front in the multi-objective setting. By contrast, BOFormer directly tackles multi-objective optimization and addresses the inherent identifiability issue. Therefore, we consider the above OptFormer and NAP as orthogonal directions to ours. Moreover, the proposed BOFormer can also benefit from the learned neural process in NAP and other variants of NPs [44, 45] as surrogate models beyond GPs.

Due to the page limit, we defer the related works on sequence modeling for RL to Appendix B.2.

## 3 Preliminaries

In this section, we present the formulation of MOBO and the background of general RL. Throughout this paper, we let $\Delta(\mathcal{Z})$ denote the set of all probability distributions over a set $\mathcal{Z}$ and use $[K]$ as a shorthand for $\{1, \cdots, K\}$.

### 3.1 Multi-Objective Bayesian Optimization

The goal of MOBO is to design an algorithm that sequentially takes samples from the input domain $\mathbb{X} \subset \mathbb{R}^d$ to jointly optimize a black-box vector-valued function $\boldsymbol{f} : \mathbb{X} \to \mathbb{R}^K$, under a sampling budget $T \in \mathbb{N}$. For ease of exposition, we also write $\boldsymbol{f}(x) := (f_1(x), \cdots, f_K(x))$ as the tuple of the $K$ scalar objective functions, for each $x \in \mathbb{X}$. At each step $t$, the algorithm selects a sample point $x_t \in \mathbb{X}$ and observes the corresponding function values $\boldsymbol{y}_t := (y_t^{(1)}, \cdots, y_t^{(K)})$, where $y_t^{(i)} = f_i(x_t) + \varepsilon_{t,i}$ is the noisy observation of the $i$-th entry of the function output and $\varepsilon_{t,i}$'s are i.i.d. zero-mean Gaussian noises. For notational convenience, we use $\mathcal{F}_t := \{(x_i, \boldsymbol{y}_i)\}_{i \in [t]}$ to denote the observations up to $t$.

**Pareto Front and Hypervolume:** To construct a (partial) ordering over the points of the input domain, we say that $\boldsymbol{f}(x)$ *dominates* $\boldsymbol{f}(x')$ if $f_i(x) \geq f_i(x')$ for all $i \in [K]$ and $f_j(x) > f_j(x')$ for at least one element $j$. For simplicity, we write $x \succ x'$ if $\boldsymbol{f}(x)$ dominates $\boldsymbol{f}(x')$. Based on this, the *Pareto front* (denoted by $\mathcal{X}^*$) is defined as the subset of $\mathbb{X}$ that cannot be dominated by any other point in $\mathbb{X}$, i.e., $\mathcal{X}^* := \{x \in \mathbb{X} | x' \nsucc x, \forall x' \in \mathbb{X}\}$. An alternative description of the goal of MOBO is to discover the Pareto front. Accordingly, MOBO algorithms are typically evaluated from the perspective of *hypervolume*, which offers a natural performance metric for capturing the inherent trade-off among different objective functions. Specifically, given a reference point $\boldsymbol{u} \in \mathbb{R}^K$ and any subset $\mathcal{X} \subseteq \mathbb{X}$, the hypervolume of $\mathcal{X}$ is defined as [46]:

$$\mathrm{HV}(\mathcal{X}; \boldsymbol{u}) := \lambda\Big( \bigcup_{\boldsymbol{y} \in \mathbb{R}^K} \Big\{ x' \big| \boldsymbol{f}(x) \succ \boldsymbol{y} \succ \boldsymbol{u}, x \in \mathcal{X} \Big\} \Big),$$

where $\lambda(\cdot)$ is the $K$-dimensional Lebesgue measure. In practice, the reference point can be configured as $\boldsymbol{u} = (\min_{x \in \mathbb{X}} f_1(x), \cdots, \min_{x \in \mathbb{X}} f_k(x))$. To evaluate a policy, we consider the *simple regret* defined as $\mathcal{R}(t) := \mathrm{HV}(\mathbb{X}) - \mathrm{HV}(\mathcal{X}_t)$, which measures the overall performance of the samples up to time step $t$. For brevity, we simply use $\mathrm{HV}(\mathcal{X})$ as a shorthand for $\mathrm{HV}(\mathcal{X}; \boldsymbol{u})$ in the sequel.

**Gaussian Process as a Surrogate Model:** To maximize hypervolume in a sample-efficient manner, MOBO imposes a function prior through GP, which serves as a surrogate probabilistic model for capturing the underlying structure of the objective functions. Specifically, as a Bayesian approach, the GP assumes that for each objective function $f_i(\cdot)$, the function values at any set of input points form a multivariate Gaussian distribution, which can be fully characterized by a mean function and a covariance kernel. Therefore, under a GP prior, given the observations $\mathcal{F}_t$ up to time $t$, the posterior predictive distribution of each $f_i(x)$ ($x \in \mathbb{X}$) remains Gaussian and can be written as $\mathcal{N}(\mu_t^{(i)}(x), \sigma_t^{(i)}(x)^2)$, where $\mu_t^{(i)}(x) := \mathbb{E}[f_i(x) | \mathcal{F}_t]$ and $\sigma_t^{(i)}(x) := \sqrt{\mathbb{V}[f_i(x) | \mathcal{F}_t]}$ can be derived in closed form through matrix operations [47]. For notational convenience, we let $\boldsymbol{\mu}_t(x) := (\mu_t^{(1)}(x), \cdots, \mu_t^{(K)}(x))$ and $\boldsymbol{\sigma}_t(x) := (\sigma_t^{(1)}(x), \cdots, \sigma_t^{(K)}(x))$.

**Acquisition Functions:** With the help of GPs, one natural way to address BO is through planning as in optimal control (e.g., via dynamic programming). However, finding the exact optimal policy for BO (either single- or multi-objective) is known to be computationally intractable in general due to the curse of dimensionality. To tackle this issue, BO resorts to *index-type* strategies induced by an acquisition function $\Upsilon(\boldsymbol{\mu}_t(x), \boldsymbol{\sigma}_t(x))$, which takes the posterior mean and variance as input and outputs an indicator for quantifying the usefulness of each potential candidate sample point $x \in \mathbb{X}$, typically based on some handcrafted design criteria. For example, the celebrated Expected Hypervolume Improvement (EHVI) method constructs an AF as $\Upsilon_{\mathrm{EHVI}}(\boldsymbol{\mu}_t(x), \boldsymbol{\sigma}_t(x)) := \mathbb{E}[\mathrm{HV}(\mathcal{X}_t \cup \{x\}) - \mathrm{HV}(\mathcal{X}_t) | \mathcal{F}_t]$, which involves a multi-dimensional integral with respect to the posterior distribution characterized by $\boldsymbol{\mu}_t(x)$ and $\boldsymbol{\sigma}_t(x)$.

### 3.2 General RL in Non-Markovian Environments

To achieve RL without Markovianity, several generalizations of the standard Markov decision process (MDP) have been proposed, such as the classic partially-observable MDPs [48, 49], the early works on general RL [50–52], and the more recent attempts on RL for arbitrary environments [16, 53, 54]. In this paper, we consider the general RL formulation in [16, 53] to address policy learning beyond Markovianity.

**Environment:** The general interaction protocol of the agent and the environment can be described as follows. Let $\mathcal{A}$ and $\mathcal{O}$ denote the set of actions and observations, respectively. At each time $t \in \mathbb{N}$, the agent first receives a new observation $O_t \in \mathcal{O}$ from the environment and takes an action $A_t \in \mathcal{A}$ based on the history $H_t := (A_0, O_1, A_1, \cdots, A_{t-1}, O_t)$ observed so far. For simplicity, we let the initial history $H_0$ be empty. We also define the set of all $n$-step histories as $\mathcal{H}^{(n)} := (\mathcal{A} \times \mathcal{O})^n$ and accordingly define the set of all finite histories as $\mathcal{H} := \bigcup_{n \geq 0} \mathcal{H}^{(n)}$. The transition dynamics of the environment is captured by the *transition function* $p : \mathcal{H} \times \mathcal{A} \to \Delta(\mathcal{O})$, which determines the transition probability $p(o|h, a) \equiv \mathbb{P}(O_{t+1} = o | H_t = h, A_t = a)$ of observing $o$ upon applying action $a$ under history $h$. Moreover, let $r : \mathcal{H} \times \mathcal{A} \times \mathcal{O} \to [-r_{\max}, r_{\max}]$ denote the reward function. Notably, the reward function $r$ in non-Markovian environments is allowed to be history-dependent and hence better suits the MOBO problems. Let $\gamma \in [0, 1)$ denote the discount factor for the rewards.

**Policies and Value Functions:** The agent specifies its strategy through a *policy* $\pi : \mathcal{H} \to \Delta(\mathcal{A})$, which maps each history to a probability distribution over the action set. Let $\Pi$ denote the set of all policies. Similar to the MDP setting, we define value functions that reflect the long-term benefit of following a policy $\pi$. Given a $\tau$-step history $h \in \mathcal{H}$,

$$V^\pi(h) := \mathbb{E}_\pi\Big[\sum_{t=\tau}^{\infty} \gamma^{t-\tau} r(H_t, A_t, O_{t+1}) \Big| H_\tau = h\Big],$$

$$Q^\pi(h,a) := \mathbb{E}_\pi\Big[\sum_{t=\tau}^{\infty} \gamma^{t-\tau} r(H_t, A_t, O_{t+1}) \Big| H_\tau = h, A_\tau = a\Big].$$

Moreover, we extend the definitions of the optimal value functions in MDPs to the non-Markovian setting as

$$V^*(h) := \sup_{\pi\in\Pi} V^\pi(h),\ Q^*(h,a) := \sup_{\pi\in\Pi} Q^\pi(h,a). \tag{1}$$

The proposition below offers a generalized version of the Bellman optimality equations and characterizes $V^*$ and $Q^*$.

**Proposition 3.1** (Dong et al. [16])**.** *The pair of* $(V^*, Q^*)$ *is the unique solution to the following system of equations:*

$$V(h) = \max_{a'\in\mathcal{A}} Q(h,a') \tag{2}$$

$$Q(h,a) = \mathbb{E}_{o\sim p(\cdot|h,a), h'\equiv(h,a,o)}\big[r(h,a,o) + \gamma V(h')\big], \tag{3}$$

*where* $V : \mathcal{H} \to \mathbb{R}$ *and* $Q : \mathcal{H}\times\mathcal{A} \to \mathbb{R}$ *are bounded real-valued functions.*

# 4 Methodology

## 4.1 Generalized DQN for Non-Markovian Problems

Motivated by the optimality equations in (2)-(3), we convert these fundamental properties into a learning algorithm.

**Loss Function:** To learn $Q^*$, we adapt the loss function of the standard DQN to the generalized non-Markovian version by minimizing the residual of the optimality equation. Let $Q_\theta(h,a)$ denote the parameterized Q-function. Then, the loss function of the generalized DQN is designed as

$$\mathbb{E}_{(h,a,o)\sim\mathcal{D}}\left[\Big(r(h,a,o) + \gamma \max_{a'\in\mathcal{A}} Q_{\bar\theta}(h',a') - Q_\theta(h,a)\Big)^2\right], \tag{4}$$

where $\mathcal{D}$ is the underlying distribution of the observed histories during training, $h' = (h,a,o)$ is the history for the next Q-value, and $Q_{\bar\theta}$ is a copy of $Q_\theta$ with parameters frozen.

**Remark 4.1.** The above loss function bears some resemblance to that of the POMDP variant of DQN, such as Deep Recurrent Q-Networks (DRQN) in [55]. Despite this, one fundamental difference is: The POMDP formulation presumes that there exists a hidden true state, which determines the transitions and the reward function, and the hidden state is to be learned and deciphered by recurrent neural networks in DRQN. By contrast, Generalized DQN does not make this presumption and involves only the history of observations and actions.

**Direct Implementation of Generalized DQN:** To implement (4), one natural design is to leverage sequence modeling (e.g., Transformers) and directly use the full observations as the input of the sequence models. This design principle is widely adopted in Transformer-based RL [56–58] for various popular RL benchmark tasks (e.g., locomotion and robot arm manipulation in MuJoCo [59]). In the context of learning AFs for MOBO, one can apply this design principle and extend the representation design of AF for SOBO (cf. Figure 1) to the MOBO setting, and this amounts to taking the posterior distributions of all $K$ objective functions at all the domain points along with $\{y_t^{(i)*} := \text{argmax}_{j\le t-1}\, y_j^{(i)}\}_{i=i}^K$ the best function values observed so far as the per-step observation, i.e., $o \equiv \{\mu^{(i)}(x), \sigma^{(i)}(x), y^{(i)*}\}_{x\in\mathcal{X}, i\in[K]}$. While being a natural variant of Transformer-based RL, this implementation of the Generalized DQN framework can be problematic in MOBO for

two reasons: (i) *Limited cross-domain transferability*: As the observation representation is domain-dependent under this design, the learned model is tied closely to the training domain and has very limited transferability. As a result, retraining or customization is needed for every task at deployment. (ii) *Scalability issue in sequence length and memory requirement:* Under this design, the sequence length would grow linearly with the number of domain points and pose a stringent requirement on the hardware memory for training. Indeed, the domain size is at least on the order of thousands in practical BO problems (e.g., circuit design [4] and hyperparameter optimization [1]).

To tackle the above issues, we propose an alternative design that better substantiates the Generalized DQN framework for MOBO with domain-agnostic representations and several practical enhancements, as detailed in Section 4.2.

### 4.2 BOFormer: An Enhanced Implementation of Generalized DQN

To avoid the issues of the direct implementation, we propose BOFormer, which is built on the following enhancements. The pseudo code is in Algorithm 1 in the Appendix.

**$Q$-Augmented Representation:** Define

$$y_t^{(i)*} := \operatorname*{argmax}_{1\le j\le t} y_i^{(i)}, \forall i \in [1, \cdots, K]$$

as the best observed function value of $j$-th objective at time $t$. Moreover, for each domain point $x \in \mathbb{X}$, let $o_t(x)$ denote the observation for $x$ as

$$o_t(x) \equiv \left\{\mu_t^{(i)}(x), \sigma_t^{(i)}(x), y_t^{(i)*}, \frac{t}{T}\right\}_{i\in[K]}.$$

Moreover, in BOFormer, we use the normalized hypervolume improvement as the reward, i.e.,

$$r_t := \frac{\mathrm{HV}(\mathcal{X}_t) - \mathrm{HV}(\mathcal{X}_{t-1})}{\mathrm{HV}(\mathcal{X}^*) - \mathrm{HV}(\mathcal{X}_t)}.$$

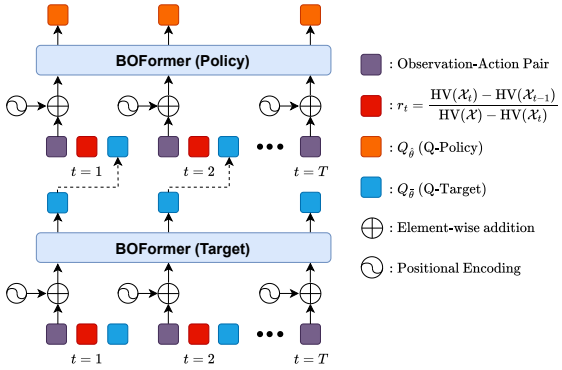



Figure 2: BOFormer comprises two distinct networks as shown above: The upper network functions as the policy network, utilizing the historical data and the Q-value predicted by the target network to estimate the Q-values for action selection. The lower network serves as the target network, responsible for constructing Q-values for past observation-action pairs.

Then, $h_t$, the history up to time $t$, is the concatenation of past observation-action pair representation defined as follows:

$$h_t = \left\{\mu_j^{(i)}(x_j), \sigma_j^{(i)}(x_j), y_{j-1}^{(i)*}, j/t, r_i, Q_{\bar{\theta}}\right\}_{i\in[k], j\in[t-1]}. \tag{5}$$

Notably, under this design, the representation is domain-agnostic and memory-efficient in the sense that its dimension does not increase with the domain size.

**BOFormer as an Acquisition Function for MOBO:** The model structure of BOFormer is provided in Figure 2. Denote $Q_\theta(\cdot) : \mathcal{H}^{(t-1)} \times \mathcal{O} \to \mathbb{R}$ to be the function of BOFormer parameterized by $\theta$ and let $\widehat{\theta}$ represent the parameters of BOFormer. The selected point $x_t$ satisfies that $x_t := \operatorname{argmax}_{x\in\mathbb{X}} Q_{\widehat{\theta}}(h_t, o_t(x))$.

Then, $Q_{\bar{\theta}}$ considered in $h_t$ can be implemented by a target network, as is commonly done in Deep Q-learning. In non-Markovian version, $Q_{\bar{\theta}}$ can be defined recursively, where

$$Q_{\bar{\theta}}(h_t, o_t(x_t)) := Q_{\bar{\theta}}\left(\{o_i(x_i), r_i, Q_{\bar{\theta}}(h_{i-1}, o_{i-1}(x_{i-1}))\}_{i=1}^{t-1}, o_t(x_t)\right).$$

**Off-Policy Learning and Prioritized Trajectory Replay Buffer:** We extend the concept of Prioritized Experience Replay (PER) [60] and introduce the Prioritized Trajectory Replay Buffer (PTRB). The detailed modifications are as follows: (i) Elements pushed into this buffer are entire trajectories $\tau = \{o_i(x_i), r_i\}_{i=1}^T$. (ii) The TD-error considered in PER is replaced by $\delta(Q_{\theta_t}, \tau)$, which is the summation of the TD-error of the policy network for all transitions in this trajectory, i.e.,

$$\delta(Q, \tau) := \sum_{i=1}^{T-1} \left(Q\left(h_i, o_i(x_i)\right) - \left(r_i + \gamma \max_{x\in\mathbb{X}} Q_{\bar{\theta}}(h_{i+1}, o_{i+1}(x))\right)\right)^2. \tag{6}$$

Let $B$ denote the batch sampled from PTRB. The loss function of BOFormer is defined as $L(\theta) := \sum_{\tau \in B} \delta(Q_\theta, \tau)$.

# 5 Experiments

In this section, we evaluate the proposed BOFormer against popular MOBO methods on both synthetic and real hyperparameter optimization problems in YAHPO Gym [61]. Unless stated otherwise, we report the average attained hypervolume over 30 evaluation episodes in the main text. Due to the space limit, all the statistics (including percentiles and standard deviation) are provided in the Appendix.

**Popular Benchmark Methods.** We compare BOFormer with various classic and state-of-the-art benchmark methods, including: (i) *Rule-based methods*: NEHVI [33], ParEGO [19], NSGA-II [62], HVKG [30], and JES [12, 39]. Regarding NEHVI, ParEGO, and HVKG, we use the differentiable Monte-Carlo version, namely $q$NEHVI, $q$ParEGO, and $q$HVKG, provided by BoTorch [34]. (ii) *Learning-based methods*: Given that BOFormer is the first learning-based MOBO method, we consider the direct multi-objective extension of FSAF [14], which achieves state-of-the-art results in SOBO. To showcase the design of BOFormer, we also adapt a popular RL Transformers, namely Decision Transformer (DT) [56], to the MOBO setting. Moreover, we also include Q-Transformer (QT) [58], a more recent Transformer design RL that uses a similar DQN loss (termed Autoregressive Discrete Q-Learning in their paper) and can be viewed as a variant of BOFormer without Q-augmented representation. To further demonstrate the competitiveness of BOFormer, we also compare it with OptFormer [42], a recent Transformer-based method designed specifically for hyperparameter optimization. All the learning-based methods are trained on GP functions under with the lengthscales drawn randomly from $[0.1, 0.4]$ for fairness. The detailed configuration is provided in Appendix A.

***Q: Does BOFormer achieve sample-efficient MOBO on a variety of optimization problems?***

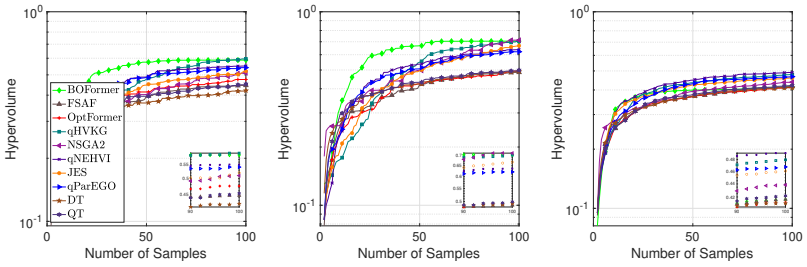
**Synthetic Functions:** We answer this question by first evaluating BOFormer extensively on a diverse collection of synthetic black-box functions: (i) Combinations of functions with many local optima, including *Ackley-Rastrigin (ARa)* and *Ackley-Schwefel-Rastrigin (ASRa)*; (ii) Combination of smooth functions, including *Branin-Currin (BC)* and *Dixon-Rosenbrock (DR)*; (iii) Combination of non-smooth and smooth functions, including *Ackley-Rosenbrock (AR)* and *Ackley-Rosenbrock-Sphere (ARS)*. Figure 3 shows the averaged hypervolume on synthetic problems. Based on Figure 3, we can observe that BOFormer constantly achieves the smallest or among the smallest regrets among all methods and demonstrates a faster convergence rate in the early stages of testing. In the cases of Branin and Currin, the performance of BOFormer is not as expected, which we attribute to the larger length scales of these functions compared to others. Then, QT, a variant of the generalized DQN but without Q-augmented representation, does not perform well. This indicates that sequence modeling itself does not necessarily guarantee an efficient search for the Pareto front, and Q-augmented representation are needed for solving MOBO, as showcased by the proposed BOFormer. Please see Appendix D for the standard deviation and the percentiles of the final hypervolume.

**Real-World Multi-Objective Hyperparameter Optimization:** We proceed with evaluations on four benchmark problems provided in the YAHPO Gym [61], including: (i) LCBench, introduced by [63]. (ii) Random Bot V2 with classifier glmnet (rbv2-glmnet). (iii) Random Bot V2 with classifier svm (rbv2-svm). (iv) Random Bot V2 with classifier xbgboost (rbv2-xbgboost), where the objective functions of (i) are testing accuracy and negative cross entropy, and the objective functions of (ii-iv) are testing accuracy and the area under ROC curve. Again, from Figure 4, we can observe that BOFormer remains the best or among the best in all the tasks. This result demonstrates the wide applicability of the proposed BOFormer. The detailed standard deviation and the percentiles of the final hypervolume for YAHPO dataset is provided in Appendix D.

***Q: The comparison between BOFormer and OptFormer.*** From Figures 3-4, we observe that the performance of BOFormer surpasses that of OptFormer. We conjecture that the reasons are two-fold: (i) OptFormer takes a supervised learning perspective to learn the context that describes the HPO information while BOFormer leverages non-Markovian RL for better long-term planning. (ii) OptFormer reinterprets HPO as a language modeling problem in a text-to-text manner. This approach necessitates that the training dataset closely resembles the testing domain, and this requirement does not hold here (e.g., dimensionality of the training domain differs from that of the testing domain).

***Q: A Study on the Effect of Sequence Length on BOFormer.*** We also conducted an ablation study comparing the Markovian BOFormer (window size $\omega = 1$) and non-Markovian BOFormer (window size $\omega > 1$) in Appendix D.3.

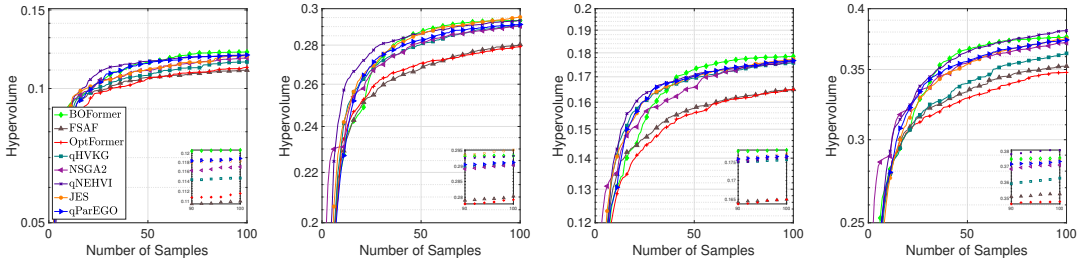
***Q: A Study on Computational Efficiency.*** We provide computation times per step in Table 1. BOFormer and qHVKG are competitive in final hypervolume (Tables 2-5), with BOFormer having shorter computation time.




(a) AR (b) ARa (c) BC

(d) DR (e) ARS (f) ASRa

Figure 3: Averaged attained hypervolume under synthetic objective functions.




(a) lcbench (b) rbv2-glmnet (c) rbv2-svm (d) rbv2-xbgboost

Figure 4: Averaged attained hypervolume under the YAHPO Gym datasets.

## 6 Conclusion

In this paper, we address MOBO problems from the perspective of RL-based AF by identifying and tackling the inherent hypervolume identifiability issue. We achieve this goal by first presenting a generalized DQN framework and implementing it through BOFormer, which leverages the sequence modeling capability of Transformers and incorporates multiple enhancements for MOBO. Our experimental results show that BOFormer is indeed a promising approach for general-purpose multi-objective black-box optimization.

## References


[1] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(1):2475–2483, 2022.

[2] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

[3] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *International Conference on Artificial Intelligence and Statistics*, pages 528–536, 2017.

[4] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Batch Bayesian Optimization via Multi-Objective Acquisition Ensemble for Automated Analog Circuit Design. In *International Conference on Machine Learning*, pages 3306–3314, 2018.

[5] Jinzhu Zhou, Zhanbiao Yang, Yu Si, Le Kang, Haitao Li, Mei Wang, and Zhiya Zhang. A trust-region parallel Bayesian optimization method for simulation-driven antenna design. *IEEE Transactions on Antennas and Propagation*, 69(7):3966–3981, 2020.

[6] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: An Efficient Bayesian Optimization Library for Materials Science. *Materials Discovery*, 4:18–21, 2016.

[7] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS central science*, 4(2):268–276, 2018.

[8] Michael Emmerich and Jan-willem Klinkenberg. The computation of the expected improvement in dominated hypervolume of Pareto front approximations. *Rapport technique, Leiden University*, 2008.

[9] Kaifeng Yang, Michael Emmerich, André Deutz, and Thomas Bäck. Multi-objective Bayesian global optimization using expected hypervolume improvement gradient. *Swarm and Evolutionary Computation*, 44:945–956, 2019.

[10] Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective Bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501, 2016.

[11] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[12] Ben Tu, Axel Gandy, Nikolas Kantas, and Behrang Shafei. Joint entropy search for multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, 35:9922–9938, 2022.

[13] Michael Volpp, Lukas P Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in Bayesian optimization. In *International Conference on Learning Representations*, 2020.

[14] Bing-Jing Hsieh, Ping-Chun Hsieh, and Xi Liu. Reinforced few-shot acquisition function learning for Bayesian optimization. *Advances in Neural Information Processing Systems*, 34:7718–7731, 2021.

[15] Alexander Shmakov, Avisek Naug, Vineet Gundecha, Sahand Ghorbanpour, Ricardo Luna Gutierrez, Ashwin Ramesh Babu, Antonio Guillen, and Soumyendu Sarkar. RTDK-BO: High dimensional Bayesian optimization with reinforced transformer deep kernels. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1–8, 2023.

[16] Shi Dong, Benjamin Van Roy, and Zhengyuan Zhou. Simple agent, complex environment: Efficient reinforcement learning with agent states. *Journal of Machine Learning Research*, 23(1):11627–11680, 2022.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[18] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

[19] Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[20] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[21] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5): 773–791, 2016.

[22] Qingfu Zhang, Wudong Liu, Edward Tsang, and Botond Virginas. Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3): 456–474, 2009.

[23] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible framework for multi-objective Bayesian optimization using random scalarizations. In *Uncertainty in Artificial Intelligence*, pages 766–776, 2020.

[24] Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective Bayesian optimization over high-dimensional search spaces. In *Uncertainty in Artificial Intelligence*, pages 507–517, 2022.

[25] Michael TM Emmerich, André H Deutz, and Jan Willem Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *IEEE Congress of Evolutionary Computation (CEC)*, pages 2147–2154, 2011.

[26] Iris Hupkens, André Deutz, Kaifeng Yang, and Michael Emmerich. Faster exact algorithms for computing expected hypervolume improvement. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 65–79. Springer, 2015.

[27] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[28] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted-metric selection. In *International Conference on Parallel Problem Solving From Nature*, pages 784–794. Springer, 2008.

[29] Victor Picheny. Multiobjective optimization using Gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

[30] Sam Daulton, Maximilian Balandat, and Eytan Bakshy. Hypervolume knowledge gradient: A lookahead approach for multi-objective Bayesian optimization with partial information. In *International Conference on Machine Learning*, pages 7167–7204, 2023.

[31] Takashi Wada and Hideitsu Hino. Bayesian optimization for multi-objective optimization and multi-point search. *arXiv preprint arXiv:1905.02370*, 2019.

[32] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, 33:9851–9864, 2020.

[33] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems*, 34:2187–2200, 2021.

[34] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, 33:21524–21538, 2020.

[35] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Predictive entropy search for multi-objective Bayesian optimization with constraints. *Neurocomputing*, 361:50–68, 2019.

[36] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635, 2017.

[37] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Output space entropy search framework for multi-objective Bayesian optimization. *Journal of Artificial Intelligence Research*, 72:667–715, 2021.

[38] Shinya Suzuki, Shion Takeno, Tomoyuki Tamura, Kazuki Shitara, and Masayuki Karasuyama. Multi-objective Bayesian optimization using Pareto-frontier entropy. In *International Conference on Machine Learning*, pages 9279–9288, 2020.

[39] Carl Hvarfner, Frank Hutter, and Luigi Nardi. Joint entropy search for maximally-informed Bayesian optimization. *Advances in Neural Information Processing Systems*, 35:11494–11506, 2022.

[40] Syrine Belakaria, Aryan Deshwal, Nitthilan Kannappan Jayakodi, and Janardhan Rao Doppa. Uncertainty-aware search framework for multi-objective Bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10044–10052, 2020.

[41] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[42] Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, et al. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068, 2022.

[43] Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou Ammar. End-to-end meta-Bayesian pptimisation with Transformer Neural Processes. *Advances in Neural Information Processing Systems*, 2023.

[44] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713, 2018.

[45] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2018.

[46] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[47] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press, 2006.

[48] George E Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[49] Tommi Jaakkola, Satinder Singh, and Michael Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. *Advances in Neural Information Processing systems*, 7, 1994.

[50] Tor Lattimore, Marcus Hutter, and Peter Sunehag. The sample-complexity of general reinforcement learning. In *International Conference on Machine Learning*, pages 28–36, 2013.

[51] Jan Leike. *Nonparametric general reinforcement learning*. PhD thesis, The Australian National University (Australia), 2016.

[52] Sultan J Majeed. *Abstractions of General Reinforcement Learning: An Inquiry into the Scalability of Generally Intelligent Agents*. PhD thesis, The Australian National University (Australia), 2021.

[53] Xiuyuan Lu, Benjamin Van Roy, Vikranth Dwaracherla, Morteza Ibrahimi, Ian Osband, Zheng Wen, et al. Reinforcement learning, bit by bit. *Foundations and Trends® in Machine Learning*, 16(6):733–865, 2023.

[54] Michael Bowling, John D Martin, David Abel, and Will Dabney. Settling the reward hypothesis. In *International Conference on Machine Learning*, pages 3003–3020, 2023.

[55] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium Series*, 2015.

[56] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097, 2021.

[57] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems*, 34:1273–1286, 2021.

[58] Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR, 2023.

[59] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[60] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[61] Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. Yahpo gym-an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International Conference on Automated Machine Learning*, pages 3–1. PMLR, 2022.

[62] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[63] Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3079–3090, 2021.

[64] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] Juergen Schmidhuber. Reinforcement Learning Upside Down: Don't Predict Rewards–Just Map Them to Actions. *arXiv preprint arXiv:1912.02875*, 2019.

[66] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *International Conference on Machine Learning*, pages 27042–27059. PMLR, 2022.

[67] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936, 2022.

[68] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2021.

## Acknowledgements

This material is based upon work partially supported by the National Science and Technology Council (NSTC), Taiwan under Contract No. 112-2628-E-A49-023 and Contract No. 112-2634-F-A49-001-MBK, also partially supported by the NVIDIA Taiwan R&D Center, and also supported by the Higher Education Sprout Project of the National Yang Ming Chiao Tung University and Ministry of Education (MOE), Taiwan. We also thank the National Center for High-performance Computing (NCHC) for providing computational and storage resources.

## A Detailed Training Configuration

### A.1 Environment

In each testing or training episode, the agent interacts with the environment for a total of $T = 100$ time steps, and the observed function values are subject to noise $\epsilon \sim N(0, 0.1)$. For learning-based algorithms, the environment consists of $K$ GP functions and returns the GP posterior mean and variance after each sampling. The kernel is randomly chosen from the RBF or Matérn-5/2, and the length scale is sampled from Uniform$(0.1, 0.4)$.

During testing, the environment computes the log marginal likelihood based on all domain points and the corresponding $K$ function values. The function value benchmark and real-world function are scaled by perturbation noise, which is sampled from $N(0, 0.1)$.

### A.2 Hyperparameter of Learning-Based Approaches

- **BOFormer:** hidden size: 128, learning rate: $10^{-5}$, weight decay: $10^{-5}$, $r_{\text{demo:}}$ 5, batch size: 8, # of attention layer: 8, # of head of attention layer: 4, window size $w = 31$, dropout: 0.1, buffer size: 64.
- **DT**[*]**:** hidden size: 128, learning rate: $10^{-4}$, weight decay: $10^{-4}$, batch size: 16, # of attention layer: 3, # of head of attention layer: 2, embed dim: 500, dropout: 0.1, warmup steps: 10, max length: 100.
- **QT:** hidden size: 128, learning rate: $10^{-5}$, weight decay: $10^{-5}$, $r_{\text{demo:}}$ 5, batch size: 8, # of attention layer: 8, # of head of attention layer: 4, window size $w = 21$, dropout: 0.1, buffer size: 64.
- **FSAF:** alpha: 0.8, hidden size: 100, learning rate: 0.01, batch size: 128, few shot step: 5, # of particles: 5, total task: 3, size of meta data: 100, use demo: True, early terminate: False, select type: average.
- **OptFormer:** string length: 128, learning rate: $10^{-2}$, weight decay: $10^{-2}$, batch size: 1, window size $w = 10$
- **Common Hyperparameter:** optimizer: Adam [64], $\epsilon$-greedy rate: 0.1

### A.3 Hyperparameter of Rule-Based Approaches

- **qEHVI:** q: 1
- **qParEgo:** q: 1, acquisition function: Expected Improvement
- **JES:** # of samples: 64, estimation type: LB
- **NSGA-II:** population size: 10, # of generations: 10, sampling: random
- **USeMO:** acquisition function: Expected Improvement, batch size: 1

## B Additional Related Work

### B.1 Transformers and Sequence Modeling for RL

Given the success of sequence-to-sequence models in language processing, RL has recently been addressed through the lens of sequence modeling, especially transformers. For example, Chen et al.

[*]We reuse the open source implementation from https://github.com/jannerm/trajectory-transformer.

[56] propose Decision Transformer (DT), which addresses offline RL by mapping return-to-go to actions via transformers and thereby substantiating the concept of Upside Down RL [65]. The design of DT has been subsequently extended to various settings, including online RL [66], multi-game setting [67], and general information matching [68]. Concurrently, to tackle offline RL, Janner et al. [57] propose Trajectory Transformer (TT), which serves as a predictive dynamics model and uses beam search as a trajectory optimizer. More recently, Chebotar et al. [58] introduced Q-Transformer (QT), which is trained by offline temporal difference updates and achieves scalable representation for Q-functions by discretizing action dimensions as tokens on a Transformer, with a focus on robotic tasks. By contrast, the proposed BOFormer is built on the (online) generalized DQN framework and designed for addressing MOBO based on multiple enhancements.

### B.2 A More Detailed Comparison to Q-Transformer

Q-Transformer (QT) [58] similarly incorporates the idea of utilizing a Transformer in learning the $Q$-function. Despite this high-level design resemblance, the proposed BOFormer is different from QT in multiple aspects:

- **Problem Setting:** QT is designed mainly to address offline RL, where the performance is highly correlated to the quality of prior data, as a robotic learning approach and required to address high-dimensional action spaces. By contrast, our work introduces BOFormer, which is trained to solve MOBO by learning a generalized Q function based on the online interactions with synthetic GP functions, and hence this can be viewed as an instance of online RL.
- **Network Architecture:** QT uses state-action sequences as the input of the Transformer. By contrast, to resolve the hypervolume identifiability issue and achieve cross-domain transferability simultaneously, the proposed transformer of BOFormer implements a generalized DQN and uses the $Q$-augmented observation representation as the input of the transformer. This approach can better address the non-Markovian property in MOBO.
- **Training Algorithm:** Compared to QT, the proposed BOFormer incorporates multiple practical enhancements, including $Q$-augmented representation, reward normalization, and demo policy and prioritized trajectory replay buffer for off-policy learning.

## C Pseudo Code and Additional Implementation Details of BOFormer

### C.1 Pseudo Code

The detailed pseudo code of the training processes for BOFormer under off-policy learning and on-policy learning setting are provided in Algorithms 1 and 2, respectively.

### C.2 Additional Implementation Details of BOFormer

**Reward Signal With Normalization:** In the MOBO setting, one natural reward design is the one-step improvement in hypervolume, i.e., $\hat{r}_t := \mathrm{HV}(\mathcal{X}_t) - \mathrm{HV}(\mathcal{X}_{t-1})$. However, as the achieved hypervolume increases, the reward signal $\hat{r}_t$ can get weaker in the later stage of an episode, making it difficult to recover the whole Pareto front. To address this, we construct $r_t$, which is $\hat{r}_t$ but scaled by the difference between the current hypervolume and optimal hypervolume, as the reward signal for RL in MOBO, i.e.,

$$r_t := \frac{\mathrm{HV}(\mathcal{X}_t) - \mathrm{HV}(\mathcal{X}_{t-1})}{\mathrm{HV}(\mathcal{X}^*) - \mathrm{HV}(\mathcal{X}_t)}. \tag{7}$$

**Remark C.1.** The information about $\mathrm{HV}(\mathcal{X}^*)$ in (7) is used only during training and can be easily pre-computed or approximated given the knowledge about the domain and the black-box functions.

**Demo-Policy-Guided Exploration:** To facilitate the off-policy learning of BOFormer, one natural approach is to adopt a behavior policy with randomized exploration (e.g., epsilon-greedy) for collecting trajectories from the environment. However, such a randomized exploration scheme can be very inefficient as the sampling budget in MOBO is usually much smaller than the domain size (i.e., the number of actions). To better guide the exploration, we propose to use a demo policy, which is

possibly sub-optimal but of sufficient strength in exploring regions near the Pareto front. In practice, we set $r_{\text{demo}}$ to be the probability of using a demo policy induced by an off-the-shelf AF for MOBO (e.g., EHVI) in this training episode. The training processes of BOFormer for off-policy learning and on-policy learning are provided in Algorithms 1 and 2, respectively.

**Algorithm 1** Off-Policy BOFormer

**Input:** $\boldsymbol{\theta}_1$, Training Episodes $E$, Time Horizon $T$, Demo Rate $r_{\text{demo}}$, Target Rate $r_{\text{target}}$, Target Network $Q_{\bar{\theta}}$, Batch Size $B$, Replay buffer $\mathcal{B}$, environment $\mathcal{E}$.
**for** $e = 1, 2, \cdots, E$ **do**
  $\mathcal{E}$.reset()
  $s_1^e = \mathcal{E}$.state
  demo = random.binomial($r_{\text{demo}}$)
  $\pi_t^e$ = ExpectedHypervolumeImprovement
  **for** $t = 1, 2, \cdots, T$ **do**
    **if** demo **then**
      $x_t^e = \pi_t^e(s_t^e)$
    **else**
      **for** $i = 1, 2, \cdots, t-1$ **do**
        Compute $Q_{\bar{\theta}}(h_i^e, o_i^e(x_i^e)) = Q_{\bar{\theta}}\left(\left\{o_j^e(x_j^e), r_j^e, Q_{\bar{\theta}}(h_{j-1}^e, o_{j-1}^e(x_{j-1}^e))\right\}_{j=1}^{i-1}, o_i^e(x_i^e)\right)$.
      **end for**
      Select $x_t^e := \operatorname{argmax}_{x \in \mathbb{X}} Q_{\widehat{\theta}}(h_t^e, o_t^e(x))$
    **end if**
    $s_{t+1}^e, r_t^e = \mathcal{E}$.step($x_t^e$)
  **end for**
  **if** e mod $r_{\text{target}} == 0$ **then**
    $Q_{\bar{\theta}} = Q_{\boldsymbol{\theta}_e}$
  **end if**
  $\mathcal{B}$.append($\tau_e = \{o_i^e(x_i^e), r_i^e\}_{i=1}^T$)
  Sample B batches from $\mathcal{B}$.
  **for** $b = 1, 2, \cdots, B$ **do**
    Compute $Q_{\bar{\theta}}^b(h_1^b, o_1^b(x_1^b)) = Q_{\bar{\theta}}\left(o_1^b(x_1^b)\right)$
    **for** $i = 1, 2, \cdots, T-1$ **do**
      Compute $Q_{\widehat{\theta}}^b(h_i^b, o_i^b(x_i^b)) = Q_{\widehat{\theta}}\left(\left\{o_i^b(x_j^b), r_j^b, Q_{\bar{\theta}}(h_{j-1}^b, o_{j-1}^b(x_{j-1}^b))\right\}_{j=1}^{i-1}, o_i^b(x_i^b)\right)$.
      Compute $Q_{\bar{\theta}}^b(h_i^b, o_i^b(x_i^b)) = Q_{\bar{\theta}}\left(\left\{o_j^b(x_j^b), r_j^b, Q_{\bar{\theta}}(h_{j-1}^b, o_{j-1}^b(x_{j-1}^b))\right\}_{j=1}^{i-1}, o_i^b(x_i^b)\right)$.
    **end for**
  **end for**
  Loss($\theta$) = $\sum_{b=1}^{B} \sum_{t=1}^{T-1} \left(Q_{\widehat{\theta}}^b(h_i^b, o_i^b(x_i^b)) - (r_t^b + \max_{x \in \mathbb{X}} Q_{\bar{\theta}}^b(h_i^b, o_i^b(x)))\right)^2$
  $\boldsymbol{\theta}_{e+1} = \operatorname{argmin}_{\boldsymbol{\theta}} \text{Loss}(\theta)$
**end for**

# D Detailed Experimental Results

Due to the space limit, in this section, we provide all the detailed experimental results as follows:

## D.1 Final hypervolume of Synthetic and YAHPO Problems

**Final hypervolume of Synthetic Problems:** The detailed results of final hypervolume of the six synthetic problems are shown in Table 2-3. We can observe that BOFormer constantly achieves the smallest or among the highest final hypervolume among all the MOBO algorithms.

**Final hypervolume of YAHPO Problems:** The detailed results of final hypervolume of the four YAHPO problems are shown in Table 4-5. Again, we can observe that BOFormer constantly achieves the smallest or among the smallest simple regret among all the MOBO algorithms throughout the sampling episode.

**Algorithm 2** On-Policy BOFormer

**Input:** $\boldsymbol{\theta}_1$, Training Episodes $E$, Time Horizon $T$, environment $\mathcal{E}$,
**for** $e = 1, 2, \cdots$,E **do**
  $\mathcal{E}$.reset()
  $s_1 = \mathcal{E}$.state
  **for** $t = 1, 2, \cdots, T$ **do**
    Select $x_t := \text{argmax}_{x \in \mathbb{X}} Q_{\theta_t}(h_t, o_t(x))$
    $o_{t+1}(x), r_t = \mathcal{E}$.step($x_t$)
    $\theta_{t+1} = \underset{\boldsymbol{\theta}}{\text{argmin}} \left( Q_\theta(h_t, o_t(x_t)) - (r_t + \gamma \max_{x \in \mathbb{X}} Q_{\boldsymbol{\theta}_t}(h_{t+1}, o_{t+1}(x))) \right)^2$
  **end for**
**end for**

### D.2 Detailed Regret Statistics

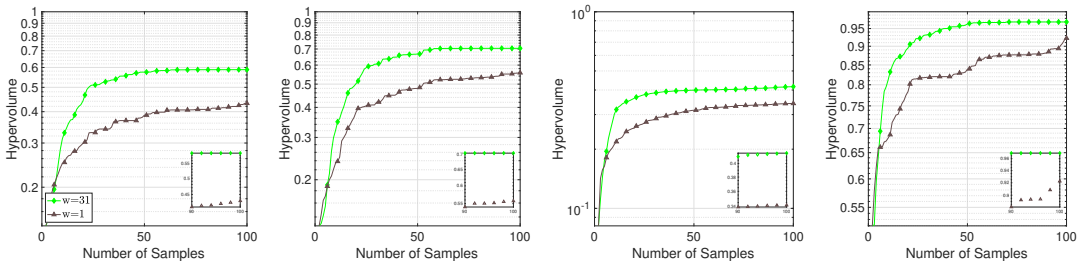
Due to the space limit, in this section, we further provide all the detailed statistics of the final simple regrets:

**Regret Statistics of Synthetic Problems:** The results are provided in Table 2-3.

**Regret Statistics of YAHPO Gym Problems:** The results are provided in Table 4-5.

### D.3 A Study on the Effect of Sequence Length on BOFormer

As mentioned in the main text, we also conduct an experiment on evaluating how the sequence length would affect the hypervolume performance of BOFormer. Figure 5 below shows that the hypervolume of non-Markovian BOFormer is better than Markovian BOFormer.




(a) AR (b) ARa (c) BC (d) DR

Figure 5: Difference length of transformer used in BOFormer

### D.4 A Study on Computational Efficiency

In this experiment, we provide the computation time per step for each method in Table 1. We observed that both BOFormer and qHVKG are competitive methods in terms of final hypervolume as shown in Table 2-5, with BOFormer demonstrating lower computation time than qHVKG.

Table 1: Average inference time for one trial

| Algs | time(s) |
|---|---|
| BOFormer | 0.3540 |
| qNEHVI | 0.1005 |
| JES | 0.3776 |
| FSAF | 0.0428 |
| DT | 0.0201 |
| NSGA-II | 0.00016 |
| qParEGO | 0.0906 |
| OptFormer | 7.5222 |
| qHVKG | 2.8061 |
| QT | 0.3518 |

Table 2: Mean, standard deviation, and the percentiles of final hypervolume for learning-based methods under synthetic functions. The best value among all the learning-based and rule-based methods is highlighted in bold and underlined.

| | | AR | ARa | BC | DR | ARS | ASR |
|---|---|---|---|---|---|---|---|
| BOFormer | Mean | 0.5880 | 0.7046 | 0.4160 | **0.9693** | 0.6226 | **0.4384** |
| | Std. | 0.0608 | 0.1067 | 0.0208 | 0.0635 | 0.0695 | 0.0691 |
| | Q.90 | **0.6673** | 0.7939 | 0.4374 | **1.0536** | 0.7058 | 0.5012 |
| | Q.75 | 0.6239 | 0.7789 | 0.4269 | **1.0123** | **0.6801** | **0.4894** |
| | Q.50 | **0.5977** | **0.7587** | 0.4188 | **0.9560** | **0.6282** | **0.4716** |
| | Q.25 | **0.5768** | 0.5717 | 0.4006 | **0.9283** | **0.5939** | 0.3620 |
| | Q.10 | 0.4844 | 0.5374 | 0.3900 | 0.9066 | 0.5073 | 0.3359 |
| FSAF | Mean | 0.4454 | 0.4861 | 0.4164 | 0.9100 | 0.4972 | 0.3518 |
| | Std. | 0.0647 | 0.0834 | 0.0264 | 0.0985 | 0.1175 | 0.0685 |
| | Q.90 | 0.5310 | 0.5989 | 0.4459 | 0.9776 | 0.6152 | 0.4428 |
| | Q.75 | 0.4889 | 0.5351 | 0.4349 | 0.9589 | 0.5773 | 0.3885 |
| | Q.50 | 0.4521 | 0.4785 | 0.4134 | 0.9225 | 0.4718 | 0.3319 |
| | Q.25 | 0.3996 | 0.4338 | 0.3983 | 0.9007 | 0.4294 | 0.3071 |
| | Q.10 | 0.3627 | 0.3967 | 0.3909 | 0.8495 | 0.3694 | 0.2892 |
| OptFormer | Mean | 0.4752 | 0.4938 | 0.4124 | 0.9336 | 0.4387 | 0.2807 |
| | Std. | 0.0703 | 0.0890 | 0.0269 | 0.0677 | 0.0623 | 0.0532 |
| | Q.90 | 0.5432 | 0.5380 | 0.4501 | 1.0125 | 0.5100 | 0.3284 |
| | Q.75 | 0.5106 | 0.5263 | 0.4322 | 0.9810 | 0.4661 | 0.3093 |
| | Q.50 | 0.4760 | 0.4949 | 0.4056 | 0.9397 | 0.4276 | 0.2736 |
| | Q.25 | 0.4561 | 0.4305 | 0.3930 | 0.9057 | 0.3933 | 0.2508 |
| | Q.10 | 0.3791 | 0.3923 | 0.3821 | 0.8343 | 0.3673 | 0.2243 |
| DT | Mean | 0.4207 | 0.4935 | 0.4105 | 0.9241 | 0.4469 | 0.3045 |
| | Std. | 0.0797 | 0.0756 | **0.0206** | 0.0504 | 0.0787 | 0.0535 |
| | Q.90 | 0.5005 | 0.5953 | 0.4382 | 0.9870 | 0.4918 | 0.3473 |
| | Q.75 | 0.4860 | 0.5321 | 0.4252 | 0.9626 | 0.4736 | 0.3252 |
| | Q.50 | 0.4336 | 0.4876 | 0.4073 | 0.9211 | 0.4466 | 0.3058 |
| | Q.25 | 0.3434 | 0.4702 | 0.3929 | 0.8824 | 0.4091 | 0.2673 |
| | Q.10 | 0.3192 | 0.3666 | 0.3895 | 0.8592 | 0.3579 | 0.2447 |
| QT | Mean | 0.4519 | 0.4989 | 0.4217 | 0.9035 | 0.4686 | 0.3218 |
| | Std. | 0.0573 | 0.0855 | 0.0259 | 0.0527 | 0.0841 | 0.0537 |
| | Q.90 | 0.5241 | 0.5961 | 0.4502 | 0.9569 | 0.5891 | 0.3972 |
| | Q.75 | 0.4844 | 0.5386 | 0.4382 | 0.9272 | 0.5309 | 0.3389 |
| | Q.50 | 0.4475 | 0.4700 | 0.4237 | 0.9128 | 0.4425 | 0.3115 |
| | Q.25 | 0.4226 | 0.4436 | 0.4018 | 0.8771 | 0.4078 | 0.2896 |
| | Q.10 | 0.3773 | 0.4259 | 0.3926 | 0.8325 | 0.3646 | 0.2690 |

Table 3: Mean, standard deviation, and the percentiles of final hypervolume for rule-based methods under synthetic functions. The best value among all the learning-based and rule-based methods is highlighted in bold and underlined.

| | | AR | ARa | BC | DR | ARS | ASR |
|---|---|---|---|---|---|---|---|
| qHVKG | Mean | **0.5956** | 0.7029 | 0.4795 | 0.9402 | **0.6275** | 0.4284 |
| | Std. | **0.0478** | 0.0708 | 0.0208 | 0.0480 | 0.0644 | 0.0533 |
| | Q.90 | 0.6564 | 0.7828 | 0.5094 | 1.0041 | **0.7092** | 0.4763 |
| | Q.75 | **0.6252** | 0.7422 | 0.4897 | 0.9802 | 0.6676 | 0.4574 |
| | Q.50 | 0.5899 | 0.7122 | 0.4725 | 0.9402 | 0.6147 | 0.4464 |
| | Q.25 | 0.5586 | **0.6640** | 0.4676 | 0.9037 | 0.5921 | **0.4032** |
| | Q.10 | **0.5387** | 0.6138 | 0.4611 | 0.8824 | **0.5642** | 0.3669 |
| NSGA2 | Mean | 0.5106 | **0.7132** | 0.4385 | 0.9578 | 0.5774 | 0.3840 |
| | Std. | 0.1327 | 0.1084 | 0.0433 | 0.0452 | 0.0984 | 0.1087 |
| | Q.90 | 0.6426 | **0.8250** | 0.4944 | 1.0179 | 0.7055 | **0.5032** |
| | Q.75 | 0.6082 | **0.7897** | 0.4772 | 0.9835 | 0.6208 | 0.4652 |
| | Q.50 | 0.5247 | 0.7274 | 0.4331 | 0.9527 | 0.5938 | 0.4057 |
| | Q.25 | 0.4585 | 0.6606 | 0.4050 | 0.9276 | 0.5487 | 0.3038 |
| | Q.10 | 0.3177 | **0.6183** | 0.4009 | **0.9102** | 0.4340 | 0.2037 |
| qNEHVI | Mean | 0.5507 | 0.6368 | **0.4911** | 0.9301 | 0.5865 | 0.4210 |
| | Std. | 0.0487 | 0.0740 | 0.0210 | **0.0447** | 0.0572 | **0.0439** |
| | Q.90 | 0.6067 | 0.7222 | **0.5151** | 0.9866 | 0.6522 | 0.4866 |
| | Q.75 | 0.5823 | 0.6864 | **0.5062** | 0.9633 | 0.6144 | 0.4477 |
| | Q.50 | 0.5514 | 0.6716 | **0.4937** | 0.9245 | 0.5926 | 0.4148 |
| | Q.25 | 0.5230 | 0.5815 | **0.4789** | 0.9042 | 0.5609 | 0.3862 |
| | Q.10 | 0.4967 | 0.5378 | **0.4629** | 0.8660 | 0.5114 | **0.3721** |
| JES | Mean | 0.5206 | 0.6676 | 0.4607 | 0.9197 | 0.5706 | 0.3883 |
| | Std. | 0.0774 | 0.0781 | 0.0232 | 0.0535 | 0.0662 | 0.0565 |
| | Q.90 | 0.6057 | 0.7587 | 0.4827 | 0.9731 | 0.6628 | 0.4678 |
| | Q.75 | 0.5702 | 0.7042 | 0.4735 | 0.9495 | 0.6030 | 0.4247 |
| | Q.50 | 0.5300 | 0.6786 | 0.4645 | 0.9156 | 0.5628 | 0.3840 |
| | Q.25 | 0.4978 | 0.6224 | 0.4432 | 0.8899 | 0.5359 | 0.3467 |
| | Q.10 | 0.4425 | 0.5716 | 0.4327 | 0.8706 | 0.5217 | 0.3186 |
| qParEGO | Mean | 0.5412 | 0.6212 | 0.4675 | 0.9228 | 0.5626 | 0.3884 |
| | Std. | 0.0633 | **0.0704** | 0.0258 | 0.0507 | **0.0341** | 0.0563 |
| | Q.90 | 0.6113 | 0.7155 | 0.4978 | 0.9656 | 0.6093 | 0.4664 |
| | Q.75 | 0.5929 | 0.6669 | 0.4886 | 0.9484 | 0.5852 | 0.4233 |
| | Q.50 | 0.5449 | 0.6225 | 0.4665 | 0.9147 | 0.5572 | 0.3734 |
| | Q.25 | 0.5073 | 0.5680 | 0.4563 | 0.8983 | 0.5361 | 0.3483 |
| | Q.10 | 0.4738 | 0.5321 | 0.4329 | 0.8649 | 0.5250 | 0.3230 |

Table 4: Mean, standard deviation, and the percentiles of final hypervolume for learning-based methods under hyperparameter optimization scenarios. The best value among all the learning-based and rule-based methods is highlighted in bold and underlined.

| | | lcbench | rbv2-xgboost | rbv2-svm | rbv2-glmnet |
|---|---|---|---|---|---|
| BOFormer | Mean | 0.1064 | 0.3495 | **0.1403** | 0.2364 |
| | Std. | 0.0003 | 0.0041 | 0.0005 | 0.0013 |
| | Q.90 | 0.1168 | **0.3974** | 0.1538 | 0.2535 |
| | Q.75 | 0.1161 | **0.3923** | **0.1532** | 0.2524 |
| | Q.50 | 0.1120 | 0.3697 | **0.1494** | **0.2479** |
| | Q.25 | 0.1033 | 0.3357 | 0.1363 | **0.2396** |
| | Q.10 | **0.0932** | 0.2813 | 0.1183 | 0.2142 |
| FSAF | Mean | 0.1007 | 0.3186 | 0.1331 | 0.2231 |
| | Std. | **0.0002** | 0.0024 | 0.0005 | 0.0012 |
| | Q.90 | 0.1095 | 0.3523 | 0.1466 | 0.2403 |
| | Q.75 | 0.1085 | 0.3500 | 0.1450 | 0.2374 |
| | Q.50 | 0.1060 | 0.3332 | 0.1401 | 0.2351 |
| | Q.25 | 0.0986 | 0.3020 | 0.1290 | 0.2218 |
| | Q.10 | 0.0860 | 0.2787 | 0.1188 | 0.2054 |
| OptFormer | Mean | 0.0996 | 0.3316 | 0.1309 | 0.2200 |
| | Std. | 0.0003 | 0.0025 | 0.0004 | 0.0013 |
| | Q.90 | 0.1101 | 0.3625 | 0.1438 | 0.2386 |
| | Q.75 | 0.1080 | 0.3590 | 0.1428 | 0.2368 |
| | Q.50 | 0.1057 | 0.3468 | 0.1358 | 0.2337 |
| | Q.25 | 0.0982 | 0.3273 | 0.1273 | 0.2242 |
| | Q.10 | 0.0839 | 0.2959 | 0.1155 | 0.1893 |

Table 5: Mean, standard deviation, and the percentiles of final hypervolume for rule-based methods under hyperparameter optimization scenarios. The best value among all the learning-based and rule-based methods is highlighted in bold and underlined.

| | | lcbench | rbv2-xgboost | rbv2-svm | rbv2-glmnet |
|---|---|---|---|---|---|
| qHVKG | Mean | 0.1035 | 0.3448 | 0.1380 | 0.2292 |
| | Std. | 0.0003 | 0.0032 | 0.0004 | 0.0014 |
| | Q.90 | 0.1140 | 0.3801 | 0.1511 | 0.2500 |
| | Q.75 | 0.1127 | 0.3786 | 0.1488 | 0.2472 |
| | Q.50 | 0.1089 | 0.3671 | 0.1447 | 0.2406 |
| | Q.25 | 0.1020 | 0.3350 | 0.1355 | 0.2328 |
| | Q.10 | 0.0881 | 0.2810 | 0.1224 | 0.2037 |
| NSGA2 | Mean | 0.1048 | 0.3514 | 0.1376 | 0.2336 |
| | Std. | 0.0002 | **0.0022** | **0.0004** | **0.0009** |
| | Q.90 | 0.1153 | 0.3796 | 0.1501 | 0.2497 |
| | Q.75 | 0.1136 | 0.3762 | 0.1485 | 0.2480 |
| | Q.50 | 0.1092 | 0.3683 | 0.1431 | 0.2439 |
| | Q.25 | 0.1032 | 0.3450 | 0.1354 | 0.2334 |
| | Q.10 | 0.0870 | 0.2904 | 0.1140 | 0.1957 |
| qNEHVI | Mean | **0.1076** | 0.3532 | 0.1395 | **0.2364** |
| | Std. | 0.0003 | 0.0033 | 0.0005 | 0.0014 |
| | Q.90 | 0.1176 | 0.3913 | 0.1509 | **0.2540** |
| | Q.75 | 0.1170 | 0.3871 | 0.1493 | **0.2525** |
| | Q.50 | **0.1146** | **0.3739** | 0.1469 | 0.2477 |
| | Q.25 | **0.1092** | 0.3436 | 0.1384 | 0.2392 |
| | Q.10 | 0.0905 | **0.2984** | **0.1260** | **0.2185** |
| JES | Mean | 0.1061 | 0.3529 | 0.1394 | 0.2328 |
| | Std. | 0.0003 | 0.0032 | 0.0005 | 0.0015 |
| | Q.90 | 0.1177 | 0.3935 | 0.1531 | 0.2532 |
| | Q.75 | 0.1141 | 0.3885 | 0.1520 | 0.2505 |
| | Q.50 | 0.1108 | 0.3736 | 0.1476 | 0.2453 |
| | Q.25 | 0.1039 | 0.3438 | 0.1358 | 0.2332 |
| | Q.10 | 0.0915 | 0.2970 | 0.1198 | 0.2025 |
| qParEGO | Mean | 0.1062 | **0.3537** | 0.1402 | 0.2338 |
| | Std. | 0.0003 | 0.0030 | 0.0006 | 0.0015 |
| | Q.90 | **0.1178** | 0.3874 | **0.1545** | 0.2530 |
| | Q.75 | **0.1174** | 0.3838 | 0.1528 | 0.2522 |
| | Q.50 | 0.1137 | 0.3724 | 0.1489 | 0.2478 |
| | Q.25 | 0.1020 | **0.3489** | **0.1409** | 0.2351 |
| | Q.10 | 0.0848 | 0.2970 | 0.1113 | 0.1965 |