Max-value Entropy Search for Multi-Objective Bayesian Optimization

Syrine Belakaria, Aryan Deshwal, Janardhan Rao Doppa
School of EECS, Washington State University
{syrine.belakaria, aryan.deshwal, jana.doppa}@wsu.edu

Abstract

We consider the problem of multi-objective (MO) blackbox optimization using expensive function evaluations, where the goal is to approximate the true pareto-set of solutions by minimizing the number of function evaluations. For example, in hardware design optimization, we need to find the designs that trade-off performance, energy, and area overhead using expensive computational simulations. In this paper, we propose a novel approach referred as *Max-value Entropy Search for Multi-objective Optimization (MESMO)* to solve this problem. MESMO employs an output-space entropy based acquisition function to efficiently select the sequence of inputs for evaluation to quickly uncover high-quality pareto-set solutions. We also provide theoretical analysis to characterize the efficacy of MESMO. Our experiments on several synthetic and real-world benchmark problems show that MESMO consistently outperforms the state-of-the-art algorithms.

1 Introduction

Many engineering and scientific applications involve making design choices to optimize multiple objectives. Some examples include tuning the knobs of a compiler to optimize performance and efficiency of a set of software programs; and designing new materials to optimize strength, elasticity, and durability. There are two common challenges in solving this kind of optimization problems: **1)** The objective functions are unknown and we need to perform expensive experiments to evaluate each candidate design choice. For example, performing computational simulations and physical lab experiments for compiler optimization and material design applications respectively. **2)** The objectives are conflicting in nature and all of them cannot be optimized simultaneously. Therefore, we need to find the *Pareto optimal* set of solutions. A solution is called Pareto optimal if it cannot be improved in any of the objectives without compromising some other objective. The overall goal is to approximate the optimal Pareto set by minimizing the number of function evaluations.

Bayesian Optimization (BO) [22] is an effective framework to solve blackbox optimization problems with expensive function evaluations. The key idea behind BO is to build a cheap surrogate model (e.g., Gaussian Process [28]) using the real experimental evaluations; and employ it to intelligently select the sequence of function evaluations using an acquisition function, e.g., expected improvement (EI). There is a large body of literature on single-objective BO algorithms [22] and their applications including hyper-parameter tuning of machine learning methods [24, 12]. However, there is relatively less work on the more challenging problem of BO for multiple objective functions [7] as discussed in the related work section.

Prior work on multi-objective BO is lacking in the following ways. Many algorithms reduce the problem to single-objective optimization by designing appropriate acquisition functions, e.g., expected improvement in Pareto hypervolume [11, 5]. Unfortunately, this choice is sub-optimal as it can potentially lead to aggressive exploitation behavior. Additionally, algorithms to optimize Pareto Hypervolume (PHV) based acquisition functions scale poorly as the number of objectives and

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

dimensionality of input space grows. Other method relies on *input space entropy* based acquisition function [7] to select the candidate inputs for evaluation. However, it is computationally expensive to approximate and optimize this acquisition function.

In this paper, we propose a novel and principled approach referred as **M***ax-value* **E***ntropy* **S***earch for* **M***ulti-objective* **O***ptimization* (MESMO) to overcome the drawbacks of prior work. MESMO employs an *output space entropy* based acquisition function to select the candidate inputs for evaluation. The key idea is to evaluate the input that maximizes the information gain about the optimal Pareto front in each iteration. Output space entropy search has many advantages over algorithms based on input space entropy search: a) allows much tighter approximation; b) significantly cheaper to compute; and c) naturally lends itself to robust optimization. Indeed, our experiments demonstrate these advantages of MESMO. Our work is inspired by the recent success of single-objective BO algorithms based on the idea of optimizing output-space information gain [26, 9], which are shown to be most efficient and robust among a family of information-theoretic acquisition functions [6, 8]. Specifically, we extend the max-value entropy search approach [26] to the challenging multi-objective setting.

Contributions. The main contributions of this paper are:

- Developing a principled approach referred as MESMO to solve multi-objective blackbox optimization problems. MESMO employs an output space entropy based acquisition function to efficiently select the sequence of candidate inputs for evaluation.
- Theoretical analysis of the MESMO algorithm in terms of asymptotic regret bounds.
- Comprehensive experiments over diverse synthetic and real-world benchmark problems to show accuracy and efficiency improvements over existing methods.

2 Background and Problem Setup

Bayesian Optimization (BO) Framework. BO is a very efficient framework to solve global optimization problems using *black-box evaluations of expensive objective functions*. Let $\mathfrak{X} \subseteq \Re^d$ be an input space. In single-objective BO formulation, we are given an unknown real-valued objective function $f : \mathfrak{X} \mapsto \Re$, which can evaluate each input $\mathbf{x} \in \mathfrak{X}$ to produce an evaluation $y = f(\mathbf{x})$. Each evaluation $f(\mathbf{x})$ is expensive in terms of the consumed resources. The main goal is to find an input $\mathbf{x}^* \in \mathfrak{X}$ that approximately optimizes f by performing a limited number of function evaluations. BO algorithms learn a cheap surrogate model from training data obtained from past function evaluations. They intelligently select the next input for evaluation by trading-off exploration and exploitation to quickly direct the search towards optimal inputs. The three key elements of BO framework are:

1) Statistical Model of the true function $f(x)$. *Gaussian Process (GP)* [28] is the most commonly used model. A GP over a space $\mathfrak{X}$ is a random process from $\mathfrak{X}$ to $\Re$. It is characterized by a mean function $\mu : \mathfrak{X} \mapsto \Re$ and a covariance or kernel function $\kappa : \mathfrak{X} \times \mathfrak{X} \mapsto \Re$. If a function f is sampled from GP(μ, κ), then $f(x)$ is distributed normally $\mathcal{N}(\mu(x), \kappa(x, x))$ for a finite set of inputs from $x \in \mathcal{X}$.

2) Acquisition Function (α) to score the utility of evaluating a candidate input $\mathbf{x} \in \mathfrak{X}$ based on the statistical model. Some popular acquisition functions in the single-objective literature include expected improvement (EI), upper confidence bound (UCB), predictive entropy search (PES) [8], and max-value entropy search (MES) [26].

3) Optimization Procedure to select the best scoring candidate input according to α depending on statistical model. DIRECT [10] is a very popular approach for acquisition function optimization.

Multi-Objective Optimization (MOO) Problem. Without loss of generality, our goal is to minimize real-valued objective functions $f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_K(\mathbf{x})$, with $K \geq 2$, over continuous space $\mathfrak{X} \subseteq \Re^d$. Each evaluation of an input $\mathbf{x} \in \mathfrak{X}$ produces a vector of objective values $\mathbf{y} = (y^1, y^2, \cdots, y^K)$ where $y^i = f_i(x)$ for all $i \in \{1, 2, \cdots, K\}$. We say that a point $\mathbf{x}$ *Pareto-dominates* another point $\mathbf{x}'$ if $f_i(\mathbf{x}) \leq f_i(\mathbf{x}')\ \forall i$ and there exists some $j \in \{1, 2, \cdots, K\}$ such that $f_j(\mathbf{x}) < f_j(\mathbf{x}')$. The optimal solution of MOO problem is a set of points $\mathcal{X}^* \subset \mathfrak{X}$ such that no point $\mathbf{x}' \in \mathfrak{X} \setminus \mathcal{X}^*$ Pareto-dominates a point $\mathbf{x} \in \mathcal{X}^*$. The solution set $\mathcal{X}^*$ is called the optimal *Pareto set* and the corresponding set of function values $\mathcal{Y}^*$ is called the optimal *Pareto front*. Our goal is to approximate $\mathcal{X}^*$ by minimizing the number of function evaluations.

3 Related work

There is a family of model based multi-objective BO algorithms that reduce the problem to single-objective optimization. ParEGO method [11] employs random scalarization for this purpose: scalar weights of K objective functions are sampled from a uniform distribution to construct a single-objective function and expected improvement is employed as the acquisition function to select the next input for evaluation. ParEGO is simple and fast, but more advanced approaches often outperform it. Many methods optimize the Pareto hypervolume (PHV) metric [5] that captures the quality of a candidate Pareto set. This is done by extending the standard acquisition functions to PHV objective, e.g., expected improvement in PHV (EHI) [5] and probability of improvement in PHV (SUR)[17]. Unfortunately, algorithms to optimize PHV based acquisition functions scale very poorly and are not feasible for more than two objectives. SMSego is a relatively faster method [19]. To improve scalability, the gain in hypervolume is computed over a limited set of points: SMSego finds those set of points by optimizing the posterior means of the GPs. A common drawback of this family of algorithms is that reduction to single-objective optimization can potentially lead to more exploitation behavior resulting in sub-optimal solutions.

PAL [31] and PESMO [7] are principled algorithms based on information theory. PAL tries to classify the input points based on the learned models into three categories: Pareto optimal, non-Pareto optimal, and uncertain. In each iteration, it selects the candidate input for evaluation towards the goal of minimizing the size of uncertain set. PAL provides theoretical guarantees, but it is only applicable for input space $\mathfrak{X}$ with finite set of discrete points. PESMO [7] relies on input space entropy based acquisition function and iteratively selects the input that maximizes the information gained about the optimal Pareto set $\mathcal{X}^*$. Unfortunately, optimizing this acquisition function poses significant challenges: a) requires a series of approximations, which can be potentially sub-optimal; and b) optimization, even after approximations, is expensive c) performance is strongly dependent on the number of Monte-Carlo samples. In comparison, our proposed output space entropy based acquisition function overcomes the above challenges, and allows efficient and robust optimization. More specifically, the time complexities of acquisition function computation in PESMO and MESMO ignoring the time to solve cheap MO problem that is common for both algorithms are $\mathcal{O}(SKm^3)$ and $\mathcal{O}(SK)$ respectively, where S is the number of Monte-Carlo samples, K is the number of objectives, and m is the size of the sample Pareto set in PESMO. Additionally, as demonstrated in our experiments, MESMO is very robust and performs very well even with one sample.

4 MESMO Algorithm for Multi-Objective Optimization

In this section, we explain the technical details of our proposed MESMO algorithm. We first mathematically describe the output space entropy based acquisition function and provide an algorithmic approach to efficiently compute it. Subsequently, we theoretically analyze MESMO in terms of asymptotic regret bounds.

Surrogate models. Gaussian processes (GPs) are shown to be effective surrogate models in prior work on single and multi-objective BO [8, 27, 26, 25, 7]. Similar to prior work [7], we model the objective functions $f_1, f_2, \cdots, f_K$ using K independent GP models $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_K$ with zero mean and i.i.d. observation noise. Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{t-1}$ be the training data from past $t-1$ function evaluations, where $\mathbf{x}_i \in \mathfrak{X}$ is an input and $\mathbf{y}_i = \{y_i^1, y_i^2, \cdots, y_i^K\}$ is the output vector resulting from evaluating functions $f_1, f_2, \cdots, f_K$ at $\mathbf{x}_i$. We learn surrogate models $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_K$ from $\mathcal{D}$.

Output space entropy based acquisition function. Input space entropy based methods like PESMO [7] selects the next candidate input $\mathbf{x}_t$ (for ease of notation, we drop the subscript in below discussion) by maximizing the information gain about the optimal Pareto set $\mathcal{X}^*$. The acquisition function based on input space entropy is given as follows:

$$\alpha(\mathbf{x}) = I(\{\mathbf{x}, \mathbf{y}\}, \mathcal{X}^* \mid D) \tag{4.1}$$

$$= H(\mathcal{X}^* \mid D) - \mathbb{E}_y[H(\mathcal{X}^* \mid D \cup \{\mathbf{x}, \mathbf{y}\})] \tag{4.2}$$

$$= H(\mathbf{y} \mid D, \mathbf{x}) - \mathbb{E}_{\mathcal{X}^*}[H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{X}^*)] \tag{4.3}$$

Information gain is defined as the expected reduction in entropy $H(.)$ of the posterior distribution $P(\mathcal{X}^* \mid D)$ over the optimal Pareto set $\mathcal{X}^*$ as given in Equations 4.2 and 4.3 (resulting from symmetric property of information gain). This mathematical formulation relies on a very expensive

and high-dimensional ($m \cdot d$ dimensions) distribution $P(\mathcal{X}^* \mid D)$, where m is size of the optimal Pareto set $\mathcal{X}^*$. Furthermore, optimizing the second term in r.h.s poses significant challenges: a) requires a series of approximations [7] which can be potentially sub-optimal; and b) optimization, even after approximations, is expensive c) performance is strongly dependent on the number of Monte-Carlo samples.

To overcome the above challenges of computing input space entropy based acquisition function, we take an alternative route and propose to maximize the information gain about the optimal **Pareto front** $\mathcal{Y}^*$. This is equivalent to expected reduction in entropy over the Pareto front $\mathcal{Y}^*$, which relies on a computationally cheap and low-dimensional ($m \cdot K$ dimensions, which is significantly less than $m \cdot d$ as $K \ll d$ in practice) distribution $P(\mathcal{Y}^* \mid D)$. Our acquisition function that maximizes the information gain between the next candidate input for evaluation $\mathbf{x}$ and Pareto front $\mathcal{Y}^*$ is given as:

$$\alpha(\mathbf{x}) = I(\{\mathbf{x}, \mathbf{y}\}, \mathcal{Y}^* \mid D) \tag{4.4}$$
$$= H(\mathcal{Y}^* \mid D) - \mathbb{E}_y[H(\mathcal{Y}^* \mid D \cup \{\mathbf{x}, \mathbf{y}\})] \tag{4.5}$$
$$= H(\mathbf{y} \mid D, \mathbf{x}) - \mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}^*)] \tag{4.6}$$

The first term in the r.h.s of equation 4.6 (entropy of a factorizable K-dimensional gaussian distribution $P(\mathbf{y} \mid D, \mathbf{x})$) can be computed in closed form as shown below:

$$H(\mathbf{y} \mid D, \mathbf{x}) = \frac{K(1 + \ln(2\pi))}{2} + \sum_{i=1}^{K} \ln(\sigma_i(\mathbf{x})) \tag{4.7}$$

where $\sigma_i^2(\mathbf{x})$ is the predictive variance of i^{th} GP at input $\mathbf{x}$. The second term in the r.h.s of equation 4.6 is an expectation over the Pareto front $\mathcal{Y}^*$. We can approximately compute this term via Monte-Carlo sampling as shown below:

$$\mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}^*)] \simeq \frac{1}{S} \sum_{s=1}^{S} [H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*)] \tag{4.8}$$

where S is the number of samples and $\mathcal{Y}_s^*$ denote a sample Pareto front. The main advantages of our acquisition function are: computational efficiency and robustness to the number of samples. Our experiments demonstrate these advantages over input space entropy based acquisition function.

There are two key algorithmic steps to compute Equation 4.8: 1) How to compute Pareto front samples $\mathcal{Y}_s^*$?; and 2) How to compute the entropy with respect to a given Pareto front sample $\mathcal{Y}_s^*$? We provide solutions for these two questions below.

1) Computing Pareto front samples via cheap multi-objective optimization. To compute a Pareto front sample $\mathcal{Y}_s^*$, we first sample functions from the posterior GP models via random fourier features [8, 20] and then solve a cheap multi-objective optimization over the K sampled functions.

Sampling functions from posterior GP. Similar to prior work [8, 7, 26], we employ random fourier features based sampling procedure. We approximate each GP prior as $\tilde{f} = \phi(\mathbf{x})^T\theta$, where $\theta \sim N(0, \mathbf{I})$. The key idea behind random fourier features is to construct each function sample $\tilde{f}(\mathbf{x})$ as a finitely parametrized approximation: $\phi(\mathbf{x})^T\theta$, where θ is sampled from its corresponding posterior distribution conditioned on the data $\mathcal{D}$ obtained from past function evaluations: $\theta|\mathcal{D} \sim N(\mathbf{A^{-1}\Phi^T y}_n, \sigma^2\mathbf{A^{-1}})$, where $\mathbf{A} = \mathbf{\Phi^T\Phi} + \sigma^2\mathbf{I}$ and $\Phi^T = [\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_{t-1})]$.

Cheap MO solver. We sample $\tilde{f}_i$ from GP model $\mathcal{M}_i$ for each of the K functions as described above. A *cheap* multi-objective optimization problem over the K sampled functions $\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f}_k$ is solved to compute sample Pareto front $\mathcal{Y}_s^*$. This cheap multi-objective optimization also allows us to capture the interactions between different objectives. We employ the popular NSGA-II algorithm [3] to solve the MO problem with cheap objective functions noting that any other algorithm can be used to similar effect.

2) Entropy computation with a sample Pareto front. Let $\mathcal{Y}_s^* = \{\mathbf{z}_1, \cdots, \mathbf{z}_m\}$ be the sample Pareto front, where m is the size of the Pareto front and each $\mathbf{z}_i = \{z_i^1, \cdots, z_i^K\}$ is a K-vector evaluated at the K sampled functions. The following inequality holds for each component y^j of the K-vector $\mathbf{y} = \{y^1, \cdots, y^K\}$ in the entropy term $H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*)$:

$$y^j \leq \max\{z_1^j, \cdots z_m^j\} \quad \forall j \in \{1, \cdots, K\} \tag{4.9}$$

The inequality essentially says that the j^{th} component of $\mathbf{y}$ (i.e., y^j) is upper-bounded by a value obtained by taking the maximum of j^{th} components of all m K-vectors in the Pareto front $\mathcal{Y}_s^*$. This inequality can be proven by a contradiction argument. Suppose there exists some component y^j of $\mathbf{y}$ such that $y^j > \max\{z_1^j, \cdots z_m^j\}$. However, by definition, $\mathbf{y}$ is a non-dominated point because no point dominates it in the jth dimension. This results in $\mathbf{y} \in \mathcal{Y}_s^*$ which is a contradiction. Therefore, our hypothesis that $y^j > \max\{z_1^j, \cdots z_m^j\}$ is incorrect and inequality 4.9 holds.

By combining the inequality 4.9 and the fact that each function is modeled as a GP, we can model each component y^j as a truncated Gaussian distribution since the distribution of y^j needs to satisfy $y^j \leq \max\{z_1^j, \cdots z_m^j\}$. Furthermore, a common property of entropy measure allows us to decompose the entropy of a set of independent variables into a sum over entropies of individual variables [2]:

$$H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*) \simeq \sum_{j=1}^{K} H(y^j | D, \mathbf{x}, \max\{z_1^j, \cdots z_m^j\}) \tag{4.10}$$

Equation 4.10 and the fact that the entropy of a truncated Gaussian distribution[14] can be computed in closed form gives the following mathematical expression for the entropy term $H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*)$. We provide the complete details of the derivation in the **Appendix**.

$$H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*) \simeq \sum_{j=1}^{K} \left[\frac{(1 + \ln(2\pi))}{2} + \ln(\sigma_j(\mathbf{x})) + \ln \Phi(\gamma_s^j(\mathbf{x})) - \frac{\gamma_s^j(\mathbf{x})\phi(\gamma_s^j(\mathbf{x}))}{2\Phi(\gamma_s^j(\mathbf{x}))} \right] \tag{4.11}$$

where $\gamma_s^j(x) = \frac{y_s^{j*} - \mu_j(\mathbf{x})}{\sigma_j(\mathbf{x})}$, $y_s^{j*} = \max\{z_1^j, \cdots z_m^j\}$, and ϕ and Φ are the p.d.f and c.d.f of a standard normal distribution respectively. By combining equations 4.7 and 4.11 with Equation 4.6, we get the final form of our acquisition function as shown below:

$$\alpha(\mathbf{x}) \simeq \frac{1}{S} \sum_{s=1}^{S} \sum_{j=1}^{K} \left[\frac{\gamma_s^j(\mathbf{x})\phi(\gamma_s^j(\mathbf{x}))}{2\Phi(\gamma_s^j(\mathbf{x}))} - \ln \Phi(\gamma_s^j(\mathbf{x})) \right] \tag{4.12}$$

A complete description of the MESMO algorithm is given in Algorithm 1. The blue colored steps correspond to computation of our output space entropy based acquisition function via sampling.

Algorithm 1 MESMO Algorithm

Input: input space $\mathfrak{X}$; K blackbox objective functions $f_1(x), f_2(x), \cdots, f_K(x)$; and maximum no. of iterations T_{max}

Initialize Gaussian process models $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_K$ by evaluating at N_0 initial points
for each iteration $t = N_0 + 1$ to T_{max} **do**
  Select $\mathbf{x}_t \leftarrow \arg max_{\mathbf{x} \in \mathfrak{X}} \ \alpha_t(\mathbf{x})$, where $\alpha_t(.)$ is computed as:
    for each sample $s \in 1, \cdots, S$:
      Sample $\tilde{f}_i \sim \mathcal{M}_i, \quad \forall i \in \{1, \cdots, K\}$
      $\mathcal{Y}_s^* \leftarrow$ Pareto front of *cheap* multi-objective optimization over $(\tilde{f}_1, \cdots, \tilde{f}_K)$
    Compute $\alpha_t(.)$ based on the S samples of $\mathcal{Y}_s^*$ as given in Equation 4.12
  Evaluate $\mathbf{x}_t$: $\mathbf{y}_t \leftarrow (f_1(\mathbf{x}_t), \cdots, f_K(\mathbf{x}_t))$
  Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_t, \mathbf{y}_t)\}$
  Update models $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_K$
  $t \leftarrow t + 1$
end for
return Pareto front of $f_1(x), f_2(x), \cdots, f_K(x)$ based on $\mathcal{D}$

4.1 Theoretical Analysis

In this section, we provide a theoretical analysis for the behavior of MESMO algorithm. Multi-objective optimization literature has multiple metrics to assess the quality of Pareto front approximation. The two commonly employed metrics include Pareto Hypervolume indicator [29] and R_2 indicator[18]. R_2 indicator is a natural extension of the cumulative regret measure in single-objective

BO as proposed in the well-known work by Srinivasan et al., [25] to prove convergence results. Prior work [17] has shown that R_2 and Pareto Hypervolume indicator show similar behavior. Indeed, our experiments validate this claim for MESMO. Therefore, we present the theoretical analysis of MESMO with respect to R_2 indicator. Let $\mathbf{x}^*$ be a point in the optimal Pareto set $\mathcal{X}^*$. Let $\mathbf{x}_t$ be a point selected for evaluation by MESMO at the t^{th} iteration. Let $R(\mathbf{x}^*) = \|R^1, \cdots, R^K\|$, where $R^j = \sum_{t=1}^{T'}(f_j(\mathbf{x}^*) - f_j(\mathbf{x}_t))$ and $\|.\|$ is the norm of the K-vector. We discuss asymptotic bounds for this measure over the input set $\mathfrak{X}$.

Theorem 1. *Let P be a distribution over vector $[y^{1*}, \cdots, y^{K*}]$, where each y^{j*} is the maximum value for function f_j among the vectors in the Pareto front obtained by solving the cheap multi-objective optimization problem over sampled functions from the K Gaussian process models. Let the observation noise for function evaluations be i.i.d $\mathcal{N}(0, \sigma)$ and $w = Pr[\left(y^{1*} > f_1(x^*)\right), \cdots, \left(y^{K*} > f_K(x^*)\right)]$. If $\mathbf{x}_t$ is the candidate input selected by MESMO at the t^{th} iteration according to 4.12 and $[y^{1*}, \cdots, y^{K*}]$ is drawn from P, then with probability atleast $1 - \delta$, in $T' = \sum_{i=1}^{T} \log_w \frac{\delta}{2\pi_i}$ number of iterations*

$$R(\mathbf{x}^*) = \sqrt{\sum_{j=1}^{K}\left(\left(v_{t^*}^j + \zeta_T\right)^2\left(\frac{2T\gamma_T^j}{log(1+\sigma^{-2})}\right)\right)} \tag{4.13}$$

where $\zeta_T = (2\log(\pi_T/\delta))^{1/2}$, $\pi_i > 0$, and $\sum_{i=1}^{T}\frac{1}{\pi_i} \leq 1$, $v_{t^*}^j = \max_t v_t^j$ with $v_t^j = \min_{x\in\mathfrak{X}} \frac{y^{j*} - \mu_{j,t-1}(\mathbf{x})}{\sigma_{j,t-1}(\mathbf{x})}$, and γ_T^j is the maximum information gain about function f_j after T function evaluations.

We provide details of the proof in the **Appendix**. The key message of this result is that since each term R^j in $R(\mathbf{x}^*)$ grows sub-linearly in the asymptotic sense, $R(\mathbf{x}^*)$ which is defined as the norm also grows sub-linearly.

5 Experiments and Results

In this section, we describe our experimental setup, present results of MESMO on diverse synthetic and real-world benchmarks, and compare MESMO with existing methods.

5.1 Experimental Setup

Multi-objective BO algorithms. We compare MESMO with existing methods described in the related work: ParEGO [11], PESMO [7], SMSego [19], EHI [5], and SUR [17]. We employ the code for these methods from the BO library Spearmint[1]. For methods requiring PHV computation, we employ the *PyGMO* library[2]. According to PyGMO documentation, the algorithm from [15] is employed for PHV computation. We did not include PAL [31] as it is known to have similar performance as SMSego [7] and works only for finite discrete input space.

Statistical models. We use a GP based statistical model with squared exponential (SE) kernel in all our experiments. The hyper-parameters are estimated after every 5 function evaluations. We initialize the GP models for all functions by sampling initial points at random from a Sobol grid. This initialization procedure is same as the one in-built in the Spearmint library.

Synthetic benchmarks. We construct two synthetic multi-objective benchmark problems using a combination of commonly employed benchmark functions for single-objective optimization[3]. We also employ two benchmarks from the general multi-objective optimization literature [16, 4]. We provide the complete details of these MO benchmarks below.

1) BC-2,2: We evaluate two benchmark functions Branin and Currin. The dimension of input space d is 2.

2) PRDZPS-6,6: We evaluate six benchmark functions, namely, Powell, Rastrigin, Dixon, Zakharov, Perm, and SumSquares. The dimension of input space d is 6.

[1] https://github.com/HIPS/Spearmint/tree/PESM

[2] https://esa.github.io/pygmo/

[3] https://www.sfu.ca/ ssurjano/optimization.html

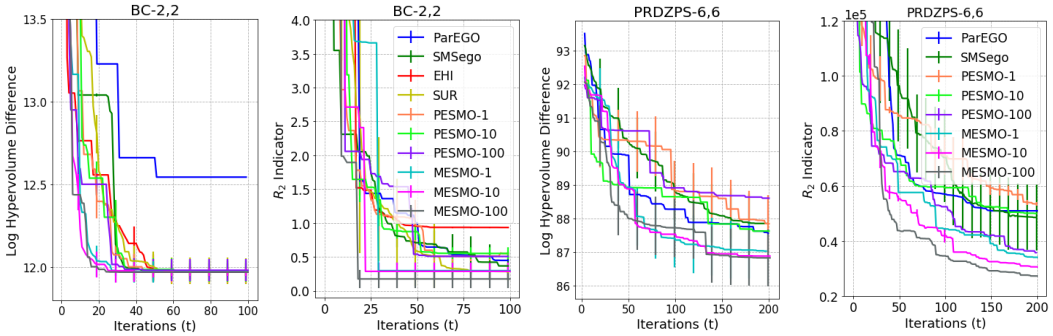

Figure 1: Results of different multi-objective BO algorithms including MESMO on synthetic benchmarks. The log of the hypervolume difference and the R_2 Indicator are shown with different number of function evaluations. The mean and variance of 10 different runs are plotted. The title of each figure refers to the name of benchmark. (Figures better seen in color.)

3) OKA2-2,3: We evaluate two functions defined in [16]. The dimension of input space d is 3.

4) DTLZ1-4,5: We evaluate four functions defined in [4]. The dimension of input space d is 5.

Real-world benchmarks. We employed four real-world benchmarks with data available at [31, 21].

1) Hyper-parameter tuning of neural networks. In this benchmark, our goal is to find a neural network with high accuracy and low prediction time. We optimize a dense neural network over the MNIST dataset [13]. Hyper-parameters include the number of hidden layers, the number of neurons per layer, the dropout probability, the learning rate, and the regularization weight penalties l_1 and l_2. We employ 10K instances for validation and 50K instances for training. We train the network for 100 epochs for evaluating each candidate hyper-parameter values on validation set. We apply a logarithm function to error rates due to their very small values.

2) SW-LLVM compiler settings optimization. SW-LLVM is a data set with 1024 compiler settings [23] determined by d=10 binary inputs. The goal of this experiment is to find a setting of the LLVM compiler that optimizes the memory footprint and performance on a given set of software programs. Evaluating these objectives is very costly and testing all the compiler settings takes days.

3) SNW sorting network optimization. The data set SNW was first introduced by [30]. The goal is to optimize the area and throughput for the synthesis of a field-programmable gate array (FPGA) platform. The input space consists of 206 different hardware design implementations of a sorting network. Each design is defined by $d = 4$ input variables.

4) Network-on-chip (NOC) optimization. The design space of NoC dataset [1] consists of 259 implementations of a tree-based network-on-chip. Each configuration is defined by $d = 4$ variables: width, complexity, FIFO, and multiplier. We optimize energy and runtime of application-specific integrated circuits (ASICs) on the Coremark benchmark workload [1].

Evaluation metrics. We employ two common metrics used in practice.

1) The Pareto hypervolume (PHV) is commonly employed to measure the quality of a given Pareto front [29]. PHV is defined as the volume between a reference point and the given Pareto front. After each iteration t , we report the difference between the hypervolume of the ideal Pareto front $(\mathcal{Y}^*)$ and hypervolume of the estimated Pareto front $(\mathcal{Y}_t)$ by a given algorithm.

$$PHV_{diff} = PHV(\mathcal{Y}^*) - PHV(\mathcal{Y}_t) \tag{5.1}$$

2) R_2 **Indicator** is the average distance the ideal Pareto front $(\mathcal{Y}^*)$ and the estimated Pareto front $(\mathcal{Y}_t)$ by a given algorithm [18]. R_2 is a distance based metric that degenerates to the regret metric presented in the theoretical analysis.

5.2 Results and Discussion

We run all experiments 10 times. The mean and variance of the PHV and R_2 metrics across different runs are reported as a function of the number of iterations.

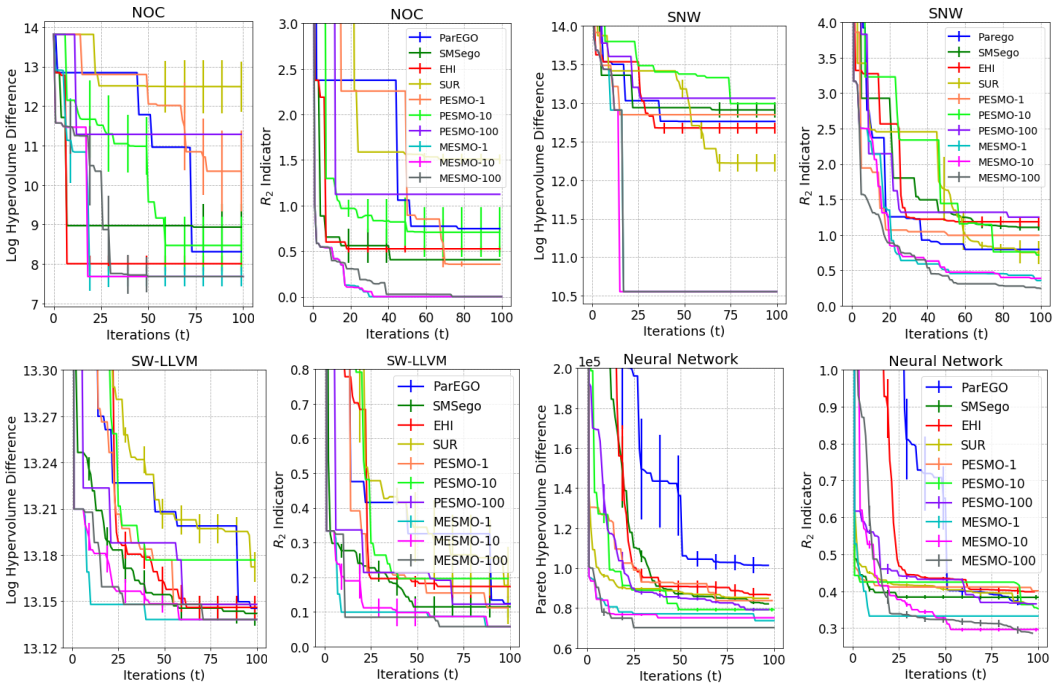

Figure 2: Results of different multi-objective BO algorithms including MESMO on real-world benchmarks. The log of the hypervolume difference and R_2 Indicator are shown with different number of function evaluations. The mean and variance of 10 different runs are plotted. The title of each figure refers to the name of real-world benchmark. (Figures better seen in color.)

MESMO vs. State-of-the-art. We evaluate the performance of MESMO and PESMO with different number of Monte-Carlo samples for acquisition function optimization. Figure 1 and Figure 2 show the results of all multi-objective BO algorithms including MESMO for synthetic and real-world benchmarks respectively. We present additional results of synthetic benchmarks in Figure 3 of the **Appendix**. We make the following empirical observations: 1) MESMO consistently performs better than all baselines and also converges much faster. For blackbox optimization problems with expensive function evaluations, faster convergence has practical benefits as it allows the end-user or decision-maker to stop early. 2) Rate of convergence of MESMO slighly varies with different number of Monte-Carlo samples. However, in all cases, MESMO performs better than baseline methods. 3) The convergence rate of PESMO is dramatically affected by the number of Monte-Carlo samples: 100 samples lead to better results than 10 and 1. In contrast, *MESMO maintains a better performance consistently even with a single sample!*. The results strongly demonstrate that MESMO is much more robust to the number of Monte-Carlo samples than PESMO. 4) Performance of ParEGO is very inconsistent. In some cases, it is comparable to MESMO, but performs poorly on many other cases. This is expected due to random scalarization.

Comparison of acquisition function optimization time. We compare the runtime of acquisition function optimization for different multi-objective BO algorithms including MESMO and PESMO (w/ different number of Monte-Carlo samples). We do not account for the time to fit GP models since it is same for all the algorithms. We measure the average acquisition function optimization time across all iterations. We run all experiments on a machine with the following configuration: Intel i7-7700K CPU @ 4.20GHz with 8 cores and 32 GB memory. Table 1 shows the time in seconds two for synthetic benchmarks. We present additional time comparison results in Figure 4 of the **Appendix**. We fix the input space dimensions to $d = 5$ and vary the number of objective functions to show how different algorithms scale with increasing number of objectives. We make the following observations: 1) The acquisition function optimization time of MESMO is significantly smaller than PESMO for the same number of samples. The difference between corresponding times grow significantly as the number of samples increase. 2) MESMO with one sample is comparable to ParEGO, which relies on scalarization to reduce to acquisition function optimization in single-objective BO. 3) The time

for PESMO and SMSego increases significantly as the number of objectives grow from two to six, whereas the corresponding growth in time is relatively small for MESMO.

Table 1: Average acquisition function optimization time in seconds.

MO Algorithm	BC-2,2	PRDZPS-6,6	MO Algorithm	BC-2,2	PRDZPS-6,6
MESMO-1	3.5±0.34	4.56±0.71	PESMO-1	13.6±3.2	110.4±17.8
MESMO-10	24.4±5.75	38.65± 0.65	PESMO-10	115.23±17.1	614.27±44
MESMO-100	242.434± 8.9	377.53± 4.29	PESMO-100	1128.3±15.3	6092.96±53.1
ParEGO	3.2± 1.6	5.3 ± 2.3	SMSego	80.5± 2.1	300.43 ± 35.7

6 Summary and Future Work

We introduced a novel and principled approach referred as MESMO to solve multi-objective Bayesian optimization problems. The key idea is to employ an output space entropy based acquisition function to efficiently select inputs for evaluation. Our comprehensive experimental results on both synthetic and real-world benchmarks showed that MESMO yields consistently better results than state-of-the-art methods, and is more efficient and robust than methods based on input space entropy search. Future work includes applying MESMO to solve novel engineering and scientific applications.

Acknowledgements. The authors gratefully acknowledge the support from National Science Foundation (NSF) grants IIS-1845922 and OAC-1910213. The views expressed are those of the authors and do not reflect the official policy or position of the NSF.

References

[1] Oscar Almer, Nigel Topham, and Björn Franke. A learning-based approach to the automated design of mpsoc networks. In *International Conference on Architecture of Computing Systems*, pages 243–258. Springer, 2011.

[2] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley and Sons, 2012.

[3] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, T Meyarivan, and A Fast. Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[4] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*, pages 105–145. Springer, 2005.

[5] Michael Emmerich and Jan-willem Klinkenberg. The computation of the expected improvement in dominated hypervolume of pareto front approximations. *Technical Report, Leiden University*, 34, 2008.

[6] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research (JMLR)*, 13(Jun):1809–1837, 2012.

[7] Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1492–1501, 2016.

[8] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pages 918–926, 2014.

[9] Matthew W Hoffman and Zoubin Ghahramani. Output-space predictive entropy search for flexible global optimization. In *NIPS workshop on Bayesian Optimization*, 2015.

[10] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[11] 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.

[12] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research (JMLR)*, 18(1):826–830, 2017.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Joseph Victor Michalowicz, Jonathan M Nichols, and Frank Bucholtz. *Handbook of differential entropy*. Chapman and Hall/CRC, 2013.

[15] Krzysztof Nowak, Marcus Märtens, and Dario Izzo. Empirical performance of the approximation of the least hypervolume contributor. In *International Conference on Parallel Problem Solving From Nature*, pages 662–671. Springer, 2014.

[16] Tatsuya Okabe, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. On test functions for evolutionary multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 792–802. Springer, 2004.

[17] Victor Picheny. Multi-objective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

[18] Victor Picheny, Tobias Wagner, and David Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, 2013.

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

[20] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

[21] Amar Shah and Zoubin Ghahramani. Pareto frontier learning with expensive correlated objectives. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1919–1927, 2016.

[22] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[23] Norbert Siegmund, Sergiy S Kolesnikov, Christian Kästner, Sven Apel, Don Batory, Marko Rosenmüller, and Gunter Saake. Predicting performance via automated feature-interaction detection. In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, pages 167–177, 2012.

[24] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[25] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[26] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, 2017.

[27] Zi Wang, Bolei Zhou, and Stefanie Jegelka. Optimization as estimation with gaussian processes in bandit settings. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1022–1031, 2016.

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

[29] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Ithaca: Shaker, 1999.

[30] Marcela Zuluaga, Peter Milder, and Markus Püschel. Computer generation of streaming sorting networks. In *Proceedings of Design Automation Conference (DAC)*, pages 1241–1249, 2012.

[31] Marcela Zuluaga, Guillaume Sergent, Andreas Krause, and Markus Püschel. Active learning for multi-objective optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 462–470, 2013.