NEW FRONTIERS IN ADAPTIVE EXPERIMENTAL DESIGN FOR MULTI-OBJECTIVE OPTIMIZATION

By

SYRINE BELAKARIA

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2023

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of SYRINE BELAKARIA find it satisfactory and recommend that it be accepted.

Janardhan Rao Doppa, Ph.D., Chair

Diane Cook, Ph.D.

Ananth Kalyanaraman, Ph.D.

Yolanda Gil, Ph.D.

Subbarao Kambhampati, Ph.D.

ACKNOWLEDGEMENTS

The outcome of my work has required a lot of guidance, assistance, and inspiration from many people to whom I would like to express special appreciation:

First and foremost, I heartily thank my advisor Prof. Jana Doppa, who believed in my potential and gave me the opportunity to work on various topics. Without his guidance and dedicated involvement, this work would have never been accomplished. I am thankful to him for involving me in several projects, and collaborations, and teaching me many other aspects of academic life. Working closely with him has been a valuable learning experience that contributed to my personal and professional growth. Beyond his scientific advice, Prof. Jana always offered unwavering support during the most mentally challenging moments of my PhD. Thank you for being a patient mentor and a great human being.

I have been fortunate to work with many amazing collaborators. Prof Yue Cao was an exemplary collaborator. I enjoyed working with him and Derek on applying Bayesian optimization to hard power engineering problems. Thank you for your continuous encouragement and support. To Prof. Partha Pande and Prof. Deuk Heo, I learned a lot about hardware design, analog circuit design, and their practical constraints from you. Thank you for contributing to my professional growth and understanding of fruitful academic collaborations. To Rishit Sheth and Nicolo Fusi who have been great mentors at Microsoft research, I enjoyed working with them on prac-

tical problems and learning more about the modeling side of Bayesian optimization. A special Thank you Prof. Yolanda Gil for her support and her advice.

I have been truly fortunate to work with Aryan Deshwal. He has been an invaluable collaborator and research partner, most of my work is a result of research discussions and brainstorming together. I learned a lot from him and with him about all aspects of research, from technical knowledge to software and most importantly enjoying every step of our collaboration. I look forward to continuing our work together.

My time at Washington State University would not have been the same without the supportive and thriving environment in our research group, the friendship of Alaleh Ahmadian and Iman Mirzadeh who made the hardship of the pandemic bearable, and the tremendous emotional support of Taha.

I gratefully acknowledge the support from National Science Foundation (NSF) grants IIS-1845922, OAC-1910213, and SII-2030159. I was also generously supported by the IBM PhD Fellowship for the last two years of my PhD studies (2021-2023). The views expressed in this dissertation are those of the authors and do not reflect the official policy or position of the NSF and IBM.

Finally, thank you to everyone who contributed to my personal and professional growth during my long academic journey.

NEW FRONTIERS IN ADAPTIVE EXPERIMENTAL DESIGN FOR MULTI-OBJECTIVE OPTIMIZATION

Abstract

by Syrine Belakaria, Ph.D.
Washington State University
May 2023

Chair: Janardhan Rao Doppa

The problem of adaptively selecting a sequence of experiments to achieve a goal (aka *adaptive experimental design*) arises in many real-world settings. A canonical example is the active learning paradigm where we need to iteratively collect labeled data to build predictors with high accuracy. Motivated by the challenges faced by scientists and engineers, this dissertation studies adaptive experimental design algorithms for the purpose of solving a large-class of multi-objective optimization (MOO) problems. Such MOO problems enable many science and engineering applications including drug design, protein engineering, design of materials, and hardware design. For example, in drug design optimization, we need to find drugs that trade-off effectiveness, safety, and cost by performing expensive experiments to evaluate each candidate drug. Similarly, in hardware design optimization, we need to find the

designs that trade-off performance, energy, and area using expensive computational simulations to mimic the real hardware.

We have the ability to evaluate any candidate input according to the target objectives by performing a costly experiment, where the cost is measured by the resources consumed by the experiment (physical or computational). Our overall goal is to approximate the optimal Pareto set of solutions by minimizing the total resource cost of conducted experiments. The key challenge is how to select the sequence of experiments under uncertainty. This dissertation develops a suite of novel reasoning algorithms based on the principles of information gain per unit resource cost and uncertainty reduction for adaptive experimental design to solve MOO problems. We appropriately instantiate these principles to derive efficient algorithms for the following MOO problem settings, most of which are studied for the first time: 1) The most basic *single-fidelity setting*, where experiments are expensive and accurate, and we can conduct a single experiment in each iteration; 2) The *batch setting* where a batch of experiments can be conducted in parallel to accelerate the search process; 3) The *constrained setting* where we cannot evaluate constraints to identify feasible inputs without performing experiments; 4) The *discrete multi-fidelity setting* where experiments can vary in the amount of resources consumed and their evaluation accuracy; and 5) The *continuous-fidelity setting*, where continuous function approximations result in a huge space of experiments. 6) The *budget-aware setting* where a limited

resource budget constraint is enforced requiring us to take a planning approach. Experiments on synthetic and real-world benchmarks from a diverse set of engineering and industrial domains demonstrate that our algorithms significantly improve resource efficiency over prior methods to produce high-quality Pareto solutions.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Dedication

I dedicate this thesis to:

My beloved parents, Mohamed Belakaria and Jamila Abbes, who has been my guiding light, and source of inspiration. Your love, encouragement and unwavering support throughout my academic journey have been the foundation of my success. You have always believed in me. Your faith has given me the strength to persevere through the toughest of times. I am forever grateful for your sacrifices and for the countless prayers for my success. This thesis is a testament to your love and dedication.

Taha, my husband, and best friend. Your patience and unconditional support have been the driving force behind my PhD success. Thank you for all the unforgettable memories we made together, and for the love and care you provide me.

CHAPTER ONE

INTRODUCTION

We have witnessed significant progress in building intelligent machines over the last decade through advances in artificial intelligence (AI) and machine learning (ML). Some examples include accurate classifiers for image data, effective machine translation and speech recognition systems for popular languages, the AlphaGo system [171] that beat the world's best human player for the game of Go, and the AlphaFold system [107] that predict a protein's 3D structure from its amino acid sequence. The key ingredient for building these successful systems is the availability of large amounts of supervised training data (aka Big-data regime). However, there are many real-world problems where we have little to no training data (aka Small-data regime) where the current AI methods are lacking. This dissertation focuses on one class of important small-data problems referred to as *goal-driven adaptive experimental design*.

Goal-Driven Adaptive Experimental Design. The problem of adaptively selecting a sequence of experiments to achieve a goal arises in many real-world settings. We provide three concrete examples with varying goals to illustrate this problem.

- In the active learning paradigm, we need to iteratively collect labeled data from a human expert to build predictors with high accuracy [160]. Each experiment corresponds to collecting the output label of an unlabeled input (e.g., image) from a human. Since each labeling query is expensive in terms of human ef-

fort, the overall goal is to minimize the number of labeling queries to build an accurate predictor.

- In A/B testing for marketing purposes, we need to decide which version of an advertisement is more effective in attracting potential customers. Each experiment corresponds to deploying an advertisement for some fraction of end-users [16]. Since collecting information about user preference and their level of interaction is costly in terms of time and financial impact on the business, the goal is to uncover the suitable advertisement with the minimum number of tests and with minimal business loss.

- In black-box function optimization (e.g., hyper-parameter optimization of ML models), we need to find the hyper-parameter configuration with the highest accuracy on the validation data [172]. Each experiment corresponds to training the ML model with a candidate hyper-parameter configuration and evaluating the accuracy of the trained model on the validation data. Since each experiment is expensive in terms of computational resources, the overall goal is to minimize the total computational cost to find the best hyperparameter configuration.

The common challenge for all three adaptive experimental design problems is to develop an effective decision-making strategy to select one or more candidate experiments in each iteration. This strategy should be *adaptive* by accounting for the data collected from past experiments and *goal-driven* to optimize for the target goal.

Uncertainty sampling (selecting the unlabeled input for which the classifier trained from the past labeled data is most uncertain about) is a typical strategy for active learning [168]. Multi-arm bandit algorithms [141] are commonly used for A/B testing problems where the key idea is to trade off exploration (selecting choices that weren't tried enough) and exploitation (selecting choices that performed well in the past). Bayesian optimization [165] is an effective framework to solve expensive black-box optimization problems. The key idea is to build a surrogate statistical model from past experimental data and use its prediction/uncertainty for unknown inputs in the search space to decide the next input for evaluation.

Optimizing Expensive Black-box Functions. Real-world expensive function optimization problems can be formalized using the following key elements: input search space (continuous, discrete, hybrid), the number of objectives (single or multiple), the fidelity of experiments (vary in expense and accuracy of evaluation), unconstrained or constrained, and blackbox or greybox optimization. We can instantiate a specific problem setting by selecting one choice for each element.

- **Search space.** The input search space $\mathfrak{X}$ can be *continuous* (all design variables are continuous), *discrete* (e.g., sets, sequences, graphs) or *hybrid* (mixture of discrete and continuous design variables).

- **Number of objectives.** In the *single-objective setting*, we assume an unknown real-valued objective function $f : \mathfrak{X} \mapsto \Re$, which can evaluate each candidate

design $x \in \mathfrak{X}$ (also called an experiment) to produce an outcome $y = f(x)$. The overall goal is to find an input $x \in \mathfrak{X}$ that approximately optimizes f by conducting a limited number of experiments and observing their outcomes. In *multi-objective optimization setting*, we need to optimize multiple real-valued objective functions.

- **Number of fidelities.** In the *single-fidelity* setting, each experimental evaluation is expensive and accurate. In the *multi-fidelity* setting, we can perform experiments that vary in the amount of resources consumed and the accuracy of evaluation.

- **Unconstrained vs. Constrained.** In the *unconstrained* setting, all candidate inputs are valid. However, in the *constrained* setting, we will have invalid inputs in the search space and we may not know the validity of input without performing an experiment.

- **Black-box vs. Grey-box optimization.** In the *black-box* setting, we only model the mapping between candidate inputs $x \in \mathfrak{X}$ and their evaluation $y = F(x)$. In the *grey-box* setting, we can model potentially useful side-information and leverage it to formally reason about the selection of experiments.

Multi-Objective Optimization of Expensive Black-box Functions. Motivated by the challenges faced by scientists and engineers, this dissertation studies

adaptive experimental design algorithms for the purpose of solving a large-class of MOO problems where we need to optimize multiple expensive-to-evaluate objective functions. Many engineering and scientific applications involve making design choices to optimize multiple objectives. Some examples include designing drugs/vaccines to optimize efficacy, cost, and safety; designing new materials to optimize strength, elasticity, and durability; and designing hardware to optimize performance, power, and area [34, 66] There are a few common challenges in solving these kinds of multi-objective optimization problems:

1. The objective functions are unknown and we need to perform expensive experiments to evaluate each candidate design choice, where the expense is measured in terms of the consumed resources (physical or computational). For example, performing computational simulations and physical lab experiments for hardware 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.

3. The solutions may need to satisfy black-box constraints, which cannot be evaluated without performing experiments. For example, in drug/vaccine design

applications, we need to find designs that trade-off efficacy and cost while satisfying safety constraints.

4. We have the ability to perform multi-fidelity experiments (discrete or continuous) to evaluate objective functions via cheaper approximations, which vary in the amount of resources consumed and their accuracy. For example, in hardware design optimization, we can use multi-fidelity simulators for design evaluations. We want to leverage this additional freedom to reduce the overall resource cost for optimization.

5. We might have access to expert knowledge about the structure or behavior of objective functions. For example, in hyper-parameter tuning of iterative machine learning models, the validation error function follows a monotonic and exponentially decaying shape with respect to the number of epochs. This *side information* can be leveraged to develop a more effective grey-box optimization approach.

6. The resource budget available to find the optimized designs is limited. Therefore, it may be necessary to take a planning view to select the sequence of experiments. This can be achieved through a non-myopic procedure, which takes into account the remaining budget and the future candidate experiments in each iteration.

Real-world MOO problems come with two or more of the above challenges. The adaptive experimental design algorithms need to approximate the optimal Pareto set while minimizing the total resource cost of conducted experiments.

The overarching goal of this dissertation is to develop AI based reasoning methods for adaptive experimental design to solve a large-class of MOO problems arising in science, engineering, and industrial domains where there is a need to select expensive experiments to optimize complex design spaces under resource budget.

The key challenge in solving this class of adaptive experiment design problems is to select the sequence of experiments to approximate the optimal Pareto set in a resource-efficient manner. This problem is an instance of sequential decision-making under uncertainty, where we need to reason about the resources spent in conducting an experiment and the value of information gathered towards the goal of optimization.

Bayesian Optimization (BO) [166] is an effective framework to solve black-box optimization problems with expensive function evaluations. The key idea behind BO is to build a cheap surrogate statistical model, e.g., Gaussian Process (GP) [189], using the real experimental data; and employ it to intelligently select the sequence of experiments or function evaluations using an acquisition/utility function, e.g., expected improvement (EI) and upper-confidence bound (UCB). There is a large body of literature on single-objective BO algorithms [166] and their applications including hyperparameter tuning of machine learning methods [172, 121]. However, there is relatively

less work on the more challenging problem of BO for multiple objective functions (first and second challenges) [83], only one prior work on the constrained multi-objective optimization problem (third challenge), and no prior work on multi-objective optimization in the multi-fidelity, grey-box, and non-myopic settings (fourth, fifth, and sixth challenges).

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 [117, 67]. This can potentially lead to aggressive exploitation behavior. Additionally, previously developed algorithms to optimize Pareto Hypervolume (PHV) based acquisition functions scale poorly as the number of objectives and the dimensionality of input space grows. More recent work [43] improves the scalability of PHV-based approaches. There are also methods that rely on *input space entropy* based acquisition function [83] to select the candidate inputs for evaluation. However, it is computationally expensive to approximate and optimize this acquisition function.

Summary of Dissertation Research. *This dissertation develops a suite of novel reasoning algorithms for adaptive experimental design to solve a large-class of MOO (single-fidelity, constrained, multi-fidelity, and budget-constrained) problems. Our algorithms are based on the principles of uncertainty reduction and information gain per unit resource cost.*

First, we propose a general framework for solving MOO problems based on the principle of *output space entropy (OSE)* search [185, 87]. The key idea is to select the input and fidelity vector (one for each objective function, if applicable) that maximizes the information gain per unit resource cost about the optimal Pareto front in each iteration. Output space entropy search has many advantages over algorithms based on input space entropy search [17]: a) it allows much tighter approximation; b) it is cheaper to compute; and c) it naturally lends itself to robust optimization with respect to the number of Monte Carlo samples used for acquisition function computation. We appropriately instantiate the OSE principle to derive efficient algorithms for solving four qualitatively different MOO problems: the most basic single-fidelity setting [17], MOO with black-box constraints [26], discrete multi-fidelity setting [21], and continuous-fidelity setting [26].

Second, we propose an uncertainty-aware search framework for multi-objective optimization (USeMO) that enables a generalization of single-objective Bayesian optimization to the multi-objective setting. The key insight is to perform a two-stage search procedure to improve the accuracy and computational-efficiency for selecting candidate inputs for evaluation. First, the algorithm solves a cheap MO optimization problem defined in terms of the acquisition functions (one for each unknown objective) to identify a list of promising candidates. Second, it selects the best candidate from this list based on a measure of uncertainty. Unlike prior methods, USeMO has

several advantages: a) Does not reduce to a single objective optimization problem; b) The reduction formulation behind USeMO allows us to leverage a variety of acquisition functions designed for single objective BO; c) Computationally-efficient to solve MOO problems with many objectives; and d) Improved uncertainty management via two-stage search procedure to select the candidate inputs for evaluation. The versatility of the USeMO framework allowed us to extend it to handle black-box, white-box, and greybox constraints [18]. We also propose a generalization of USeMO in two ways: 1) Automatically selecting acquisition function from a given library in each iteration using a multi-arm bandit approach, and 2) Selecting a diverse batch of experiments for parallel evaluation in each iteration by configuring Determinantal Point Processes (DPPs) [123] in the output space.

Finally, to address the challenge of resource budget constraint, we develop a novel approach for budget-aware BO using a non-myopic reasoning procedure. The key idea is to use the submodularity property of the lower bound of the Bellman equation to approximate the lookahead horizon and adapt its length to the remaining budget. We apply this budget-aware method for hyper-parameter optimization of iterative machine learning algorithms. We extend this approach to the multi-objective setting where we approximate the lookahead horizon via a batch multi-objective optimization approach by exploiting the same submodularity property [43].

Comprehensive experiments on diverse synthetic and real-world benchmarks demon-

strate that our MOO algorithms are computationally-efficient, and significantly improve the resource-efficiency to produce high-quality Pareto solutions than the state-of-the-art algorithms.

1.1 Technical Contributions

The main contribution of this dissertation is the development and evaluation of a suite of multi-objective Bayesian optimization (BO) algorithms to significantly push the frontiers of the adaptive experimental design research area. Our algorithms are based on the principles of information gain per unit resource cost and uncertainty reduction. Specific contributions include:

- Development of two approaches to solve the most basic MOO problem in the single-fidelity setting, where experiments are expensive and accurate
  - MESMO: Max-value Entropy Search for Multi-Objective Bayesian Optimization [17].
  - USEMO: Uncertainty-Aware Search framework for Multi-Objective Bayesian Optimization [22]
  - First theoretical analysis to prove a sub-linear regret bound for multi-objective BO setting [17, 22].
- Development of an approach (generalization of USEMO) to solve MOO problems by selecting a batch of diverse experiments for parallel evaluations using

determinantal point processes.

  - PDBO: Pareto front-Diverse Batch Multi-Objective Bayesian Optimization [29]

- Development of two approaches (extensions of MESMO and USEMO algorithms) to handle MOO problems with black-box constraints, which cannot be evaluated without performing experiments.

  - MESMOC: Max-value Entropy Search for Multi-Objective BO with Constraints [26, 20]

  - USEMOC: Uncertainty-Aware Search framework for Multi-Objective BO with Constraints [18]

- Development of multi-fidelity optimization approaches to solve MOO problems by appropriately leveraging the available side information.

  - MF-OSEMO algorithm to solve MOO problems in the discrete multi-fidelity setting, where experiments can vary in the amount of resources consumed and their evaluation accuracy [21].

  - iMOCA algorithm to solve MOO problems in the continuous-fidelity setting, where continuous function approximations result in a huge space of experiments with varying cost [19, 26].

- Development of non-myopic optimization algorithms when the available resource budget is limited by viewing the problem from a planning perspective.
  - A budget-aware approach to solve hyper-parameter optimization problems for iterative machine learning algorithms with structured responses in the form of learning curves. We also leverage the side information in the form of the structure of the objective functions through appropriate modeling and reasoning tools [28].
  - A non-myopic optimization approach to solve general MOO problems when the resource budget is limited.
- Applying our algorithms to diverse real-world problems in engineering and industrial domains in close collaboration with domain experts [200, 23, 93, 55, 195, 143]. Experimental evaluation on diverse synthetic and real-world benchmark problems to demonstrate the effectiveness of the proposed algorithms over existing MOO algorithms.
- Open-source release of the software associated with published research so far: MESMO[1], USEMO[2] MESMOC[3], MF-OSEMO[4], iMOCA[5], and BAPI [6].

[1] `https://github.com/belakaria/MESMO`
[2] `https://github.com/belakaria/USeMO`
[3] `https://github.com/belakaria/MESMOC`
[4] `https://github.com/belakaria/MF-OSEMO`
[5] `https://github.com/belakaria/iMOCA`
[6] `https://github.com/belakaria/BAPI`

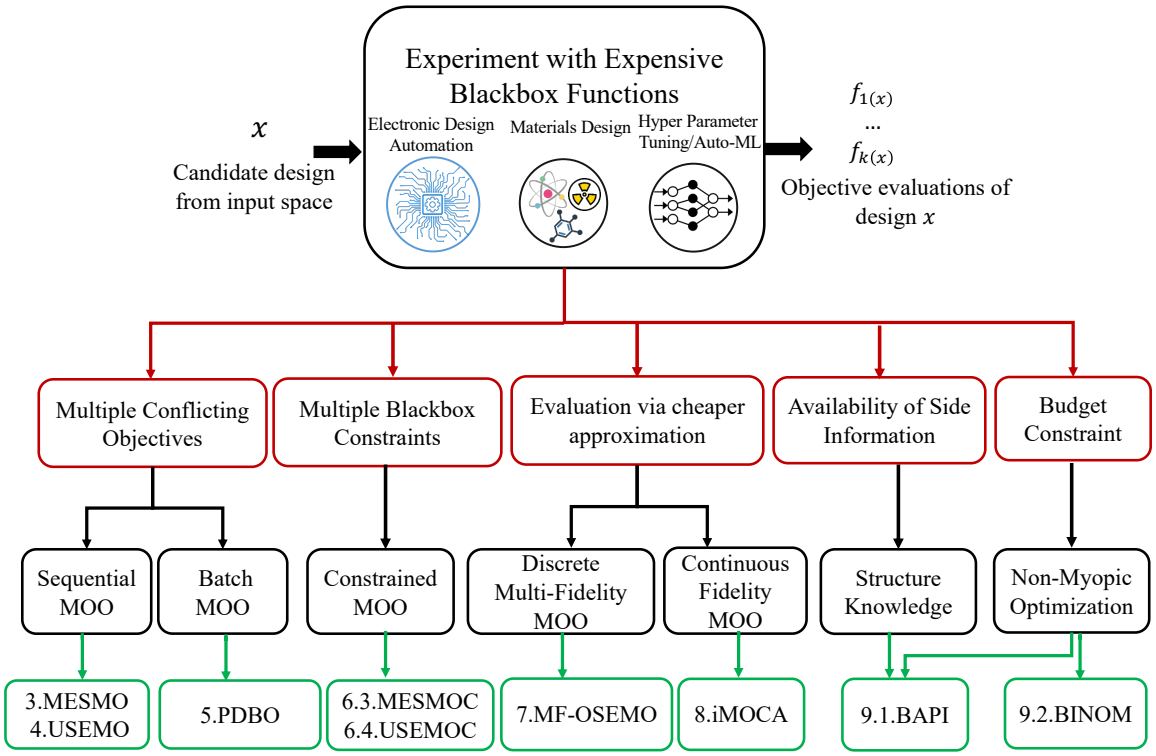

Figure 1.1: Overview of the multi-objective optimization (MOO) problem settings studied in this dissertation and the proposed MOO algorithms along with Chapter numbers where they are described.

1.2 Outline of the Thesis

The remaining part of the dissertation is organized as follows.

In Chapter 2, we first provide an overview of the problem setup for different MOO problem settings studied in this dissertation. Next, we discuss the necessary background material on Bayesian optimization for both single-objective and multi-objective settings. Finally, we define the evaluation metrics to measure the efficacy of adaptive experimental design algorithms to solve MOO problems.

In Chapter 3, we address the most basic MOO problem in the single-fidelity setting, where the goal is to optimize multiple black-box objective functions. To solve this problem, we propose an algorithm referred to as **M***ax-value* **E***ntropy* **S***earch for* **M***ulti-objective* **O***ptimization* (MESMO)[17]. We mathematically describe the output space entropy-based acquisition function behind MESMO and provide an algorithmic approach to efficiently compute it. We also provide theoretical analysis to characterize the efficacy of MESMO and present experimental results on several real-world and synthetic benchmark problems.

In Chapter 4, we present the Uncertainty-Aware Search Framework for Multi-Objective Bayesian Optimization (USeMO), which is a wrapper approach that can be instantiated with any acquisition function for single-object optimization. USeMO addresses the same problem setting as in Chapter 3. First, we provide an overview of USeMO followed by the details of its two main components. Subsequently, we provide a theoretical analysis of USeMO in terms of asymptotic regret bounds. Finally, we provide the experimental evaluation of USeMO on several synthetic and real-world problems.

In Chapter 5, we describe an algorithmic approach referred to as PDBO which generalizes the USeMO framework from Chapter 4 in two ways. First, it automatically selects a single-objective acquisition from a given library in each iteration using a multi-arm bandit strategy. In contrast, USeMO uses the same acquisition function

which is specified by the practitioner in each BO iteration. Second, it allows to select a batch of inputs for evaluation in each iteration for discovering a diverse Pareto front using determinantal point processes configured for the output space. We provide experimental results of PDBO and compare with prior methods on several benchmark problems in terms of both quality and diversity of MOO solutions.

In Chapter 6, we address the MOO problem with constraints, where the goal is to optimize multiple real-valued objective functions while satisfying several black-box constraints over the input space. To solve this problem, we propose two algorithms, namely, Max-value Entropy Search for Multi-objective Optimization with Constraints (MESMOC) and Uncertainty aware Search Framework for Multi-Objective Bayesian Optimization with Constraints (USeMOC), which are generalizations of MESMO and USeMO respectively. We first describe the problem setup and discuss prior work. Next, we explain the technical details of both MESMOC and USeMOC algorithms followed by their experimental results on real-world benchmarks.

In Chapter 7, we address the discrete multi-fidelity version of the MOO problem, where we have access to multiple fidelities for each objective function that vary in the amount of resources consumed and the accuracy of evaluation. To solve this problem, we propose an algorithm referred to as **M***ulti-***F***idelity* **O***utput* **S***pace* **E***ntropy Search for* **M***ulti-objective* **O***ptimization* (MF-OSEMO), which is a generalization of MESMO. We first describe the complete details related to the multi-fidelity MOO

problem. Subsequently, we explain the details of MF-OSEMO algorithm with two mathematically different approximations of the corresponding output space entropy-based acquisition function.

In Chapter 8, we address the continuous-fidelity MOO problem, where we have access to alternative functions through which we can evaluate cheaper approximations of objective functions by varying a continuous fidelity variable (e.g., number of epochs or size of the training set in hyper-parameter tuning problems). To solve this problem, we propose an algorithm referred to as **i***nformation-Theoretic* **M***ulti-Objective Bayesian* **O***ptimization with* **C***ontinuous* **A***pproximations* (iMOCA), which is a generalization of MF-OSEMO. We first describe the complete details related to the continuous-fidelity MOO problem. Subsequently, we explain the details of iMOCA algorithm with two mathematically different approximations of the corresponding output space entropy-based acquisition function.

In Chapter 9, we address the problem of non-myopic multi-objective Bayesian optimization. Non-myopic algorithms are particularly well-suited for small resource budgets as they take a planning perspective and reason about different experimental plans in a look-ahead manner. We start by proposing a solution to the budgeted non-myopic Bayesian optimization, then we show how to incorporate side information from the hyper-parameter optimization problem to build a planning approach for iterative machine learning models. Subsequently, we propose a more generic solution

for non-myopic multi-objective optimization problems.

Finally, in Chapter 10, we provide a summary of the dissertation, lessons learned, and list some important future research directions.

CHAPTER TWO

PROBLEM SETUP AND BACKGROUND

In this Chapter, we first formally define the MOO problem settings that are addressed in this dissertation. Next, we provide an overview of the generic Bayesian optimization framework in the single-objective and multi-objective settings. Finally, we define the metrics to evaluate the effectiveness of the proposed MOO algorithms.

2.1 Overview of Multi-Objective Optimization

Multi-objective optimization (MOO) problems can be formalized in terms of the following key elements: number of objectives, necessity to satisfy black-box constraints, availability of cheaper approximations or fidelities (discrete/continuous) for function evaluations, and availability of side information. Below we provide a brief overview of different MOO problem settings that are addressed in this dissertation noting that a more detailed problem setup is specified under the respective chapters.

Basic Multi-objective Optimization Problem. The goal is to maximize real-valued objective functions $f_1(\mathbf{x}), \cdots, f_K(\mathbf{x})$, 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 $\mathbf{y} = (y_1, \cdots, y_K)$ where $y_i = f_i(\mathbf{x})$ for all $i \in \{1, \cdots, K\}$. If a budget constraint on the resource cost is specified, we refer to this variant as *Budget-aware* MOO.

Batch Multi-objective Optimization Problem. This setting differs from the

basic multi-objective optimization problem described above in one aspect: we are allowed to select B inputs for parallel evaluation in each iteration instead of one input per iteration. Our overall goal is to uncover a high-quality and diverse Pareto front while minimizing the total number of expensive function evaluations.

MOO Problem with Constraints. This is a generalization of the basic MOO problem (i.e., no constraints) where the goal is to maximize the K real-valued objective functions while satisfying L black-box constraints of the form $C_1(\mathbf{x}) \geq 0, \cdots, C_L(\mathbf{x}) \geq 0$. Each evaluation of an input $\mathbf{x} \in \mathfrak{X}$ produces a vector of objective values and constraint values $\mathbf{y} = (y_{f_1}, \cdots, y_{f_K}, y_{c_1} \cdots y_{c_L})$ where $y_{f_j} = f_j(\mathbf{x})$ for all $j \in \{1, \cdots, K\}$ and $y_{c_i} = C_i(\mathbf{x})$ for all $i \in \{1, \cdots, L\}$.

MOO Problem with Discrete Multi-fidelity Experiments. This is a general version of the MOO problem, where we have access to M_j fidelities for each function f_j that vary in the amount of resources consumed and the accuracy of evaluation. The evaluation of an input $\mathbf{x} \in \mathfrak{X}$ with fidelity vector $\mathbf{m} = [m_1, \cdots, m_K]$ produces an evaluation vector of K values denoted by $\mathbf{y^m} \equiv [y_1^{(m_1)}, \cdots, y_K^{(m_K)}]$, where $y_j^{(m_j)} = f_j^{(m_j)}(\mathbf{x})$ for all $j \in \{1, \cdots, K\}$.

MOO Problem with Continuous-fidelity Experiments. In this general version of the multi-fidelity setting, we have access to $g_i(\mathbf{x}, z_i)$ where g_i is an alternative function through which we can evaluate cheaper approximations of f_i by varying the fidelity variable $z_i \in \mathcal{Z}$ (continuous function approximations). The evaluation of an

input $\mathbf{x} \in \mathfrak{X}$ with fidelity vector $\mathbf{z} = [z_1, \cdots, z_K]$ produces an evaluation vector of K values denoted by $\mathbf{y} \equiv [y_1, \cdots, y_K]$, where $y_i = g_i(\mathbf{x}, z_i)$ for all $i \in \{1, \cdots, K\}$.

Multi-Objective Optimization Solution. 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*. The goal of multi-objective BO is to approximate $\mathcal{X}^*$ while minimizing the number of function evaluations.

Table 2.1 contains the mathematical notations used throughout the dissertation.

2.2 Background on Bayesian Optimization

This dissertation studies adaptive experimental design algorithms for solving the above-mentioned MOO problems within the framework of Bayesian optimization (BO). Hence, we provide the necessary background on BO for the reader.

Bayesian Optimization is a very efficient framework to solve global optimization problems using *black-box evaluations of expensive objective functions*. Without loss of generality, let $\mathfrak{X} \subseteq \Re^d$ be an input space. In the 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. For example, in hardware design optimization application, $\mathbf{x}$ corresponds to a candidate hardware design with some choices

to input variables, and $f(\mathbf{x})$ corresponds to running a computationally-expensive simulation to mimic the real hardware. 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* (inputs for which the statistical model has high uncertainty) and *exploitation* (inputs for which the model has high prediction value) to quickly direct the search toward optimal inputs. The three key elements of BO framework are:

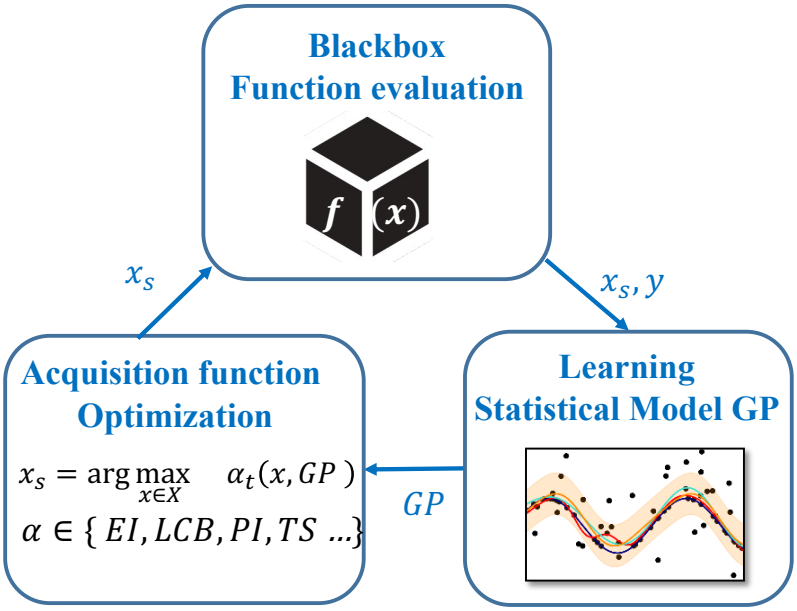

Figure 2.1: Overview of the Bayesian optimization framework for the single objective function setting. In each iteration, we use the statistical model trained from past data to select the next input $\mathbf{x}_s$ for function evaluation using the acquisition strategy and update the mode using the new training example $(\mathbf{x}_s, y)$.

1) Statistical Model of the true function $f(\mathbf{x})$. *Gaussian Process (GP)* [189]

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 $m : \mathfrak{X} \mapsto \Re$ and a covariance or kernel function $\kappa : \mathfrak{X} \times \mathfrak{X} \mapsto \Re$. If a function f is sampled from $\mathcal{GP}(m, \kappa)$, then $f(\mathbf{x})$ is distributed normally $\mathcal{N}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}))$ for a set of inputs from $\mathbf{x} \in X$. The predictive mean and uncertainty of a GP for a new input $\mathbf{x}_n \in \mathfrak{X}$ is defined as:

$$\mu(\mathbf{x}_n) = \kappa_{\mathbf{x}_n,X}[\kappa_{X,X} + \sigma^2 I]^{-1}(Y - m(X)) + m(\mathbf{x}_n)$$

$$\sigma^2(\mathbf{x}_n) = \kappa_{\mathbf{x}_n,\mathbf{x}_n} - \kappa_{\mathbf{x}_n,X}[\kappa_{X,X} + \sigma^2 I]^{-1}\kappa_{X,\mathbf{x}_n}$$

where $\kappa_{\mathbf{x}_n,\mathbf{x}_n} = \kappa(\mathbf{x}_n, \mathbf{x}_n)$, $\kappa_{X,X} = \kappa(X, X)$, $\kappa_{\mathbf{x}_n,X} = [\kappa(\mathbf{x}_n, \mathbf{x}_i)]_{\forall i}$, X is the set of evaluated inputs and Y is their corresponding function values.

Kernel functions. A kernel function measures the similarity between a pair of inputs $(\mathbf{x}, \mathbf{x}')$ and allows GPs to model correlations between function values for different inputs. Examples of kernel functions that are commonly used over continuous input spaces are:

$$\text{Radial Basis Function (RBF):} \quad \kappa(\mathbf{x}, \mathbf{x}') = \sigma_f \cdot exp(\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}) \tag{2.1}$$

$$\text{Matern} \quad 5/2: \quad \kappa(\mathbf{x}, \mathbf{x}') = \sigma_f(1 + \frac{\sqrt{5}||\mathbf{x} - \mathbf{x}'||}{l} + \frac{5||\mathbf{x} - \mathbf{x}'||^2}{3l^2})exp(-\frac{||\mathbf{x} - \mathbf{x}'||}{l}) \tag{2.2}$$

Each kernel possesses the hyperparameter σ_f , the signal variance, which is a scale factor controlling the range of the function values represented by a GP. The RBF

kernel and Matern kernel contain a length-scale hyperparameter l that controls the closeness/similarity of $\mathbf{x}$ and $\mathbf{x}'$. l is also known as the smoothness parameter. GPs can also be used for discrete/hybrid spaces once we define an appropriate kernel [65, 54]. Some examples include diffusion kernels for categorical spaces [57, 52] and hybrid spaces [58], dictionary kernels for high-dimensional discrete/hybrid spaces [62], Kendall and Mallow kernels [61] for permutation spaces, structure-coupled kernel for rich structured spaces such as molecules [50].

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 BO literature include expected improvement (EI), upper confidence bound (UCB), lower confidence bound (LCB), Thompson sampling (TS), and max-value entropy search (MES) [185]. For the sake of completeness, we formally define the acquisition functions employed in Chapters 4 and 5:

$$UCB(\mathbf{x}) = \mu(\mathbf{x}) + \beta^{1/2}\sigma(\mathbf{x}) \tag{2.3}$$

$$LCB(\mathbf{x}) = \mu(\mathbf{x}) - \beta^{1/2}\sigma(\mathbf{x}) \tag{2.4}$$

$$TS(\mathbf{x}) = f(\mathbf{x}) \; with \; f(.) \sim \mathcal{GP} \tag{2.5}$$

$$EI(\mathbf{x}) = \sigma(\mathbf{x})(\gamma\Phi(\gamma) + \phi(\gamma)), \; \gamma = \frac{\tau - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \tag{2.6}$$

3) Optimization Procedure to select the best scoring candidate input according

to acquisition function α depending on the statistical model. DIRECT [105] is a very popular approach for acquisition function optimization.

$$\mathbf{x}_s \leftarrow \arg max_{\mathbf{x} \in \mathfrak{X}} \ \alpha(\mathbf{x})$$

Multi-objective Bayesian Optimization. To handle the multi-objective setting, BO for single-objective optimization can be extended using the same three key elements introduced in section 2.2. However, there are two main challenges: **1)** How to build statistical models for multiple expensive-to-evaluate objective functions?; and **2)** How to design a good acquisition function to capture the trade-off between multiple objectives? To overcome the first challenge, most existing work uses independent Gaussian process (GP) for each objective. We provide more details about the surrogate modeling strategy used in our proposed solutions for each of the problem settings in the respective chapters. To overcome the second challenge, new acquisition functions are designed to tackle the challenges of multi-objective problems. In this dissertation, we propose several acquisition functions to address the problems introduced in Section 2.1. Figure 2.2 illustrates an overview of the Bayesian optimization process in the multi-objective setting.

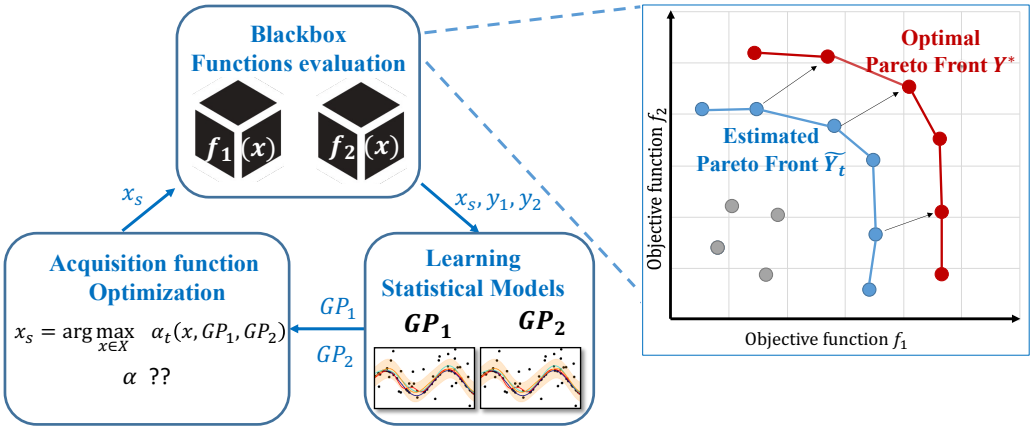

Figure 2.2: Overview of the Bayesian optimization process for two objective functions (K=2). We learn one GP surrogate model for each expensive-to-evaluate objective function. In each iteration, the acquisition strategy α reasons about the utility of evaluating an unknown input using the information from the GP models for two objective functions and selects the input $\mathbf{x}_s$ with highest utility for function evaluation. The evaluation of input $\mathbf{x}_s$ produces (y_1, y_2) and this new training example is used to update the statistical models GP_1, GP_2.

2.3 Evaluation Metrics

To evaluate the proposed MOO algorithms, we employed two known metrics to measure the quality of Pareto solutions: the Pareto hypervolume indicator and the R_2 indicator. Additionally, we propose a new Pareto diversity metric to evaluate the diversity of the Pareto front.

Pareto hypervolume (PHV) is commonly employed to measure the quality of a given Pareto front [202]. PHV is defined as the volume between a reference point and the given Pareto front (set of non-dominated points). The hypervolume indicator can be used as a performance metric in two different ways. After each iteration t: 1) We

report the hypervolume of the estimated Pareto front ($\mathcal{Y}_t$) by a given algorithm, and 2) 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.

$$PHV_{diff} = PHV(\mathcal{Y}^*) - PHV(\mathcal{Y}_t) \tag{2.7}$$

The left part of Figure 2.3 provides an illustration of the hypervolume metric.

R_2 **Indicator** is the average distance between the ideal Pareto front ($\mathcal{Y}^*$) and the estimated Pareto front ($\mathcal{Y}_t$) by a given algorithm [157]. R_2 is a distance-based metric that degenerates to the regret metric presented in our theoretical analysis of regret bounds. The right part of Figure 2.3 provides an illustration of the R_2 metric.

Diversity of Pareto Front: In multi-objective optimization, our goal is to find high-quality Pareto fronts. However, in several real-world problems, practitioners might highly care about the diversity of the Pareto front. The diversity of the Pareto front has not been formally evaluated in any previous work. As one of our contributions, we introduce an evaluation metric to measure the diversity of the Pareto front.

Diversity is an important criterion for many optimization problems. Prior work on batch BO both in the single-objective and multi-objective settings focused on evaluating diversity with respect to the input space [91]. However, in most real-world MOO problems, diversity in the input space does not necessarily reflect diversity in the output space. There is little to no work on understanding, formalizing, and

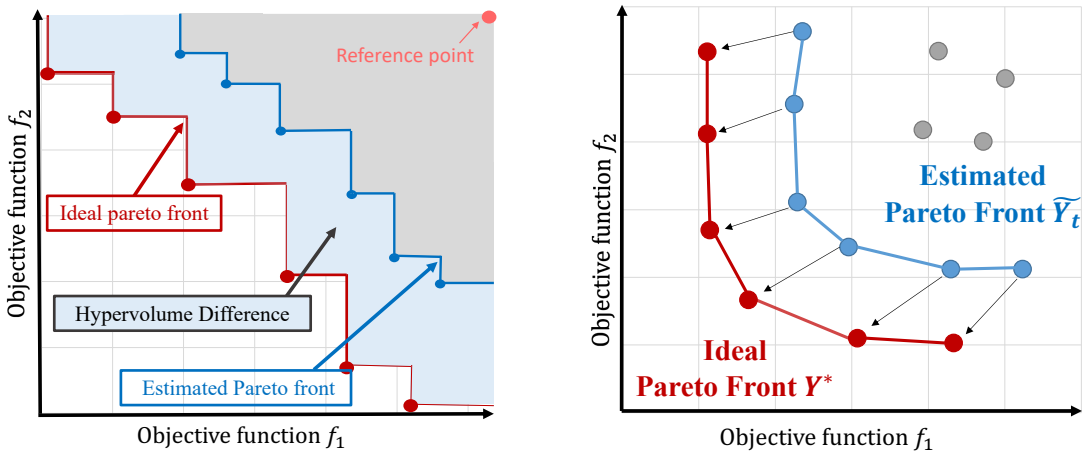

Figure 2.3: Illustration of the Pareto hypervolume and the R_2 metrics for two objective functions. The blue points correspond to the Pareto front estimated by a given algorithm. The red points correspond to the optimal Pareto front. *Left:* The gray volume is its corresponding Pareto hypervolume. The blue area represents the Pareto hypervolume difference metric for this example. *Right:* The black arrows represent the distances that are averaged to compute the R_2 metric.

measuring Pareto front diversity in MOO. In most cases, finding a more diverse set of points in the output space leads to a higher hypervolume [203]. However, a higher hypervolume indicator does not necessarily correspond to a more diverse Pareto front. [118] is the only prior work that proposed a diversity-guided approach for batch MOO. However, the diversity of the produced Pareto front was not evaluated.

To fill this gap, we propose an evaluation metric to assess the *Diversity of the Pareto Front (DPF)*. Given a Pareto font $\mathcal{Y}_t$, DPF is the average pairwise distance between points (i.e., output vectors) on Pareto front. It is important to clarify that the pairwise distances are computed in the output space between different vector pairs $(\mathbf{y}, \mathbf{y}') \in \mathcal{Y}_t$, unlike previously used metrics to assess input space diversity in

Notation	Definition
$\mathbf{x}, \mathbf{y}, \mathbf{f}, \mathbf{m}$	bold notation represents vectors
$\mathbf{x}$	input vector of d dimensions
$[n]$	set of first n natural numbers $\{1, 2, \cdots, n\}$
$f_1, f_2, \cdots, f_K$	true objective functions
$C_1, C_2, \cdots, C_L$	Constraints functions
$\tilde{f}_j$	function sampled from the highest fidelity of the jth Gaussian process model
$\mathcal{X}^*$	true Pareto set
$\mathfrak{X}$	Input Search space
I	Information gain
$\mathcal{Y}^*$	true Pareto front of the objective functions $[f_1, f_2, \cdots, f_K]$
$\mathcal{Y}^*_s$	Pareto front of the sampled functions $[\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f}_K]$
$\mathcal{Y}_t$	Pareto front at iteration t
$\mathcal{GP}_j$	Gaussian process modeling function/constraint j

Table 2.1: The general mathematical notations used throughout the dissertation.

the single-objective setting [5]. The proposed diversity measure and the hypervolume metric complement each other and together represent the merits of the Pareto front.

$$DPF(\mathcal{Y}_t) = \frac{\sum_{(i,j)\in\mathcal{I}} ||\mathbf{y_i} - \mathbf{y_j}||}{|\mathcal{I}|} \quad \text{with} \quad \mathcal{I} = \{(i,j) \forall i, j \in \{1 \cdots t\}, i < j, (\mathbf{y_i}, \mathbf{y_j}) \in \mathcal{Y}_t\}$$

CHAPTER THREE

OUTPUT SPACE ENTROPY SEARCH FOR MULTI-OBJECTIVE BO

In this chapter, we address the most basic MOO problem in the single-fidelity setting, where the goal is to optimize multiple black-box objective functions by approximating the true Pareto-set of solutions while minimizing the number of function evaluations. For example, in hardware design optimization, we need to find designs that trade-off performance, energy, and area overhead using expensive computational simulations [34, 66, 114, 101, 51, 142, 102, 128, 129, 167, 41, 113, 36, 103, 132, 10, 40, 9, 11, 97, 93, 95, 96, 98].

We propose a novel approach referred to 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. 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) is significantly cheaper to compute; and c) naturally lends itself to robust optimization. We also provide theoretical analysis in terms of asymptotic regret bounds to characterize the efficacy of MESMO. Our experiments on several synthetic and real-world benchmark problems show that MESMO consistently outperforms state-

of-the-art algorithms.

3.1 Problem Setup

Basic Multi-Objective Optimization Problem. The goal is to maximize 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 (also called an experiment) of an input $\mathbf{x} \in \mathfrak{X}$ produces a vector of objective values $\mathbf{y} = (y_1, y_2, \cdots, y_K)$ where $y_i = f_i(\mathbf{x})$ for all $i \in \{1, 2, \cdots, K\}$. We say that an point $\mathbf{x}$ *Pareto-dominates* another point $\mathbf{x}'$ if $f_i(\mathbf{x}) \geq f_i(\mathbf{x}') \, \forall i$ 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*. The goal of multi-objective BO is to approximate $\mathcal{X}^*$ while minimizing the number of expensive function evaluations. In the application of hardware design optimization, $\mathbf{x} \in \mathfrak{X}$ is a candidate hardware design; evaluation of design $\mathbf{x}$ to get output objectives such as power, performance, and area involve performing computationally-expensive simulation to mimic the real hardware; and our goal is to find the optimal Pareto set of hardware designs to trade-off power, performance, and area. Table 2.1 contains all the mathematical notations used in this section.

Surrogate Models. Gaussian processes (GPs) are shown to be effective surrogate

models in prior work on single and multi-objective BO [84, 186, 185, 174, 83]. Similar to prior work [83], we model the objective functions $f_1, f_2, \cdots, f_K$ using K independent GP models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ 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 $\mathbf{y}_i = \{y_1^i, y_2^i, \cdots, y_K^i\}$ is the output vector resulting from evaluating functions $f_1, f_2, \cdots, f_K$ at $\mathbf{x}_i$. We learn surrogate models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ from $\mathcal{D}$.

3.2 Related Work

There is a family of model-based multi-objective BO algorithms that reduce the problem to single-objective optimization. The ParEGO method [117] 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 [67] 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) [67] and probability of improvement in PHV (SUR) [155] also referred to as EHVI and HVPI respectively [80]. 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 [158]. To improve scalability, the gain in hypervolume is computed over a limited set of points: SMSego finds those sets 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 [206], PESMO [83], and the concurrent works USeMO [22] and MESMO [17] 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 the uncertain set. PAL provides theoretical guarantees, but it is only applicable for input space $\mathfrak{X}$ with a finite set of discrete points. USeMO is a general framework that iteratively generates a cheap Pareto front using the surrogate models and then selects the input with the highest uncertainty for evaluation. PESMO [83] 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) it requires a series of approximations, which can be potentially sub-optimal; b) the optimization, even after approximations, is expensive; and c) the performance is strongly dependent on the number of Monte-Carlo samples. In comparison, our proposed output space entropy-based acquisition function partially

overcomes the above challenges, and allows efficient and robust optimization with respect to the number of samples used for acquisition function computation. More specifically, the time complexities of acquisition function computation in PESMO and MESMO ignoring the time to solve the 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. In fact, PESMO formulation relies on an expensive and high-dimensional ($l \cdot d$ dimensions) distribution over the input space, where l is the size of the optimal Pareto set $\mathcal{X}^*$ while MESMO relies on a computationally cheap and low-dimensional distribution over the output space ($l \cdot K$ dimensions, which is considerably less than l·d as $K \ll d$ in practice). Additionally, Belakaria et al. [17] demonstrated that MESMO is very robust and performs very well even with one sample.

To overcome the shortcoming of the existing work, we propose two approaches: **1)** Our proposed output space entropy-based acquisition function, MESMO, 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.

MESMO is also very robust and performs very well even with one sample. The details of MESMO are provided in section 3.3. **2)** Our proposed uncertainty-aware approach, USEMO, provides a flexible and efficient wrapper framework by enabling practitioners to leverage approaches from the single objective BO literature to solve multi-objective BO problems. The details of USeMO are provided in Chapter 4.

3.3 Max-value Entropy Search for Multi-Objective BO

Input space entropy-based methods like PESMO [83] selects the next candidate input $\mathbf{x}_t$ (for ease of notation, we drop the subscript in the 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:

$$\alpha(\mathbf{x}) = I(\{\mathbf{x}, \mathbf{y}\}, \mathcal{X}^* \mid D) \tag{3.1}$$

$$= H(\mathcal{X}^* \mid D) - \mathbb{E}_y[H(\mathcal{X}^* \mid D \cup \{\mathbf{x}, \mathbf{y}\})] \tag{3.2}$$

$$= H(\mathbf{y} \mid D, \mathbf{x}) - \mathbb{E}_{\mathcal{X}^*}[H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{X}^*)] \tag{3.3}$$

Information gain is defined as the expected reduction in entropy $H(.)$[1] of the posterior distribution $P(\mathcal{X}^* \mid D)$ over the optimal Pareto set $\mathcal{X}^*$ as given in equations (3.2) and (3.3) (resulting from the symmetric property of information gain). This mathematical formulation relies on an expensive and high-dimensional ($l \cdot d$ dimensions)

[1]The conditioning on D and $\mathbf{x}$ in $H(\mathbf{y} \mid D, \mathbf{x})$ is on fixed values and not random variables

distribution $P(\mathcal{X}^* \mid D)$, where l is the size of the optimal Pareto set $\mathcal{X}^*$. Furthermore, optimizing the second term in r.h.s poses significant challenges: a) it requires a series of approximations [83] which can be potentially sub-optimal; and b) the optimization, even after approximations, is expensive c) the 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 the expected reduction in entropy over the Pareto front $\mathcal{Y}^*$, which relies on a computationally cheap and low-dimensional ($l \cdot K$ dimensions, which is considerably less than $l \cdot d$ 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:

$$\alpha(\mathbf{x}) = I(\{\mathbf{x}, \mathbf{y}\}, \mathcal{Y}^* \mid D) \tag{3.4}$$

$$= H(\mathcal{Y}^* \mid D) - \mathbb{E}_y[H(\mathcal{Y}^* \mid D \cup \{\mathbf{x}, \mathbf{y}\})] \tag{3.5}$$

$$= H(\mathbf{y} \mid D, \mathbf{x}) - \mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}^*)] \tag{3.6}$$

The first term in the r.h.s of equation (3.6) (entropy of a factorizable K-dimensional

Gaussian distribution $P(\mathbf{y} \mid D, \mathbf{x})$) can be computed in closed form as shown below:

$$H(\mathbf{y} \mid D, \mathbf{x}) = \frac{K(1 + \ln(2\pi))}{2} + \sum_{j=1}^{K} \ln(\sigma_j(\mathbf{x})) \tag{3.7}$$

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 (3.6) is an expectation over the Pareto front $\mathcal{Y}^*$. We can approximately compute this term via Monte-Carlo sampling as shown below:

$$\mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}^*)] \simeq \frac{1}{S} \sum_{s=1}^{S} [H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*)] \tag{3.8}$$

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 the input space entropy-based acquisition function.

There are two key algorithmic steps to compute equation (3.8). We want to know: 1) how to compute Pareto front samples $\mathcal{Y}_s^*$?; and 2) and how to compute the entropy with respect to a given Pareto front sample $\mathcal{Y}_s^*$? We provide solutions for these two questions.

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 [84, 159] and then solve a cheap

multi-objective optimization over the K sampled functions.

Sampling functions from posterior GP. Similar to prior work [84, 83, 185], 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 parameterized 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 $\mathcal{GP}_i$ 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 [47] 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 $\mathcal{Y}^*_s = \{\mathbf{v}^1, \cdots, \mathbf{v}^l\}$ be the sample Pareto front, where l is the size of the Pareto front and each $\mathbf{v}^i = \{v^i_1, \cdots, v^i_K\}$ 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 $H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*)$:

$$y_j \leq y_{j_s}^* \quad \forall j \in \{1, \cdots, K\} \tag{3.9}$$

where $y_{j_s}^* = \max\{v_j^1, \cdots v_j^l\}$. 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 l 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 $y_j > y_{j_s}^*$. 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 $y_j > y_{j_s}^*$ is incorrect and inequality (3.9) holds.

By combining the inequality (3.9) and the fact that each function is modeled as a GP, we can approximate each component y_j as a truncated Gaussian distribution since the distribution of y_j needs to satisfy $y_j \leq y_{j_s}^*$. 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 [38]:

$$H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*) = \sum_{j=1}^{K} H(y_j | D, \mathbf{x}, y_{j_s}^*) \tag{3.10}$$

The r.h.s is a summation over entropies of K variables $\{y_1, \cdots, y_K\}$. The probability distribution of each variable y_j is a truncated Gaussian with upper bound $y_{j_s}^*$

[137]. The differential entropy for each y_j is given as:

$$H(y_j \mid D, \mathbf{x}, \mathcal{Y}_s^*) \simeq \left[\frac{(1+\ln(2\pi))}{2} + \ln(\sigma_j(\mathbf{x})) + \ln \Phi(\gamma_s^j(\mathbf{x})) - \frac{\gamma_s^j(\mathbf{x})\phi(\gamma_s^j(\mathbf{x}))}{2\Phi(\gamma_s^j(\mathbf{x}))}\right] \tag{3.11}$$

Equation (3.10) and equation (3.11) give the following expression of $H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*)$.

$$H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*) \simeq \sum_{j=1}^{K} \left[\frac{(1+\ln(2\pi))}{2} + \ln(\sigma_j(\mathbf{x})) + \ln \Phi(\gamma_s^j(\mathbf{x})) - \frac{\gamma_s^j(\mathbf{x})\phi(\gamma_s^j(\mathbf{x}))}{2\Phi(\gamma_s^j(\mathbf{x}))}\right] \tag{3.12}$$

where $\gamma_s^j(\mathbf{x}) = \frac{y_{js}^* - \mu_j(\mathbf{x})}{\sigma_j(\mathbf{x})}$, and ϕ and Φ are the p.d.f and c.d.f of a standard normal distribution respectively. By combining equations (3.7) and (3.12) with equation (3.6), we get the final form of our acquisition function as shown below:

$$\alpha(\mathbf{x}) \simeq \frac{1}{S} \sum_{s=1}^{S} \sum_{j=1}^{K} \left[\frac{\gamma_s^j(\mathbf{x})\phi(\gamma_s^j(\mathbf{x}))}{2\Phi(\gamma_s^j(\mathbf{x}))} - \ln \Phi(\gamma_s^j(\mathbf{x}))\right] \tag{3.13}$$

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

Algorithm 1 MESMO Algorithm

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

Initialize Gaussian process models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$ 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 $\tilde{f}_j \sim \mathcal{GP}_j, \quad \forall j \in \{1, \cdots, K\}$
      $\mathcal{Y}_s^* \leftarrow$ 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 (3.13)
  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 $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$
  $t \leftarrow t + 1$
end for
return Pareto front of $f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_K(\mathbf{x})$ based on $\mathcal{D}$

3.4 Theoretical Analysis

In this section, we provide a theoretical analysis of the behavior of the MESMO algorithm. Multi-objective optimization literature has multiple metrics to assess the quality of Pareto front approximation. The two commonly employed metrics include the Pareto Hypervolume indicator [202] and R_2 indicator[157]. 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., [174] to prove convergence results. Prior work [155] 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

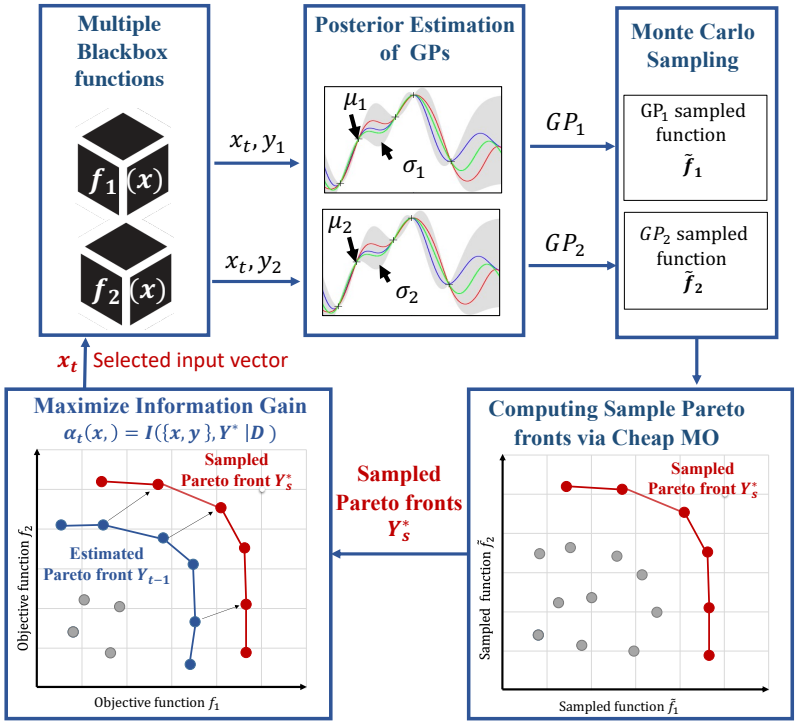

Figure 3.1: Overview of the MESMO algorithm for two objective functions (K=2). We build statistical models $\mathcal{GP}_1$, $\mathcal{GP}_2$ for the two objective functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ respectively. First, we sample functions from the statistical models. We compute sample Pareto fronts by solving a cheap MO problem over the sampled functions. Second, we select the best candidate input x_t that maximizes the information gain. Finally, we evaluate the functions for x_t to get (y_1, y_2) and update the statistical models using the new training example.

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 the 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 $w = Pr[(y_1^* > f_1(x^*)), \cdots, (y_K^* > f_K(x^*))]$. If $\mathbf{x}_t$ is the candidate input selected by MESMO at the t^{th} iteration according to 3.13 and $[y_1^*, \cdots, y_K^*]$ is drawn from P, then with probability at least $1 - \delta$, in $T' = \sum_{i=1}^{T} \log_w \frac{\delta}{2\pi_i}$ number of iterations

$$R(\mathbf{x}^*) = \sqrt{\sum_{j=1}^{K} \left(\left(v_{t^*}^j + \zeta_T \right)^2 \left(\frac{2T\gamma_T^j}{log(1+\sigma^{-2})} \right) \right)} \tag{3.14}$$

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 Appendix A. 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.

3.5 Experiments and Results

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

3.5.1 Experimental Setup

Multi-objective BO algorithms. We compare MESMO with existing methods described in the related work: ParEGO [117], PESMO [83], SMSego [158], EHI [67], and SUR [155]. We employ the code for these methods from the BO library Spearmint[2]. For methods requiring PHV computation, we employ the *PyGMO* library[3]. According to PyGMO documentation, the algorithm from [149] is employed for PHV computation. We did not include PAL [206] as it is known to have similar performance as SMSego [83] and works only for finite discrete input space. The code for our method is available at (github.com/belakaria/MESMO).

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 the same as the one in-built into 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[4]. We also employ two benchmarks from the general multi-objective optimization literature [152, 49]. We provide the complete details of these

[2]https://github.com/HIPS/Spearmint/tree/PESM
[3]https://esa.github.io/pygmo/
[4]https://www.sfu.ca/ ssurjano/optimization.html

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: Powell, Rastrigin, Dixon, Zakharov, Perm, and SumSquares. The dimension of input space d is 6.

3) OKA2-2,3: We evaluate two functions defined in [152]. The dimension of input space d is 3.

4) DTLZ1-4,5: We evaluate four functions defined in [49]. The dimension of input space d is 5.

Real-world benchmarks. We employed four real-world benchmarks with data available at [206, 164].

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 [127]. 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 [170] 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 [205]. 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 [4] 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 [4].

Evaluation metrics. We employ two common metrics. The *Pareto hypervolume (PHV)* metric measures the quality of a given Pareto front [202] and the R_2 metric degenerates to the regret metric presented in our theoretical analysis. We provide a detailed definition of the metrics in section 2.3

3.5.2 *Results and Discussion*

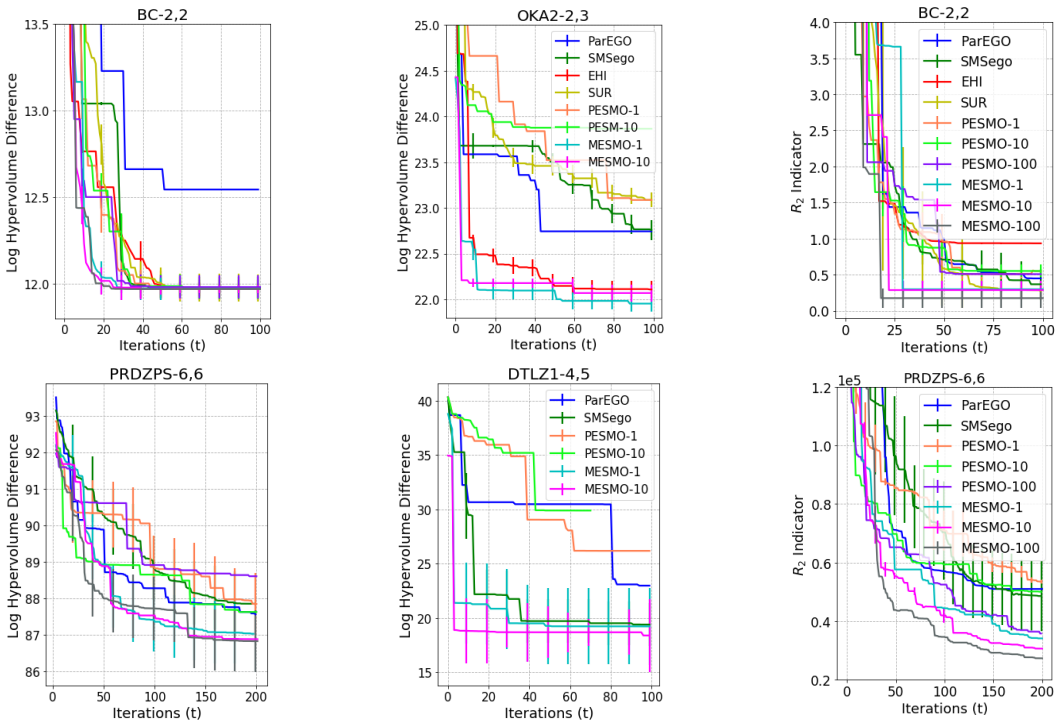

Figure 3.2: Results of different multi-objective BO algorithms including MESMO on synthetic benchmarks. The log of the hypervolume difference and the R_2 Indicator are shown with different number of function evaluations. The mean and variance of 10 different runs are plotted.

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.

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 3.2 and 3.3 show the results of all multi-objective BO algorithms including MESMO for synthetic and real-world benchmarks respectively. We make

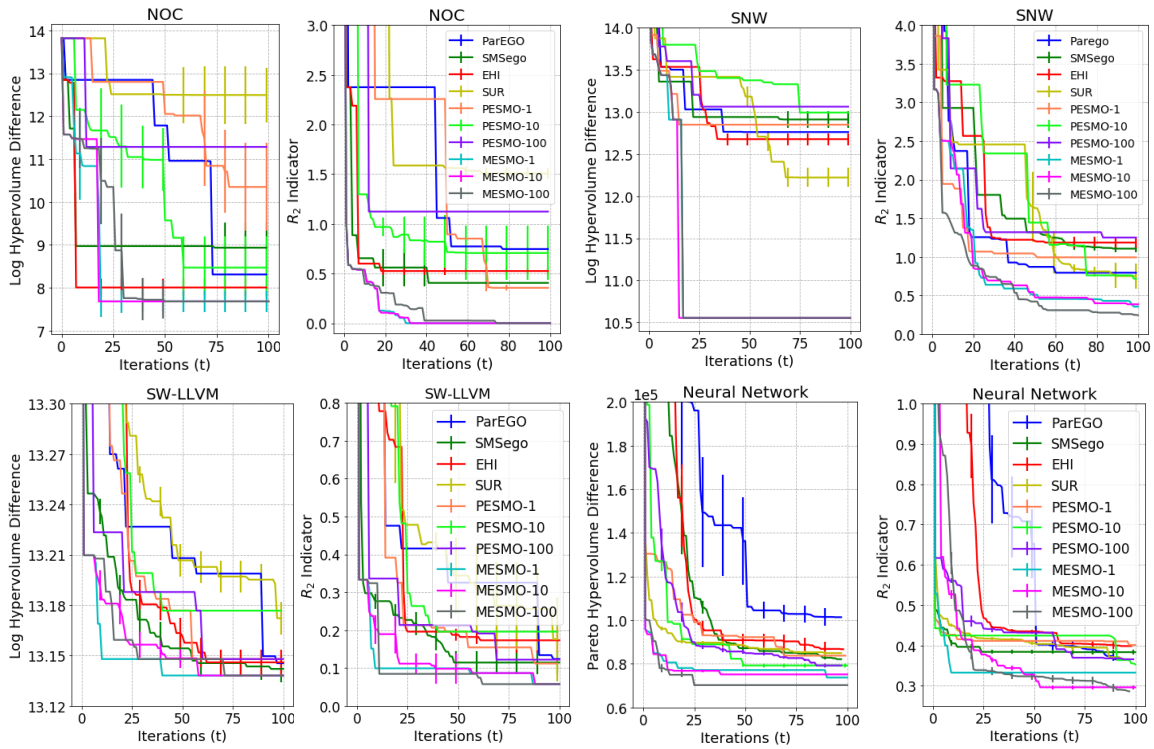

Figure 3.3: Results of different multi-objective BO algorithms including MESMO on real-world benchmarks. The log of the hypervolume difference and R_2 Indicator are shown with different number of function evaluations. The mean and variance of 10 different runs are plotted.

the 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 slightly 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 perfor-*

mance 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 3.1 shows the time in seconds two for synthetic benchmarks. We present additional time comparison results in Figure 3.4. 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 3.1: 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

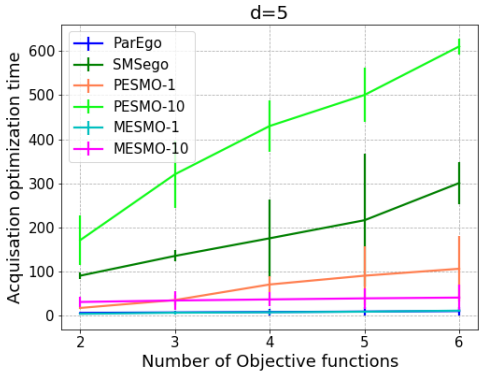

Figure 3.4: Results for acquisition function optimization time of different multi-objective BO algorithms including MESMO with increasing number of objective functions for fixed input space dimension $d = 5$.

3.6 Summary

We introduced a novel and principled approach referred to as MESMO to solve multi-objective Bayesian optimization problems. The key idea is to employ an output space entropy-based acquisition function (select the input with maximum information

gain about the optimal Pareto front) 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.

CHAPTER FOUR

UNCERTAINTY-AWARE SEARCH FOR MULTI-OBJECTIVE BO

In this chapter, we address the problem of multi-objective black-box optimization using expensive function evaluations, where the goal is to approximate the true Pareto set of solutions while 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 simulations.

We propose a novel uncertainty-aware search framework referred to as USeMO to efficiently select the sequence of inputs for evaluation to solve this problem. The key insight behind USeMO is a two-stage search procedure to improve the accuracy and computational-efficiency of sequential decision-making under uncertainty for selecting candidate inputs for evaluation. USeMO selects the inputs for evaluation as follows. First, it solves a cheap MO optimization problem defined in terms of the acquisition functions (one for each unknown objective) to identify a list of promising candidates. Second, it selects the best candidate from this list based on a measure of uncertainty. USeMO has several advantages: a) Does not reduce to single objective optimization problem; b) Allows to leverage a variety of acquisition functions designed for single objective BO; c) Computationally-efficient to solve MO problems with many objectives; and d) Improved uncertainty management via two-stage search procedure to select the candidate inputs for evaluation. We also provide theoretical analysis to

characterize the efficacy of our approach. We provide an experimental evaluation of USeMO on several synthetic and real-world problems.

4.1 Overview of USeMO Framework

As shown in Figure 4.1, USeMO is an iterative algorithm that involves four key steps. First, We build statistical models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ for each of the K objective functions from the training data in the form of past function evaluations. Second, we select a set of promising candidate inputs $\mathcal{X}_p$ by solving a cheap MO optimization problem defined using the statistical models. Specifically, multiple objectives of the cheap MO problem correspond to $\text{AF}(\mathcal{GP}_1, \mathbf{x}), \text{AF}(\mathcal{GP}_2, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x})$ respectively. Any standard acquisition function AF from single-objective BO (e.g., EI, TS) can be used for this purpose. The Pareto set $\mathcal{X}_p$ corresponds to the inputs with different trade-offs in the utility space for K unknown functions. Third, we select the best candidate input $\mathbf{x}_s \in \mathcal{X}_p$ from the Pareto set that maximizes some form of uncertainty measure for evaluation. Fourth, the selected input $\mathbf{x}_s$ is used for evaluation to get the corresponding function evaluations: $y_1{=}f_1(\mathbf{x}_s)$, $y_2{=}f_2(\mathbf{x}_s), \cdots, y_K{=}f_K(\mathbf{x}_s)$. The next iteration starts after the statistical models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ are updated using the new training example: input is $\mathbf{x}_s$ and output is $(y_1, y_2, \cdots, y_K)$. Algorithm 2 provides the algorithmic pseudocode for USeMO.

Advantages. USeMO has many advantages over prior methods. **1)** Provides flexibility to plug in any acquisition function for single-objective BO. This allows us to

leverage existing acquisition functions including EI, TS, and LCB. **2)** Unlike methods that reduce to single-objective optimization, USeMO has a better mechanism to handle uncertainty via a two-stage procedure to select the next candidate for evaluation: Pareto set obtained by solving cheap MO problem contains all promising candidates with varying trade-offs in the utility space and the candidate with maximum uncertainty from this list is selected. **3)** Computationally efficient to solve MO problems with many objectives.

Algorithm 2 USeMO Framework

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

1: Initialize training data of function evaluations $\mathcal{D}$
2: Initialize statistical models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ from $\mathcal{D}$
3: **for** each iteration $t = 1$ to T_{max} **do**
4: // Solve cheap MO problem with objectives $\text{AF}(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x})$ to get candidate inputs
5: $\mathcal{X}_p \leftarrow \min_{x \in \mathfrak{X}} (\text{AF}(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x}))$
6: // Pick the candidate input with maximum uncertainty
7: Select $\mathbf{x}_{t+1} \leftarrow \arg ma\mathbf{x}_{x \in \mathcal{X}_p} U_{\beta_t}(\mathbf{x})$
8: Evaluate $\mathbf{x}_{t+1} : \mathbf{y}_{t+1} \leftarrow (f_1(\mathbf{x}_{t+1}), \cdots, f_K(\mathbf{x}_{t+1}))$
9: Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})\}$
10: Update models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ using $\mathcal{D}$
11: $t \leftarrow t + 1$
12: **end for**
13: **return** Pareto set and Pareto front of $\mathcal{D}$

4.2 Key Algorithmic Components of USeMO

The two main algorithmic components of USeMO framework are: selecting most promising candidate inputs by solving a cheap MO problem and picking the best

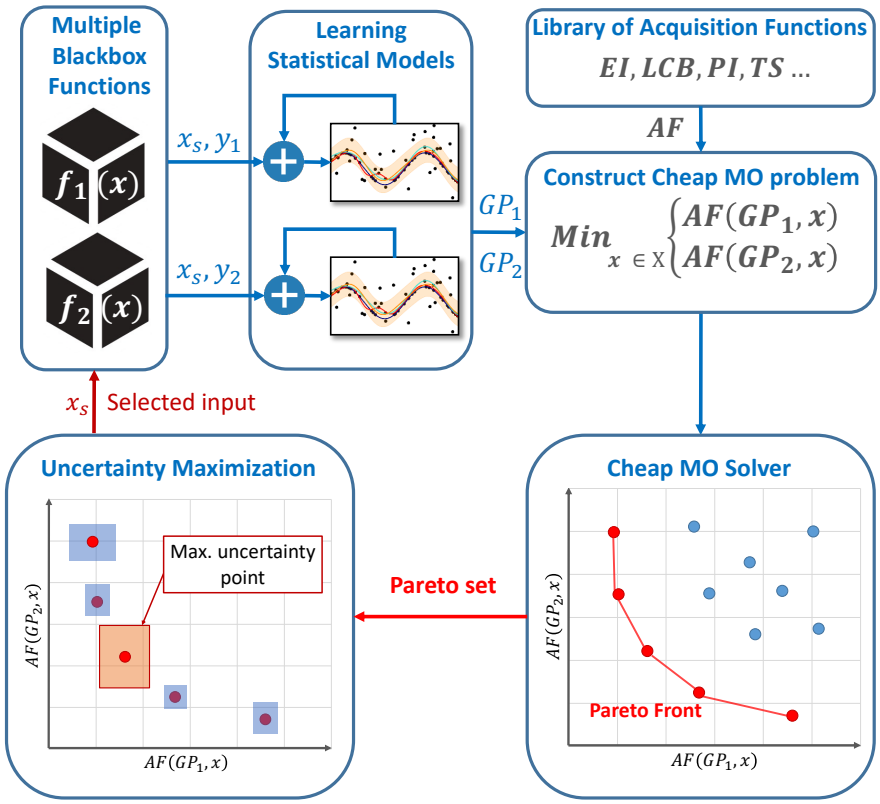

Figure 4.1: Overview of the USeMO framework for two objective functions (K=2). We build statistical models $\mathcal{GP}_1$, $\mathcal{GP}_2$ for the two objective functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$. In each iteration, we perform the following steps. First, we construct a cheap MO problem using the statistical models $\mathcal{GP}_1$ and $\mathcal{GP}_2$ and an input acquisition function AF: $\min_{x \in \mathfrak{X}}$ $(\text{AF}(\mathcal{GP}_1, \mathbf{x}), \text{AF}(\mathcal{GP}_2, \mathbf{x}))$ and employ a cheap MO solver to find the promising candidate inputs in the form of Pareto set. Second, we select the best candidate input $\mathbf{x}_s$ from the Pareto set based on a measure of uncertainty. Finally, we evaluate the functions for $\mathbf{x}_s$ to get $\mathbf{y}_s$=(y_1, y_2) and update the statistical models using the new training example.

candidate via uncertainty maximization. We describe their details below.

Selection of promising candidate inputs. We employ the statistical models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ towards the goal of selecting promising candidate inputs as fol-

lows. Given an acquisition function AF (e.g., EI), we construct a cheap multi-objective optimization problem with objectives $\text{AF}(\mathcal{GP}_1, \mathbf{x}), \text{AF}(\mathcal{GP}_2, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x})$, where $\mathcal{GP}_i$ is the statistical model for the unknown function f_i. Since we present the framework as minimization for the sake of technical exposition, all AFs will be minimized. The Pareto set $\mathcal{X}_p$ obtained by solving this cheap MO problem represents the most promising candidate inputs for evaluation.

$$\mathcal{X}_p \leftarrow \min_{x \in \mathfrak{X}} \left(\text{AF}(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x})\right) \tag{4.1}$$

Each acquisition function $\text{AF}(\mathcal{GP}_i, \mathbf{x})$ is dependent on the corresponding surrogate model $\mathcal{GP}_i$ of the unknown objective function f_i. Hence, each acquisition function will carry the information of its associated objective function. As iterations progress, using more training data, the models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ will better mimic the true objective functions $f_1, f_2, \cdots, f_K$. Therefore, the Pareto set of the acquisition function space (solution of Equation 4.1) becomes closer to the Pareto set of the true functions $\mathcal{X}^*$ with increasing iterations. Intuitively, the acquisition function $\text{AF}(\mathcal{GP}_i, \mathbf{x})$ corresponding to unknown objective function f_i tells us the utility of a point x for optimizing f_i. The input minimizing $\text{AF}(\mathcal{GP}_i, \mathbf{x})$ has the highest utility for f_i, but may have a lower utility for a different function f_j $(j \neq i)$. The utility of inputs for evaluation of f_j is captured by its own acquisition function $\text{AF}(\mathcal{GP}_j, \mathbf{x})$. Therefore, there is a trade-off in the utility space for all K different functions. The Pareto set $\mathcal{X}_p$

obtained by simultaneously optimizing acquisition functions for all K unknown functions will capture this utility trade-off. As a result, each input $x \in \mathcal{X}_p$ is a promising candidate for evaluation towards the goal of solving MOO problem. USeMO employs the same acquisition function for all K objectives. The main reason is to give equivalent evaluation for all functions in the Pareto front (PF) at each iteration. If we use different AFs for different objectives, the sampling procedure would be different. Additionally, the values of various AFs can have considerably different ranges. Thus, this can result in an unbalanced trade-off between functions in the cheap PF leading to the same unbalance in our final PF.

Cheap MO solver. We employ the popular NSGA-II algorithm [47] to solve the MO problem with cheap objective functions noting that any other algorithm can be used to similar effect. NSGA-II evaluates the cheap objective functions at several inputs and sorts them into a hierarchy of sub-groups based on the ordering of Pareto dominance. The similarity between members of each sub-group and their Pareto dominance is used by the algorithm to move towards more promising parts of the input space.

Picking the best candidate input. We need to select the best input from the Pareto set $\mathcal{X}_p$ obtained by solving the cheap MO problem. All inputs in $\mathcal{X}_p$ are promising in the sense that they represent the trade-offs in the utility space corresponding to different unknown functions. It is critical to select the input that will

guide the overall search towards the goal of quickly approximating the true Pareto set $\mathcal{X}^*$. We employ a uncertainty measure defined in terms of the statistical models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ to select the most promising candidate input for evaluation. In single-objective optimization case, the learned model's uncertainty for an input can be defined in terms of the variance of the statistical model. For multi-objective optimization case, we define the uncertainty measure as the volume of the uncertainty hyper-rectangle.

$$U_{\beta_t}(\mathbf{x}) = VOL(\{(LCB(\mathcal{GP}_i, \mathbf{x}), UCB(\mathcal{GP}_i, \mathbf{x})\}_{i=1}^{k}) \tag{4.2}$$

where LCB($\mathcal{GP}_i, \mathbf{x}$) and UCB($\mathcal{GP}_i, \mathbf{x}$) represent the lower confidence bound and upper confidence bound of the statistical model $\mathcal{GP}_i$ for an input x as defined in equations 2.4 and 2.5; and β_t is the parameter value to trade-off exploitation and exploration at iteration t. We employ the adaptive rate recommended by [174] to set the β_t value depending on the iteration number t. We measure the uncertainty volume measure for all inputs $x \in \mathcal{X}_p$ and select the input with maximum uncertainty for function evaluation.

$$\mathbf{x}_{t+1} = \arg ma\mathbf{x}_{x \in \mathcal{X}_p} U_{\beta_t}(\mathbf{x}) \tag{4.3}$$

4.3 Theoretical Analysis

In this section, we provide a theoretical analysis for the behavior of USeMO approach. MOO literature has multiple metrics to assess the quality of Pareto front approximation. Most commonly employed metrics include Pareto Hypervolume (PHV) indicator [202], R_2 indicator, and epsilon indicator [155]. Both epsilon and R_2 metrics are instances of distance-based regret, a natural generalization of the regret measure for single-objective problems. We consider the case of LCB acquisition function and extend the cumulative regret measure for single-objective BO proposed in the well-known work by Srinivasan et al., [174] to prove convergence results. However, our experimental results show the generality of USeMO with different acquisition functions including TS and EI. Prior work [155] has shown that R_2, *epsilon*, and PHV indicator show similar behavior. Indeed, our experiments validate this claim for UseMO. We present the theoretical analysis of USeMO in terms of asymptotic regret bounds. Since the point selected in the proof is arbitrary, it holds for all points. Hence, the regret bound can be easily adapted for both epsilon and R_2 metrics.

Let $\mathbf{x}^*$ be a point in the optimal Pareto set $\mathcal{X}^*$. Let $\mathbf{x}_t$ be a point in the Pareto set $\mathcal{X}_t$ estimated by USeMO approach by solving cheap MO problem at the t^{th} iteration. Let $R(\mathbf{x}^*) = \|R_1, \cdots, R_K\|$, where $R_i = \sum_{t=1}^{T_{max}} (f_i(\mathbf{x}_t) - f_i(\mathbf{x}^*))$ and $\|.\|$ is the norm of the K-vector and T_{max} is the maximum number of iterations. We discuss asymptotic bounds for this measure using GP-LCB as an acquisition function over the input set

$\mathcal{X}$. We provide proof details in Appendix B.

Lemma 1 Given $\delta \in (0,1)$ and $\beta_t = 2log(|\mathcal{X}|\pi^2 t^2/6\delta)$, the following holds with probability $1-\delta$:

$$|f_i(\mathbf{x}) - \mu_{i,t-1}(\mathbf{x})| \leq \beta_t^{1/2}\sigma_{i,t-1}(\mathbf{x}) \tag{4.4}$$

$$\text{for all} 1 \leq i \leq k, \mathbf{x} \in \mathcal{X}, \text{and } t \geq 1 \tag{4.5}$$

Theorem 1 If $\mathcal{X}_t$ is the Pareto set obtained by solving the cheap multi-objective optimization problem at t-th iteration, then the following holds with probability $1-\delta$,

$$R(\mathbf{x}^*) \leq \sqrt{\sum_{i=1}^{k} CT_{max}\beta_{T_{max}}\gamma^i_{T_{max}}} \tag{4.6}$$

where C is a constant and $\gamma^i_{T_{max}}$ is the maximum information gain about function f_i after T_{max} iterations. Essentially, this theorem suggests that since each term R_i in $R(\mathbf{x}^*)$ grows sub-linearly in the asymptotic sense, $R(\mathbf{x}^*)$ which is defined as the norm also grows sub-linearly. To the best of our knowledge, this is the first work to prove a *sub-linear regret* for multi-objective BO setting. We proved this result using the same AF for all objectives. This is a strong theoretical-proof that USeMO is already the best in this setting. This is one of the strong reasons that justify the use of single AF within USeMO framework.

4.4 Experiments and Results

In this section, we describe our experimental setup and present results of USeMO on diverse benchmarks.

4.4.1 Experimental Setup

Multi-objective BO algorithms. We compare USeMO with existing methods including ParEGO [117], PESMO [83], SMSego [158], EHI [67], and SUR [155]. We employ the code for these methods from the BO library Spearmint[1]. We present the results of USeMO with EI and TS acquisition functions — USeMO-TS and USeMO-EI — noting that results show similar trend with other acquisition functions. We did not include PAL [206] as it is known to have similar performance as SMSego [83] and works only for finite discrete input space. The code for our method is available at (github.com/belakaria/USeMO).

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 10 function evaluations. We initialize the GP models for all functions by sampling initial points at random from a Sobol grid using the in-built procedure in the Spearmint library. GPs are fitted using normalized objective function values to guarantee that all objectives are within the same range.

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

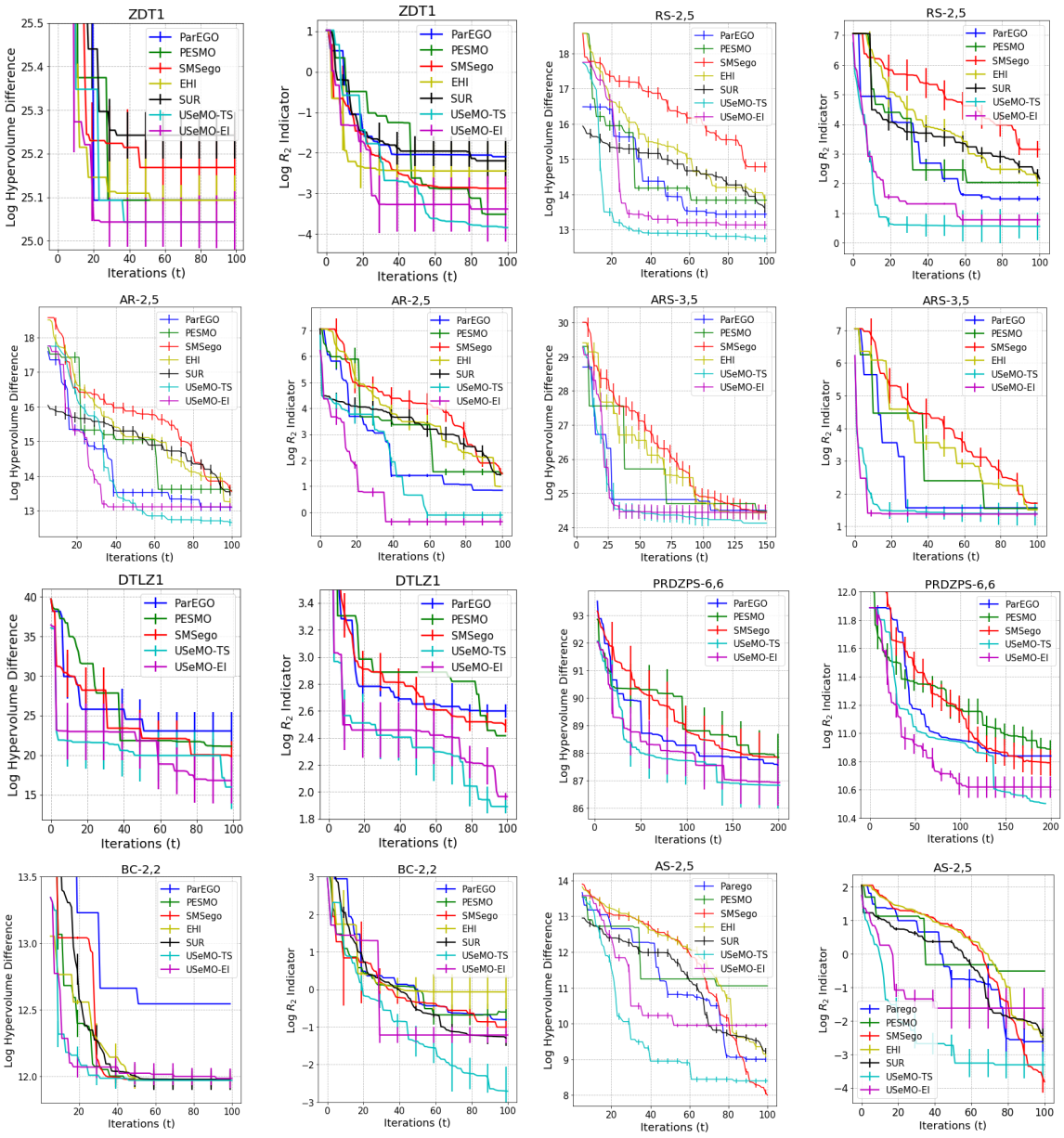

Figure 4.2: Results of different multi-objective BO algorithms including USeMO on synthetic benchmarks. The log of the hypervolume difference and log of R_2 Indicator are shown with different number of function evaluations (iterations). The mean and variance of 10 different runs are plotted. The tile of each figure refers to the benchmark name defined in Table 4.1.

Cheap MO solver. We employ the popular NSGA-II algorithm to solve the cheap MO problem noting that other solvers can be used to similar effect. For NSGA-II, the most important parameter is the number of function calls. We experimented with values varying from 1,000 to 20,000. We noticed that increasing this number does not result in any performance improvement for USeMO. Therefore, we fixed it to 1500 for all our experiments.

Name	Benchmark functions	K	d
BC-2,2	Branin-Currin	2	2
ZDT1	Zitzler,Deb,Thiele	2	4
AS-2,5	Ackley-Sphere	2	5
AR-2,5	Ackley-Rosenbrock	2	5
RS-2,5	Rosenbrock-Sphere	2	5
ARS-3,5	Ackley-Rosenbrock-Sphere	3	5
DTLZ1	Deb,Thiele,Laumanns,Zitzler	4	3
PRDZPS-6,6	Powell-Rastrigin-Dixon Zakharov-Perm-SumSquares	6	6

Table 4.1: Details of synthetic benchmarks: Name, benchmark functions, no. of objectives K, and input dimension d.

Synthetic benchmarks. We construct several synthetic multi-objective (MO) benchmark problems using a combination of commonly employed benchmark functions for single-objective optimization[2] and two of the known general MO benchmarks. We

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

provide the complete details of these MO benchmarks in Table 4.1. Due to space constraints, we present some of the results in the appendix

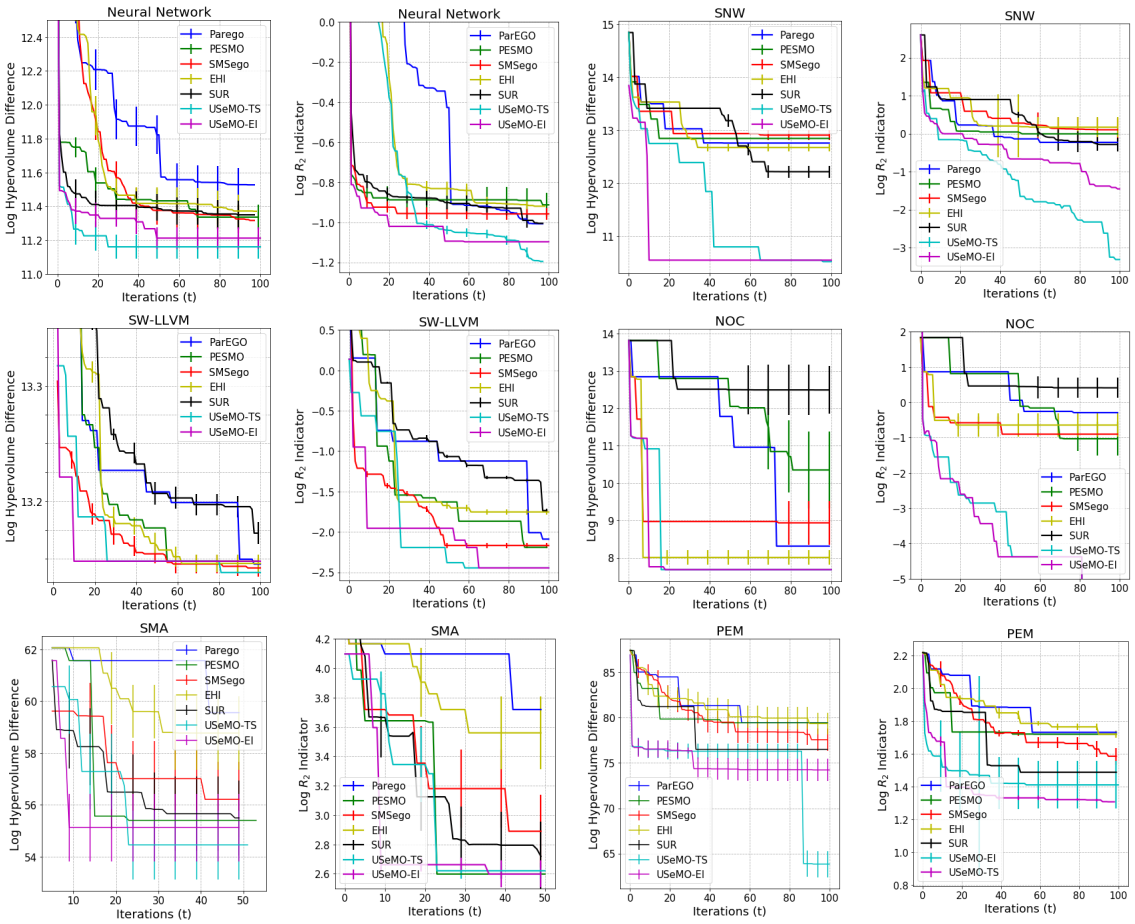

Figure 4.3: Results of different multi-objective BO algorithms including USeMO on real-world benchmarks. The log of the hypervolume difference and Log R_2 Indicator are shown with different number of function evaluations (iterations). The mean and variance of 10 different runs are plotted. The tile of each figure refers to the name of real-world benchmarks.

Real-world benchmarks. We employed six diverse real-world benchmarks for our experiments.

1) Hyper-parameter tuning of neural networks. Our goal is to find a neural

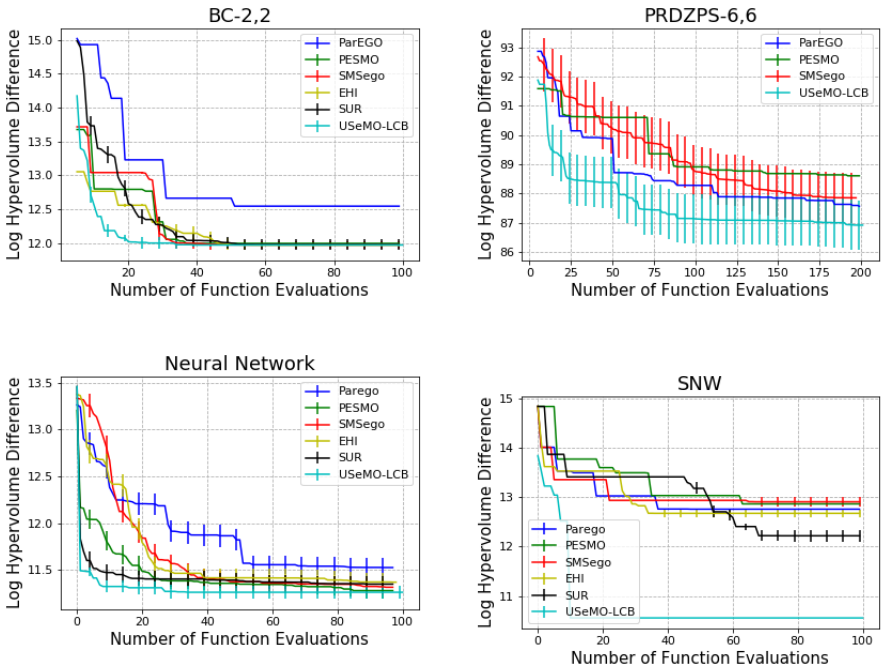

Figure 4.4: Comparison of USeMO-LCB with baseline algorithms. We plot the log of the hypervolume difference for synthetic and real-world benchmark problems as a function of the number of evaluations. The mean and variance of 10 different runs are plotted. The figure title refers to the benchmark name defined in the experiments section. (Better seen in color).

network with high accuracy and low prediction time. We optimize a dense neural network over the MNIST dataset [127]. 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 small values.

2) SW-LLVM compiler settings optimization. SW-LLVM is a data set with 1024 compiler settings [170] 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 settings takes over 20 days.

3) SNW sorting network optimization. The data set SNW was first introduced by [205]. 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 [4] 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.

5) Shape memory alloys (SMA) optimization. The materials dataset SMA consists of 77 different design configurations of shape memory alloys [78]. The goal is to optimize thermal hysteresis and transition temperature of alloys. Each design is defined by $d = 6$ input variables (e.g., atomic size of the alloying elements including metallic radius and valence electron number).

6) Piezo-electric materials (PEM) optimization. PEM is a materials dataset consisting of 704 configurations of Piezoelectric materials [78]. The goal is to optimize piezoelectric modulus and bandgap of these material designs. Each design configuration is defined by $d = 7$ input variables (e.g., ionic radii, volume, and density).

Evaluation metrics. We employ two common metrics. The *Pareto hypervolume (PHV)* metric measures the quality of a given Pareto front [202] and the R_2 metric degenerates to the regret metric presented in our theoretical analysis. We provide a detailed definition of the metrics in section 2.3

4.4.2 Results and Discussion

USeMO vs. State-of-the-art. We evaluate the performance of USeMO with different acquisition functions including TS, EI, and LCB. Due to space constraints, we show the results for USeMO with TS and EI, two very different acquisition functions, to show the generality and robustness of our approach. We also provide more results with LCB acquisition function in figure 4.4. Figure 4.2, 4.4 and 4.3 show the results of all multi-objective BO algorithms including USeMO for synthetic and real-world benchmarks respectively. We make the following empirical observations: 1) USeMO 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 USeMO varies with different acquisition functions (i.e., TS

and EI), but both cases perform better than baseline methods. 3) The convergence rate of PESMO becomes slower as the dimensionality of input space grows for a fixed number of objectives, whereas USeMO maintains a consistent convergence behavior. 4) Performance of ParEGO is very inconsistent. In some cases, it is comparable to USeMO, but performs poorly on many other cases. This is expected due to random scalarization.

Uncertainty maximization vs. random selection. Recall that USeMO needs to select one input for evaluation from the promising candidates obtained by solving a cheap MO problem. We compare uncertainty maximization and random policy for selection in figure 4.5 . We observe that uncertainty maximization performs better than random policy. However, in some cases, random policy is competitive, which shows that all candidates from the solution of cheap MO problem are promising and improve the efficiency.

Comparison of acquisition function optimization time. We compare the runtime of acquisition function optimization for different multi-objective BO algorithms including USeMO. 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 4.2 shows the time in seconds for synthetic benchmarks. We can see that USeMO

scales significantly better than state-of-the-art method PESMO. USeMO is comparable to ParEGO, which relies on scalarization to reduce to acquisition optimization in single-objective BO. The time for PESMO and SMSego increases significantly as the number of objectives grow beyond two.

MO Algorithms / Benchmarks	USeMO	PESMO	ParEGO	SMSego
BC-2,2	4.1 ± 0.7	13.6±3.2	4.2± 1.6	80.5± 2.1
ZDT1	5 ± 0.3	14.1±2.1	4.8± 1.2	84± 6.7
RS-2,5	5.3±1.4	16.9±1.9	5.7± 1.1	90.2±8.2
ARS-3,5	7.0 ±1.5	34.8±12.6	6.7±1.4	135.0 ±12.4
DTLZ1	9 ±2.4	63.6±10.1	8.2±0.9	215 ±16.2
PRDZPS-6,6	13.9± 1.1	110.4±17.8	12.3 ± 2.3	300.43 ± 35.7

Table 4.2: Acquisition function optimization time in secs.

4.5 Summary

We introduced a novel framework referred to as USeMO to solve multi-objective Bayesian optimization problems. The key idea is a two-stage search procedure to improve the accuracy and efficiency of sequential decision-making under uncertainty for selecting inputs for evaluation. Our experimental results on diverse benchmarks showed that USeMO yields consistently better results than state-of-the-art methods and scales gracefully to large-scale MO problems.

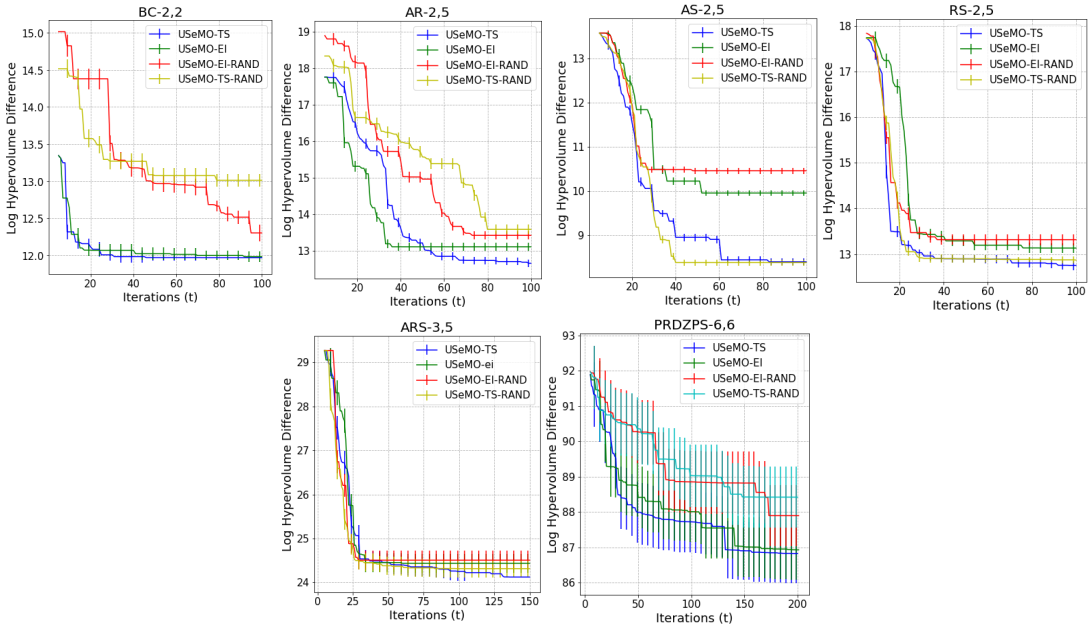

Figure 4.5: Comparison of USeMO with uncertainty maximization and random policy for selecting the best input from Pareto set obtained by solving cheap MO problem. We plot the log of the hypervolume difference for several synthetic benchmark problems as a function of the number of evaluations. The mean and variance of 10 different runs are plotted. The figure title refers to the benchmark name defined in table 4.1. (Better seen in color).

CHAPTER FIVE

PARETO FRONT-DIVERSE BATCH MULTI-OBJECTIVE BO

In many applications, practitioners care about the diversity of solutions, and we have the infrastructure to perform parallel experiments to accelerate the discovery of high-quality solutions. In this Chapter, motivated by these observations, we consider the problem of multi-objective optimization (MOO) of expensive black-box functions with the goal of discovering high-quality and diverse Pareto fronts where we are allowed to evaluate a batch of inputs in parallel.

We solve this problem in the framework of Bayesian optimization (BO) and propose a novel and effective approach referred to as **P***areto front-***D***iverse Batch Multi-Objective* **BO** *(PDBO)*. PDBO tackles two important challenges: 1) How to automatically select the best acquisition in each iteration of BO, and 2) How to select a diverse batch of inputs by considering multiple objectives. We propose principled solutions to address these two challenges. First, PDBO employs a multi-armed bandit approach to select one acquisition function from a given library. We solve a cheap MOO problem by assigning the selected acquisition function for each expensive objective function to obtain a candidate set of inputs for evaluation. Second, it utilizes Determinantal Point Processes (DPPs) to choose a Pareto-front-diverse batch of inputs for evaluation from the candidate set obtained from the first step. The key parameters for the methods behind these two steps are updated after each round of

function evaluations. Our experiments on multiple MOO benchmarks demonstrate that PDBO outperforms prior methods in terms of both the quality and diversity of Pareto solutions. [1]

5.1 Problem setup

Batch Multi-Objective Optimization (MOO). We consider a multi-objective optimization problem where the goal is to simultaneously optimize multiple conflicting functions. Let $\mathfrak{X} \subset \mathbb{R}^d$ be the input space of d design variables, where each candidate input $\mathbf{x} \in \mathfrak{X}$ is a d-dimensional input vector. And let $\{f_1, \cdots, f_K\}$ with $K \geq 2$ be the objective functions defined over the input space $\mathfrak{X}$ where $f_1(\mathbf{x}), \cdots, f_K(\mathbf{x}) : \mathfrak{X} \to \mathbb{R}$. We denote the functions evaluation at an input $\mathbf{x}$ as $\mathbf{y} = [y_1, \cdots, y_K]$, where $y_i = f_i(\mathbf{x})$ for all $i \in \{1, \cdots, K\}$. Without loss of generality, we assume minimization for all K objective functions. The optimal solution of the MOO problem is a set of inputs $\mathcal{X}^* \subset \mathfrak{X}$ such that no input $\mathbf{x}' \in \mathfrak{X} \setminus \mathcal{X}^*$ Pareto-dominates another input $\mathbf{x} \in \mathcal{X}^*$. A point $\mathbf{x}$ Pareto-dominates another point $\mathbf{x}'$ if and only if $\forall j : f_j(\mathbf{x}) \leq f_j(\mathbf{x}')$ and $\exists j : f_j(\mathbf{x}) < f_j(\mathbf{x}')$. The set of input solutions $\mathcal{X}^*$ is called the optimal *Pareto set* and the corresponding set of function values $\mathcal{Y}^*$ is called the optimal *Pareto front*. We are allowed to select B inputs for parallel evaluation in each iteration, and our overall goal is to uncover a high-quality and diverse Pareto front while minimizing the total number of expensive function evaluations.

[1] This work was co-led with Alaleh Ahmadian

5.2 Related Work

In this section, we describe the closely related work to the problem setup and proposed approach in this chapter.

Multi-Objective Bayesian Optimization. Compared to single-objective BO, there is relatively less work on multi-objective BO (MOBO). Prior work builds on the insights from single-objective BO methods [165, 84, 185, 90] to develop MOBO methods. Some of the recent work on MOBO include Predictive Entropy Search for Multi-objective Bayesian Optimization (PESMO) [83], Max-value Entropy Search for Multi-Objective Bayesian optimization (MESMO) [17], Multi-Objective Regionalized Bayesian Optimization (MORBO) [45], Uncertainty-aware Search framework for Multi-Objective Bayesian Optimization [22], Pareto-Frontier Entropy Search (PFES) [176], and Expected Hypervolume Improvement [68, 67]. Each of these methods has been shown to perform well on a variety of MOO problems. We provide a more thorough description of existing work in the MOO in section 3.2.

Batch Multi-Objective Optimization. The batch BO problem in the multi-objective setting is even much less studied. Diversity-Guided Efficient Multi-Objective Optimization (DGEMO) [118] approximates and analyzes a piecewise-continuous Pareto set representation which allows the algorithm to introduce a batch selection strategy that optimizes for both hypervolume improvement and diversity of selected samples. However, Konakovic Lukovic et al. [118] did not study or eval-

uate the Pareto-front diversity of the produced solutions. qEHVI (Differentiable Expected Hypervolume Improvement) [43] is an exact computation of the joint EHVI of q new candidate points (up to Monte-Carlo integration error). While previous EHVI formulations rely on gradient-free acquisition function optimization or approximated gradients, qEHVI computes the exact gradients of the MC estimator via auto-differentiation. qPAREGO is a novel extension of ParEGO [117, 43] that supports parallel evaluation and constraints. More recent work [134] proposed an approach to address MOO problems with continuous/infinite Pareto fronts by approximating the whole Pareto set via a continuous manifold. This approach enables a better preference-based exploration strategy for practitioners compared to prior work [1, 154, 12]. However, it is typically unknown to the user if the Pareto front is dense/continuous, especially in expensive function settings where the data is limited. *It is not known whether any of the proposed batch methods produce diverse Pareto fronts or not, as they were not evaluated on diversity metrics.* We perform an experimental evaluation to answer this question. We propose a novel approach ADAM-BO that utilizes Determinantal Point Processes to solve the batch multi-objective problem. Our proposed approach finds a diverse Pareto front while maintaining the quality of selected designs.

DPPs for Batch Single-Objective BO. Determinantal Point Processes are elegant probabilistic models that were first studied and introduced by [32, 31]. They charac-

terize the property of repulsion in a set of vectors and are well-suited for the selection of a diverse subset of inputs from a predefined set. There has been previous work on using DPPs for selecting a batch of inputs for evaluation in the single-objective BO literature [111, 144, 187]. However, *to the best of our knowledge, there is no work on using DPPs for multi-objective BO to uncover diverse Pareto fronts.*

Adaptive Acquisition Function Selection. There has been a plethora of research on finding efficient and reliable acquisition functions (AFs). However, prior work has shown that no single acquisition function is universally efficient and consistently outperforms all others [94]. GP-Hedge [85] proposed to use a portfolio of acquisition functions. The optimization of each AF will nominate an input, and the algorithm will select one of them for evaluation using the selection probabilities. The GP-Hedge method uses the Hedge strategy [73], a multi-arm bandit method designed to choose one action amongst a set of different possibilities using selection probabilities calculated based on the reward (performance given by function values) collected from previous evaluations.Vasconcelos et al. [181] extended Hoffman et al. [85] by proposing to use discounted cumulative reward and Vasconcelos et al. [182] suggested using Thompson sampling to automatically set the hedge hyperparameter η. The adaptive acquisition function selection was not extended previously to the context of multi-objective optimization. We propose a multi-arm bandit strategy with a novel reward function to dynamically select one acquisition function from a given library for multi-

objective optimization problems.

5.3 Proposed PDBO Algorithm

In this section, We start by providing an overview of the proposed PDBO algorithm accompanied by an illustrative figure that summarizes the overall workflow. Next, we explain our algorithms for two key components of PDBO, namely, adaptive acquisition function selection via a multi-arm bandit strategy and diverse batch selection via determinantal point processes for multi-objective output space diversity.

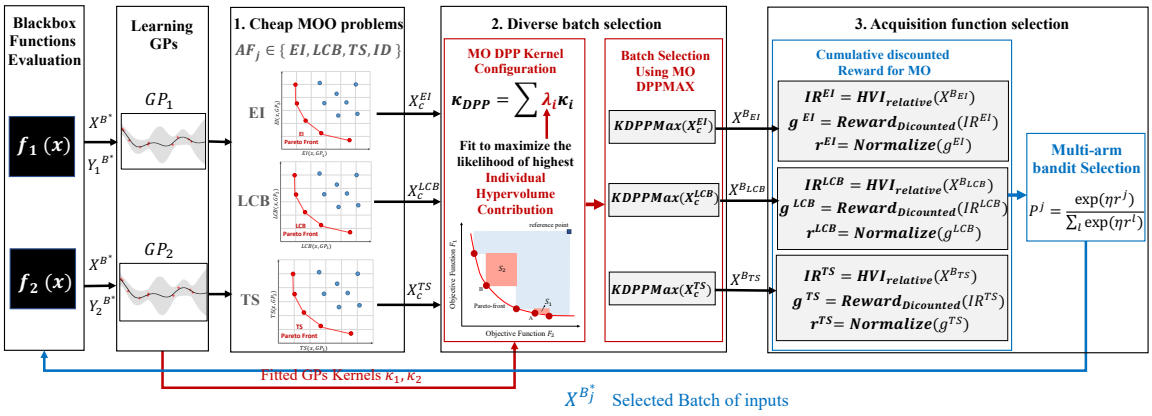

Figure 5.1: Overview of PDBO algorithm illustrating its key components explained in Section 5.3

Overview of PDBO. PDBO is an iterative algorithm. It introduces novel methods for selecting varying acquisition functions and for bringing the diversity of inputs into the multi-objective BO setting. The method builds K independent Gaussian processes $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$ as surrogates for each of the objective functions. Its three

key steps at each iteration t to select B inputs for evaluation are:

1. Solving multiple cheap MOO problems: PDBO takes as input a portfolio of notable acquisition functions, $\mathcal{P} = \{\text{AF}_1, \cdots, \text{AF}_M\}$ from the single-objective BO literature. It constructs M cheap MOO problems, each corresponding to one acquisition function. The multiple objectives defining the cheap MOO problems are acquisition functions respectively corresponding to the K objective functions. Solving cheap MOO problems will generate M cheap Pareto-sets of solutions $\mathcal{X}_c^1 \cdots \mathcal{X}_c^M$.

2. Diverse batch selection: From each cheap Pareto set $\mathcal{X}_c^j$, a batch $X_t^{B_j} \subset \mathcal{X}_c^j$ of B inputs is selected using a diversity-aware approach based on determinantal point processes (DPPs). Importantly, the adapted DPP is configured to favor the diversity in the output space and to handle multiple objective settings by using a principally fitted convex combination of the kernels of the K Gaussian processes. The convex combination scalars are strategically set to maximize the likelihood of selecting a diverse subset of inputs with respect to the Pareto front.

3. Acquisition function selection: From $\{X_t^{B_j};\ \forall\ j \in [1, \cdots, M]\}$, only one nominated subset would be selected using a multi-arm bandit strategy. The keys to this selection are probabilities $p^1, \cdots, p^j$, one for each acquisition function to capture their performance based on past iterations. p^j is the probability of selecting the batch generated by the acquisition function AF_j, defined in equation 5.5. These probabilities are updated based on the discounted cumulative reward r^j of each of the respective

acquisition functions. The reward values r^j are updated based on the quality of the batches nominated by the respective acquisition functions.

Algorithm 3 Pareto front-Diverse Batch Multi-Objective BO (PDBO)

Input: $\mathfrak{X}$ input space; $\{f_1, \cdots, f_K\}$, K black-box objective functions; $\mathcal{P} = \{\text{AF}_1, \cdots, \text{AF}_M\}$ portfolio of acquisition functions; B batch size; and T_{max} number of iterations

Initialize data $\mathcal{D}_0 = \{\mathcal{X}_0, \mathcal{Y}_0\}$ with N_0 initial points
for each iteration $t \in [1, T_{max}]$ **do**
  Fit statistical models $\mathcal{GP}_1, \cdots, \mathcal{GP}_k$ using $\mathcal{D}_{t-1}$
  for each acquisition function $\text{AF}_j \in \mathcal{P}$ **do**
    $\mathcal{X}_c^j \leftarrow \arg min_{\mathbf{x} \in \mathfrak{X}} (\text{AF}_j(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}_j(\mathcal{GP}_k, \mathbf{x}))$ // Solve cheap MOO problem
    $X_t^{B_j} \leftarrow$ DPP-Select$(\mathcal{X}_c^j, \{\mathcal{GP}_1, \cdots, \mathcal{GP}_k\}, \mathcal{D}_{t-1})$ // Select a batch of inputs from $\mathcal{X}_c^j$ using DPPs
  end for
  $\text{AF}_{j^*} \leftarrow$ Adaptive-AF-Select$(\mathcal{D}_{t-1}, \{X_{t-1}^{B_j}\}_{j=1}^M)$ // Select an AF using previously aggregated data
  $X_t^B = X_t^{B_{j^*}}$ // Choose the batch nominated by AF_{j^*}
  $Y_t^B \leftarrow \{[f_1(\mathbf{x}), \cdots, f_k(\mathbf{x})];\ \forall\ \mathbf{x} \in X_t^B\}$ // Evaluate objective functions for batch of inputs X_t^B
  $\mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\} \leftarrow \{\mathcal{X}_{t-1}, \mathcal{Y}_{t-1}\} \cup \{(X_t^B, Y_t^B)\}$
end for
return Pareto set $\mathcal{X}_{T_{max}}$ and Pareto front $\mathcal{Y}_{T_{max}}$

Algorithm 3 provides a pseudocode with high-level steps of the PDBO approach. The DPP-Select and Adaptive-AF-Select represent the second and third key steps. The details of these methods and their corresponding pseudocodes are provided in Sections 5.3.1 and 5.3.2, respectively.

5.3.1 Multi-arm Bandit Strategy for Adaptive Acquisition Function Selection

In this section, we propose a multi-arm bandit approach to *adaptively* select one acquisition function (AF) from a given library of AFs in each iteration of PDBO.

Multi-arm Bandit Formulation. We are given a portfolio of M acquisition functions $\mathcal{P} = \{\text{AF}_1, \cdots, \text{AF}_M\}$ and our goal is to adaptively select one AF in each

iteration of PDBO. Each acquisition function in $\mathcal{P}$ corresponds to one arm, and we need to select an arm based on the performance of past selections for solving the MOO problem. Inspired by the previous work on acquisition function selection and algorithm selection in the single objective setting [85, 181, 182], we propose an adaptive acquisition function selection approach for the multi-objective setting (see Algorithm 2). We explain the two main steps of this approach below.

Nominating Promising Candidates via Cheap MOO. In each PDBO iteration, we employ the updated statistical models $\{\mathcal{GP}_1 \cdots \mathcal{GP}_K\}$ and the portfolio $\mathcal{P}$ to generate M sets of candidate points. For each acquisition function AF_j, the algorithm constructs a cheap MOO problem with the objectives defined as $\text{AF}_j(\mathcal{GP}_1, \mathbf{x}) \cdots \text{AF}_j(\mathcal{GP}_K, \mathbf{x})$. Assuming minimization, the cheap MOO generate M Pareto-sets of solutions $\mathcal{X}_c^1 \cdots \mathcal{X}_c^M$ (one for each acquisition function) defined as:

$$\mathcal{X}_c^j \leftarrow \arg\min_{\mathbf{x} \in \mathfrak{X}} \{\text{AF}_j(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}_j(\mathcal{GP}_K, \mathbf{x})\} \tag{5.1}$$

We employ the algorithm proposed by Deb et al. [47] to solve cheap MOO problems defined in 5.1. From each $\mathcal{X}_c^j$, a batch $X_t^{B_j} \subset \mathcal{X}_c^j$ of B points is selected in iteration t using a diversity-aware approach described in Section 5.3.2. We denote the function evaluations of inputs in $X_t^{B_j}$ by $Y_t^{B_j}$.

Multi-Objective Reward Update. We employ the relative hypervolume improvement as the quality metric to define our reward. The Pareto hypervolume captures

the quality of nominated batches from a Pareto-dominance perspective and carries information about the represented trade-off between the multiple objectives.

Defining the immediate reward as the raw Pareto hypervolume of the nominated batch $IR_t^j = HV(Y_t^j)$ can lead to an undesirable assessment of the suitable acquisition function since a batch of points can have a large hypervolume value at iteration t but does not provide a significant improvement over the previous Pareto front $\mathcal{Y}_{t-1}$ while another batch nominated in iteration $t-1$ may have a smaller hypervolume $HV(Y_{t-1}^j)$ yet provides a higher improvement over $\mathcal{Y}_{t-2}$. Additionally, initial iterations might provide drastic hypervolume improvements even if the selected points are not optimal. To mitigate these issues, we use the relative hypervolume improvement as the immediate reward instead of the hypervolume. In each iteration, t of the BO algorithm, the immediate reward IR_t^j for each acquisition function AF_j; $\forall\ j \in \{1 \cdots M\}$ is defined as follows.

$$IR_t^j = \frac{HV(\tilde{\mathcal{Y}}_{t-1} \cup \tilde{Y}_t^j) - HV(\tilde{\mathcal{Y}}_{t-1})}{HV(\tilde{\mathcal{Y}}_{t-1})} \tag{5.2}$$

where $\tilde{\mathcal{Y}}_{t-1}$ is the Pareto front at iteration $t-1$ and $\tilde{Y}_t^j$ is the evaluation of the batch of points X_t^j nominated by AF_j computed using the *predictive mean of the updated GP based statistical models.*

As the optimization progresses, the statistical models provide a better representation of the objective functions, and the batches nominated by each acquisition

function become more informative about the quality of its selections. Therefore, the impact of the early iterations may become irrelevant later. Consequently, we employ a *discounted cumulative reward* for each acquisition function AF_j at iteration t is defined as g_t^j.

$$g_t^j = \gamma g_{t-1}^j + IR_t^j = \sum_{t' \leq t} \gamma^{t'-1} IR_{t'}^j \tag{5.3}$$

Where γ is a decay rate that trades off past and recent improvements. The use of the decay rate can lead to equal or comparable rewards in advanced iterations, causing the algorithm to select the acquisition function randomly. To address this problem, the discounted cumulative reward g_t^j should be normalized [181]. The rewards at the first iteration are all initialized to zero and then updated at each iteration t using the following expression:

$$r_t^j = \frac{g_t^j - g_{max}^j}{g_{max}^j - g_{min}^j} \tag{5.4}$$

where $g_{max}^j = max(\{g_{t'}^j, t' \in [1,t]\})$ and $g_{min}^j = \min(\{g_{t'}^j, t' \in [1,t]\})$.

Finally, the probability of selection of each AF at each iteration t is calculated using equation 5.5.

$$p_t^j = \frac{exp(\eta r_t^j)}{\sum_{l=1}^{M} exp(\eta r_t^l)} \; for \; j = 1 \ldots M \tag{5.5}$$

Remark. It is important to note that we are using a full information multi-arm bandit strategy that requires the reward to be updated for all possible actions (i.e., for all acquisition functions) at each iteration. Since we evaluate only the batch nominated by the selected AF, we achieve this by computing the reward using the

Algorithm 4 ADAPTIVE-AF-SELECT

Input: training data $\mathcal{D}_t$; Batches nominated by different AFs $\{X_t^{B_j}, \forall j \in \{1 \cdots M\}\}$

if $t == 0$ **then**
  $r_t^j = 0$ for $j = 1 \cdots M$
else
  Compute rewards: r_t^j for $j = 1 \cdots M$ using Equation 5.4
  Update probabilities: $p_t^j = \frac{exp(\eta r_t^j)}{\sum_{l=1}^{M} exp(\eta r_t^l)}$ for $j = 1 \cdots M$
end if
Select an acquisition function AF_{j^*} according to the probabilities $\{p_t^j\}_{j=1}^M$
return AF_{j^*}

predictive mean functions of the *updated* surrogate models. For this reason, we solve a cheap MOO problem for each $\text{AF}_j \in \mathcal{P}$ even though the acquisition function is selected based on the data from the previous iterations. Algorithm 4 provides the pseudocode of the adaptive acquisition function selection based on the estimated rewards and probabilities.

5.3.2 Determinantal Point Processes for Batch Selection

In this section, we explain our approach to select a batch of diverse inputs by configuring DPPs to promote *output space diversity*.

Determinantal Point Processes. (DPPs) [123] are probabilistic measures that characterize the property of repulsion in a set of vectors. DPPs are well-suited to model samples of a diverse subset of k points from a predefined set of n of points. Given a similarity function over a pair of points, DPPs assign a high probability of selection to the most diverse subsets according to the similarity function. The similarity function is typically defined as a kernel. Formally, given a DPP kernel

defined over a set S of n elements, the k-DPP distribution is defined as selecting a subset S' of size k with $S' \subset S$ with probability proportional to the determinant of the kernel:

$$Pr(S') = \frac{det(\kappa(S'))}{\sum_{|s|=k} det(\kappa(s))} \tag{5.6}$$

[111] introduced and formalized the use of DPP in the context of batch BO for the single-objective setting. Given the surrogate GP of the objective function, the covariance of the GP is used as the similarity function for the DPP. The approach selects the first point in the batch by maximizing the UCB acquisition function. Next, it creates a set of points referred to as relevance region by bounding the search space with the maximizer of the LCB acquisition function and manually discretizing the bounded space into a grid of n points. DPP selects the remaining $(k-1)$ points out of the n points in the relevance region.

[187, 151, 144] used similar techniques to apply DPPs to high-dimensional and discrete spaces. There exist two approaches to selecting a diverse subset with a fixed size via DPP: 1) Choosing the subset that maximizes the determinant, typically referred to as *DPP-max*; and 2) Sampling with the determinantal probability measure referred to as *DPP-sample*. In this paper, we will focus on DPP-max. Although selecting the subset that maximizes the determinant is an NP-Hard problem, several approximations were proposed [147]. A greedy strategy [111] provides an approximate

solution and was adopted in several Bayesian optimization papers [187].

Limitations of Prior Work and Challenges for MOO. We list the key limitations of prior methods for DPP-based batch selection in the single-objective setting as they are applicable to the multi-objective setting too. **L1)** How can we overcome the limitation of selecting the first point separately regardless of the DPP diversity? **L2)** How can we prevent the potential limitation of under-explored search space caused by the discretization of the space to create the relevance region set?

The key challenges to employing DPPs for batch selection in the multi-objective optimization setting include **C1)** How to define a kernel that captures the diversity for multiple objectives given that we have K separate surrogate models and their corresponding kernel? **C2)** How can the DPP kernel capture the Pareto front diversity and the trade-off between the objectives without compromising the Pareto quality of selected points?

DPPs for Multi-objective BO. Below, we propose principled methods to overcome the limitations of prior work on DPPs for batch selection (**L1** and **L2**) and address the two challenges (**C1** and **C2**).

Multi-objective Relevance Region. Our proposed algorithm naturally mitigates the two limitations of the single-objective DPP approach. Recall that the first step of PDBO algorithm (section 5.3.1) proposes to generate cheap approximate Pareto-sets which capture the trade-offs between the objectives in the utility space and might

include, with high probability, optimal points [22, 118]. We consider the cheap Pareto sets as the multi-objective relevance region. Our approach allows for generating the relevance region without manually discretizing the search space. Also, the full batch is selected from the multi-objective relevance region leading to a better diversity among all the points in the batch.

Multi-objective DPP Kernel Fitting. To overcome the challenges of using DPPs in the MOO setting, we build a new kernel κ_{dpp} that is defined as a convex combination of the K kernels of the statistical models (GPs) representing each of the black-box objective functions. Let $\Lambda = [\lambda_1, \cdots, \lambda_K]$ be a vector of size K where each λ_i corresponds to the convex combination scalar associated with kernel κ_i of the objective function f_i. The DPP kernel κ_{DPP} is defined as:

$$\kappa_{DPP} = \sum_{i=1}^{K} \lambda_i \cdot \kappa_i \quad st. \sum_{i=1}^{i=K} \lambda_i = 1 \tag{5.7}$$

The hyperparameters of the kernels $\kappa_i \; \forall \; i \in \{1, 2, \cdots K\}$ are fixed during the fitting of the Gaussian processes. In order to set the convex combination scalars Λ in a principled manner that promotes diverse batch selection, we propose to set the Λ to the values that maximize the log marginal likelihood of selecting points with the highest individual hypervolume contribution. The individual hypervolume contribution (HVC) of each point in the evaluated Pareto front $\mathcal{Y}_t$ (via evaluated training data D_t) is the reduction in hypervolume if the point is removed from the Pareto front.

HVC is considered a Pareto front (PF) diversity indicator [45]. Points in crowded regions of the Pareto front have smaller HVC values. Therefore, more Pareto front points with high HVC indicate more output space coverage and consequently, higher PF diversity. We provide an illustration in Figure 5.2.

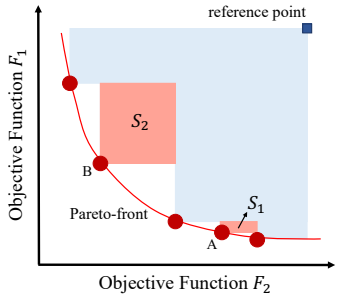

Figure 5.2: Sections S_1 and S_2 show the individual hypervolume contribution of points A and B, respectively. This figure shows that points with higher individual hypervolume contributions are also more diverse.

$$HVC(\mathbf{y}) = HV(\mathcal{Y}_t) - HV(\mathcal{Y}_t \setminus \{\mathbf{y}\}) \; \forall \; \mathbf{y} \in \mathcal{Y}_t \tag{5.8}$$

Given equation 5.8, we can construct the training set for the fitting of Λ. Let $\mathbf{C}_t$ $= [HVC(\mathbf{y}); \; \forall \; \mathbf{y} \in \mathcal{Y}_t]$ be a vector of the individual hypervolume contributions of currently evaluated points $\mathcal{X}_t \in \mathcal{D}_t$.

$$\Lambda^* = argmin_{\Lambda \in [0,1]^K} \log p(\mathbf{C}_t|\mathcal{X}_t) \qquad s.t \sum_{i=1}^{K} \lambda_i = 1 \tag{5.9}$$

$$\text{where } \log p(\mathbf{C}_t|\mathcal{X}_t) = -\frac{1}{2}\mathbf{C}_t^T \kappa_{DPP}^{-1} \mathbf{C}_t - \frac{1}{2}\log|\kappa_{DPP}| - \frac{n}{2}\log 2\pi$$

Algorithm 5 provides the pseudo-code for building the DPP kernel and selecting the diverse batch from a given candidate/cheap Pareto set $\mathcal{X}_c$.

Algorithm 5 DPP-SELECT

Input: cheap Pareto set $\mathcal{X}_c$; surrogate models $\mathcal{GP}_1, \cdots, \mathcal{GP}_k$; data $\mathcal{D}_t$.

1: $\mathbf{C}_t = [HVC(\mathbf{y}), \ \forall\, \mathbf{y} \in \mathcal{Y}_t]$ // Calculate the individual hypervolume contribution for each input $\in \mathcal{D}_t$
2: $\kappa_{DPP} = \sum_{i=1}^{K} \lambda_i \cdot \kappa_i \ st. \ \sum_{i=1}^{K} \lambda_i = 1$ // Construct κ_{DPP} as a convex combination of function kernels
3: $\Lambda^* = \arg min_{\Lambda \in [0,1]^\kappa} \log p(\mathbf{C}_t | \mathcal{X}_t) \quad s.t \sum_{i=1}^{K} \lambda_i = 1$ // Select λ_i values by maximizing the LML
4: Use the fitted κ_{DPP} kernel to select the most diverse points X_t^B from the cheap Pareto set $\mathcal{X}_c$ via DPP-max
5: **return** the selected B inputs, X_t^B

5.4 Experiments and Results

In this section, we provide experimental details and compare PDBO to the relevant state-of-the-art methods on multiple MOO benchmarks and varying batch sizes. We evaluate all methods using the hypervolume indicator and the Pareto front diversity (PFD) measure explained in section 5.1.

Benchmarks. We conduct experiments on benchmarks with varying numbers of input and output dimensions to show the versatility and flexibility of our method. We use ZDT-1 (d=25, k=2), ZDT-2 (d=4, k=2), ZDT-3 (d=12, k=2) [204], DTLZ-1 (d=10, k=4), DTLZ-3 (d=9, k=4), DTLZ-5 (d=12, k=6) [49] and the gear train design problem (d=4, k=3) [46]. In the ablation studies, we provide additional experiments with ZDT-1, ZDT-2, and ZDT-3 where we vary the number of input space dimensions, additional experiments with DTLZ-1, DTLZ-3, and DTLZ-5 where we

vary the input and output dimensions.

Baselines. We compare our method to the available state-of-the-art batch multi-objective optimization methods. DGEMO, qEHVI, qPAREGO, and USEMO-EI. We also include NSGA-II as the evolutionary algorithm baseline and random point selection. We set the hyperparameters of our method to $\gamma = 0.7$ and $\tau = 4$ as recommended by Hoffman et al. [85], Vasconcelos et al. [181]. We define the acquisition functions portfolio as $\mathcal{P}$=$\{EI, TS, UCB, ID\}$

Experimental Setup. All experiments are initialized with five random inputs/evaluations and run for at least 250 function evaluations. We conduct experiments with four different batch sizes $B \in \{2, 4, 8, 16\}$ and adjust the number of iterations accordingly. For instance, when using a batch size of two, we run the algorithm for 125 iterations. Each experiment is repeated 25 times, and we report the average and standard deviation of the hypervolume indicator and the PFD metric. To solve the constrained optimization problem in the DPP algorithm, we utilize an implementation of the Byrd-Omojokun Trust-Region Sequential Quadratic Programming (SQP) method [124, 148] from the Python SciPy library [183]. For baselines, we use the codes and hyperparameters provided in the open-source repositories of DGEMO [2] and Botorch [3]. The reference point for each benchmark is included in table 5.1.

[2]https://github.com/yunshengtian/DGEMO
[3]https://github.com/pytorch/botorch

Problem Name	Reference Point
ZDT-1	[11.0, 11.0]
ZDT-2	[11.0, 11.0]
ZDT-3	[11.0, 11.0]
DTLZ-1	[400.0, ..., 400.0]
DTLZ-3	[10000.0, ..., 10000.0]
DTLZ-5	[10.0, ..., 10.0]
Gear Train Design	[6.6764, 59.0, 0.4633]

Table 5.1: The multi-objective functions used for the experiments and their reference points

5.4.1 Results and Discussion

Our experimental results in Figure 5.3 demonstrates that PDBO outperforms all the baseline methods in most experiments with respect to the Hypervolume indicator and provides a competitive performance on the others. Additionally, PDBO outperforms all baselines with respect to the Pareto-front diversity metric.

We compare PDBO with state-of-the-art methods introduced in section 5.4 in terms of the Pareto front diversity. It is noteworthy that aside from DGEMO, none of the other baseline methods have a claim of Pareto front diversity. Figure 5.4 shows that PDBO outperforms all existing methods on the Pareto front diversity measure.

Advantages. PDBO is fast and effective in producing high-quality and diverse Pareto fronts. While outperforming the baseline methods, it can also be used with any number of input and output dimensions as well as being flexible to run with any batch size. The two state-of-the-art methods are DGEMO and qEHVI. The

DGEMO method fails to run for experiments with more than three objective functions as the graph cut algorithm consistently crashes (same observation was made by [44]). qEHVI fails to run with batch sizes higher than eight as the method becomes extremely memory-consuming even with GPUs. Therefore, PDBO's ability to easily run with any input and output dimensions as well as any batch size is an advantage for practitioners. PDBO is capable of proactively creating a diverse Pareto front while improving or maintaining the quality of Pareto front.

Ablation Studies. We provide additional results of ablation studies. Since the proposed algorithm relies on two main contributions (adaptive acquisition function selection and multi-objective batch selection via DPPs), we study the contribution of each component to the overall performance by running them separately.

Merits of the adaptive acquisition function selection strategy. We show that the adaptive acquisition function selection method introduced in section 5.3.1 outperforms the variant of PDBO with static acquisition functions from the portfolio. To isolate the impact of this part from the batch selection, we built this ablation with respect to the USEMO baseline due to its flexibility and effectiveness with a wide variety of different acquisition functions. We use a batch size of one and use a portfolio $\mathcal{P} = \{EI, TS, LCB, ID\}$. We run USEMO-UCB, USEMO-TS, USEMO-ID, and USEMO-EI as baselines. Then, we show the effectiveness of our adaptive acquisition function selection method by adding the adaptive selection approach introduced in

5.3.1 to USEMO. Therefore we are only comparing the static acquisition functions choice to the adaptive acquisition function selection. The results of this ablation, reported in figure 5.5, show that the multi-arm bandit strategy always performs better than the other acquisition functions or converges to the performance as the best-performing acquisition function.

Merits of the DPP-based batch selection for MOO. Similar to the previous ablation, we use the USEMO-EI baseline to isolate the impact of the DPP selection from the adaptive acquisition function selection. USEMO-EI solves a cheap MOO between K EI acquisition function using NSGA-II which produces an approximate cheap Pareto set. It later selects the next input to evaluate using an uncertainty metric. We substitute the input selection strategy proposed in USEMO with our DPP selection strategy and compare both strategies. We perform the ablation using several batch sizes $B = \{2, 4, 8, 16\}$. The results reported in figures 5.6, 5.7 and 5.8, show that the proposed DPP selection strategy (referred to as DPP-EI) outperforms the USEMO selection strategy on the diversity measure while improving the hypervolume quality.

5.5 Summary

We proposed the Pareto front-Diverse Batch Multi-Objective BO (PDBO) method based on the BO framework. It employs a full information multi-arm bandit algorithm with discounted reward to adaptively select the most suitable acquisition function in each iteration. We additionally proposed an appropriate reward based on the relative

hypervolume contribution of each acquisition function and a multi-objective DPP approach configured to select a batch of Pareto-diverse inputs for evaluation in each iteration. Experimental results on multiple benchmarks demonstrate that PDBO outperforms prior methods in terms of both diversity and quality of the Pareto-front solutions.

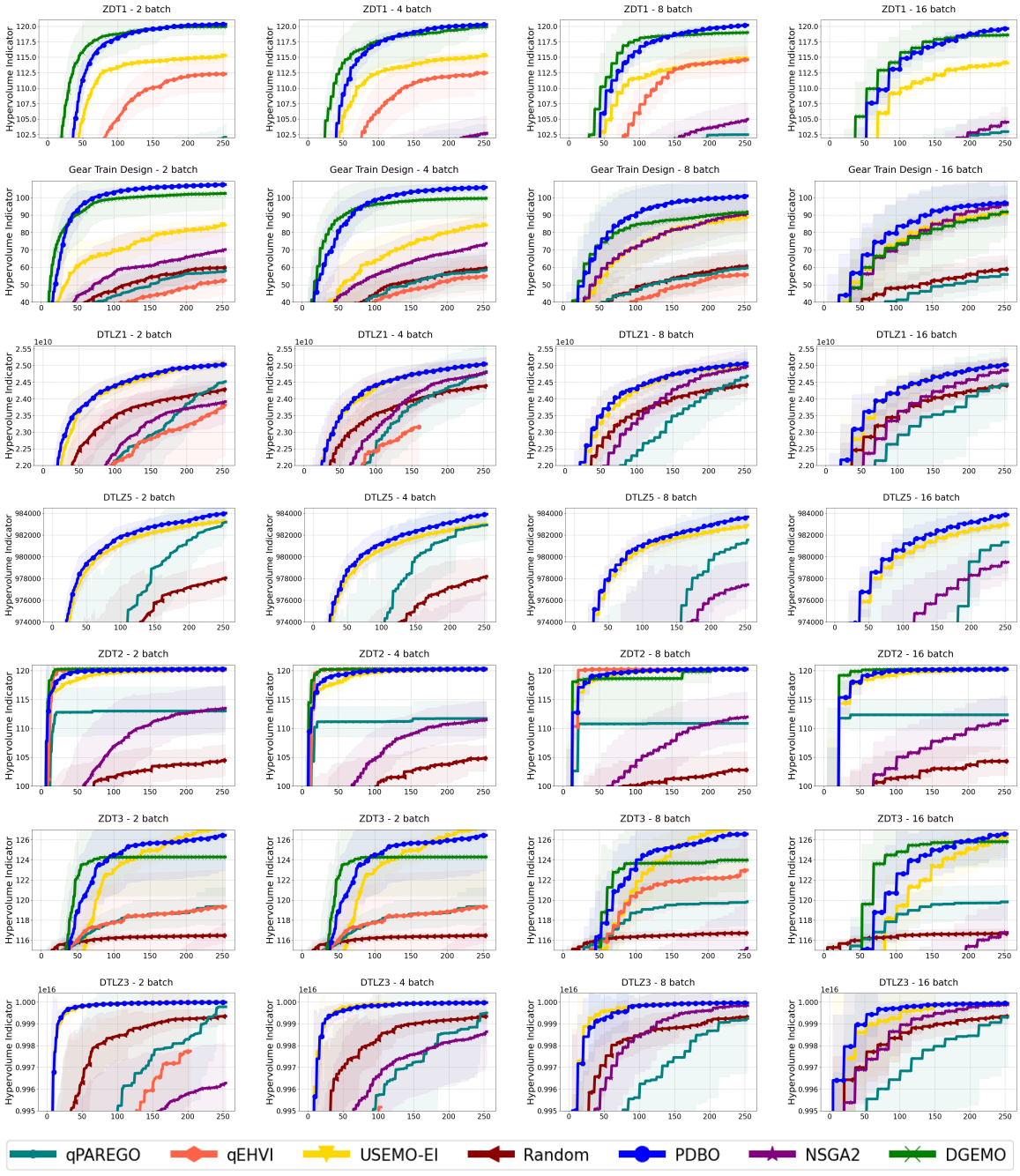

Figure 5.3: Hypervolume results evaluated on multiple benchmarks and batch sizes

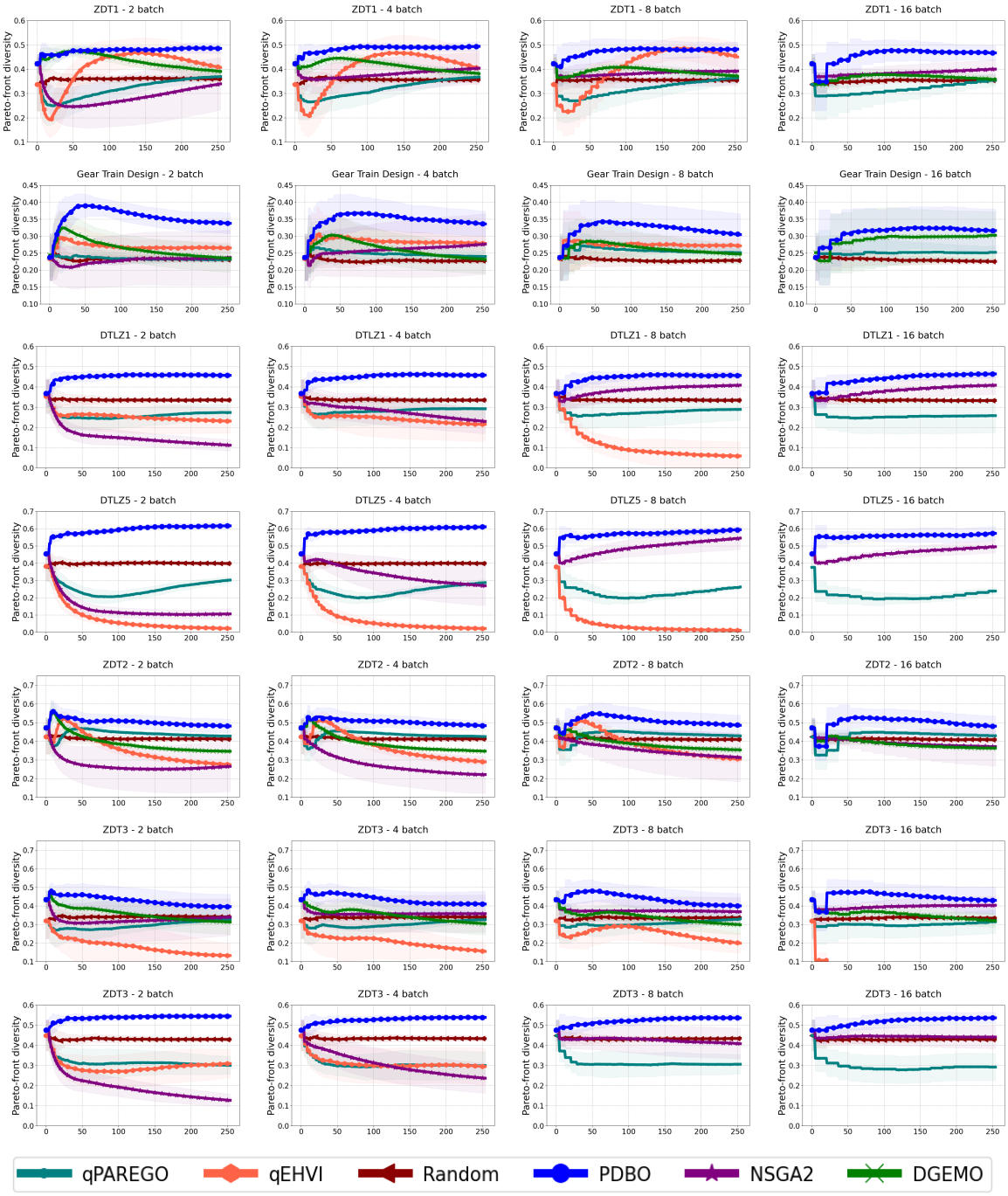

Figure 5.4: DPF results for all benchmarks

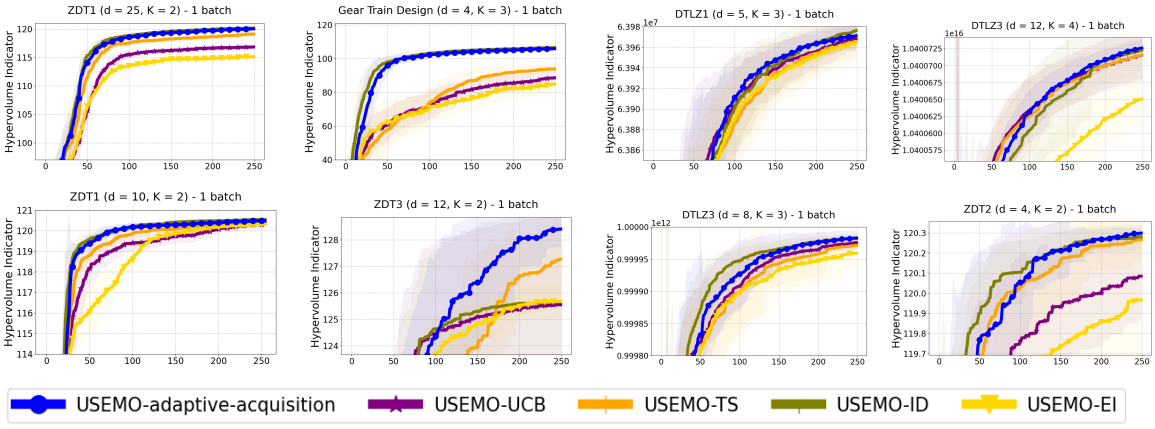

Figure 5.5: Acquisition functions comparison results

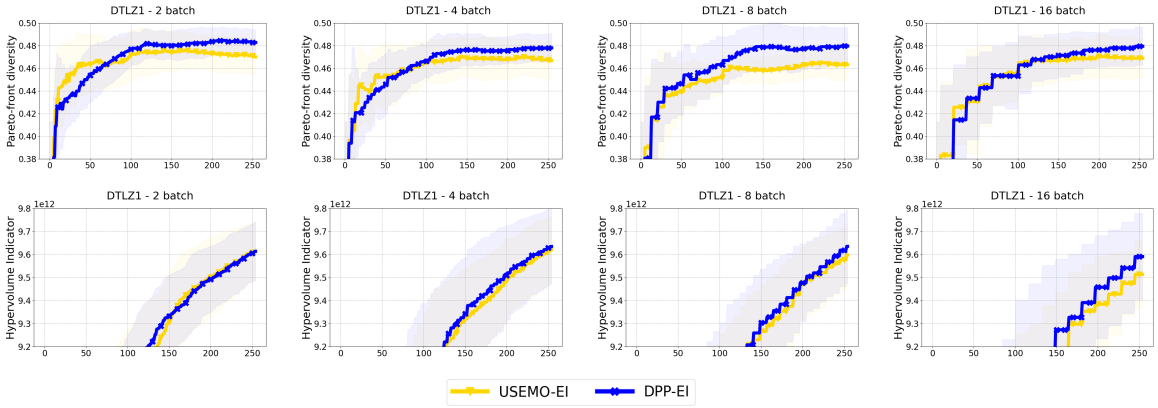

Figure 5.6: DPF and hypervolume results for DTLZ1 problem, d = 13, K = 5

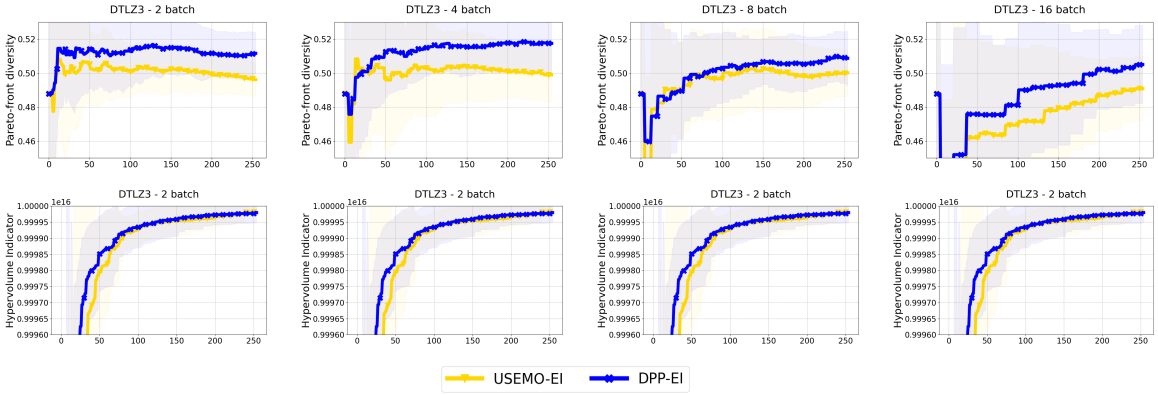

Figure 5.7: DPF and hypervolume results - DTLZ3 problem, d = 8, K = 4

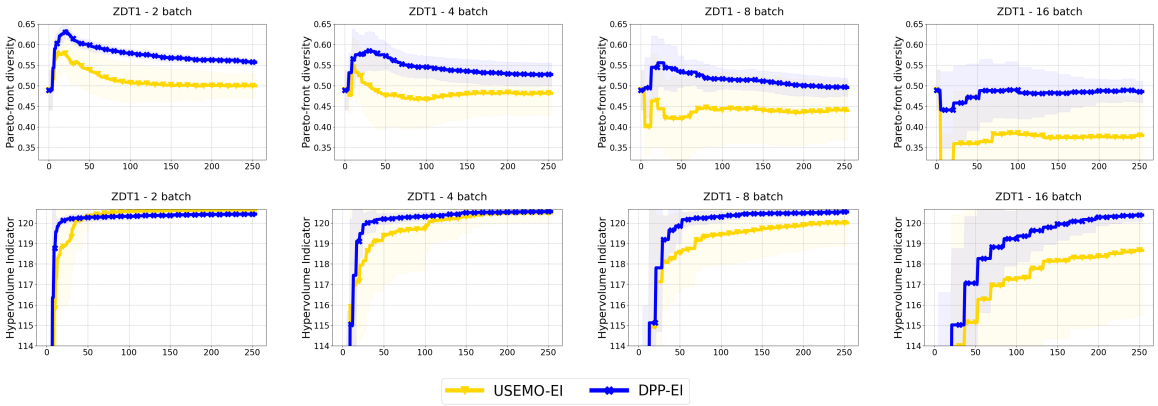

Figure 5.8: DPF and hypervolume results - ZDT1 problem, d = 3, K = 2

CHAPTER SIX

CONSTRAINED MULTI-OBJECTIVE BO

In this Chapter, we address the MOO problem with constraints, where the goal is to optimize multiple real-valued objective functions while satisfying several black-box constraints over the input space. For example, in aviation power system design applications, we need to find the designs that trade-off total energy and mass while satisfying specific thresholds for motor temperature and voltage of cells. This optimization requires performing expensive computational simulations to evaluate designs.

To solve this problem, we propose two algorithms, namely, Max-value Entropy Search for Multi-objective Optimization with Constraints (MESMOC) and Uncertainty aware Search Framework for Multi-Objective Bayesian Optimization with Constraints (USeMOC), which are generalizations of MESMO and USeMO respectively. MESMOC employs an output-space entropy-based acquisition function to efficiently select the sequence of inputs for evaluation to uncover high-quality pareto-set solutions while satisfying constraints. USeMOC selection method consists of solving a cheap constrained MO optimization problem via surrogate models of the true functions to identify the most promising candidates and picking the best candidate based on a measure of uncertainty. In what follows, we first describe the problem setup and discuss prior work. Then, we explain the technical details of both MESMOC and USeMOC algorithms. We applied the proposed algorithms to the design of a multi-

output switched-capacitor voltage regulator and the design of the power system of an unmanned areal vehicle.

6.1 Problem Setup

MOO Problem with Constraints This is a generalization of the basic MOO problem, where we need to satisfy some black-box constraints. Our goal is to maximize real-valued objective functions $f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_K(\mathbf{x})$, with $K \geq 2$, while satisfying L black-box constraints of the form $C_1(\mathbf{x}) \geq 0, C_2(\mathbf{x}) \geq 0, \cdots, C_L(\mathbf{x}) \geq 0$ over continuous space $\mathfrak{X} \subseteq \Re^d$. Each evaluation of an input $\mathbf{x} \in \mathfrak{X}$ produces a vector of objective values and constraint values $\mathbf{y} = (y_{f_1}, y_{f_2}, \cdots, y_{f_K}, y_{c_1} \cdots y_{c_L})$ where $y_{f_j} = f_j(\mathbf{x})$ for all $j \in \{1, 2, \cdots, K\}$ and $y_{c_i} = C_i(\mathbf{x})$ for all $i \in \{1, 2, \cdots, L\}$.We say that a valid input $\mathbf{x}$ (satisfies all constraints) *Pareto-dominates* another input $\mathbf{x}'$ if $f_j(\mathbf{x}) \geq f_j(\mathbf{x}') \, \forall j$ and there exists some $j \in \{1, 2, \cdots, K\}$ such that $f_j(\mathbf{x}) > f_j(\mathbf{x}')$. The goal of multi-objective BO with constraints is to approximate the Pareto set over valid inputs $\mathcal{X}^*$ while minimizing the number of expensive function evaluations. For example, in electric aviation power system design applications, we need to find designs that trade off total energy and mass while satisfying specific thresholds for motor temperature and voltage of cells. Table 2.1 contains all the mathematical notations used in this section.

Surrogate Models. We model the objective functions and black-box constraints by independent GP models $\mathcal{GP}_{f_1}, \mathcal{GP}_{f_2}, \cdots, \mathcal{GP}_{f_K}$ and $\mathcal{GP}_{c_1}, \mathcal{GP}_{c_2}, \cdots, \mathcal{GP}_{f_K}$ 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 $\mathbf{y}_i = \{y_{f_1}^i, \cdots, y_{f_K}^i, y_{c_1}^i, \cdots y_{c_L}^i\}$ is the output vector resulting from evaluating the objective functions and constraints at $\mathbf{x}_i$. We learn surrogate models from $\mathcal{D}$.

6.2 Related Work

There exists very limited prior work to address constrained MOO problems [75, 70]. PESMOC [75] is the current state-of-the-art method for this problem setting. PESMOC extends the information-theoretic approach PESMO that relies on the principle of input space entropy search to the constrained setting. As a consequence, it inherits the drawbacks of PESMO. MESMOC+ [71] is a concurrent work that also employs the principle of output space entropy search to solve constrained multi-objective optimization problems. However, our proposed MESMOC approach uses a completely different approximation of the information gain leading to a different expression of the acquisition function. This method employs a series of complex mathematical approximations based on Assumed Density Filtering (ADF).

To overcome the shortcomings of the existing work, we propose two approaches: 1) Our proposed MESMOC algorithm uses the truncated Gaussian distribution approximation that results in a closed-form expression, fast, and easy to implement acquisition function. Additionally, the ADF based method [71] considers blackbox constraints only in the acquisition function definition while MESMOC addresses the

constraints both in the acquisition function expression and in the acquisition function optimization process to ensure the selection of valid inputs. 2) Our proposed uncertainty aware approach, USEMOC, provides a flexible and fast framework enabling practitioners to leverage approaches from the single-objective BO literature and also handles both white-box and grey-box constraints.

6.3 Max-value Entropy Search for MOO with Constraints

6.3.1 MESMOC Acquisition Function

To overcome the challenges of computing input space entropy-based acquisition function, MESMO [17] proposed to maximize the information gain about the optimal **Pareto front**. However, MESMO did not address the challenge of constrained Pareto front. We propose an extension of MESMO's acquisition function to maximize the information gain between the next candidate input for evaluation $\mathbf{x}$ and constrained Pareto front $\mathcal{Y}^*$ given as:

$$\alpha(\mathbf{x}) = I(\{\mathbf{x}, \mathbf{y}\}, \mathcal{Y}^* \mid D) = H(\mathbf{y} \mid D, \mathbf{x}) - \mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}^*)] \tag{6.1}$$

In this case, the output vector $\mathbf{y}$ is $K+L$ dimensional: $\mathbf{y} = (y_{f_1}, y_{f_2}, \cdots, y_{f_K}, y_{c_1} \cdots y_{c_L})$ where $y_{f_j} = f_j(\mathbf{x})$ for all $j \in \{1, 2, \cdots, K\}$ and $y_{c_i} = C_i(\mathbf{x})$ for all $i \in \{1, 2, \cdots, L\}$. Consequently, the first term in the r.h.s of equation (6.1), entropy of a factorizable

$(K+L)$-dimensional Gaussian distribution $P(\mathbf{y} \mid D, \mathbf{x}$, can be computed in closed form as shown below:

$$H(\mathbf{y} \mid D, \mathbf{x}) = \frac{(K+L)(1+\ln(2\pi))}{2} + \sum_{j=1}^{K} \ln(\sigma_{f_j}(\mathbf{x})) + \sum_{i=1}^{L} \ln(\sigma_{c_i}(\mathbf{x})) \tag{6.2}$$

where $\sigma^2_{f_j}(\mathbf{x})$ and $\sigma^2_{c_i}(\mathbf{x})$ are the predictive variances of j^{th} function and i^{th} constraint GPs respectively at input $\mathbf{x}$.

The l.h.s of equation (6.1) can be decomposed in a similar way to equation (3.8). There are two key algorithmic steps to compute this part of the equation: 1) The first is how to compute Pareto front samples $\mathcal{Y}^*_s$?; and 2) The second is 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 and constraints from the posterior GP models via random Fourier features [84, 159] and then solve a cheap constrained multi-objective optimization over the K sampled functions and L sampled constraints.

Cheap MO solver. We sample $\tilde{f}_i$ from GP model $\mathcal{GP}_{f_j}$ for each of the K functions and $\tilde{C}_i$ from GP model $\mathcal{GP}_{c_i}$ for each of the L constraints. A *cheap* constrained multi-objective optimization problem over the K sampled functions $\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f}_K$ and the L sampled constraints $\tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_L$ is solved to compute the sample Pareto

front $\mathcal{Y}_s^*$. We employ the popular constrained NSGA-II algorithm [47, 48] to solve the constrained MO problem with cheap sampled objective functions and constrained.

2) Entropy Computation with a Sample Pareto Front. Let $\mathcal{Y}_s^* = \{\mathbf{v}^1, \cdots, \mathbf{v}^l\}$ be the sample Pareto front, where l is the size of the Pareto front and each $\mathbf{v}^i$ is a $(K+L)$-vector evaluated at the K sampled functions and L sampled constraints $\mathbf{v}^i = \{v^i_{f_1}, \cdots, v^i_{f_K}, v^i_{c_1}, \cdots, v^i_{c_L}\}$. The following inequality holds for each component y_j of the $(K+L)$-vector $\mathbf{y} = \{y_{f_1}, \cdots, y_{f_K}, y_{c_1}, \cdots y_{c_L}\}$ in the entropy term $H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*)$:

$$y_j \leq \max\{v_j^1, \cdots v_j^l\} \quad \forall j \in \{f_1, \cdots, f_K, c_1, \cdots, c_L\} \tag{6.3}$$

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 l $(K+L)$-vectors in the Pareto front $\mathcal{Y}_s^*$. This inequality had been proven by a contradiction for MESMO [17] for $j \in \{f_1, \cdots, f_K\}$. We assume the same for $j \in \{c_1, \cdots, c_L\}$.

By combining the inequality (6.3) and the fact that each function is modeled as an independent GP, we can approximate each component y_j as a truncated Gaussian distribution since the distribution of y_j needs to satisfy $y_j \leq \max\{v_j^1, \cdots v_j^l\}$. Let $y_s^{c_i *} = \max\{v_{c_i}^1, \cdots v_{c_i}^l\}$ and $y_s^{f_j *} = \max\{v_{f_j}^1, \cdots v_{f_j}^l\}$. 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 [38]:

$$H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*) = \sum_{j=1}^{K} H(y_{f_j}|D, \mathbf{x}, y_s^{f_j*}) + \sum_{i=1}^{C} H(y_{c_i}|D, \mathbf{x}, y_s^{c_i*}) \tag{6.4}$$

The r.h.s is a summation over entropies of $(K+L)$-variables $\mathbf{y} = \{y_{f_1}, \cdots, y_{f_K}, y_{c_1}, \cdots y_{c_L}\}$. The differential entropy for each y_j is the entropy of a truncated Gaussian distribution [137] and given by the following equations:

$$H(y_{f_j}|D, \mathbf{x}, y_s^{f_j*}) \simeq \left[\frac{(1+\ln(2\pi))}{2} + \ln(\sigma_{f_j}(\mathbf{x})) + \ln \Phi(\gamma_s^{f_j}(\mathbf{x})) - \frac{\gamma_s^{f_j}(\mathbf{x})\phi(\gamma_s^{f_j}(\mathbf{x}))}{2\Phi(\gamma_s^{f_j}(\mathbf{x}))}\right] \tag{6.5}$$

$$H(y_{c_i}|D, \mathbf{x}, y_s^{c_i*}) \simeq \left[\frac{(1+\ln(2\pi))}{2} + \ln(\sigma_{c_i}(\mathbf{x})) + \ln \Phi(\gamma_s^{c_i}(\mathbf{x})) - \frac{\gamma_s^{c_i}(\mathbf{x})\phi(\gamma_s^{c_i}(\mathbf{x}))}{2\Phi(\gamma_s^{c_i}(\mathbf{x}))}\right] \tag{6.6}$$

Consequently we have:

$$\begin{aligned} H(\mathbf{y} \mid D, \mathbf{x}, \mathcal{Y}_s^*) \simeq & \sum_{j=1}^{K} \left[\frac{(1+\ln(2\pi))}{2} + \ln(\sigma_{f_j}(\mathbf{x})) + \ln \Phi(\gamma_s^{f_j}(\mathbf{x})) - \frac{\gamma_s^{f_j}(\mathbf{x})\phi(\gamma_s^{f_j}(\mathbf{x}))}{2\Phi(\gamma_s^{f_j}(\mathbf{x}))}\right] \\ & + \sum_{i=1}^{L} \left[\frac{(1+\ln(2\pi))}{2} + \ln(\sigma_{c_i}(\mathbf{x})) + \ln \Phi(\gamma_s^{c_i}(\mathbf{x})) - \frac{\gamma_s^{c_i}(\mathbf{x})\phi(\gamma_s^{c_i}(\mathbf{x}))}{2\Phi(\gamma_s^{c_i}(\mathbf{x}))}\right] \end{aligned} \tag{6.7}$$

where $\gamma_s^{c_i}(\mathbf{x}) = \frac{y_s^{c_i*} - \mu_{c_i}(\mathbf{x})}{\sigma_{c_i}(\mathbf{x})}$, $\gamma_s^{f_j}(\mathbf{x}) = \frac{y_s^{f_j*} - \mu_{f_j}(\mathbf{x})}{\sigma_{f_j}(\mathbf{x})}$, ϕ and Φ are the p.d.f and c.d.f of a standard normal distribution respectively. By combining equations (6.2) and (6.7) with equation (6.1), we get the final form of our acquisition function as shown below:

$$\alpha(\mathbf{x}) \simeq \frac{1}{S} \sum_{s=1}^{S} \left[\sum_{j=1}^{K} \frac{\gamma_s^{f_j}(\mathbf{x}) \phi(\gamma_s^{f_j}(\mathbf{x}))}{2\Phi(\gamma_s^{f_j}(\mathbf{x}))} - \ln \Phi(\gamma_s^{f_j}(\mathbf{x})) + \sum_{i=1}^{L} \frac{\gamma_s^{c_i}(\mathbf{x}) \phi(\gamma_s^{c_i}(\mathbf{x}))}{2\Phi(\gamma_s^{c_i}(\mathbf{x}))} - \ln \Phi(\gamma_s^{c_i}(\mathbf{x})) \right] \tag{6.8}$$

Finding feasible initial designs. The acquisition function defined in equation 6.8 will build constrained Pareto front samples $\mathcal{Y}_s^*$ by sampling functions and constraints from the Gaussian process posterior. The posterior of the GP is built based on the current training data $\mathcal{D}$. The truncated Gaussian approximation defined in Equations 6.5 and 6.6 requires the upper bound $y_s^{f_j*}$ and $y_s^{c_i*}$ to be defined. However, in the early Bayesian optimization iterations of the algorithm, the configurations evaluated may not include any feasible design parameters. This is especially true for scenarios where the fraction of feasible design configurations in the entire design space is very small. In such cases, the sampling process of the constrained Pareto fronts $\mathcal{Y}_s^*$ is susceptible to failure because the surrogate models did not gather any knowledge about feasible regions of the design space *yet*. Consequently, the upper bounds $y_s^{f_j*}$ and $y_s^{c_i*}$ are not well-defined and the acquisition function in 6.8 is not well-defined. Intuitively, the algorithm should first aim at identifying feasible design configurations

by maximizing the probability of satisfying all the constraints. We define a special case of our acquisition function for such challenging scenarios as shown below:

$$\alpha_{prob}(\mathbf{x}) = \prod_{i=1}^{L} Pr(C_i(\mathbf{x}) \geq 0) \tag{6.9}$$

This acquisition function enables an efficient feasibility search due to its exploitation characteristics [74, 3]. Given that the probability of constraint satisfaction is binary (0 or 1), the algorithm will be able to quickly prune unfeasible regions of the design space and move to other promising regions until it identifies feasible design configurations. This approach will enable a more efficient search over feasible regions later and accurate computation of the acquisition function.

Acquisition function optimization. Given that the constraints are black-box, selecting a valid input might still be challenging even with a well-designed acquisition function. In order to increase the chances of selecting a valid input, we constrain the acquisition function optimization with the predictive mean of the constraints.

$$\mathbf{x} \leftarrow \arg ma\mathbf{x}_{\mathbf{x}\in\mathfrak{X}} \ \alpha(\mathbf{x}) \tag{6.10}$$

$$\mathbf{s.t}(\mu_{c_1} \geq 0, \cdots, \mu_{c_L} \geq 0)$$

A complete description of the MESMOC algorithm is given in Algorithm 6.

Algorithm 6 MESMOC Algorithm

Input: input space $\mathfrak{X}$; K blackbox functions $f_1(\mathbf{x}), \cdots, f_K(\mathbf{x})$; L blackbox constraints $C_1(\mathbf{x}), \cdots, C_L(\mathbf{x})$; and maximum no. of iterations T_{max}

Initialize Gaussian process models $\mathcal{GP}_{f_1}, \mathcal{GP}_{f_2}, \cdots, \mathcal{GP}_{f_K}$ and $\mathcal{GP}_{c_1}, \mathcal{GP}_{c_2}, \cdots, \mathcal{GP}_{c_L}$ by evaluating at N_0 initial points
for each iteration $t = N_0 + 1$ to T_{max} **do**
  if feasible design parameters $\mathbf{x}_{feasible} \notin \mathcal{D}$ **then**
    Select design parameters $\mathbf{x}_t \leftarrow \arg ma\mathbf{x}_{\mathbf{x}\in\mathcal{X}}\ \alpha_{prob}(\mathbf{x})$ # *eq. 6.9*
  else
    Select $\mathbf{x}_t \leftarrow \arg ma\mathbf{x}_{\mathbf{x}\in\mathfrak{X}}\ \alpha_t(\mathbf{x})$
      s.t $(\mu_{c_1} \geq 0, \cdots, \mu_{c_L} \geq 0)$
    $\alpha_t(.)$ is computed as:
      for each sample $s \in 1, \cdots, S$:
        Sample $\tilde{f}_j \sim \mathcal{GP}_{f_j}, \quad \forall j \in \{1, \cdots, K\}$
        Sample $\tilde{C}_i \sim \mathcal{GP}_{c_i}, \quad \forall i \in \{1, \cdots, L\}$
        // Solve *cheap* MOO over $(\tilde{f}_1, \cdots, \tilde{f}_K)$ constrained by $(\tilde{C}_1, \cdots, \tilde{C}_L)$
        $\mathcal{Y}_s^* \leftarrow \arg ma\mathbf{x}_{x\in\mathcal{X}}(\tilde{f}_1, \cdots, \tilde{f}_K)$
          s.t $(\tilde{C}_1 \geq 0, \cdots, \tilde{C}_L \geq 0)$
      Compute $\alpha_t(.)$ based on the S samples of $\mathcal{Y}_s^*$ as given in equation (6.8)
  end if
  Evaluate $\mathbf{x}_t$; $\mathbf{y}_t \leftarrow (f_1(\mathbf{x}_t), \cdots, f_K(\mathbf{x}_t), C_1(\mathbf{x}_t), \cdots, C_L(\mathbf{x}_t))$
  Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_t, \mathbf{y}_t)\}$
  Update models $\mathcal{GP}_{f_1}, \mathcal{GP}_{f_2}, \cdots, \mathcal{GP}_{f_K}$ and $\mathcal{GP}_{c_1}, \mathcal{GP}_{c_2}, \cdots, \mathcal{GP}_{c_L}$
  $t \leftarrow t + 1$
end for
return Pareto front of $f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_K(\mathbf{x})$ based on $\mathcal{D}$

6.3.2 Experiments and Results of MESMOC

Experimental Setup: In this section, we compare MESMOC with PESMOC [75], the state-of-the-art BO algorithm for solving constrained MO problems and MESMOC+ [71], the concurrent approach which also relies on the same principle of output space entropy search. Due to lack of BO approaches for constrained MO setting, we also compare to known genetic algorithms: NSGA-II [47] and MOEAD

[198]. However, they require large number of function evaluations to converge which is not practical for the optimization of expensive functions. We employ a GP based statistical model with squared exponential (SE) kernel in all our experiments. The hyper-parameters are estimated after every five function evaluations (iterations). We initialize the GP models for all functions by sampling the initial points at random. We employ the code for PESMOC and MESMOC+ from the BO library Spearmint[1]. We employ NSGA-II and MOEAD from the Platypus library[2]. Our code for MESMOC is available at the following Github repository [3]. We provide additional details about the algorithms parameters, libraries, and computational resources in the Appendix C.2.2.

Real-world Engineering Design Problems

Below we provide the details of the two real-world problems and associated optimization tasks that are employed for our experimental evaluation.

1) Electrified Aviation Power System Design. We consider optimizing the design of electrified aviation power system of unmanned aerial vehicle (UAV) via a time-based static simulation. The UAV system architecture consists of a central Li-ion battery pack, hex-bridge DC-AC inverters, PMSM motors, and necessary wiring [24, 27]. Each candidate input consists of a set of 5 (d=5) variable design parameters

[1]github.com/EduardoGarrido90/Spearmint
[2]platypus.readthedocs.io/en/latest/getting-started.html#installing-platypus
[3]github.com/belakaria/MESMOC

such as the battery pack configuration (battery cells in series, battery cells in parallel) and motor size (number of motors, motor stator winding length, motor stator winding turns). We minimize two objective functions: mass and total energy. This problem has 5 black-box constraints:

$$C_0 : \text{Maximum final depth of discharge} \leq 75\%$$

$$C_1 : \text{Minimum cell voltage} \geq 3V$$

$$C_2 : \text{Maximum motor temperature} \leq 125°C$$

$$C_3 : \text{Maximum inverter temperature} \leq 120°C$$

$$C_5 : \text{ Maximum modulation index} \leq 1.3$$

2) Analog Circuit Optimization Domain. We consider optimizing the design of a multi-output switched-capacitor voltage regulator via Cadence circuit simulator that imitates the real hardware [25]. This circuit relies on a dynamic frequency switching clock. Each candidate circuit design is defined by 33 input variables (d=33). The first 24 variables are the width, length, and unit of the eight capacitors of the circuit $W_i, L_i, M_i \ \forall i \in 1 \cdots 8$. The remaining input variables are four output voltage references $V_{ref_i} \ \forall i \in 1 \cdots 4$ and four resistances $R_i \ \forall i \in 1 \cdots 4$ and a switching frequency f. We optimize nine objectives: maximize efficiency Eff, maximize four output voltages $V_{o_1} \cdots V_{o_4}$, and minimize four output ripples $OR_1 \cdots OR_4$. Our problem has a

total of nine constraints. Since some of the constraints have upper bounds and lower bounds, they are defined in the problem by 14 different constraints:

$$C_0 : Cp_{total} \simeq 20nF \; with \; Cp_{total} = \sum_{i=1}^{8} (1.955 W_i L_i + 0.54(W_i + L_i)) M_i$$

$$C_1 \; to \; C_4 : V_{o_i} \geq V_{ref_i} \; \forall \in 1 \cdots 4$$

$$C_5 \; to \; C_8 \; : \quad OR_{lb} \leq OR_i \; \forall i \; \in 1 \cdots 4$$

$$C_9 \; to \; C_{12} \; : \quad OR_i \leq OR_{ub} \; \forall i \; \in 1 \cdots 4$$

$$C_{13} : Eff \leq 100\%$$

where OR_{lb} and OR_{ub} are the predefined lower-bound and upper-bound of OR_i respectively. Cp_{total} is the total capacitance of the circuit.

Results and Discussion We evaluate the performance of our algorithm and the baselines using the Pareto hypervolume (PHV) metric. PHV is a commonly employed metric to measure the quality of a given Pareto front [202]. We provide a detailed definition of the metrics in section 2.3. Figure 6.1 shows that MESMOC outperforms existing baselines. It recovers a better Pareto front with a significant gain in the number of function evaluations. Both of these experiments are motivated by real-world engineering applications where further analysis of the designs in the Pareto front is crucial.

1) Electrified Aviation Power System Design. In this setting, the input space

is discrete with 250,000 combinations of design parameters. Out of the entire design space, only 9% of design combinations passed all the constraints and only five points are in the optimal Pareto front. From a domain expert perspective, satisfying all the constraints is critical. Hence, the results reported for the hypervolume include only points that satisfy all the constraints. Despite the hardness of the problem, 90% (180 out of 200 inputs) of the designs selected by MESMOC satisfy all the constraints while for MESMOC+, PESMOC, MOEAD, and NSGA-II, this was 49% (98 out of 200), 1.5% (3 out of 200 inputs), 9.5% (19 out of 200 inputs), and 7.5% (15 out of 200 inputs) respectively. MESMOC was not able to recover all the five points from the optimal Pareto front. However, it was able to closely approximate the optimal Pareto front and recover better designs than the baselines.

2) Analog Circuit Design Optimization. In this setting, the input space is continuous, consequently there is an infinite number of candidate designs. From a domain expert perspective, satisfying all the constraints is not critical and is impossible to achieve. The main goal is to satisfy most of the constraints (and getting close to satisfying the threshold for violated constraints) while reaching the best possible objective values. Therefore, the results reported for the hypervolume include all the evaluated points. In this experiment, the efficiency of circuit is the most important objective function. The table in Figure 2 shows the optimized circuit parameters from different algorithms.

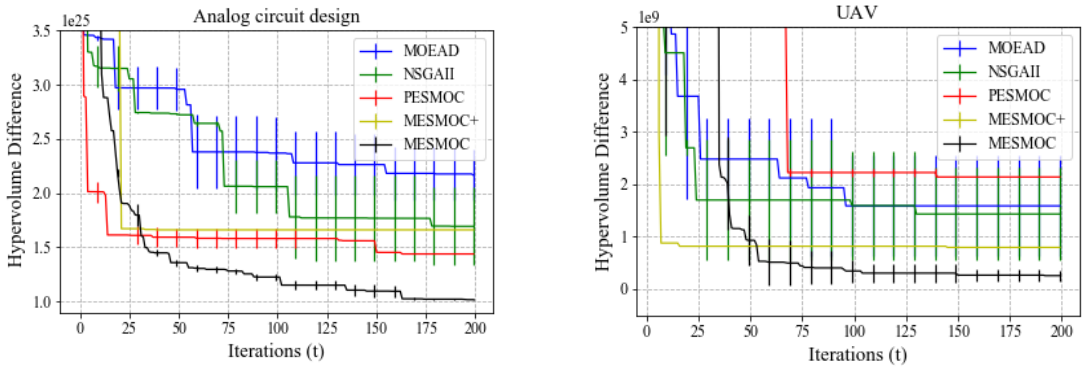

Figure 6.1: Results of different constrained multi-objective algorithms including MESMOC. The PHV metric is shown as a function of the number function evaluations.

SPECS	NSGA-II		PESMOC		MESMOC	
V_{ref1}(V)	0.6	0.5	0.52	0.53	0.63	0.52
V_{ref2}(V)	0.55	0.62	0.55	0.61	0.51	0.53
V_{ref3}(V)	1.06	1.06	1.07	1.12	1.05	1.13
V_{ref4}(V)	1.07	1.09	1.09	1.06	1.05	1.06
V_{o1}(mV)	699.6	713.1	677.10	760.60	**678.40**	**551.62**
V_{o2} (mV)	700.4	712.2	690.70	725.70	**520.61**	**632.80**
V_{o3} (V)	1.10	1.06	*1.08*	1.15	**1.12**	**1.16**
V_{o4} (V)	1.09	1.09	*1.08*	0.99	**1.14**	**1.08**
Eff (%)	73.26	71.85	**76.20**	**74.82**	***88.81***	***88.53***

Figure 6.2: Comparison table of optimized circuit parameters obtained from different algorithms (designs are selected from the Pareto set prioritized by efficiency)

All algorithms can generate design parameters for the circuit that meets the voltage reference requirements. The optimized circuit using MESMOC can achieve the highest conversion efficiency of 88.81% (12.61% improvement when compared with PESMOC with fixed frequency optimization and 17.86% improvement when compared with NSGA-II) with similar output ripples. The circuit with optimized pa-

rameters can generate the target output voltages within the range of 0.52V to 0.76V (1/3x ratio) and 0.99V to 1.17V (2/3x ratio) under the loads varying from 14 Ohms to 1697 Ohms.

6.4 Uncertainty-aware Search Framework for Constrained MOO

In this section, we propose an extension of USeMO framework to handle constraints. Constraints in real-world problems usually take one of the following three forms: [**Type 1**] C_i is a function of the input x and can be expressed declaratively (i.e., white-box constraint); [**Type 2**] C_i is black-box constraint; and [**Type 3**] C_i is a function of both the input x and a combination of the black-box objective functions. The proposed algorithm USeMOC is capable of handling the three types of constraints. In the following, we provide an overview of USeMOC followed by the details of its two main components and an experimental evaluation of an analog circuit design problem.

6.4.1 Overview of USeMOC Framework

USeMOC is an iterative algorithm that involves four key steps. First, We build statistical models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ for each of the K objective functions from the training data in the form of past function evaluations. Second, we select a set of promising candidate inputs $\mathcal{X}_p$ by solving a constrained cheap MO optimization problem defined using the statistical models. Specifically, multiple objectives of the

cheap MO problem correspond to $\text{AF}(\mathcal{GP}_1, \mathbf{x}), \text{AF}(\mathcal{GP}_2, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x})$ respectively. Any standard acquisition function AF from single-objective BO (e.g., EI, LCB) can be used for this purpose. Additionally, if the constraint is black-box, it is modeled by a GP $\mathcal{GP}_{c_i}$ and its predictive mean $\mu_{c_i}(\mathbf{x})$ is used instead for reasoning. If the value of a constraint C_i depends on the evaluation of one (or more) objectives $f_i(\mathbf{x})$, we employ the predictive mean $\mu_i(\mathbf{x})$. The Pareto set $\mathcal{X}_p$ corresponds to the inputs with different trade-offs in the utility space for K unknown functions satisfying the constraints. Third, we select the best candidate input $\mathbf{x}_s \in \mathcal{X}_p$ from the Pareto set that maximizes some form of uncertainty measure for evaluation. Fourth, the selected input $\mathbf{x}_s$ is used for evaluation to get the corresponding function evaluations: $y_1{=}f_1(\mathbf{x}_s)$, $y_2{=}f_2(\mathbf{x}_s), \cdots, y_K{=}f_K(\mathbf{x}_s)$. Algorithm 7 provides the algorithmic pseudocode for USeMOC.

Advantages. USeMOC has many advantages over prior methods. **1)** Provides flexibility to plug-in any acquisition function for single-objective BO. This allows us to leverage existing acquisition functions including EI and LCB. **2)** Computationally-efficient to solve constrained MO problems with many objectives. **3)** Can handle all three types of constraints (i.e., type 1, type 2, and type 3).

6.4.2 Key Algorithmic Components of USeMOC

The two main algorithmic components of USeMOC framework are: selecting the most promising candidate inputs by solving a cheap constrained MO problem and

Algorithm 7 USeMOC Framework

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

1: Initialize training data of function evaluations $\mathcal{D}$
2: Initialize statistical models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K, , \mathcal{GP}_{c_0}, \cdots, \mathcal{GP}_{c_m}$ from $\mathcal{D}$
3: **for** each iteration t=1 to T_{max} **do**
4: // Solve constrained cheap MO problem with objectives $\text{AF}(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x})$ to get candidate inputs
5: $\mathcal{X}_p \leftarrow \arg min_{x \in \mathcal{X}} (\text{AF}(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x}))$ **s.t** $\mu_{c_0}, \cdots, \mu_{c_m}, C_{m+1} \cdots, C_L$
6: // Pick the candidate input with maximum uncertainty
7: Select $\mathbf{x}_{t+1} \leftarrow \arg max_{x \in \mathcal{X}_p} U_{\beta_t}(\mathbf{x})$
8: Evaluate $\mathbf{x}_{t+1}$: $\mathbf{y}_{t+1} \leftarrow (f_1(\mathbf{x}_{t+1}), \cdots, f_K(\mathbf{x}_{t+1}))$, $C_{t+1} \leftarrow (C_0(\mathbf{x}_{t+1}), \cdots, C_L(\mathbf{x}_{t+1}))$
9: Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_{t+1}, \mathbf{y}_{t+1}, C_{t+1})\}$
10: Update models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K, \mathcal{GP}_{c_0}, \cdots, \mathcal{GP}_{c_m}$ using $\mathcal{D}$
11: $t \leftarrow t + 1$
12: **end for**
13: **return** Pareto set and Pareto front of $\mathcal{D}$

picking the best candidate via uncertainty maximization. We describe their details below.

Selection of promising candidates. We employ the statistical models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$ towards the goal of selecting promising candidate inputs as follows. Given an acquisition function AF (e.g., EI), we construct a constrained cheap multi-objective optimization problem with objectives $\text{AF}(\mathcal{GP}_1, \mathbf{x}), \cdots, \text{AF}(\mathcal{GP}_K, \mathbf{x})$, where $\mathcal{GP}_i$ is the statistical model for the unknown function f_i and $C_1, C_2, \cdots, C_L$ are the problem constraints. Generally, there are three types of constraints:

- [**Type 1**] C_i is a function of the input x and can be expressed declaratively (white-box constraint). Such a constraint will take the same form as in the

cheap MO problem.

- **[Type 2]** C_i is black-box constraint. In this case, it will be modeled by an independent GP $\mathcal{GP}_{c_i}$ and predictive mean of its model μ_{c_i} can be used in the optimization process.

- **[Type 3]** C_i is a function of both the input x and a combination of the black-box objective functions. Since verifying constraint C_i depends on the evaluation of one (or more) objectives f_j, we employ the predictive mean(s) $\mu_i(\mathbf{x})$ instead.

Without loss of generality and for the sake of notation in Algorithm 7, we suppose that the first m constraints with $0 \leq m \leq L$ are black-box.

Since we present the framework as minimization, all AFs will be minimized. The Pareto set $\mathcal{X}_p$ obtained by solving this cheap constrained MO problem represents the most promising candidate inputs for evaluation.

$$\mathcal{X}_p \leftarrow \arg min_{x \in \mathcal{X}} \left(\textsc{Af}(\mathcal{GP}_1, \mathbf{x}), \cdots, \textsc{Af}(\mathcal{GP}_K, \mathbf{x})\right) \tag{6.11}$$
$$\textbf{s.t } \mu_{c_0}, \cdots, \mu_{c_m}, C_{m+1} \cdots, C_L$$

Each acquisition function $\textsc{Af}(\mathcal{GP}_i, \mathbf{x})$ is dependent on the corresponding surrogate model $\mathcal{GP}_i$ of the unknown objective function f_i. Hence, each acquisition function will carry the information of its associated objective function. As iterations progress, using more training data, the models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ will better mimic the true

objective functions $f_1, f_2, \cdots, f_K$. Therefore, the Pareto set of the acquisition function space (solution of Equation 6.11) becomes closer to the Pareto set of the true functions $\mathcal{X}^*$ with increasing iterations.

Intuitively, the acquisition function $\text{AF}(\mathcal{GP}_i, \mathbf{x})$ corresponding to unknown objective function f_i tells us the utility of a point x for optimizing f_i. The input minimizing $\text{AF}(\mathcal{GP}_i, \mathbf{x})$ has the highest utility for f_i, but may have a lower utility for a different function f_j $(j \neq i)$. The utility of inputs for evaluation of f_j is captured by its own acquisition function $\text{AF}(\mathcal{GP}_j, \mathbf{x})$. Therefore, there is a trade-off in the utility space for all K different functions. The Pareto set $\mathcal{X}_p$ obtained by simultaneously optimizing acquisition functions for all K unknown functions will capture this utility trade-off. As a result, each input $\mathbf{x} \in \mathcal{X}_p$ is a promising candidate for evaluation towards the goal of solving MOO problem. USeMOC employs the same acquisition function for all K objectives. The main reason is to give equivalent evaluation for all functions in the Pareto front (PF) at each iteration. If we use different AFs for different objectives, the sampling procedure would be different. Additionally, the values of various AFs can have considerably different ranges. Thus, this can result in an unbalanced trade-off between functions in the cheap PF leading to the same unbalance in our final PF.

Cheap constrained MO solver. We employ the constrained version of the popular NSGA-II algorithm ([47, 48]) to solve the MO problem with cheap objective functions

and cheap constraints noting that any other algorithm can be used.

Picking the best candidate input. We need to select the best input from the Pareto set $\mathcal{X}_p$ obtained by solving the cheap MO problem. All inputs in $\mathcal{X}_p$ are promising in the sense that they represent the trade-offs in the utility space corresponding to different unknown functions. It is critical to select the input that will guide the overall search towards the goal of quickly approximating the true Pareto set $\mathcal{X}^*$. We employ a uncertainty measure defined in terms of the statistical models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ to select the most promising candidate input for evaluation. In single-objective optimization case, the learned model's uncertainty for an input can be defined in terms of the variance of the statistical model. For multi-objective optimization case, we define the uncertainty measure as the volume of the uncertainty hyper-rectangle.

$$U_{\beta_t}(\mathbf{x}) = VOL(\{(LCB(\mathcal{GP}_i, \mathbf{x}), UCB(\mathcal{GP}_i, \mathbf{x})\}_{i=1}^{k}) \tag{6.12}$$

where $\text{LCB}(\mathcal{GP}_i, \mathbf{x})$ and $\text{UCB}(\mathcal{GP}_i, \mathbf{x})$ represent the lower confidence bound and upper confidence bound of the statistical model $\mathcal{GP}_i$ for an input x as defined in equations 2.4 and 2.5; and β_t is the parameter value to trade-off exploitation and exploration at iteration t. We employ the adaptive rate recommended by ([174]) to set the β_t value depending on the iteration number t. We measure the uncertainty volume measure for all inputs $x \in \mathcal{X}_p$ and select the input with maximum uncertainty for function

evaluation.

$$\mathbf{x}_{t+1} = \arg ma\mathbf{x}_{x \in \mathcal{X}_p} U_{\beta_t}(\mathbf{x}) \tag{6.13}$$

6.4.3 Experiments and Results of USeMOC

In this section, we discuss the experimental evaluation of USeMOC and prior methods on a real-world analog circuit design optimization task. The code for USeMOC is available in github repository: github.com/belakaria/USEMOC.

Analog circuit optimization domain. We consider optimizing the design of a multi-output switched-capacitor voltage regulator via Cadence circuit simulator that imitates the real hardware [25]. Each candidate circuit design is defined by 32 input variables (d=32). The first 24 variables are the width, length, and unit of the eight capacitors of the circuit $W_i, L_i, M_i \; \forall i \in 1 \cdots 8$. The remaining input variables are four output voltage references $V_{ref_i} \; \forall i \in 1 \cdots 4$ and four resistances $R_i \; \forall i \in 1 \cdots 4$. We optimize nine objectives: maximize efficiency Eff, maximize four output voltages $V_{o_1} \cdots V_{o_4}$, and minimize four output ripples $OR_1 \cdots OR_4$. Our problem has a total

of fourteen constraints:

$$C_0 : Cp_{total} \simeq 20nF \; with \; Cp_{total} = \sum_{i=1}^{8} (1.955 W_i L_i + 0.54(W_i + L_i)) M_i$$

$$C_1 \; to \; C_4 : V_{o_i} \geq V_{ref_i} \; \forall \in 1 \cdots 4$$

$$C_5 \; to \; C_8 \; : \quad OR_{lb} \leq OR_i \; \forall i \; \in 1 \cdots 4$$

$$C_9 \; to \; C_{12} \; : \quad OR_i \leq OR_{ub} \; \forall i \; \in 1 \cdots 4$$

$$C_{13} : Eff \leq 100\%$$

where OR_{lb} and OR_{ub} are the predefined lower-bound and upper-bound of OR_i respectively. Cp_{total} is the total capacitance of the circuit. In this problem, C_0 is a white-box constraint (Type 1), while remaining constraints are combinations of black-box objectives (Type 3).

Multi-objective BO algorithms. We compare USeMOC with the existing BO method PESMOC. Due to lack of BO approaches for constrained MO, we compare to known genetic algorithms (NSGA-II and MOEAD). However, they require large number of function evaluations to converge which is not practical for optimization of expensive functions.

Evaluation Metrics. To measure the performance of baselines and USeMOC, we employ two different metrics, one measuring the accuracy of solutions and another one measuring the efficiency in terms of the number of simulations. 1) Pareto hypervolume

(PHV) is a commonly employed metric to measure the quality of a given Pareto front [202]. After each iteration t (or number of simulations), we measure the PHV for all algorithms. We evaluate all algorithms for 100 circuit simulations. 2) Percentage gain in simulations is the fraction of simulations our BO algorithm USeMOC is saving to reach the PHV accuracy of solutions at the convergence point of baseline algorithm employed for comparison.

Results and Discussion. We evaluate the performance of USeMOC with two different acquisition functions (EI and LCB) to show the generality and robustness of our approach. We also provide results for the percentage gain in simulations achieved by USeMOC when compared to each baseline method. Figure 6.3 shows the PHV metric achieved by different multi-objective methods including USeMOC as a function of the number of circuit simulations. We make the following observations: 1) USeMOC with both EI and LCB acquisition functions perform significantly better than all baseline methods. 2) USeMOC is able to uncover a better Pareto solutions than baselines using significantly less number of circuit simulations. This result shows the efficiency of our approach. Table 6.1 shows that USeMOC achieves percentage gain in simulations w.r.t baseline methods ranging from 90 to 93%.

The analog circuit is implemented in the industry-provided process design kit (PDK) and shows better efficiency and output ripples. Since MOEAD is the best performing baseline optimization method, we use it for the rest of the experimental analysis. Ta-

ble 6.4 illustrates the simulated performance of circuit optimized by MOEAD (best baseline) and USeMOC-EI (best variant of our proposed algorithm). Results of both algorithms meet the voltage reference and ripple requirements (100mV) . Compared to MOEAD, the optimized circuit with USeMOC-EI can achieve a higher conversion efficiency of 76.2 % (5.25 % higher than MOEAD, highlighted in red color) with similar output ripples. The optimized circuits can generate the target output voltages within the range of 0.52V-0.61V (1/3x ratio) and 1.07V-1.12V (2/3x ratio) under the loads varying from 14 Ohms to 1697 Ohms (highlighted in black and green colors). Thus, the capability of USeMOC to optimize the parameters of circuit under different output voltage/current conditions is clearly validated. Future work includes improving USeMOC to solve more challenging problems [24, 27].

Method	MOEAD	NSGA-II	PESMOC
Gain in simulations	90.7%	93.3%	92.5%

Table 6.1: Percentage gain in simulations achieved by our USeMOC compared to baselines.

6.5 Summary

We introduced two principled approaches to solving multi-objective Bayesian optimization problems with constraints. The first approach is named MESMOC where the key idea is to employ an output space entropy-based acquisition function to efficiently select valid inputs for evaluation, to enforce constraints during the acquisition

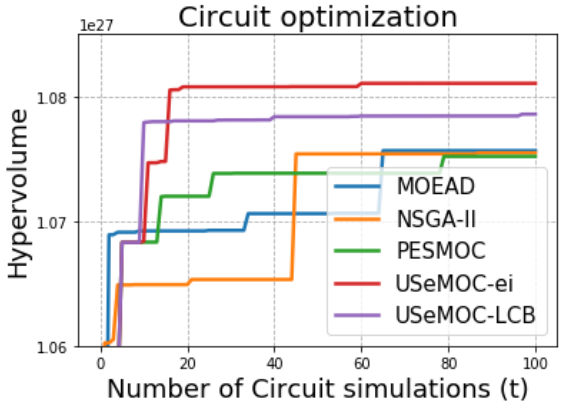

Figure 6.3: Results of different multi-objective algorithms including USeMOC. The hypervolume metric is shown as a function of the number of circuit design simulations.

SPECS	MOEAD			USEMOC-EI		
V_{ref1}(V)	0.53	0.6	0.6	*0.52*	0.53	0.56
V_{ref2}(V)	0.55	0.51	0.59	*0.55*	0.61	0.57
V_{ref3}(V)	1.14	1.06	1.07	*1.07*	1.12	1.11
V_{ref4}(V)	1.22	1.16	1.14	*1.09*	1.06	1.1
R_1(Ohm)	144	1668	1012	**207**	**1198**	**619**
R_2(Ohm)	758	620	559	**306**	**1697**	**89**
R_3(Ohm)	247	66	10	**67**	**1379**	**70**
R_4(Ohm)	222	144	1830	**42**	**14**	**301**
V_{o1}(mV)	551.5	702.18	775.01	**677.10**	**760.60**	**656.9**
V_{o2} (mV)	612.2	671.01	912.22	**690.70**	**725.70**	**569.4**
V_{o3} (V)	1.17	1.09	867.96	***1.08***	**1.15**	**1.14**
V_{o4} (V)	1.117	1.12	1.12	***1.08***	0.99	**1.13**
OR1(mV)	52.69	11.39	0.496	2.50	4.20	11.30
OR2(mV)	9.32	4.52	0.984	3.50	5.00	15.90
OR3(mV)	64.96	96.57	28.199	58.9	87.1	75.7
OR4(mV)	4.75	4.80	1.05	80.7	25.1	74.3
Eff (%)	70.95	65.94	64.61	***76.2003***	***74.82***	***73.71***

Figure 6.4: Comparison table of optimized circuit parameters obtained by MOEAD and USeMOC-EI (designs are selected from the Pareto set prioritized by efficiency)

function optimization, and to accelerate the discovery of initial valid inputs by maximizing the probability of constraint satisfaction. The second approach is named USeMOC where the key idea is to solve a cheap constrained multi-objective problem

over acquisition functions to find promising candidates and then select the input with the highest uncertainty for evaluation. We provided real-world experiments on analog circuit design and unmanned aviation power systems design optimization problems and demonstrated the efficacy of our proposed algorithms.

CHAPTER SEVEN

DISCRETE FIDELITY MULTI-OBJECTIVE BO

In this Chapter, we study the novel problem of blackbox optimization of multiple objectives via multi-fidelity function evaluations that vary in the amount of resources consumed and their accuracy. The overall goal is to approximate the true Pareto set of solutions by minimizing the resources consumed for function evaluations. For example, in power system design optimization, we need to find designs that trade-off cost, size, efficiency, and thermal tolerance using multi-fidelity simulators for design evaluations.

We propose a novel approach referred as **M***ulti-***F***idelity* **O***utput* **S***pace* **E***ntropy Search for* **M***ulti-objective* **O***ptimization (MF-OSEMO)* to solve this problem. The key idea is to select the sequence of candidate input and fidelity-vector pairs that maximize the information gain about the true Pareto front per unit resource cost. We provide two qualitatively different approximations to efficiently compute the entropy, which is a key step for MF-OSEMO. These approximations make different trade-offs in terms of accuracy and computational-efficiency: one has a closed-form expression and another employs numerical integration. We evaluate both approximations on several synthetic and real-world benchmark problems.

7.1 Problem Setup

Discrete Multi-Fidelity MOO Problem. This is a general version of the MOO problem, where we have access to M_j fidelities for each function f_j that vary in the amount of resources consumed and the accuracy of evaluation. The evaluation of an input $\mathbf{x} \in \mathfrak{X}$ with fidelity vector $\mathbf{m} = [m_1, m_2, \cdots, m_K]$ produces an evaluation vector of K values denoted by $\mathbf{y^m} \equiv [y_1^{(m_1)}, \cdots, y_K^{(m_K)}]$, where $y_j^{(m_j)} = f_j^{(m_j)}(x)$ for all $j \in \{1, 2, \cdots, K\}$. Let $\lambda_j^{(m_j)}$ be the cost of evaluating i^{th} function f_j at $m_j \in [M_j]$ fidelity, where m_j=M_j corresponds to the highest fidelity for f_j. Our goal is to approximate the optimal Pareto set $\mathcal{X}^*$ over the highest fidelities functions while minimizing the overall cost of function evaluations (experiments). For example, in power system design optimization, we need to find designs that trade-off cost, size, efficiency, and thermal tolerance using multi-fidelity simulators for design evaluations. Table 7.1 contains all the mathematical notations used in this section (MF-OSEMO).

Cost of Function Evaluations. The total normalized evaluation cost is $\lambda^{(\mathbf{m})} \equiv \sum_{j=1}^{K} \lambda_j^{(m_j)}/\lambda_j^{(M_j)}$. We normalize the total cost since the cost units can be different for different objectives (e.g. cost unit for f_1 is computation time while cost unit for f_2 could be memory space size). If the cost is known, it can be directly injected in the latter expression. However, in some real-world settings, the cost of a function evaluation can only be known after the function evaluation. For example, in hyper-parameter tuning of a neural network, the cost of the experiment is defined by the

Notation	Definition
$f_1^{(m_1)}, f_2^{(m_1)}, \cdots, f_K^{(m_K)}$	functions the m_j fidelity of the true objective functions
$\tilde{f}_j^{(m_j)}$	function sampled from jth Gaussian process model at m_jth fidelity
$M_1, M_2, \cdots, M_K$	no. of fidelities for each function
$\mathbf{m} = [m_1, m_2, \cdots, m_K]$	fidelity vector where each fidelity $m_j \in [M_j]$
$y_j^{m_j}$	jth function f_j evaluated at m_jth fidelity where $m_j \in [M_j]$
$\mathbf{y}^{\mathbf{m}}$	output vector equivalent to $[y_1^{(m_1)}, \cdots, y_K^{(m_K)}]$
$\lambda_j^{(m_j)}$	cost of evaluating jth function f_j at m_jth fidelity
$\lambda^{(\mathbf{m})}$	total normalized cost $\lambda^{(\mathbf{m})} \equiv \sum_{j=1}^{K} \left(\lambda_j^{(m_j)} / \lambda_j^{(M_j)}\right)$
$\mathcal{Y}^*$	true pareto front of the objective functions $[f_1, f_2, \cdots, f_K]$ (the highest fidelities)
$\mathcal{Y}_s^*$	Pareto front of the sampled highest fidelities $[\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f}_K]$

Table 7.1: Table describing additional mathematical notations used in this section (MF-OSEMO).

training and inference time. However, we cannot know the exact needed time until after the experiment is finalized. In this case, the cost can be modeled by an independent Gaussian process. The predictive mean can be used during the optimization. Our goal is to approximate $\mathcal{X}^*$ by minimizing the overall cost of function evaluations. Let $D = \{(\mathbf{x}_i, \mathbf{y}_i^{(\mathbf{m})})\}_{i=1}^{t-1}$ be the training data from past $t-1$ function evaluations, where $\mathbf{x}_i \in \mathfrak{X}$ is an input and $\mathbf{y}_i^{(\mathbf{m})} = [y_1^{(m_1)}, \cdots, y_K^{(m_K)}]$ is the output vector resulting from evaluating functions $f_1^{(m_1)}, f_2^{(m_2)}, \cdots, f_K^{(m_k)}$ at $\mathbf{x}_i$. Gaussian processes (GPs) are known to be effective surrogate models in prior work on single and multi-objective BO [174, 83]. We learn K surrogate models $\mathcal{GP}_1, \mathcal{GP}_2, \cdots, \mathcal{GP}_K$ from $\mathcal{D}$, where each $\mathcal{GP}_j$ corresponds to the jth function f_j. In our setting, each function has multiple fidelities. So one ideal property desired for the surrogate model of a single function is to take into account all the fidelities in a single model. Multi-fidelity GPs (MF-GP) are capable of modeling functions with multiple fidelities in a single model. Hence,

each of our surrogate model $\mathcal{GP}_j$ is a multi-fidelity GP.

Specifically, we use the MF-GP model as proposed in [112, 179]. We describe the complete details of the MF-GP model below for the sake of completeness. One key thing to note about MF-GP model is that the kernel function ($\kappa((\mathbf{x_i}, m_i), (\mathbf{x_j}, m_j))$) is dependent on both the input and the fidelity. For a given input $\mathbf{x}$, the MF-GP model returns a *vector* (one for each fidelity) of predictive mean, a *vector* of predictive variance, and a matrix of predictive covariance. The MF-GP model has two advantages. The first is that all fidelities are integrated into one single GP. The second is that difference among fidelities are adaptively estimated without any additional feature representation for fidelities. It should be noted that we employ an independent multi-fidelity GP for each function.

We describe full details of a MF-GP model for one objective function f_j (without loss of generality) below:

Let $y_j^{(1)}(\mathbf{x}), \ldots, y_j^{(M_j)}(\mathbf{x})$ represent the values obtained by evaluating the function f_j at its 1st, 2nd, $\ldots, M_j$th fidelity respectively. In a MF-GP model, each fidelity is represented by a Gaussian process and the observation is modeled as

$$y_j^{(m_j)}(\mathbf{x}) = f_j^{(m_j)}(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}}).$$

Let $f_j^{(1)} \sim GP(0, \kappa_1(\mathbf{x}, \mathbf{x}'))$ be a Gaussian process for the 1st fidelity i.e. $m_j = 1$, where $\kappa_1 : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$ is a suitable kernel. The output for successively fidelities

$m_j = 2, \ldots, M_j$ is recursively defined as

$$f_j^{(m_j)}(\mathbf{x}) = f_j^{(m_j-1)}(\mathbf{x}) + f_{j_e}^{(m_j-1)}(\mathbf{x}), \tag{7.1}$$

where, $f_{j_e}^{(m_j-1)} \sim GP(0, \kappa_e(\mathbf{x}, \mathbf{x}'))$ with $\kappa_e : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$. It is assumed that $f_{j_e}^{(m_j-1)}$ is conditionally independent from all fidelities lower than m_j. As a result, *the kernel for a pair of points evaluated at the same fidelity* becomes:

$$\kappa_{m_j}(\mathbf{x}, \mathbf{x}') \equiv \kappa_1(\mathbf{x}, \mathbf{x}') + (m_j - 1)\kappa_e(\mathbf{x}, \mathbf{x}') \tag{7.2}$$

and as a result, the output for m_jth fidelity is also modeled as a Gaussian process:

$$f_j^{(m_j)} \sim GP(0, \kappa_{m_j}(\mathbf{x}, \mathbf{x}')).$$

The kernel function for a pair of inputs evaluated at different fidelities m_j and m_j' is:

$$\kappa((\mathbf{x}, m_j), (\mathbf{x}', m_j')) = \text{cov}\left(f_j^{(m_j)}(\mathbf{x}), f_j^{(m_j')}(\mathbf{x}')\right) = \kappa_{m_j}(\mathbf{x}, \mathbf{x}')$$

where $m_j \leq m_j'$ and cov represents covariance. Using a kernel matrix $K \in \mathcal{R}^{n \times n}$ in which the p, q element is defined by $\kappa((\mathbf{x}, m_j^p), (\mathbf{x}', m_j^q))$, all fidelities $f_j^{(1)}, \ldots, f_j^{(M_j)}$ can be integrated into one common Gaussian process by which predictive mean and

variance are obtained as

$$\mu^{(m_j)}(\mathbf{x}) = K + \sigma^2_{\text{noise}} I^{-1}\mathbf{Y}, \tag{7.3}$$

$$\sigma^{2^{(m_j)}}(\mathbf{x}) = \kappa((\mathbf{x}, m_j), (\mathbf{x}, m_j)) - k_n^{(m_j)}(\mathbf{x})^\top K + \sigma^2_{\text{noise}} I^{-1} k_n^{(m_j)}(\mathbf{x}), \tag{7.4}$$

where $k_n^{(m_j)}(\mathbf{x}) \equiv (\kappa((\mathbf{x}, m_j), (\mathbf{x}_1, m_{j_1})), \ldots, \kappa((\mathbf{x}, m_j), (\mathbf{x}_n, m_{j_n})))^\top$ and $\mathbf{Y} = (y_1^{(m_{j_1})}(\mathbf{x}_1), \ldots, y_n^{(m_{j_n})}(\mathbf{x}_n))^\top$. We also define $\sigma^{2^{(m_j m'_j)}}(\mathbf{x})$ as the predictive covariance between $(\mathbf{x}, m_j)$ and $(\mathbf{x}, m'_j)$, i.e., covariance for identical $\mathbf{x}$ at different fidelities:

$$\sigma^{2(m_j m'_j)}(\mathbf{x}) = \kappa((\mathbf{x}, m_j), (\mathbf{x}, m'_j)) - k_n^{(m_j)}(\mathbf{x})^\top K + \sigma^2_{\text{noise}} I^{-1} k_n^{(m'_j)}(\mathbf{x}). \tag{7.5}$$

7.2 Related Work

Multi-fidelity Single-Objective Optimization. Acquisition functions (AFs) for single-fidelity and single-objective BO are extensively studied [166]. AFs can be broadly classified into two categories. First, *myopic* AFs rely on improving a "local" measure of utility (e.g., expected improvement). Second, *non-myopic* AFs measure the "global" utility of evaluating a candidate input for solving the black-box optimization problem (e.g., predictive entropy search). Canonical examples of myopic acquisition function include expected improvement (EI) and upper-confidence bound (UCB). EI was extended to multi-fidelity setting [89, 156, 125]. The popular GP-

UCB method [174] was also extended to multi-fidelity setting with discrete fidelities [108] and continuous fidelities [109]. Entropy based methods fall under the category of non-myopic AFs. Some examples include entropy search (ES) [82] and predictive entropy search (PES) [84]. Their multi-fidelity extensions include MT-ES [177, 116] and MF-PES [199, 136]. Unfortunately, they inherit the computational difficulties of the original ES and PES. Max-value entropy search (MES) [185] and output space predictive entropy search [87] are recent approaches that rely on the principle of output space entropy (OSE) search. Prior work [185] has shown the advantages of OSE search in terms of compute-time, robustness, and accuracy over input space entropy search methods. Recent work [173] proposed a general approach based on mutual information. Takeno et al. [179] extended MES to multi-fidelity setting and showed its effectiveness over MF-PES. MUMBO [140] extended MES to the continuous-fidelity and multi-task setting.

Multi-fidelity Multi-Objective Optimization. Prior work outside ML literature has considered domain-specific methods that employ single-fidelity multi-objective approaches in the context of multi-fidelity setting by using the lower fidelities *only as an initialization* [119, 8]. Specifically, Ariyarit and et al. [8] employs the single-fidelity algorithm based on expected hypervolume improvement acquisition function and Kontogiannis et al. [119] employs an algorithm that is very similar to SMSego. Also, both these methods model all fidelities with the same GP and assume that higher fidelity

evaluation is a sum of lower-fidelity evaluation and offset error. These are strong assumptions and may not hold in general multi-fidelity settings including the problems from our experimental evaluation. Our proposed MF-OSEMO [21] and iMOCA algorithms (generalized versions of MESMO [17] solve MOO problem in discrete and continuous-fidelity settings respectively using the principle of output space entropy search and leverage some technical ideas from the prior work on single-objective optimization. We are not aware of any prior work on generic discrete/continuous-fidelity algorithms for MOO problems in the BO literature.

7.3 MF-OSEMO Algorithm with Two Approximations

We describe our proposed acquisition function for the multi-fidelity MOO problem setting. We leverage the information-theoretic principle of output space information gain to develop an efficient and robust acquisition function. This method is applicable for the general case, where at each iteration, the objective functions can be evaluated at different fidelities.

The key idea behind the proposed acquisition function is to find the pair $\{\mathbf{x}, \mathbf{m}\}$ that maximizes the information gain about the **Pareto front of the highest fidelities (denoted by $\mathcal{Y}^*$)** per unit cost, where $\{\mathbf{x}, \mathbf{m}\}$ represents a candidate input $\mathbf{x}$ evaluated at a vector of fidelities $\mathbf{m} = [m_1, m_2, \cdots, m_K]$. This idea can be expressed

mathematically as given below:

$$\alpha(\mathbf{x}, \mathbf{m}) = I(\{\mathbf{x}, \mathbf{y}^{(\mathbf{m})}\}, \mathcal{Y}^* \mid D)/\lambda^{(\mathbf{m})} \tag{7.6}$$

where $\lambda^{(\mathbf{m})}$ is the total *normalized* cost of evaluating the objective functions at $\mathbf{m}$ and D is the data collected so far. Figure 7.1 provides an overview of the MF-OSEMO algorithm. The information gain in equation (7.6) is defined as the expected reduction in entropy $H(.)$ of the posterior distribution $P(\mathcal{Y}^* \mid D)$ as a result of evaluating $\mathbf{x}$ at fidelity vector $\mathbf{m}$:

$$I(\{\mathbf{x}, \mathbf{y}^{(\mathbf{m})}\}, \mathcal{Y}^* \mid D) = H(\mathcal{Y}^* \mid D) - \mathbb{E}_{y^{(\mathbf{m})}}[H(\mathcal{Y}^* \mid D \cup \{\mathbf{x}, \mathbf{y}^{(\mathbf{m})}\})] \tag{7.7}$$

$$= H(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}) - \mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}, \mathcal{Y}^*)] \tag{7.8}$$

equation (7.8) follows from equation (7.7) as a result of the symmetric property of information gain. The first term in the r.h.s of equation (7.8) is the entropy of a factorizable K-dimensional Gaussian distribution $P(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}))$ which can be computed in closed form as shown below:

$$H(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}) = \frac{K(1 + \ln(2\pi))}{2} + \sum_{j=1}^{K} \ln(\sigma_j^{(m_j)}(\mathbf{x})) \tag{7.9}$$

where $\sigma_j^{(m_j)}(\mathbf{x})$ is the predictive variance of j^{th} surrogate model GP_j at input $\mathbf{x}$ and

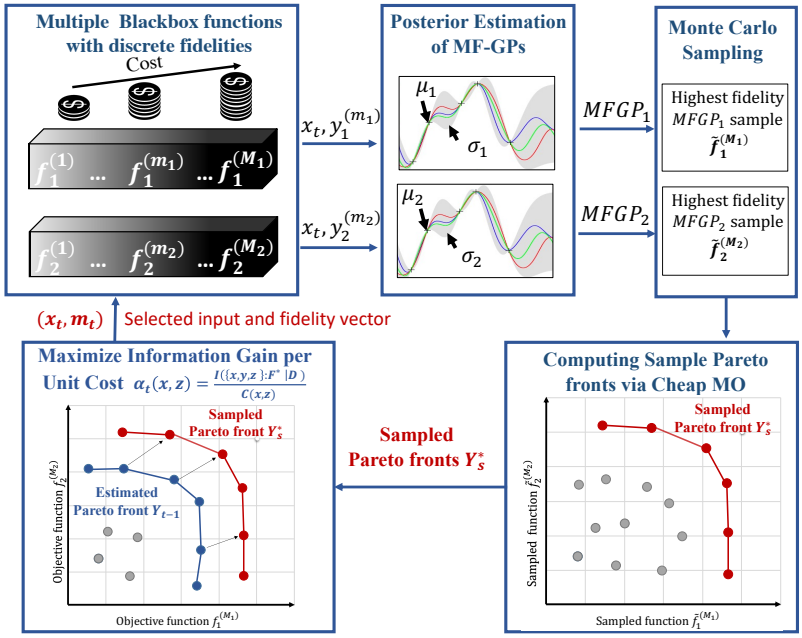

Figure 7.1: Overview of the MF-OSEMO algorithm for two objective functions (K=2). We build multi-fidelity statistical models $\mathcal{MFGP}_1$, $\mathcal{MFGP}_2$ for the two objective functions $f_1(x)$ and $f_2(x)$ with M_1 and M_2 fildelities respectively. First, we sample highest fidelity functions from the statistical models. We compute sample pareto fronts by solving a cheap MO problem over the sampled functions. Second, we select the best candidate input x_t and fidelity vector $m_t = (m_1, m_2)$ that maximizes the information gain per unit cost . Finally, we evaluate the functions for x_t at fidelities m_t to get $(y_1^{(m_1)}, y_2^{(m_2)})$ and update the statistical models using the new training example.

fidelity m_j. The second term in the r.h.s of equation (7.8) is an expectation over the Pareto front of the highest fidelities $\mathcal{Y}^*$. We can approximately compute this term via Monte-Carlo sampling as shown below:

$$\mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}, \mathcal{Y}^*)] \simeq \frac{1}{S}\sum_{s=1}^{S}[H(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}, \mathcal{Y}_s^*)] \tag{7.10}$$

where S is the number of samples and $\mathcal{Y}_s^*$ denote a sample Pareto front obtained over the highest fidelity functions sample from K surrogate models. The main advantages of our acquisition function are: cost efficiency, computational-efficiency, and robustness to the number of samples. Our experiments demonstrate these advantages over state-of-the-art single-fidelity AFs for multi-objective optimization.

There are two key algorithmic steps to compute equation (7.10). The first is computing Pareto front samples $\mathcal{Y}_s^*$; and the second is computing the entropy with respect to a given Pareto front sample $\mathcal{Y}_s^*$. We provide solutions for these two steps below.

1) Computing Pareto Front Samples via Cheap Multi-Objective optimization. To compute a Pareto front sample $\mathcal{Y}_s^*$, we first sample highest fidelity functions from the posterior MF-GP models via random Fourier features [84, 159] and then solve a cheap multi-objective optimization over the K sampled high fidelity functions. It is important to note that we are sampling only the **highest fidelity function** from each MF-GP surrogate model.

Sampling functions from the posterior of MF-GP model. Similar to prior work [84, 83, 185], we employ random Fourier features based sampling procedure. We approximate each GP prior of the highest fidelity as $\tilde{f}^{(M)} = \phi(\mathbf{x})^T\theta$, where $\theta \sim N(0, \mathbf{I})$. The key idea behind random Fourier features is to construct each function sample $\tilde{f}^{(M)}(\mathbf{x})$ as a finitely parametrized approximation: $\phi(\mathbf{x})^T\theta$, where θ is sampled

from its corresponding posterior distribution conditioned on the data D obtained from past function evaluations: $\theta|D \sim N(\mathbf{A^{-1}\Phi^T y}_n, \sigma^2\mathbf{A^{-1}})$, 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 $\tilde{f}_i^{(M_i)}$ from each surrogate model $\mathcal{MF} - \mathcal{GP}_i$ as described above. A *cheap* multi-objective optimization problem over the K sampled functions $\tilde{f}_1^{(M_1)}, \tilde{f}_2^{(M_2)}, \cdots, \tilde{f}_K^{(M_K)}$ is solved to compute the 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 [47] to solve the MO problem with cheap objective functions noting that any other algorithm can be used.

2) Entropy Computation with a Sample Pareto Front. Let $\mathcal{Y}_s^* = \{\mathbf{v}^1, \cdots, \mathbf{v}^l\}$ be the sample Pareto front, where l is the size of the Pareto front and each $\mathbf{v}^i = \{v_1^i, \cdots, v_K^i\}$ is a K-vector evaluated at the K sampled high fidelity functions. The following inequality holds for each component $y_j^{(m_j)}$ of the K-vector $\mathbf{y^{(m)}} = \{y_1^{(m_1)}, \cdots, y_K^{(m_k)}\}$ in the entropy term $H(\mathbf{y^{(m)}} \mid D, \mathbf{x}, \mathcal{Y}_s^*)$:

$$y_j^{(m_j)} \leq y_{j_s}^* \quad \forall j \in \{1, \cdots, K\} \tag{7.11}$$

where $y_{j_s}^* = \max\{v_j^1, \cdots v_j^l\}$. The inequality essentially says that the j^{th} component of $\mathbf{y^m}$ (i.e., $y_j^{m_j}$) is upper-bounded by a value obtained by taking the maximum of j^{th} components of all l vectors $\{\mathbf{v}^1, \cdots, \mathbf{v}^l\}$ in the Pareto front $\mathcal{Y}_s^*$. The proof of 7.11

can be divided into two cases:

Case I. If y_j is evaluated at its highest fidelity (i.e $m_j = M_j$), inequality (7.11) can be proven by a contradiction argument. Suppose there exists some component $y_j^{(M_j)}$ of $\mathbf{y}^{(\mathbf{M})}$ such that $y_j^{(M_j)} > y_{j_s}^*$. However, by definition, $\mathbf{y}^{(\mathbf{M})}$ is a non-dominated point because no point dominates it in the jth dimension. This results in $\mathbf{y}^{(\mathbf{M})} \in \mathcal{Y}_s^*$ which is a contradiction. Therefore, our hypothesis that $y_j^{(M_j)} > y_{j_s}^*$ is incorrect and inequality (7.11) holds.

Case II. If y_j is evaluated at one of its lower fidelities (i.e, $m_j \neq M_j$), the proof follows from the assumption that the value of lower fidelity of a objective is usually smaller than the corresponding higher fidelity, i.e., $y_j^{(m_j)} \leq y_j^{(M_j)} \leq y_{j_s}^*$. This is especially true for most real-world experiments. For example, in optimizing a neural network's accuracy with respect to its hyperparameters, a commonly employed fidelity is the number of data samples used for training. It is reasonable to believe that the accuracy is always higher for the higher fidelity (more data samples to train on) when compared to a lower fidelity (less data samples). By combining the inequality (7.11) and the fact that each function is modeled as an independent MF-GP, 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 [38]:

$$H(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}, \mathcal{Y}_s^*) = \sum_{j=1}^{K} H(y_j^{(m_j)} | D, \mathbf{x}, y_{j_s}^*) \tag{7.12}$$

The computation of equation (7.12) requires the computation of the entropy of $p(y_j^{(m_j)}|D, \mathbf{x}, y_{j_s}^*)$. This is a conditional distribution that depends on the value of m_j and can be expressed as $H(y_j^{(m_j)}|D, \mathbf{x}, y_j^{(m_j)} \leq y_{j_s}^*)$. This entropy is dealt with in two cases.

First, for $\mathbf{m_j = M_j}$, the density function of this probability is approximated by truncated Gaussian distribution and its entropy can be expressed as [137]:

$$H(y_j^{(M_j)}|D, \mathbf{x}, y_j^{(M_j)} \leq y_{j_s}^*) \simeq \frac{(1 + \ln(2\pi))}{2} + \ln(\sigma_j^{(M_j)}(\mathbf{x})) + \ln \Phi(\gamma_s^{(M_j)}(\mathbf{x})) - \frac{\gamma_s^{(M_j)}(\mathbf{x})\phi(\gamma_s^{(M_j)}(\mathbf{x}))}{2\Phi(\gamma_s^{(M_j)}(\mathbf{x}))} \tag{7.13}$$

where $\gamma_s^{(M_j)}(\mathbf{x}) = \frac{y_{j_s}^* - \mu_j^{(M_j)}(\mathbf{x})}{\sigma_j^{(M_j)}(\mathbf{x})}$, and ϕ and Φ are the p.d.f and c.d.f of a standard normal distribution respectively.

Second, for $\mathbf{m_j \neq M_j}$, the density function of $p(y_j^{(m_j)}|D, \mathbf{x}, y_{j_s}^*)$ can be computed using two different approximations as described below.

Approximation 1 (MF-OSEMO-TG): As a consequence of **Case II**, which states that $y_j^{(m_j)} \leq y_{j_s}^*$ also holds for all lower fidelities, the entropy of $p(y_j^{(m_j)}|D, \mathbf{x}, y_{j_s}^*)$ can also be approximated by the entropy of a truncated Gaussian distribution and

expressed as follow:

$$H(y_j^{(m_j)}|D,\mathbf{x},y_j^{(m_j)} \leq y_{j_s}^*) \simeq \frac{(1+\ln(2\pi))}{2} + \ln(\sigma_j^{(m_j)}(\mathbf{x})) + \ln\Phi(\gamma_s^{(m_j)}(\mathbf{x})) - \frac{\gamma_s^{(m_j)}(\mathbf{x})\phi(\gamma_s^{(m_j)}(\mathbf{x}))}{2\Phi(\gamma_s^{(m_j)}(\mathbf{x}))} \tag{7.14}$$

where $\gamma_s^{(m_j)}(\mathbf{x}) = \frac{y_{j_s}^* - \mu_j^{(m_j)}(\mathbf{x})}{\sigma_j^{(m_j)}(\mathbf{x})}$.

Approximation 2 (MF-OSEMO-NI): Although equation (7.14) is sufficient for computing the entropy for $m_j \neq M_j$, it can be improved by conditioning on a tighter inequality $y_j^{(M_j)} \leq y_{j_s}^*$ as compared to the general one, i.e., $y_j^{(m_j)} \leq y_{j_s}^*$. As we show below, this improvement comes at the expense of not obtaining a final closed-form expression, but it can be efficiently computed via numerical integration. We apply the derivation of the entropy based on numerical integration for single-objective problem, proposed in [179], for the multi-objective setting.

Now, for calculating $H(y_j^{(m_j)}|D,\mathbf{x},y_j^{(m_j)} \leq y_{j_s}^*)$ by replacing $p(y_j^{(m_j)}|D,\mathbf{x},y_j^{(m_j)} \leq y_{j_s}^*)$ with $p(y_j^{(m_j)}|D,\mathbf{x},y_j^{(M_j)} \leq y_{j_s}^*)$ and using Bayes' theorem, we have:

$$p(y_j^{(m_j)}|D,\mathbf{x},y_j^{(M_j)} \leq y_{j_s}^*) = \frac{p(y_j^{(M_j)} \leq y_{j_s}^*|y_j^{(m_j)},D,\mathbf{x})p(y_j^{(m_j)},D,\mathbf{x})}{p(y_j^{(M_j)} \leq y_{j_s}^*|D,\mathbf{x})} \tag{7.15}$$

Both the densities, $p(y_j^{(M_j)} \leq y_{j_s}^*|D,\mathbf{x})$ and $p(y_j^{(m_j)},D,\mathbf{x})$ can be obtained from the

predictive distribution of MF-GP model and is given as follows:

$$p(y_j^{(m_j)}, D, \mathbf{x}) = \frac{\phi(\gamma_j^{(m_j)}(\mathbf{x}))}{\sigma_j^{(m_j)}} \tag{7.16}$$

$$p(y_j^{(M_j)} \leq y_{j_s}^* | D, \mathbf{x}) = \Phi(\gamma_s^{(M_j)}(\mathbf{x}))) \tag{7.17}$$

where $\gamma_j^{(m_j)}(\mathbf{x}) = \frac{y_j^{(m_j)} - \mu_j^{(m_j)}(\mathbf{x})}{\sigma_j^{(m_j)}(\mathbf{x})}$. Since MF-GP represents all fidelities as one unified Gaussian process, the joint marginal distribution $p(y_j^{(M_j)}, y_j^{(m_j)} | D, \mathbf{x})$ can be immediately obtained from the posterior distribution of the corresponding model $\mathcal{GP}_j$ as given below:

$$p(y_j^{(M_j)} | y_j^{(m_j)}, \mathbf{x}, D) \sim \mathcal{N}(\mu_j(\mathbf{x}), s_j^2(\mathbf{x})) \tag{7.18}$$

where $\mu_j(\mathbf{x}) = \frac{\sigma_j^{2^{(m_j M_j)}}(\mathbf{x})(y_j^{(m_j)} - \mu_j^{m_j}(\mathbf{x}))}{\sigma_j^{2^{(m_j)}}(\mathbf{x})}$ and $s_j^2(\mathbf{x}) = \sigma_j^{2^{(M_j)}}(\mathbf{x}) - \frac{(\sigma_j^{2^{(m_j M_j)}}(\mathbf{x}))^2}{\sigma_j^{2^{(m_j)}}(\mathbf{x})}$. As a result, $p(y_j^{(M_j)} \leq y_{j_s}^* | y_j^{(m_j)}, D, \mathbf{x})$ is expressed as the cumulative distribution of the Gaussian in (7.18):

$$p(y_j^{(M_j)} \leq y_{j_s}^* | y_j^{(m_j)}, D, \mathbf{x}) = \Phi(\frac{y_{j_s}^* - \mu_j(\mathbf{x})}{s_j(\mathbf{x})}) \tag{7.19}$$

By substituting (7.16), (7.17), and 7.19 into (7.15) we get:

$$H(y_j^{(m_j)} | D, \mathbf{x}, y_j^{(M_j)} \leq y_{j_s}^*) = -\int \Psi(y_j^{(m_j)}) \log(\Psi(y_j^{(m_j)})) dy_j^{(m_j)} \tag{7.20}$$

With $\Psi(y_j^{(m_j)}) = \Phi(\frac{y_{js}^* - \mu_j(\mathbf{x})}{s_j(\mathbf{x})}) \frac{\Phi(\gamma_s^{(M_j)}(\mathbf{x})))\phi(\gamma_j^{(m_j)}(\mathbf{x}))}{\sigma_j^{(m_j)}}$. Since this integral is over one-dimension variable $y_j^{(m_j)}$, numerical integration can result in a tight approximation.

A complete description of the MF-OSEMO algorithm is given in Algorithm 8. The blue colored steps correspond to computation of our acquisition function via sampling.

7.4 Experiments and Results

In this section, we describe our experimental setup, and present results of MF-OSEMO and baseline methods.

7.4.1 Experimental Setup

Baselines. We compare MF-OSEMO with state-of-the-art single-fidelity MO algorithms: ParEGO [117], PESMO [83], SMSego [158], EHI [67], and SUR [155]. We employ the code for these methods from the BO library Spearmint[1].

Statistical models. We use MF-GP models as described in section 8.1. We employ squared exponential (SE) kernels in all our experiments. The hyper-parameters are estimated after every 5 function evaluations. We initialize the MF-GP models for all functions by sampling initial points at random from a Sobol grid. We Initialise each of the lower fidelities with 5 points and the highest fidelity with only one point.

Synthetic benchmarks. We construct two synthetic benchmark problems using a

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

Algorithm 8 MF-OSEMO Algorithm

Input: input space $\mathfrak{X}$; K blackbox objective functions where each function f_j has multiple fidelities M_j $\Big(\{f_1^{(1)}(\mathbf{x}), \cdots, f_1^{(M_1)}(\mathbf{x})\}, \cdots, \{f_K^{(1)}(\mathbf{x}), \cdots, f_K^{(M_K)}(\mathbf{x})\}\Big)$; and total budget λ_{Total}

Initialize multi-fidelity Gaussian process models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$ by evaluating at initial points D
While $\lambda_t \leq \lambda_{total}$ **do**
  for each sample $s \in 1, \cdots, S$:
    Sample highest-fidelity functions $\tilde{f}_i^{(M_i)} \sim \mathcal{GP}_i, \quad \forall i \in \{1, \cdots, K\}$
    $\mathcal{Y}_s^* \leftarrow$ Pareto front of *cheap* multi-objective optimization over $(\tilde{f}_1^{(M_1)}, \cdots, \tilde{f}_K^{(M_K)})$
  Find the next point to evaluate: select $(\mathbf{x}_t, \mathbf{m}_t) \leftarrow \arg max_{\mathbf{x} \in \mathfrak{X}, \mathbf{m}} \ \alpha_t(\mathbf{x}, \mathbf{m}, \mathcal{Y}^*)$
  Update the total cost consumed: $\lambda_t \leftarrow \lambda_t + \lambda^{(\mathbf{m_t})}$
  Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_t, \mathbf{y}_t^{\mathbf{m}})\}$
  Update models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$
  $t \leftarrow t + 1$
end while
return Pareto front and Pareto set of $f_1(x), \cdots, f_K(x)$ based on $\mathcal{D}$
Procedure $\alpha_t(\mathbf{x}, \mathbf{m}, \mathcal{Y}_s^*)$
// Computes information gain (I) about the posterior of true Pareto front ($\mathcal{Y}^*$) per unit cost as a result of evaluating $\mathbf{x}$
// I = H_1 - H_2; where H_1 = Entropy of $\mathbf{y}^{(\mathbf{m})}$ conditioned on D and $\mathbf{x}$
// and H_2 = Expected entropy of $\mathbf{y}^{(\mathbf{m})}$ conditioned on D, $\mathbf{x}$ and ($\mathcal{Y}^*$)
Set $H_1 = H(\mathbf{y}^{(\mathbf{m})} \mid D, \mathbf{x}) = K(1 + \ln(2\pi))/2 + \sum_{j=1}^{K} \ln(\sigma_j^{(m_j)}(\mathbf{x}))$ (entropy of K-factorizable Gaussian)
To compute $H_2 \simeq \frac{1}{S} \sum_{s=1}^{S} \sum_{j=1}^{K} H(y_j^{(m_j)} | D, \mathbf{x}, y_{j_s}^*)$, initialize $H_2 = 0$
for each sample $\mathcal{Y}_s^*$ **do**
  for $j \in 1 \cdots K$ **do**
    Set $y_{j_s}^*$ = maximum of jth component of all vectors in $\mathcal{Y}_s^*$
    If $m_j = M_j$ // if evaluating jth function at highest fidelity
      H_2 += $H(y_j^{(M_j)} | D, \mathbf{x}, y_j^{(M_j)} \leq y_{j_s}^*)$ (entropy of truncated Gaussian $p(y_j^{(M_j)} | D, \mathbf{x}, y_j^{(M_j)} \leq y_{j_s}^*)$)
    If $m_j \neq M_j$ // if evaluating jth function at lower fidelity
      // two approximations are provided
      If approximation = TG
        H_2 += $H(y_j^{(m_j)} | D, \mathbf{x}, y_j^{(m_j)} \leq y_{j_s}^*)$ (entropy of truncated Gaussian $p(y_j^{(M_j)} | D, \mathbf{x}, y_j^{(m_j)} \leq y_{j_s}^*)$)
      If approximation = NI
        H_2 += $H(y_j^{(m_j)} | D, \mathbf{x}, y_j^{(M_j)} \leq y_{j_s}^*)$ (entropy computed via numerical integration)
  end for
end for
Divide by number of samples: $H_2 = H_2/S$
return $(H_1 - H_2)/\lambda^{(\mathbf{m})}$

combination of commonly employed benchmark functions for multi-fidelity and single-objective optimization [2], and two of the known general MO benchmarks [79]. Their complete details are provided in Table 7.2.

Real-world benchmarks. We consider two challenging problems that are described below.

1) Rocket launching simulation. We consider the simulation study of a rocket [81] being launched from the Earth's surface. Input variables for simulation are mass of fuel, launch height, and launch angle. Output objectives are the time taken to return to Earth's surface, the angular distance travelled with respect to the centre of the Earth, and the absolute difference between the launch angle and the radius at the point of launch. However, these simulations are computationally expensive and can take up to several hours. The simulator has a tolerance parameter that can be adjusted to perform multi-fidelity simulations: small tolerance means accurate simulations, but long runtime. We employ two tolerance parameter values to create two fidelities for each objective: cost of two fidelities are 0.05 minutes and 30 minutes respectively.

2) Network-on-chip (NOC) optimization. Designing good communication infrastructure is important to improve the quality of hardware designs. This is typically done using cycle-accurate simulators that imitate the real hardware. We consider a design space of NoC dataset consisting of 1024 implementation of a network-on-chip

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

[35]. Each configuration is defined by ten input variables (d=10). We optimize two objectives: latency and energy. This benchmark has two fidelities with costs 3 mins and 45 mins respectively.

Name	k	d	Benchmark functions	p	Costs
BC	2	2	Branin	2	$[1, 10]$
			Currin	2	$[1, 10]$
SPP	3	4	Shekel	3	$[0.1, 1, 10]$
			Park 1	2	$[1, 10]$
			Park 2	2	$[1, 10]$
ZDT3	2	6	Zitzler,Deb,Thiele	2^2	$[1, 10]^2$
DTLZ1	6	5	Deb,Thiele,Laumanns,Zitzler	3^6	$[0.1, 1, 10]^6$

Table 7.2: Details of synthetic benchmarks: Name, benchmark functions, no. of objectives k, input dimension d, number fidelities p, and costs of different fidelities for each function.

7.4.2 Results and Discussion

To evaluate the performance of MF-OSEMO, we employ a common multi-objective metric used in practice. The *Pareto hypervolume (PHV)* metric measures the quality of a given Pareto front [202]. We provide a detailed definition of the metrics in section 2.3 As a function of the cost of evaluations, we report the difference between the hypervolume of the ideal Pareto front ($\mathcal{Y}^*$) and hypervolume of the best reached Pareto front estimated by optimizing the posterior mean of the models at the highest fidelities [83]. The posterior means are optimized over a randomly generated grid of

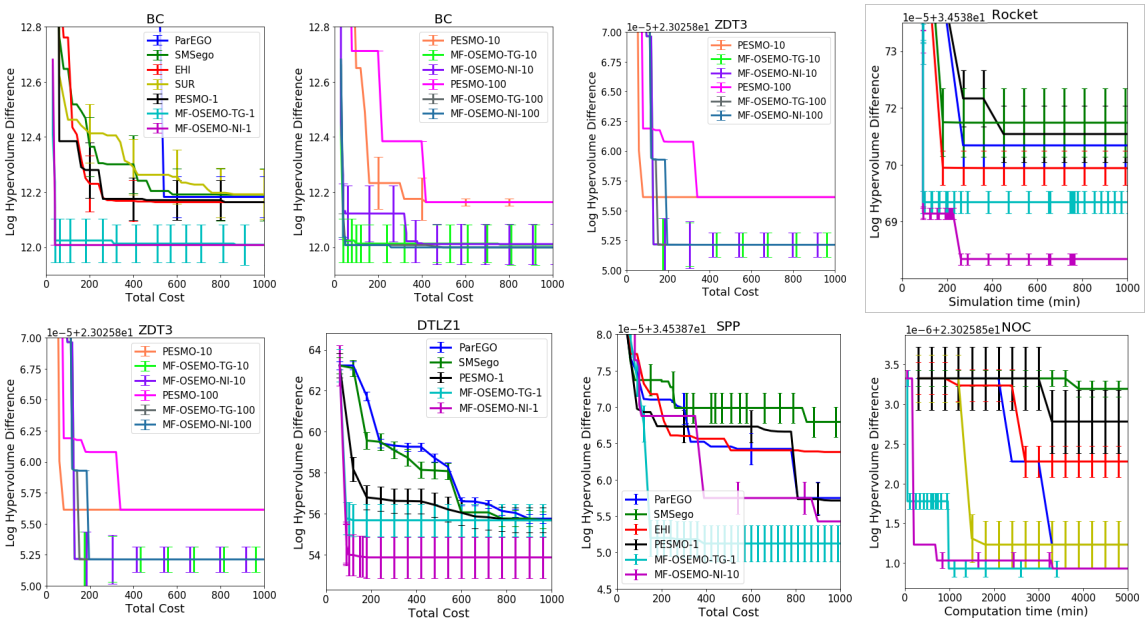

Figure 7.2: Results of MF-OSEMO and single-fidelity multi-objective BO algorithms on synthetic benchmarks and real-world problems. The log of the hypervolume difference is shown with varying cost.

10,000 points. We also provide the *cost reduction factor*, which is the ratio between the worst cost at which MF-OSEMO converges (worst case for MF-OSEMO), and the earliest cost for which any of the single-fidelity baselines converge (best case for baseline) after running all algorithms for very large costs. We run all experiments 10 times. The mean and variance of the PHV metrics across different runs are reported as a function of the total cost consumed. Since in all our experiments, the costs of different functions are on the same scale, we plot results against the sum of the costs.

MF-OSEMO vs. State-of-the-art. We compare the performance of MF-OSEMO-TG and MF-OSEMO-NI with single-fidelity MO methods. Figure 7.2 shows the re-

sults of all multi-objective BO algorithms including MF-OSEMO for synthetic and real-world benchmarks. We observe that: 1) MF-OSEMO consistently performs better than all baselines. Both the variants of MF-OSEMO converge at a much lower cost. 2) Rates of convergence of MF-OSEMO-TG and MF-OSEMO-NI are slightly different. However, in all cases, MF-OSEMO performs better than baseline methods. We notice that in few cases (e.g., both real-world benchmarks), MF-OSEMO-TG converges earlier than MF-OSEMO-NI. This demonstrates that even with loose approximation, using the MF-OSEMO-TG can provide consistently competitive results using less computation time.

Name	BC	SPP	ZDT3	DTLZ1	Rocket	NOC
λ	4.2	190	380	100	250	1200
λ_B	2000	1950	2000	800	4000	10000
Λ	99.79%	90.25%	81%	87.5%	93.75%	88%

Table 7.3: Convergence costs for MF-OSEMO and baselines, and cost reduction factor achieved by MF-OSEMO: *worst* convergence cost for MF-OSEMO λ, *best* convergence cost from all baselines methods λ_B, and cost reduction factor Λ.

Cost reduction factor. Some of the baselines will eventually converge if they are run for a much larger cost. In table 7.3, we provide the cost reduction factor to show the percentage of cost-gain achieved by using MF-OSEMO when compared to single-fidelity baselines. Although the metric gives advantage to baselines, the results in the table show a consistently high gain ranging from 81% to 99.8%.

Robustness of MF-OSEMO. We evaluate the performance of MF-OSEMO and PESMO with different number of Monte-Carlo samples (MCS). We provide results for two synthetic benchmarks BC and ZDT3 in figure 7.2 with 1, 10, and 100 MCS for PESMO, MF-OSEMO-TG, and MF-OSEMO-NI. For clarity of figures, we provided those results in two diffrent figures side by side. We notice that 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. However, MF-OSEMO-TG and MF-OSEMO-NI maintain a better performance consistently even with a single sample. These results strongly demonstrate that our proposed method is much more robust to the number of MCS.

7.5 Summary

We introduced a novel and principled approach referred as MF-OSEMO to solve multi-fidelity multi-objective Bayesian optimization problems. The key idea is to employ an output space entropy-based acquisition function to efficiently select inputs and fidelity vectors for evaluation. Our experimental results on both synthetic and real-world benchmarks showed that MF-OSEMO yields consistently better results than state-of-the-art single-fidelity methods.

CHAPTER EIGHT

CONTINUOUS FIDELITY MULTI-OBJECTIVE BO

Many real-world applications involve black-box optimization of multiple objectives using continuous function approximations that trade-off accuracy and resource cost of evaluation. For example, in rocket launching research, we need to find designs that trade off return time and angular distance using continuous-fidelity simulators (e.g., varying tolerance parameter to trade-off simulation time and accuracy) for design evaluations[196, 56]. The goal is to approximate the optimal Pareto set by minimizing the cost of evaluations.

In this Chapter, we propose a novel approach referred to as **i***nformation-Theoretic* **M***ulti-Objective Bayesian* **O***ptimization with* **C***ontinuous* **A***pproximations (iMOCA)* to solve this problem. The key idea is to select the sequence of input and function approximations for multiple objectives which maximize the information gain per unit cost for the optimal Pareto front. To efficiently compute entropy, which is an important step for iMOCA, we develop two qualitatively different approximations that make different trade-offs in terms of computation-time and accuracy. iMOCA extends the single-fidelity multi-objective algorithm MESMO and its discrete-fidelity version MF-OSEMO to the more general continuous-fidelity setting. We evaluate iMOCA and compare it to existing approaches including MESMO and MF-OSEMO.

8.1 Problem Setup

Continuous-Fidelity MOO Problem. The continuous-fidelity MOO problem is the general version of the discrete multi-fidelity setting where we have access to $g_i(\mathbf{x}, z_i)$ where g_i is an alternative function through which we can evaluate cheaper approximations of f_i by varying the fidelity variable $z_i \in \mathcal{Z}$ (continuous function approximations). Without loss of generality, let $\mathcal{Z}$=$[0, 1]$ be the fidelity space. Fidelities for each function f_i vary in the amount of resources consumed and the accuracy of evaluation, where z_i=0 and z_i^*=1 refer to the lowest and highest fidelity respectively. At the highest fidelity z_i^*, $g_i(\mathbf{x}, z_i^*) = f_i(\mathbf{x})$. The evaluation of an input $\mathbf{x} \in \mathcal{X}$ with fidelity vector $\mathbf{z} = [z_1, z_2, \cdots, z_K]$ produces an evaluation vector of K values denoted by $\mathbf{y} \equiv [y_1, y_2, \cdots, y_K]$, where $y_i = g_i(\mathbf{x}, z_i)$ for all $i \in \{1, 2, \cdots, K\}$. Let $\mathcal{C}_i(\mathbf{x}, z_i)$ be the cost of evaluating $g_i(\mathbf{x}, z_i)$. Our goal is to approximate the optimal Pareto set $\mathcal{X}^*$ over the highest fidelities functions while minimizing the overall cost of function evaluations (experiments). For example, in rocket launching research, we need to find designs that trade-off return time and angular distance using continuous-fidelity simulators (e.g., varying tolerance parameter to trade-off simulation time and accuracy) for design evaluations. Table 8.1 contains all the mathematical notations used in this section (iMOCA).

Cost of Function Evaluations. The total normalized cost of function evaluation is $\mathcal{C}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{K} (\mathcal{C}_i(\mathbf{x}, z_i)/\mathcal{C}_i(\mathbf{x}, z_i^*))$. We normalize the cost of each function by the

cost of its highest fidelity because the cost units of different objectives can be different. If the cost is known, it can be directly injected in the latter expression. However, in some real-world settings, the cost of a function evaluation can be only known after the function evaluation. In this case, the cost can be modeled by an independent Gaussian process. The predictive mean can be used during the optimization. The final goal is to recover $\mathcal{X}^*$ while minimizing the total cost of function evaluations.

Continuous-Fidelity GPs as Surrogate Models. Let $D = \{(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)\}_{i=1}^{t-1}$ be the training data from past t-1 function evaluations, where $\mathbf{x}_i \in \mathcal{X}$ is an input and $\mathbf{y}_i = [y_1, y_2, \cdots, y_K]$ is the output vector resulting from evaluating functions $g_1, g_2, \cdots, g_K$ for $\mathbf{x}_i$ at fidelities $z_1, z_2, \cdots, z_K$ respectively. We learn K surrogate statistical models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$ from $\mathcal{D}$, where each model $\mathcal{GP}_j$ corresponds to the jth function g_j. Continuous fidelity GPs (CF-GPs) are capable of modeling functions with continuous fidelities within a single model. Hence, we employ CF-GPs to build surrogate statistical models for each function. Specifically, we use the CF-GP model proposed in [109]. W.l.o.g, we assume that our functions g_j are defined over the spaces $\mathcal{X} = [0,1]^d$ and $\mathcal{Z} = [0,1]$. Let $g_j \sim \mathcal{GP}_j(0, \kappa_j)$ such that $y_j = g_j(z_j, \mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \eta^2)$ and $\kappa : (\mathcal{Z} \times \mathcal{X})^2 \to \mathbb{R}$ is the prior covariance matrix defined on the product of input and fidelity spaces.

$$\kappa_j([z_j, \mathbf{x}], [z'_j, \mathbf{x}']) = \kappa_{j\mathcal{X}}(\mathbf{x}, \mathbf{x}') \cdot \kappa_{j\mathcal{Z}}(z_j, z'_j)$$

where $\kappa_{j\mathcal{X}}, \kappa_{j\mathcal{Z}}$ are radial kernels over $\mathcal{X}$ and $\mathcal{Z}$ spaces respectively. $\mathcal{Z}$ controls the smoothness of g_j over the fidelity space to be able to share information across different fidelities. A key advantage of this model is that it integrates all fidelities into one single GP for information sharing. We denote the posterior mean and standard deviation of g_j conditioned on D by $\mu_{g_j}(\mathbf{x}, z_j)$ and $\sigma_{g_j}(\mathbf{x}, z_j)$. We denote the posterior mean and standard deviation of the highest fidelity functions $f_j(\mathbf{x}) = g_j(\mathbf{x}, z_j^*)$ by $\mu_{f_j}(x) = \mu_{g_j}(\mathbf{x}, z_j^*)$ and $\sigma_{f_j}(\mathbf{x}) = \sigma_{g_j}(\mathbf{x}, z_j^*)$ respectively. We define $\sigma^2_{g_j,f_j}(x)$ as the predictive co-variance between a lower fidelity z_j and the highest fidelity z_j^* at the same $\mathbf{x}$.

Notation	Definition
$g_1, g_2, \cdots, g_K$	General objective functions with low and high fidelities
$\tilde{g}_j$	Function sampled from the jth Gaussian process model at fidelity z_j
$z_1, z_2, \cdots, z_K$	The fidelity variables for each function
$\mathbf{z}$	Fidelities vector
$\mathbf{z}^* = [z_1^*, z_2^*, \cdots, z_K^*]$	Fidelities vector with all fidelities at their highest value
y_j	jth function g_j evaluated at fidelity z_j
$\mathbf{y} = [y_1, y_2, \cdots, y_K]$	Output vector resulting from evaluating $g_1, g_2, \cdots, g_K$ for $\mathbf{x}_i$ at fidelities $z_1, z_2, \cdots, z_K$ respectively
$\mathbf{f} = [f_1, f_2, \cdots, f_K]$	Output vector resulting from evaluating functions $f_1, f_2, \cdots, f_K$ or equivalently $g_1, g_2, \cdots, g_K$ for $\mathbf{x}_i$ at fidelities $z_1^*, z_2^*, \cdots, z_K^*$ respectively
$\mathcal{C}_j(\mathbf{x}, z_j)$	Cost of evaluating jth function g_j at fidelity z_j
$\mathcal{C}(\mathbf{x}, \mathbf{z})$	Total normalized cost $\mathcal{C}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{K} (\mathcal{C}_i(\mathbf{x}, z_i)/\mathcal{C}_i(\mathbf{x}, z_i^*))$
$\mathcal{Z}$	Fidelity space
$\mathcal{Z}_t^{(j)}$	Reduced fidelity space for function g_j at iteration t
$\mathcal{Z}_r$	Reduced fidelity space
ξ	Information gap
$\beta_t^{(j)}$	Exploration/exploitation parameter for function g_j at iteration t

Table 8.1: Mathematical notations and their associated definition used in this section (iMOCA)

8.2 iMOCA Algorithm with Two Approximations

We first describe the key idea behind our proposed iMOCA algorithm including the main challenges. Next, we present our algorithmic solution to address those challenges.

Key Idea of iMOCA: The acquisition function behind iMOCA employs principle of output space entropy search to select the sequence of input and fidelity-vector (one for each objective) pairs. iMOCA is applicable for solving MO problems in both continuous and discrete fidelity settings. *The key idea is to find the pair* $\{\mathbf{x}_t, \mathbf{z}_t\}$ *that maximizes the information gain* I *per unit cost about the Pareto front of the highest fidelities* (denoted by $\mathcal{Y}^*$), where $\{\mathbf{x}_t, \mathbf{z}_t\}$ represents a candidate input $\mathbf{x}_t$ evaluated at a vector of fidelities $\mathbf{z}_t = [z_1, z_2, \cdots, z_K]$ at iteration t. Importantly, iMOCA performs joint search over input space $\mathcal{X}$ and reduced fidelity space $\mathcal{Z}_r$ over fidelity vectors for this selection.

$$(\mathbf{x}_t, \mathbf{z}_t) \leftarrow \arg max_{\mathbf{x}\in\mathcal{X}, \mathbf{z}\in\mathcal{Z}_r} \ \alpha_t(\mathbf{x}, \mathbf{z}) \ , \text{ where } \quad \alpha_t(\mathbf{x}, \mathbf{z}) = I(\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}, \mathcal{Y}^*|D)/\mathcal{C}(\mathbf{x}, \mathbf{z}) \tag{8.1}$$

In the following sections, we describe the details and steps of our proposed algorithm iMOCA. We start by explaining the bottlenecks of continuous fidelity optimization due to the infinite size of the fidelity space followed by describing a principled approach

to reduce the fidelity space. Subsequently, we present the computational steps of our proposed acquisition function: Information gain per unit cost for each candidate input and fidelity-vector pair.

8.2.1 Approach to Reduce Fidelity Search Space

In this work, we focus primarily on MO problems with continuous fidelity space. The continuity of this space results in infinite number of fidelity choices. Thus, selecting an informative and meaningful fidelity becomes a major bottleneck. Therefore, we reduce the search space over fidelity-vector variables in a principled manner guided by the learned statistical models [109]. Our fidelity space reduction method is inspired from BOCA for single-objective optimization [109]. We apply the method in BOCA to each of the objective functions to be optimized in MO setting.

A favourable setting for continuous-fidelity methods would be for the lower fidelities g_j to be informative about the highest fidelity f_j. Let h_j be the bandwith parameter of the fidelity kernel $\kappa_{j\mathcal{Z}}$ and let $\xi : \mathcal{Z} \to [0, 1]$ be a measure of the gap in information about $g_j(., z_j^*)$ when queried at $z_j \neq z_j^*$ with $\xi(z_j) \approx \frac{\|z_j - z_j^*\|}{h_j}$ for the squared exponential kernels [109]. A larger h_j will result in g_j being smoother across $\mathcal{Z}$. Consequently, lower fidelities will be more informative about f_j and the information gap $\xi(z_j)$ will be smaller.

To determine an informative fidelity for each function in iteration t, we reduce the

space $\mathcal{Z}$ and select z_j from the subset $\mathcal{Z}_t^{(j)}$ defined as follows:

$$\mathcal{Z}_t^{(j)}(\mathbf{x}) = \{\{z_j \in \mathcal{Z}_{\setminus\{z_j^*\}}, \sigma_{g_j}(\mathbf{x}, z_j) > \gamma(z_j), \xi(z_j) > \beta_t^{(j)} \|\xi\|_\infty\} \cup \{z_j^*\}\} \tag{8.2}$$

where $\gamma(z_j) = \xi(z_j)(\frac{\mathcal{C}_j(\mathbf{x}, z_j)}{\mathcal{C}_j(\mathbf{x}, z_j^*)})^q$ and $q = \frac{1}{p_j + d + 2}$ with p_j, d the dimensions of $\mathcal{Z}$ and $\mathcal{X}$ respectively. Without loss of generality, we assume that $p_j = 1$. $\beta_t^{(j)} = \sqrt{0.5d \cdot log\ (2tl + 1)}$ is the exploration/exploitation parameter [109]. where, l is the effective L_1 diameter of $\mathcal{X}$ and is computed by scaling each dimension by the inverse of the bandwidth of the SE kernel for that dimension. We denote by $\mathcal{Z}_r = \{\mathcal{Z}_t^{(j)}, j \in \{1 \dots K\}\}$, the reduced fidelity space for all K functions.

We filter out the fidelities for each objective function at BO iteration t using the above-mentioned two conditions. We provide intuitive explanation of these conditions below.

The first condition $\sigma_{g_j}(\mathbf{x}, z_j) > \gamma(z_j)$: A reasonable multi-fidelity strategy would query the cheaper fidelities in the beginning to learn about the function g_j by consuming the least possible cost budget and later query from higher fidelities in order to gain more accurate information. Since the final goal is to optimize f_j, the algorithm should also query from the highest-fidelity. However, the algorithm might never query from higher fidelities due to their high cost. This condition will make sure that lower fidelities are likely to be queried, but not excessively and the algorithm will move toward querying higher fidelities as iterations progress. Since $\gamma(z_j)$ is monotonically

increasing in $\mathcal{C}_j$, this condition can be easily satisfied by cheap fidelities. However, if a fidelity is very far from z_j^*, then the information gap ξ will increase and hence, uninformative fidelities would be discarded. Therefore, $\gamma(z_j)$ will guarantee achieving a good trade-off between resource cost and information.

The second condition $\xi(z_j) > \beta_t^{(j)} \|\xi\|_\infty$: We recall that if the first subset of $\mathcal{Z}_t^{(j)}$ is empty, the algorithm will automatically evaluate the highest-fidelity z_j^*. However, if it is not empty, and since the fidelity space is continuous (infinite number of choices for z_j), the algorithm might query fidelities that are very close to z_j^* and would cost nearly the same as z_j^* without being as informative as z_j^*. The goal of this condition is to prevent such situations by excluding fidelities in the small neighborhood of z_j^* and querying z_j^* instead. Since $\beta_t^{(j)}$ increases with t and ξ is increasing as we move away from z_j^*, this neighborhood is shrinking and the algorithm will eventually query z_j^*.

8.2.2 Naive-CFMO: A Simple Continuous-Fidelity MO Baseline

In this section, we first describe a simple baseline approach referred to as *Naive-CFMO* to solve continuous-fidelity MO problems by combining the above-mentioned fidelity space reduction approach with existing multi-objective BO methods. Next, we explain the key drawbacks of Naive-CFMO and how our proposed iMOCA algorithm overcomes them.

A straightforward way to construct a continuous-fidelity MO method is to perform

a two-step selection process similar to the continuous-fidelity single-objective BO algorithm proposed in [109]:

Step 1) Select the input $\mathbf{x}$ that maximizes the acquisition function at the *highest fidelity*. This can be done using any existing multi-objective BO algorithm.

Step 2) Evaluate $\mathbf{x}$ at the *cheapest valid fidelity for each function* in the reduced fidelity space $\mathcal{Z}_t^{(j)}(\mathbf{x})$ computed using the reduction approach mentioned in the previous section. Since we are studying information gain based methods in this work, we instantiate Naive-CFMO using the state-of-the-art information-theoretic MESMO algorithm [17] for Step 1. Algorithm 9 shows the complete pseudo-code of Naive-CFMO.

Drawbacks of Naive-CFMO: Unfortunately, Naive-CFMO has two major drawbacks.

- The acquisition function solely relies on the highest-fidelity f_j. Therefore, it does not capture and leverage the statistical relation between different fidelities and full-information provided by the global function g_j.
- Generally, there is a dependency between the fidelity space and the input space in continuous-fidelity problems. Therefore, selecting an input that maximizes the highest-fidelity and then evaluating it at a different fidelity can result in a mismatch in the evaluation process leading to poor performance and slower convergence.

iMOCA vs. Naive-CFMO: Our proposed iMOCA algorithm overcomes the drawbacks of Naive-CFMO as follows.

- iMOCA's acquisition function maximizes the information gain per unit cost across all fidelities by capturing the relation between fidelities and the impact of resource cost on information gain.
- iMOCA performs joint search over input and fidelity space to select the input variable $\mathbf{x} \in \mathcal{X}$ and fidelity variables $\mathbf{z} \in \mathcal{Z}_r$ while maximizing the proposed acquisition function. Indeed, our experimental results demonstrate the advantages of iMOCA over Naive-CFMO.

8.2.3 Information-Theoretic Continuous-Fidelity Acquisition Function

In this section, we explain the technical details of the acquisition function behind iMOCA. We propose two approximations for the computation of information gain per unit cost.

The information gain in equation (8.1) is defined as the expected reduction in entropy $H(.)$ of the posterior distribution $P(\mathcal{Y}^*|D)$ due to evaluating $\mathbf{x}$ at fidelity vector $\mathbf{z}$. Based on the symmetric property of information gain, the latter can be rewritten as follows:

$$I(\{\mathbf{x},\mathbf{y},\mathbf{z}\},\mathcal{Y}^*|D) = H(\mathbf{y}|D,\mathbf{x},\mathbf{z}) - \mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y}|D,\mathbf{x},\mathbf{z},\mathcal{Y}^*)] \tag{8.3}$$

In equation (8.3), the first term is the entropy of a K-dimensional Gaussian distribution that can be computed in closed form as follows:

$$H(\mathbf{y}|D,\mathbf{x},\mathbf{z}) = \sum_{j=1}^{K} \ln(\sqrt{2\pi e}\ \sigma_{g_j}(\mathbf{x},z_j)) \tag{8.4}$$

In equation (8.3), the second term is an expectation over the Pareto front of the highest fidelities $\mathcal{Y}^*$. This term can be approximated using Monte-Carlo sampling:

$$\mathbb{E}_{\mathcal{Y}^*}[H(\mathbf{y}|D,\mathbf{x},\mathbf{z},\mathcal{Y}^*)] \simeq \frac{1}{S}\sum_{s=1}^{S}[H(\mathbf{y}|D,\mathbf{x},\mathbf{z},\mathcal{Y}_s^*)] \tag{8.5}$$

where S is the number of samples and $\mathcal{Y}_s^*$ denote a sample Pareto front obtained over the highest fidelity functions sampled from K surrogate models. To compute equation (8.5), we provide algorithmic solutions to construct Pareto front samples $\mathcal{Y}_s^*$ and to compute the entropy with respect to a given Pareto front sample $\mathcal{Y}_s^*$.

1) Computing Pareto Front Samples: We first sample highest fidelity functions

$\tilde{f}_1, \cdots, \tilde{f}_K$ from the posterior CF-GP models via random Fourier features [84, 159]. This is done similar to prior work [83, 185]. We solve a cheap MO optimization problem over the K sampled functions $\tilde{f}_1, \cdots, \tilde{f}_K$ using the popular NSGA-II algorithm [47] to compute the sample Pareto front $\mathcal{Y}_s^*$.

2) Entropy Computation for a Given Pareto Front Sample: Let $\mathcal{Y}_s^* = \{\mathbf{v}^1, \cdots, \mathbf{v}^l\}$ be the sample Pareto front, where l is the size of the Pareto front and each $\mathbf{v}^i = \{v_1^i, \cdots, v_K^i\}$ is a K-vector evaluated at the K sampled highest-fidelity functions. The following inequality holds for each component y_j of $\mathbf{y} = (y_1, \cdots, y_K)$ in the entropy term $H(\mathbf{y}|D, \mathbf{x}, \mathbf{z}, \mathcal{Y}_s^*)$:

$$y_j \leq f_s^{j*} \quad \forall j \in \{1, \cdots, K\} \tag{8.6}$$

where $f_s^{j*} = \max\{v_j^1, \cdots v_j^l\}$. Essentially, this inequality says that the j^{th} component of $\mathbf{y}$ (i.e., y_j) is upper-bounded by a value, which is the maximum of j^{th} components of all l vectors $\{\mathbf{v}^1, \cdots, \mathbf{v}^l\}$ in the Pareto front $\mathcal{Y}_s^*$. Inequality (8.6) holds by the same proof of (7.11). *For the ease of notation, we drop the dependency on* $\mathbf{x}$ *and* $\mathbf{z}$*. We use* f_j *to denote* $f_j(x) = g_j(x, z_j^*)$*, the evaluation of the highest fidelity at* x *and* y_j *to denote* $g_j(x, z_j)$ *the evaluation of* g_j *at a lower fidelity* $z_j \neq z_j^*$*.*

By combining the inequality (8.6) and the fact that each function is modeled as an independent CF-GP, 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 [38]:

$$H(\mathbf{y}|D,\mathbf{x},\mathbf{z},\mathcal{Y}_s^*) = \sum_{j=1}^{K} H(y_j|D,\mathbf{x},z_j,f_s^{j*}) \tag{8.7}$$

The computation of (8.7) requires the computation of the entropy of $p(y_j|D,\mathbf{x},z_j,f_s^{j*})$. This is a conditional distribution that depends on the value of z_j and can be expressed as $H(y_j|D,\mathbf{x},z_j,y_j \leq f_s^{j*})$. This entropy can be computed using two different approximations as described below.

Truncated Gaussian Approximation (iMOCA-T): As a consequence of (8.6), which states that $y_j \leq f_s^{j*}$ also holds for all fidelities, the entropy of $p(y_j|D,\mathbf{x},z_j,f_s^{j*})$ can also be approximated by the entropy of a truncated Gaussian distribution and expressed as follows:

$$H(y_j|D,\mathbf{x},z_j,y_j \leq f_s^{j*}) \simeq \ln(\sqrt{2\pi e}\ \sigma_{g_j}) + \ln\Phi(\gamma_s^{(g_j)}) - \frac{\gamma_s^{(g_j)}\phi(\gamma_s^{(g_j)})}{2\Phi(\gamma_s^{(g_j)})} \text{ where } \gamma_s^{(g_j)} = \frac{f_s^{j*}-\mu_{g_j}}{\sigma_{g_j}} \tag{8.8}$$

From equations (8.5), (8.4), and (8.8), we get the final expression of iMOCA-T as shown below:

$$\alpha_t(\mathbf{x},\mathbf{z},\mathcal{Y}^*) \simeq \frac{1}{\mathcal{C}(\mathbf{x},\mathbf{z})S}\sum_{j=1}^{K}\sum_{s=1}^{S}\left[\frac{\gamma_s^{(g_j)}\phi(\gamma_s^{(g_j)})}{2\Phi(\gamma_s^{(g_j)})} - \ln(\Phi(\gamma_s^{(g_j)}))\right] \tag{8.9}$$

Extended-skew Gaussian Approximation (iMOCA-E): Although equation (8.9) is sufficient for computing the entropy, this entropy can be mathematically interpreted and computed with a different approximation. The condition $y_j \leq f_s^{j*}$, is originally expressed as $f_j \leq f_s^{j*}$. Substituting this condition with it's original equivalent, the entropy becomes $H(y_j|D, \mathbf{x}, z_j, f_j \leq f_s^{j*})$. Since y_j is an evaluation of the function g_j while f_j is an evaluation of the function f_j, we observe that $y_j|f_j \leq f_s^{j*}$ can be approximated by an extended-skew Gaussian (ESG) distribution [140, 15]. It has been shown that the differential entropy of an ESG does not have a closed-form expression [6]. Therefore, we derive a simplified expression where most of the terms are analytical by manipulating the components of the entropy. We apply the derivation of the entropy based on ESG formulation, proposed in [140], for the multi-objective setting.

In order to simplify the calculation $H(y_j|D, \mathbf{x}, z_j, f_j \leq f_s^{j*})$, we start by deriving an expression for its probability distribution. Based on the definition of the conditional distribution of a bi-variate normal, $f_j|y_j$ is normally distributed with mean $\mu_{f_j} + \frac{\sigma_{f_j}}{\sigma_{g_j}}\tau(y_j - \mu_{g_j})$ and variance $\sigma_{f_j}^2(1-\tau)^2$, where $\tau = \frac{\sigma_{g_j,f_j}^2}{\sigma_{g_j}\sigma_{f_j}}$ is the predictive correlation between y_j and f_j. We can now write the cumulative distribution function for $y_j|f_j \leq$

f_s^{j*} as shown below:

$$P(y_j \leq u | f_j \leq f_s^{j*}) = \frac{P(y_j \leq u, f_j \leq f_s^{j*})}{P(f_j \leq f_s^{j*})} = \frac{\int_{-\infty}^{u} \phi\left(\frac{\theta - \mu_{g_j}}{\sigma_{g_j}}\right) \Phi\left(\frac{f_s^{j*} - \mu_{f_j} - \frac{\sigma_{f_j}}{\sigma_{g_j}} \tau(\theta - \mu_{g_j})}{\sqrt{\sigma_{f_j}^2 (1-\tau)^2}}\right) d\theta}{\sigma_{g_j} \Phi\left(\frac{f_s^{j*} - \mu_{f_j}}{\sigma_{f_j}}\right)}$$

Let us define the normalized variable $\Gamma_{f_s^{j*}}$ as $\Gamma_{f_s^{j*}} \sim \frac{y_j - \mu_{g_j}}{\gamma_{g_j}} | f_j \leq f_s^{j*}$. After differentiating with respect to u, we can express the probability density function for $\Gamma_{f_s^{j*}}$ as:

$$P(u) = \frac{\phi(u)}{\Phi(\gamma_s^{(f_j)})} \Phi\left(\frac{\gamma_s^{(f_j)} - \tau u}{\sqrt{1 - \tau^2}}\right)$$

which is the density of an ESG with mean and variance defined as follows:

$$\mu_{\Gamma_{f_s^{j*}}} = \tau \frac{\phi(\gamma_s^{(f_j)})}{\Phi(\gamma_s^{(f_j)})}, \sigma_{\Gamma_{f_s^{j*}}} = 1 - \tau^2 \frac{\phi(\gamma_s^{(f_j)})}{\Phi(\gamma_s^{(f_j)})} \left[\gamma_s^{(f_j)} + \frac{\phi(\gamma_s^{(f_j)})}{\Phi(\gamma_s^{(f_j)})}\right] \tag{8.10}$$

Therefore, we can express the entropy of the ESG as shown below:

$$H(\Gamma_{f_s^{j*}}) = -\int P(u) \ln(P(u)) du \tag{8.11}$$

We also derive a more simplified expression of the iMOCA-E acquisition function

based on ESG. For a fixed sample ${f_s}^j*$, $H(\Gamma_{f_s^{j*}})$ can be decomposed as follows:

$$H(\Gamma_{f_s^{j*}}) = \mathbb{E}_{u\sim\Gamma_{f_s^{j*}}}\left[-\ln(\phi(u)) + \ln(\Phi(\gamma_s^{(f_j)})) - \ln(\Phi(\frac{\gamma_s^{(f_j)} - \tau u}{\sqrt{1-\tau^2}}))\right] \tag{8.12}$$

We expand the first term as shown below:

$$\mathbb{E}_{u\sim\Gamma_{f_s^{j*}}}\left[-\ln(\phi(u))\right] = \frac{1}{2}\ln(2\pi) + \frac{1}{2}\mathbb{E}_{u\sim\Gamma_{f_s^{j*}}}\left[u^2\right] \tag{8.13}$$

From the mean and variance of $\Gamma_{f_s^{j*}}$ in equation (8.10), we get:

$$\mathbb{E}_{u\sim\Gamma_{f_s^{j*}}}\left[u^2\right] = \mu^2_{\Gamma_{f_s^{j*}}} + \sigma_{\Gamma_{f_s^{j*}}} = 1 - \tau^2\frac{\phi(\gamma_s^{(f_j)})\gamma_s^{(f_j)}}{\Phi(\gamma_s^{(f_j)})} \tag{8.14}$$

We note that the final entropy can be computed using the following expression.

$$H(y_j|D, \mathbf{x}, z_j, y_j \leq f_s^{j*}) = H(\Gamma_{f_s^{j*}}) + \ln(\sigma_{g_j}) \tag{8.15}$$

By combining equations (8.12) and (8.15), we get:

$$\begin{aligned} H(y_j|D, \mathbf{x}, z_j, f_j \leq f_s^{j*}) \simeq \ln(\sqrt{2\pi e}\ \sigma_{g_j}) + \ln(\Phi(\gamma_s^{(f_j)})) - \tau^2\frac{\phi(\gamma_s^{(f_j)})\gamma_s^{(f_j)}}{2\Phi(\gamma_s^{(f_j)})} \\ - \mathbb{E}_{u\sim\Gamma_{f_s^{j*}}}\left[\ln(\Phi(\frac{\gamma_s^{(f_j)} - \tau u}{\sqrt{1-\tau^2}}))\right] \end{aligned} \tag{8.16}$$

From equations (8.5), (8.4) and (8.16), the final expression of iMOCA-E can be

expressed as follow:

$$\alpha_t(\mathbf{x}, \mathbf{z}, \mathcal{Y}^*) \simeq \frac{1}{\mathcal{C}(\mathbf{x}, \mathbf{z})S} \sum_{j=1}^{K} \sum_{s=1}^{S} \tau^2 \frac{\gamma_s^{(f_j)} \phi(\gamma_s^{(f_j)})}{2\Phi(\gamma_s^{(f_j)})} - \ln(\Phi(\gamma_s^{(f_j)})) + \mathbb{E}_{u \sim \Gamma_{f_s^{j*}}} \left[\ln(\Phi(\frac{\gamma_s^{(f_j)} - \tau u}{\sqrt{1-\tau^2}})) \right] \tag{8.17}$$

The expression given by equation (8.17) is mostly analytical except for the last term. We perform numerical integration via Simpson's rule using $\mu_{\Gamma_{f_s^{j*}}} \mp \gamma \sqrt{\sigma(\Gamma_{f_s^{j*}})}$ as the integral limits. Since this integral is over one-dimension variable, numerical integration can result in a tight approximation with low computational cost. Complete pseudo-code of iMOCA is shown in Algorithm 10.

Generality of the two approximations: We observe that for any fixed value of $\mathbf{x}$, when we choose the highest-fidelity for each function $\mathbf{z}$=$\mathbf{z}^*$, **a)** For iMOCA-T, we will have $g_i = f_j$; and **b)** For iMOCA-E, we will have $\tau = 1$. Consequently, both equation (8.9) and equation (8.17) will degenerate to the acquisition function of MESMO optimizing only highest-fidelity functions given in equation (3.13) in section 3.3.

The main advantages of our proposed acquisition function are: cost-efficiency, computational efficiency, and robustness to the number of Monte-Carlo samples. Indeed, our experiments demonstrate these advantages over state-of-the-art single-fidelity MO algorithms.

Algorithm 9 Naive-CFMO Algorithm

Input: input space $\mathcal{X}$; K blackbox functions f_j and their continuous approximations g_j; total budget $\mathcal{C}_{total}$

Initialize continuous fidelity Gaussian process $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$ by evaluating at initial points D
While $\mathcal{C}_t \leq \mathcal{C}_{total}$ **do**
  for each sample $s \in 1, \cdots, S$:
    Sample highest-fidelity functions $\tilde{f}_j \sim \mathcal{GP}_j(., z_j^*)$
    $\mathcal{Y}_s^* \leftarrow$ Solve *cheap* MOO over $(\tilde{f}_1, \cdots, \tilde{f}_K)$
  Find the query based on $\mathcal{Y}^* = \{\mathcal{Y}_s^*, s \in \{1 \dots S\}\}$:
  // Use eq (3.13) for α_t (MESMO)
  select $\mathbf{x}_t \leftarrow \arg max_{\mathbf{x} \in \mathcal{X}} \ \alpha_t(\mathbf{x}, \mathcal{Y}^*)$
  for $j \in 1 \cdots K$ **do**
    select $z_j \leftarrow \arg min_{\mathbf{z_j} \in \mathcal{Z}_t^{(j)}(\mathbf{x_t}) \cup \{z_j^*\}} \ \mathcal{C}_i(x_t, z_j)$
  Fidelity vector $\mathbf{z}_t \leftarrow [z_1 \dots z_K]$
  Update the total cost: $\mathcal{C}_t \leftarrow \mathcal{C}_t + \mathcal{C}(\mathbf{x}_t, \mathbf{z}_t)$
  Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)\}$
  Update models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$
  $t \leftarrow t + 1$
end while
return Pareto front and Pareto set of black-box functions $f_1(x), \cdots, f_K(x)$

Algorithm 10 iMOCA Algorithm

Input: input space $\mathcal{X}$; K blackbox functions f_j and their continuous approximations g_j; total budget $\mathcal{C}_{total}$

Initialize continuous fidelity Gaussian process $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$ by initial points D
While $\mathcal{C}_t \leq \mathcal{C}_{total}$ **do**
  for each sample $s \in 1, \cdots, S$:
    Sample highest-fidelity functions $\tilde{f}_j \sim \mathcal{GP}_j(., z_j^*)$
    $\mathcal{Y}_s^* \leftarrow$ Solve *cheap* MOO over $(\tilde{f}_1, \cdots, \tilde{f}_K)$
  Find the query based on $\mathcal{Y}^* = \{\mathcal{Y}_s^*, s \in \{1 \dots S\}\}$
  // Choose one of the two approximations
  If approx = T // Use eq (8.9) for α_t (iMOCA-T)
    select $(\mathbf{x}_t, \mathbf{z}_t) \leftarrow \arg max_{\mathbf{x} \in \mathcal{X}, \mathbf{z} \in \mathcal{Z}_r} \ \alpha_t(\mathbf{x}, \mathbf{z}, \mathcal{Y}^*)$
  If approx = E // Use eq (8.17) for α_t (iMOCA-E)
    select $(\mathbf{x}_t, \mathbf{z}_t) \leftarrow \arg max_{\mathbf{x} \in \mathcal{X}, \mathbf{z} \in \mathcal{Z}_r} \ \alpha_t(\mathbf{x}, \mathbf{z}, \mathcal{Y}^*)$
  Update the total cost: $\mathcal{C}_t \leftarrow \mathcal{C}_t + \mathcal{C}(\mathbf{x}_t, \mathbf{z}_t)$
  Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)\}$
  Update models $\mathcal{GP}_1, \cdots, \mathcal{GP}_K$
  $t \leftarrow t + 1$
end while
return Pareto front and Pareto set of black-box functions $f_1(x), \cdots, f_K(x)$

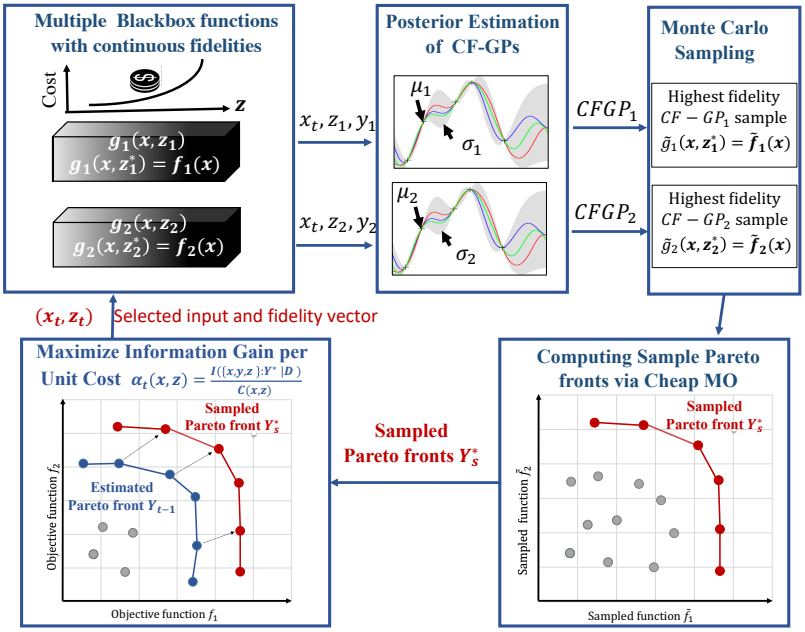

Figure 8.1: Overview of the iMOCA algorithm for two objective functions (K=2). We build Continuous fidelity statistical models $\mathcal{CFGP}_1$, $\mathcal{CFGP}_2$ for the two objective functions $f_1(x)$ and $f_2(x)$ with z_1 and z_2 fildelities respectively. First, we sample highest fidelity functions from the statistical models. We compute sample pareto fronts by solving a cheap MO problem over the sampled functions. Second, we select the best candidate input x_t and fidelity vector $z_t = (z_1, z_2)$ that maximizes the information gain per unit cost . Finally, we evaluate the functions for x_t at fidelities z_t to get (y_1, y_2) and update the statistical models using the new training example.

8.3 Experiments and Results

We present the results for iMOCA with MESMO and MF-OSEMO as baselines since iMOCA is the generalization of both MESMO and MF-OSEMO to the most general setting (continuous-fidelity).

Experimental Setup. In our experiments, we employed CF-GP models as described in section 8.1 with squared exponential kernels. We initialize the surrogate

models of all functions with the same number of points selected randomly from both lower and higher fidelities. We compare iMOCA with several baselines: six state-of-the-art single-fidelity MO algorithms (ParEGO, SMSego, EHI, SUR, PESMO, and MESMO) and one naive continuous-fidelity baseline that we proposed in Section 8.2.2. We employ the code for ParEGO, PESMO, SMSego, EHI, and SUR from the BO library Spearmint[1]. The code for all four of our algorithms are available in public Github repositories. We provide more details about the algorithms parameters, libraries, and computational resources in the Appendix C.2.2. For experiments in discrete fidelity setting, the number of fidelities is very limited. Thus, the fidelity space reduction method deem meaningless in this case. Therefore, we employ iMOCA without fidelity space reduction for those scenarios. Additionally, we compare to the state-of-the-art discrete fidelity method MF-OSEMO. MF-OSEMO has two variants: MF-OSEMO-TG and MF-OSEMO-NI. Since MF-OSEMO-TG has the same formulation as iMOCA-T and provide similar results, we compare only to MF-OSEMO-NI.

8.3.1 Synthetic Benchmarks

We evaluate our most general algorithm iMOCA and baselines on four different synthetic benchmarks. We construct two problems using a combination of benchmark functions for continuous-fidelity and single-objective optimization [175]: *Branin,Currin* (with K=2, d=2) and *Ackley, Rosen, Sphere* (with K=3, d=5). To show the effective-

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

ness of iMOCA on settings with discrete fidelities, we employ two of the known general MO benchmarks: *QV* (with K=2, d=8) and *DTLZ1* (with K=6, d=5) [79, 169]. We provide their complete details in Appendix C.2.1. The titles of plots in Fig. 8.2, Fig. 8.4, and Fig. 8.3 denote the corresponding experiments.

8.3.2 Real-world Engineering Design Optimization Problems

We evaluate iMOCA and baselines on four real-world design optimization problems from diverse engineering domains. We provide the details of these problems below.

1) Analog Circuit Design Optimization. Design of a voltage regulator via Cadence circuit simulator that imitate the real hardware [25, 88]. The simulation time can be adjusted to vary the simulation from fast and inaccurate to long and accurate. Each candidate circuit design is defined by 33 input variables (d=33). We optimize nine objectives: efficiency, four output voltages, and four output ripples. This problem has a continuous-fidelity space with cost varying from 10 mins to 120 mins.

2) Panel Structure Design for Large Vessels. The deck structure in large vessels commonly require the design of panels resisting uni-axial compression in the direction of the stiffeners [201]. We consider optimizing the trade-off between two objective functions: weight and strength of the panel. These functions depend on six input variables (d=6): one of them is the number of stiffeners used and five others relating

to the plate thickness and stiffener dimensions. This problem has a discrete fidelity setting: two fidelities with computational costs 1 min and 21 mins respectively.

3) Rocket Launching Simulation. Rocket launching studies [81] require several long and computationally-expensive simulations to reach an optimal design. In this problem, we have three input variables ($d = 3$): mass of fuel, launch height, and launch angle. The three objective functions are return time, angular distance, and difference between the launch angle and the radius at the point of launch. The simulator has a parameter that can be adjusted to perform continuous fidelity simulations. We employ the parameter range to vary the cost from 0.05 to 30 mins.

4) Network-On-Chip Design. Communication infrastructure is critical for efficient data movement in hardware chips [104, 63, 37, 42] and they are designed using cycle-accurate simulators. We consider a dataset of 1024 configurations of a network-on-chip with ten input variables (d=10) [35]. We optimize two objectives: latency and energy. This problem has two discrete fidelities with costs 3 mins and 45 mins respectively.

8.3.3 Results and Discussion

We compare iMOCA with both approximations (iMOCA-T and iMOCA-E) to all baselines. We employ two known metrics for evaluating the quality of a given Pareto front: *Pareto hypervolume (PHV)* metric and R_2 indicator. We provide a detailed definition of the metrics in section 2.3. We report both the difference in the hyper-

volume, and the average distance between an optimal Pareto front ($\mathcal{F}^*$) and the best recovered Pareto front estimated by optimizing the posterior mean of the models at the highest fidelities [83]. The mean and variance of PHV and R_2 metrics across 10 different runs are reported as a function of the total cost.

Fig. 8.2 shows the PHV results of all the baselines and iMOCA for synthetic and real-world experiments (Fig. 8.3 shows the corresponding R_2 results). We observe that iMOCA consistently outperforms all baselines. Both iMOCA-T and iMOCA-E have lower converge cost. Additionally, iMOCA-E shows a better convergence rate than iMOCA-T. This result can be explained by its tighter approximation. Nevertheless, iMOCA-T displays very close or sometimes better results than iMOCA-E. This demonstrates that even with loose approximation, using the iMOCA-T approximation can provide consistently competitive results using less computation time. For experiments with the discrete fidelity setting, iMOCA most of the times outperformed MF-OSEMO or produced very close results. It is important to note that MF-OSEMO is an algorithm designed specifically for the discrete-fidelity setting. Therefore, the competitive performance of iMOCA shows its effectiveness and generalisability.

Figure 8.4 shows the results of evaluating iMOCA and PESMO with varying number of Monte-Carlo samples $S \in \{1, 10, 100\}$. For ease of comparison and readability, we present these results in two different figures side by side. We observe that the convergence rate of PESMO is dramatically affected by the number of MC samples S.

However, iMOCA-T and iMOCA-E maintain a better performance consistently even with a single sample. These results strongly demonstrate that our method *iMOCA is much more robust to the number of Monte-Carlo samples.*

Table 8.2: *Best* convergence cost from all baselines $\mathcal{C}_B$, *Worst* convergence cost for iMOCA $\mathcal{C}$, and cost reduction factor $\mathcal{G}$.

Name	BC	ARS	Circuit	Rocket
$\mathcal{C}_B$	200	300	115000	9500
$\mathcal{C}$	30	100	55000	2000
$\mathcal{G}$	85%	66.6%	52.1%	78.9%

Cost Reduction Factor. We also provide the *cost reduction factor* for experiments with continuous fidelities, which is the percentage of gain in the convergence cost when compared to the best performing baseline (the earliest cost for which any of the single-fidelity baselines converge). Although this metric gives advantage to baselines, the results in Table 8.2 show a consistently high gain ranging from 52.1% to 85%.

8.4 Summary

We introduced a novel approach referred to as iMOCA to solve multi-objective BO problems with continuous function approximations. The key idea is to select inputs and function approximations for evaluation which maximizes the information gained per unit resource cost about the optimal Pareto front. Experimental results on diverse benchmarks showed that iMOCA consistently outperforms state-of-the-art single-fidelity methods and a naive continuous-fidelity MOO algorithm.

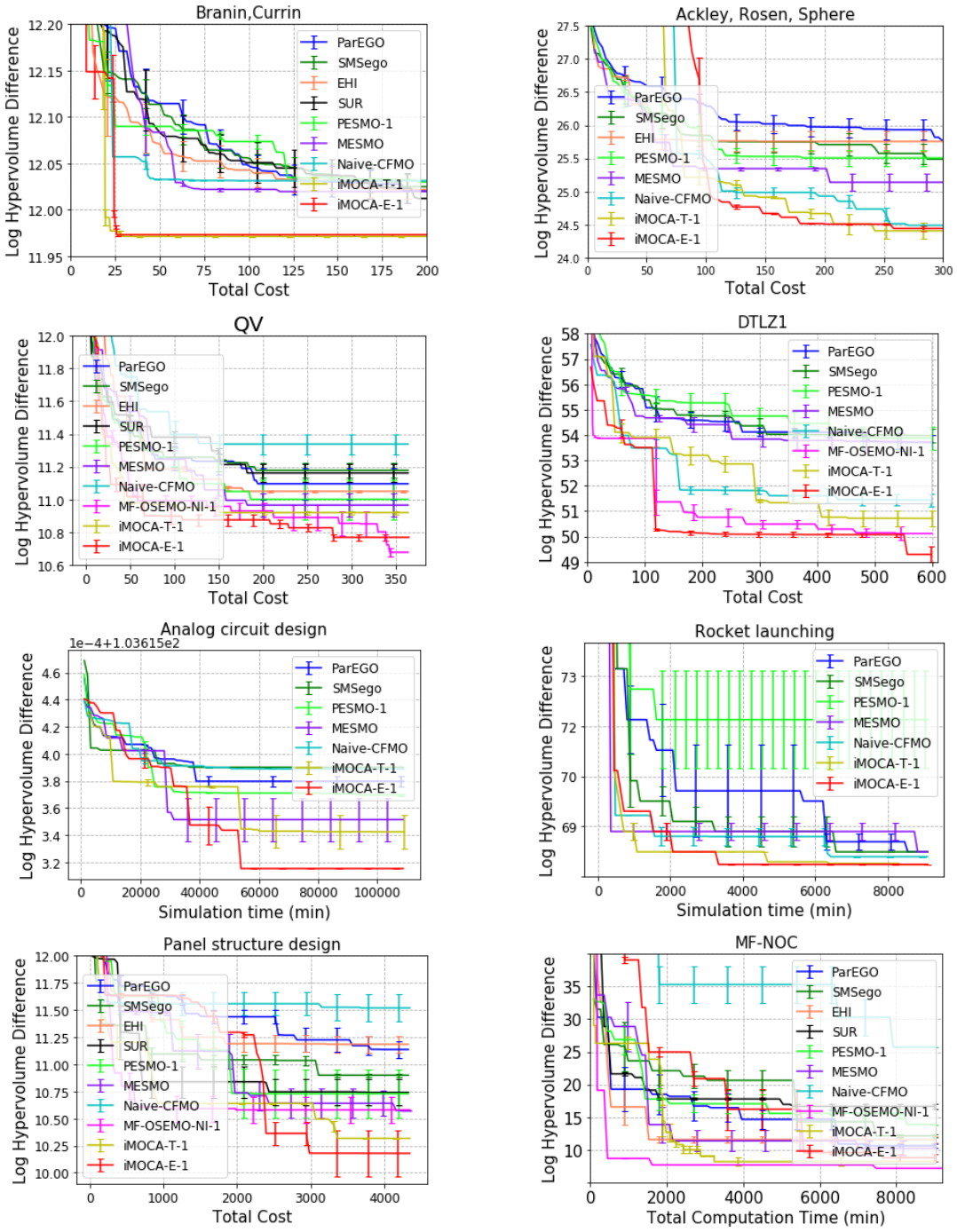

Figure 8.2: Results of iMOCA and the baselines algorithms on synthetic benchmarks and real-world problems. The PHV metric is presented against the total resource cost of function evaluations.

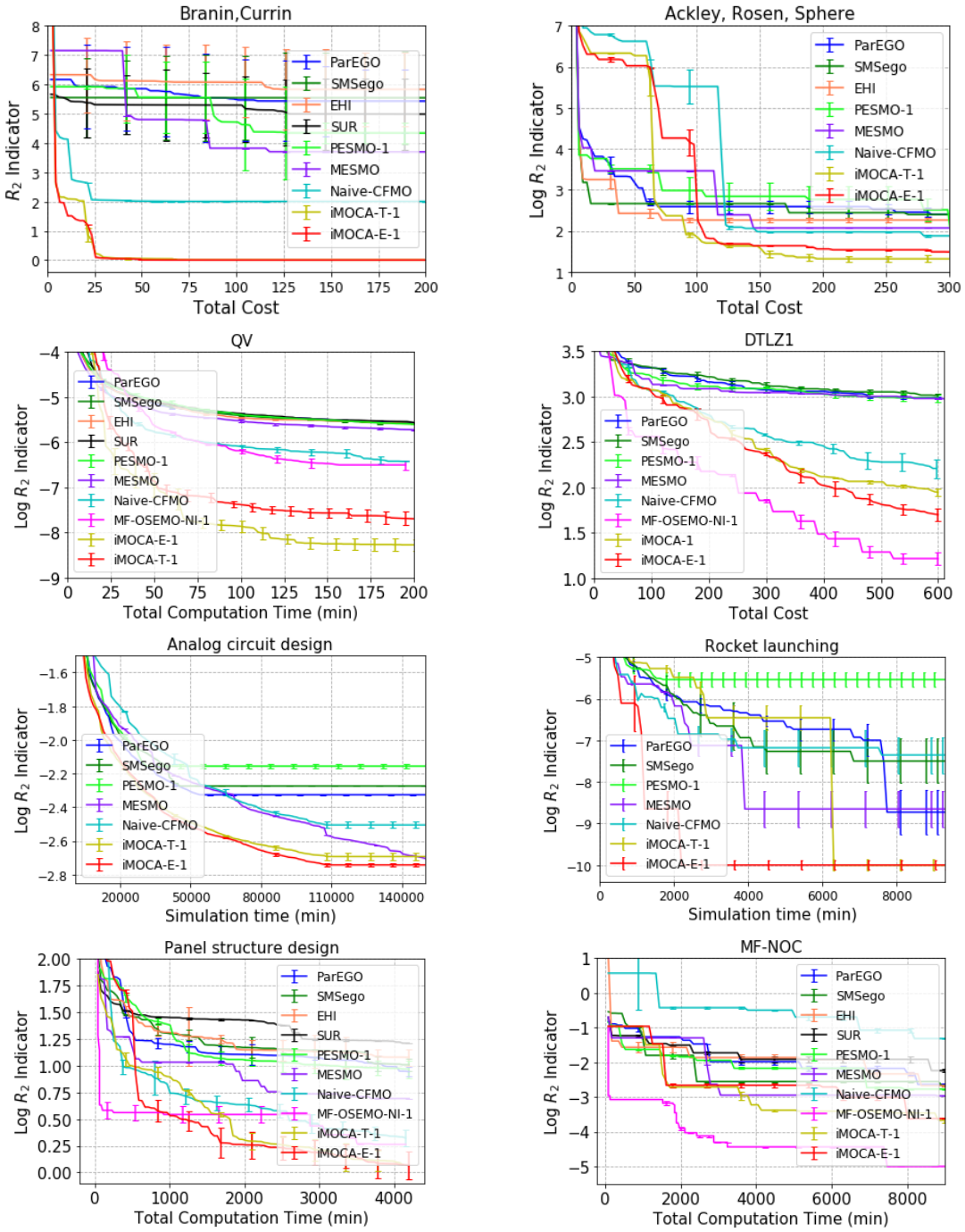

Figure 8.3: Results of iMOCA and the baselines algorithms on synthetic benchmarks and real-world problems. The R_2 metric is presented against the total resource cost of function evaluations.

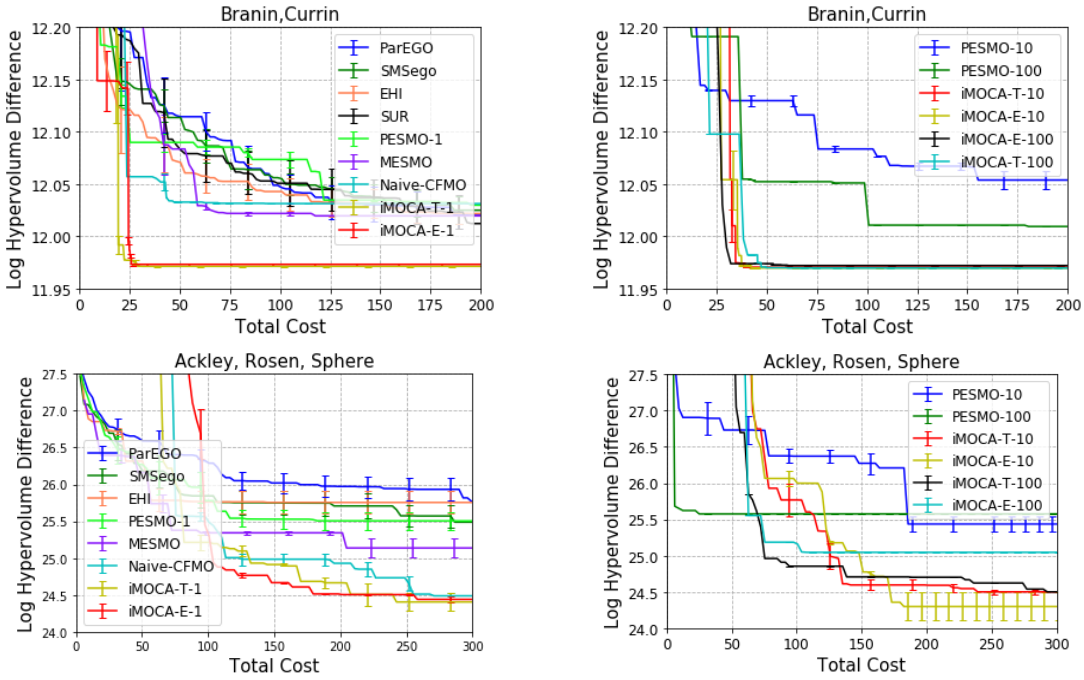

Figure 8.4: Results of synthetic benchmarks showing the effect of varying the number of Monte-Carlo samples for iMOCA, MESMO, and PESMO. The hypervolume difference is shown against the total resource cost of function evaluations.

CHAPTER NINE

BUDGET-AWARE MULTI-OBJECTIVE BO

In this Chapter, we address the problem of non-myopic multi-objective Bayesian optimization. Non-myopic algorithms are particularly well-suited for small resource budgets as they take a planning perspective and reason about different experimental plans in a look-ahead manner. We start by proposing a solution to the budgeted non-myopic Bayesian optimization. Next, we show how to incorporate side information from the hyper-parameter optimization (HPO) problem to build a planning approach for iterative machine learning models.

We propose a novel approach referred to as **B**udget-**A**ware **P**lanning for **I**terative Learners (BAPI) to solve HPO problems under a constrained cost budget. BAPI is an efficient non-myopic Bayesian optimization solution that accounts for the budget and leverages the prior knowledge about the objective function and cost function to select better configurations and to take more informed decisions during the evaluation (training). Subsequently, we propose an extension of this solution for the generic non-myopic multi-objective optimization problem. We evaluate our algorithm on diverse HPO benchmarks for iterative learners.

9.1 Budgeted Non-Myopic Bayesian optimization

Sequential experimental design with a finite number of function evaluations (finite-horizon) or a finite budget, where an agent adaptively designs a pre-specified number of experiments has been addressed in several ways. The optimal policy for this problem can be formulated as a dynamic program, which balances the inherent trade-off between exploitation and exploration. However, this optimal policy has been shown to be intractable even for simple problems with a single. Common approximations for the single objective problem include rollout and Monte-Carlo tree search. Forcibly limiting the horizon to one is also known as *myopic* Bayesian optimization, where the algorithm selects an input that maximizes some utility function at only the next iteration. All the methods discussed prior to this chapter are considered myopic Bayesian optimization. In this section, we discuss a non-myopic BO approximation to solve this problem.

9.1.1 Problem Setup

Consider the problem of sequentially optimizing a blackbox objective function f over the input space $\mathfrak{X}$ where the evaluation of each candidate input $\mathbf{x} \in \mathfrak{X}$ is expensive and where the cost c of each input is unknown before the evaluation. In the context of HPO for iterative machine learning models, each input candidate $\mathbf{z} := [\mathbf{x}, t]$, where $\mathbf{x}$ represents model/pipeline hyperparameters and $t \in \mathcal{T}=[1 \dots t_{max}]$ is the number

of training epochs. We let the objective function $f(\mathbf{x}, t)$ be defined as the accuracy [1] and the unknown cost $c(\mathbf{x}, t)$ be defined as the training time. The objective is to identify the maximum of f in a number of queries whose cumulative cost is bounded by a total budget B_T. Let $\mathcal{Z} := \mathcal{X} \times \mathcal{T}$, our problem can be stated as

$$\max_{Z \in P(\mathcal{Z})} \max_{\mathbf{z} \in Z} f(\mathbf{z}), \qquad \text{s.t.} \sum_{\mathbf{z} \in Z} c(\mathbf{z}) \leq B_T \tag{9.1}$$

where $P(\mathcal{Z})$ denotes the power set of $\mathcal{Z}$ and $Z = \{\mathbf{z}_1 \dots \mathbf{z}_h\}$ is the sequence of inputs evaluated until the budget B_T is exhausted. In other words, the optimal design $\mathbf{z}^*$ is defined as $\mathbf{z}^* \leftarrow \arg\max_{\mathcal{Z}} f(\mathbf{z})$ with $\mathbf{z}^* \in Z$ and $\sum_{\mathbf{z} \in Z} c(\mathbf{z}) \leq B_T$. The problem in Equation (9.1) is solved using a non-myopic policy, where at each iteration, the algorithm accounts for the sequence of inputs that can be evaluated within the remaining budget, i.e., the horizon h is *adaptive*. We define the non-myopic setting later in this section, which is similar to the setting considered in Lee et al. [131] and Astudillo et al. [13].

We focus on problem settings where the objective function is monotonic in the number of epochs t. Specifically, for a fixed hyperparameter $\mathbf{x}$, $f(\mathbf{x}, t) \leq f(\mathbf{x}, t')$ when $t \leq t'$. This is a reasonable assumption for iterative learners. Even if monotonicity does not hold over all training epochs, keeping track of the best model over training epochs is a standard practice [39].

[1] Any bounded metric can be used (e.g, loss, some cases of reward for RL models etc.)

Gaussian Processes GPs are characterized by a mean function m and a covariance or kernel function κ. If a function f is sampled from GP(m, κ), then $f(\mathbf{x}, t)$ is distributed normally $\mathcal{N}(m(\mathbf{x}, t), \kappa([\mathbf{x}, t], [\mathbf{x}, t]))$ for a finite set of inputs from $[\mathbf{x}, t] \in \mathcal{X} \times \mathcal{T}$. The predictive mean and uncertainty for a GP for a new input $\mathbf{z}_* \in \mathcal{Z}$ is defined as:

$$\mu(\mathbf{z}_*) = \kappa_{\mathbf{z}_*,Z}[\kappa_{Z,Z} + \sigma^2 I]^{-1}(Y - m(Z)) + m(\mathbf{z}_*)$$

$$\sigma^2(\mathbf{z}_*) = \kappa_{\mathbf{z}_*,\mathbf{z}_*} - \kappa_{\mathbf{z}_*,Z}[\kappa_{Z,Z} + \sigma^2 I]^{-1}\kappa_{Z,\mathbf{z}_*}$$

where $\kappa_{\mathbf{z}_*,\mathbf{z}_*} = \kappa(\mathbf{z}_*, \mathbf{z}_*)$, $\kappa_{Z,Z} = \kappa(Z, Z)$, $\kappa_{\mathbf{z}_*,Z} = [\kappa(\mathbf{z}_*, \mathbf{z}_i)]_{\forall i}$, Z is the set of evaluated inputs and Y is their corresponding function values. A typical choice to model blackbox functions with a temporal component is using a product kernel $\kappa([\mathbf{x}, t], [\mathbf{x}', t']) = \kappa_{\mathbf{x}}(\mathbf{x}, \mathbf{x}') \times \kappa_t(t, t')$. $\kappa_{\mathbf{x}}$, defined over the input space $\mathbf{x} \in \mathcal{X}$, is often selected to be an RBF or a Matern kernel. For the temporal component κ_t, previous work for GPs over iterative learners [178] proposed an exponential decay (ED) kernel, defined as $\kappa_t(t, t') = \beta^\alpha/(t + t' + \beta)^\alpha$, to model decreasing covariance with increasing time. However, this kernel does not guarantee that the predictive mean of GP or sampled functions would necessarily follow a desired monotonic shape. Nguyen et al. [146] argued that the use of ED is not appropriate for reinforcement learning models where the reward might follow a logistic shape and proposed the use of an RBF kernel for κ_t.

Bayesian Optimization And Non-Myopic Query Policies BO often selects the next input to evaluate that maximizes the acquisition function. Two examples of acquisition functions are expected improvement (EI) and upper confidence bound (UCB) and both are considered myopic since they only aim to maximize the function for the next query without accounting for the future queries.

We review some standard facts for optimal sequential decision-making [153, 99]. Consider having collected a set of i responses D_i and let u denote the utility of D_i for maximizing $f(\mathbf{z}) = y$, i.e., $u(D_i) = \max_{(\mathbf{z},y)\in D_i} y$. The marginal gain in utility of the query $\mathbf{z}$ w.r.t. D_i is expressed as:

$$u(y|\mathbf{z}, D_i) = u(D_i \cup (\mathbf{z}, y)) - u(D_i) \tag{9.2}$$

The one-step expected marginal gain is equivalent to the expected improvement (EI) strategy [139]:

$$\mathcal{U}_1(\mathbf{z}|D_i) = \mathbb{E}_y[u(D_i \cup (\mathbf{z}, y)) - u(D_i)|\mathbf{z}, D_i] \tag{9.3}$$

Now, consider the case where r steps are remaining. The r-steps expected marginal gain can be expressed through the Bellman recursion as [99]:

$$\mathcal{U}_r(\mathbf{z}|D_i) = \mathbb{E}_y[u(y|\mathbf{z}, D_i)] + \mathbb{E}_y[\max_{\mathbf{z}'} \mathcal{U}_{r-1}(\mathbf{z}'|D_i \cup (\mathbf{z}, y))] \tag{9.4}$$

Maximizing (9.4) w.r.t. $\mathbf{z}$ results in the optimal r-steps "lookahead" . Being a se-

quence of r nested integrals of maximizations, optimizing (9.4) is intractable for even small r.

Lower Bound To Optimal Policy The previous discussion focused on the optimality of selecting single queries. We review recent work by [99] which makes a connection between single selection and batch selection of size r, $Z = \{\mathbf{z}_1 \dots \mathbf{z}_r\}$. Assuming parallel evaluation, the optimal set of selected points Z^* maximizes the expected marginal utility of the new associated evaluations $Y = \{y_1 \dots y_r\}$:

$$Z^* = \arg\max_{Z \in \mathcal{Z}} \mathcal{U}(Z|D_i) \text{ with } \quad \mathcal{U}(Z|D_i) = \mathbb{E}_Y[u(Y|Z, D_i)] \tag{9.5}$$

Jiang et al. [99] showed that choosing a query $\mathbf{z}^* \in Z^*$ is equivalent to solving $\arg\max_{\mathbf{z}} V(\mathbf{z}|D)$ where

$$V(\mathbf{z}|D_i) = \mathbb{E}_y[u(y|\mathbf{z}, D_i)] + \max_{Z':|Z'|=r-1} \mathbb{E}_y[\mathcal{U}(Z'|D_i \cup (\mathbf{z}, y))] \tag{9.6}$$

and that the second term of (9.6) is a lower bound to the second term in (9.4):

$$\max_{Z':|Z'|=r-1} \mathbb{E}_y[\mathcal{U}(Z'|D_i \cup (\mathbf{z}, y))] \leq \mathbb{E}_y[\max_{\mathbf{z}'} \mathcal{U}_{r-1}(\mathbf{z}'|D_i \cup (\mathbf{z}, y))] \tag{9.7}$$

Given this observation, Jiang et al. [99] proposed approximating the optimal policy (9.4) by optimizing its lower bound (9.6) which is equivalent to optimizing the batch

EI known as q-EI. Jiang et al. [99] proposed using joint q-EI which is budget-unaware and scales poorly with increased dimensions [190].

9.1.2 Related Work

Non-Myopic Policies While there were early attempts at non-myopic selection for length-two horizons [e.g., 153], most work on proposing practical methods for longer horizons is very recent. Wu and Frazier [191] developed gradient estimates for two-step EI admitting gradient-based search for the optimal two-step selection. Lam et al. [126] utilized a Markov decision process (MDP) formalism and performed roll-outs with a predefined base policy to estimate the value function. GLASSES [77] approximates the solution for the optimal non-myopic selection by a combination of approximate integration given future selections and approximating the future selections by a diversity-promoting batch selection procedure from González et al. [76]).

Closely related to our work is BINOCULARS Jiang et al. [99] which was discussed in Section 9.1.1. The major differences to our proposed BAPI approach are that (a) BINOCULARS uses *joint* batch expected improvement q-EI Wang et al. [184] while we use the sequential greedy selection Wilson et al. [190], (b) BINOCULARS uses a fixed horizon that is budget agnostic while we use a budget-adaptive horizon, and (c) BINOCULARS does not take cost into account when returning its non-myopic selected query while our method does factor cost in. Lee et al. [131] consider the general cost-aware setting and frame the problem as a constrained MDP. Their method

approximately solves the intractable problem by performing rollouts of a base policy that does not adapt the horizon to the budget. Moreover, its base policy is normalized by cost leading to a bias towards low-cost queries. Building upon the efficient one-shot multi-step tree approach of Jiang et al. [100], Astudillo et al. [13] introduces cost-modeling and develops a budget-aware method. However, this method's adaptation to the horizon is post-hoc in the sense that the horizon has to be fixed in advance and cannot be adaptive to the remaining budget due to utility function formulation and optimization. This leads to an unnecessary higher dimensional optimization and a manual zero-padding technique to handle cases where the selected horizon violates the remaining budget. Our proposed solution BAPI executes a non-myopic query selection policy by wrapping standard BO in a layer of budget-aware planning and proposing a principled approach for adapting the horizon to the amount of remaining budget. The multi-objective setting have not been previously addressed in the context of non-myopic Bayesian optimization.

HPO/BO For Iterative Learning [64] proposed learning curve prediction in order to allow early termination of non-promising candidates. This approach utilizes approximate Bayesian inference w.r.t. a pre-defined finite set of learning curve models to perform extrapolation to a fixed horizon. [115] built on this method by showing Bayesian neural networks could be used for learning curve prediction. [178] proposed a hierarchical GP model for HP tuning that includes learning curve prediction upon

which decisions for exploration (freeze current and test new candidate) and exploitation (thaw current and continue learning) are based. More recently, [39] proposed an optimal stopping procedure for increasing the sample efficiency of BO and showed competitive performance with [64] for iterative learners. While this procedure obtains theoretical guarantees, it must generate a sample of large size from the GP in order to make a reliable decision. Moreover, solving for the stopping rule requires an approximate backward induction technique after each epoch. Our proposal for early termination is conceptually simpler and far less computationally demanding.

The method of [135] considers a finite set of learners modeled with freeze/thaw [178] selecting between exploration and exploitation with a heuristic ϵ-greedy rule. [193] extend the knowledge gradient acquisition function [72] to trace-valued observations that occur in multi-fidelity applications. BOIL [146] also considers trace-valued observations but compresses the trace via learned weighted sum as well as adding carefully-chosen intermediate trace values as observations. The setting for BOIL includes reinforcement learning problems with reward functions taking non-exponentially decaying shapes. Our BAPI approach uses a product kernel to jointly model correlations between HPs and epochs within the iterative training procedure. Kandasamy et al. [109] developed BOCA, an extension of UCB to the general multi-fidelity BO setting.

We note that there exist orthogonal approaches that focus on the ability to ex-

trapolate responses based on smaller datasets where the cost is varied based on the size or fraction of the dataset used for training. These methods propose algorithms based on multi-task BO [177, 116] and importance sampling [7].

Bandit Algorithms for HPO Given that the objective in (9.1) is equivalent to optimizing for simple regret, there is a large amount of relevant work within the multi-armed bandits literature. Audibert et al. [14] developed the upper confidence bound exploration (UCB-E) policy, for the best arm identification (BAI) in the budgeted setting by providing conditions under which simple regret decays exponentially with increasing budget. Hoffman et al. [86] considered linear bandits and proposed the BayesGap algorithm which is an exploration policy within budget constraints. Later, Jamieson and Talwalkar [92] analyzed successive halving as an instance of non-stochastic multi-armed bandits in the setting where the budget is greater than the number of learners. HyperBand [133] is an implementation of successive halving running this algorithm in multiple successive rounds and is a very general algorithm for HPO including non-iterative learners. Most recently, BOHB [69] modified HyperBand by utilizing BO within the successive halving procedure which guides the selection process for the learners that will be trained for longer budgets.

All the above-discussed approaches were proposed in the context of optimizing a problem with a single objective function, the accuracy of hyper-parameter configuration. Recently [163] proposed an extension of Hyperband to the multi-objective

setting. Our proposed approach leverages side information both for the accuracy and cost functions and can be incorporated with any multi-objective algorithm including MESMO and USeMO.

9.1.3 Proposed Approach

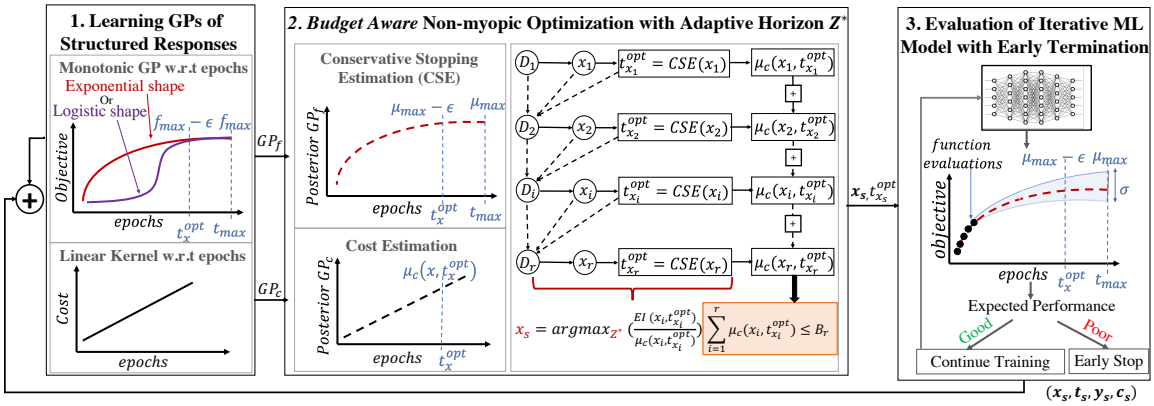

Figure 9.1: Overview of BAPI algorithm illustrating its three key components explained in section 9.1.3

In this section, we start by providing a high-level overview of the proposed BAPI algorithm and briefly explain its key components. Next, we provide complete details of each component. First, we describe how to perform an efficient budget-aware non-myopic search. Second, we explain our approach to model structured response for iterative learning which can be used to estimate conservative stopping for increased resource efficiency. Finally, after describing an alternative kernel for the cost model, we provide the full BAPI approach with all its components coherently put together.

Overview Of BAPI Algorithm

Let $\mathcal{GP}_f$ and $\mathcal{GP}_c$ be the surrogate probabilistic models learned given a set of observed data points D_i of the objective function f and the cost function c respectively. Let μ_c and σ_c^2 be the predictive mean and variance of $\mathcal{GP}_c$ and μ and σ^2 be the predictive mean and variance of $\mathcal{GP}_f$.

As shown in Figure 9.1, BAPI is a sequential algorithm with three key components listed below:

1. Learning Surrogate Models We build two surrogate models $\mathcal{GP}_f$ for the objective function and $\mathcal{GP}_c$ for the cost function by fitting independent GPs using queries evaluated in the past. We enforce shape constraint on the posterior of $\mathcal{GP}_f$ with respect to t (epoch number) to incorporate prior knowledge about the monotonicity of the function. We use a special kernel for $\mathcal{GP}_c$ to leverage our knowledge about the variability of the cost across different HPs and its linearity with respect to t.

2. Budget Aware Non-Myopic Optimization With Adaptive Horizon We perform non-myopic optimization to approximate the optimal lookahead horizon Z^* defined as the potential sequence of inputs that can be evaluated until their conservative stopping t_x^{opt} without violating the remaining budget B_r: $Z^* = \{(\mathbf{x}_1, t_{\mathbf{x}_1}^{opt}) \dots (\mathbf{x}_r, t_{\mathbf{x}_r}^{opt})\}$ such that $\sum_{\mathbf{z} \in Z^*} c(\mathbf{z}) \approx \sum_{i=1}^{r} \mu_c(\mathbf{x}_i, t_{\mathbf{x}_i}^{opt}) \leq B_r$. While constructing the horizon Z^*, each input $\mathbf{x}_i$ is selected based on its expected improvement. The associated conser-

vative stopping $t^{opt}_{\mathbf{x}_i}$ and cost $\mu_c(\mathbf{x}_i, t^{opt}_{\mathbf{x}_i})$, are estimated upon the input selection.

3. Evaluation With Early Termination From Z^* obtained from the second step, we select the input with highest expected improvement per unit cost *at its estimated conservative stopping* for evaluation. After training the model for a fraction of the maximum number of epochs, we re-estimate the performance of the input at its conservative stopping epoch. We early terminate the training if the expected performance is poor with high certainty.

Budget-Aware BO with Adaptive Horizon

Approximating the non-myopic optimization with the batch expected improvement q-EI, where the batch size q is equal to the horizon of the lookahead optimization r, is an efficient approach [99]. However, the *joint* q-EI via reparametrization trick and Monte Carlo sampling proposed in Wang et al. [184] and used in Jiang et al. [99] requires the size of the batch to be fixed and solves a *joint one-shot* optimization problem of $(d \times q)$ dimensions.

Challenges 1) In the context of budgeted non-myopic optimization, the horizon of remaining queries r is unknown: it depends on remaining budget B_r and expected costs of horizon queries $\mathbf{z}_i, i \in \{1 \dots r\}$. An efficient method should allow the horizon to be adaptive to the budget. Therefore, the *joint* q-EI is not a suitable solution. 2) Given an optimization problem with a reasonable medium-size dimension and a medium-length horizon, the dimensionality of the joint optimization problem can

significantly increase. Wilson et al. [190] showed that the performance of the joint q-EI deteriorates for large optimization dimension.

Proposed Alternative To overcome the above two challenges, we propose to employ the sequential greedy q-EI via reparametrization trick and Monte Carlo sampling proposed in Wilson et al. [190]. Wilson et al. [190] showed that q-EI is a submodular acquisition function, which guarantees a near-optimal maximization via a sequential greedy approach. This incremental version of the acquisition function has several distinct advantages over the joint one: 1) It is amenable to an adaptive horizon, where we can stop adding points to the batch based on the remaining budget. 2) It is more efficient and produces better performance when the value of $d \times q$ is high [190]. After the batch approximation returns a sequence (horizon) of inputs, we select one input to query its expensive function evaluation. We discuss an input selection strategy, that is relevant to iterative machine learning models optimization, in section 9.1.3. Note that our approximation can clearly extend to the use of any other batch acquisition function that satisfies the submodularity condition and have a sequential greedy approach, namely the q-UCB and q-PI [190].

It is important to highlight that in practice, our approach can naturally extend to parallel BO evaluation (batch setting). The user can choose more than one point from the approximated horizon and evaluate them in parallel as long as the horizon length is fairly larger than the number of selected points for evaluation. Even though in this

work we focus on the sequential setting, we enable this option in our implementation. We provide a summary of the budget-aware BO approach in Algorithm 11.

Algorithm 11 Budget Aware Non-myopic BO

Input: $\mathcal{Z}$, $f(\mathbf{z}), c(\mathbf{z})$, models $\mathcal{GP}_f, GP_c$, utility function $u(y|\mathbf{z}, D)$, a total budget B_T
Output: $D, \mathbf{z}^*, f(\mathbf{z}^*)$

1: Initialize the remaining budget $B_r \leftarrow B_T$
2: **while** $B_r \geq 0$: **do**
3: Approximate the optimal horizon via adaptive optimal batch computation Z^* of size r such that $\sum_{i=0}^{r} \mu_c(\mathbf{z}_i) \leq B_r$
4: Select a candidate input $\mathbf{z}^* \in Z^*$ and observe its evaluation $f(\mathbf{z}^*) = y^*$ and cost $c(\mathbf{z}^*) = y_c^*$
5: Update the remaining budget $B_r \leftarrow B_r - c(\mathbf{z}^*)$
6: Update data $D = D \cup \{(\mathbf{x}^*, y^*, y_c^*)\}$
7: **end while**

Bayesian Optimization Over Iterative Learners with Structured Response

In this section, we describe our proposed approaches to leverage prior knowledge about the structure of the responses, namely, the monotonicity and shape of the function f and the linearity of the cost c.

We propose to use a GP with monotonicity constraint over the t variable to model the function f. Recent work [2] proposed an efficient approach to introduce linear operator inequality constraints to GPs. Let f be the function modeled by a GP and $\mathcal{L}$ be a linear operator. The proposed approach enables the posterior prediction to account for inequality constraints defined as $a(\mathbf{z}) \leq \mathcal{L}f(\mathbf{z}) \leq b(\mathbf{z})$. The derivative operator is a linear operator. Hence, to apply monotonicity, this condition can be seen as the partial derivative of the model of f with respect to t is positive. For this condition to hold, Agrell [2] proposed to define a set of *virtual observation locations*

$Z^v = \{\mathbf{z_1}^v, \ldots, \mathbf{z_s}^v\}$ where the condition is guaranteed to be satisfied.

The posterior predictive distribution of the monotonic GP is $\mathbf{f}^*|Y, C$, which is the distribution of $\mathbf{f}^* = f(\mathbf{z}_*)$ for some new inputs $\mathbf{z}_* = [\mathbf{x}_*, t_*]$, conditioned on the observed data Y and the constraint C defined as $a(Z^v) \leq \mathcal{L}f(Z^v) \leq b(Z^v)$. The final derivation of the predictive distribution, provided by Agrell [2], is defined as follows:

$$\mathbf{f}^*|Y, C \sim \mathcal{N}(\mu^* + A(\mathbf{C} - \mathcal{L}\mu^v) + B(Y - \mu), \Sigma)$$

$$\mathbf{C} = \widetilde{C}|Y, C \sim \mathcal{TN}(\mathcal{L}\mu^v + A_1(Y - \mu), B_1, a(Z^v), b(Z^v))$$

where $\mathcal{TN}(\cdot, \cdot, a, b)$ is the truncated Gaussian $\mathcal{N}(\cdot, \cdot)$ conditioned on the hyper-rectangle $[a_1, b_1] \times \cdots \times [a_k, b_k]$, $\mu^v = m(Z^v), \mu^* = m(\mathbf{z}_*), \mu = m(Z)$. The definition of the matrices A, B, A_1, B_1 and Σ can be found in Appendix D.1. The computation of the posterior of the monotonic GP requires the definition of derivatives of the kernel function. In this work we consider monotonicity with respect to one dimension t. Therefore, we need the first order derivatives.

In cases where the function is known to be exponentially decaying (e.g., neural network training), the kernel over dimension t should be defined as an ED kernel. However, in cases where the shape of learning curve is monotonic but not necessarily exponentially decaying (e.g., cumulative and average reward for RL models), an RBF kernel with monotonicity over dimension t should be used. Leveraging monotonicity in the modeling allows flexibility and the generalization of our approach for several

types of ILs.

The specification of the location of virtual observations Z^v can have an important effect on the efficiency and scalability of the monotonic Gaussian process. Agrell [2] proposed to have a suboptimization problem to find the optimal location. The idea is to iteratively place virtual observation locations where the probability that the constraint holds is low. However, this optimization becomes suboptimal when the dimension of the problem grows. Given that our function is monotonic with respect to only one dimension, we chose to define linearly spaced locations with respect to dimension t. The number of points is defined based on the kernel:

- ED kernel: The virtual observations locations will mainly enforce the direction of the monotonicity, therefore adding only two locations is sufficient.
- RBF Kernel: The number and location of virtual observations depends on the smoothness (lengthscale) of the kernel. The distance between every two virtual observations should be smaller than the lengthscale in order to maintain the monotonicity and avoid any fluctuations.

We provide the derivatives for both kernels in Appendix D.1. We additionally provide insights about the efficient posterior computation of the monotonic GP. For more details, we refer the reader to Agrell [2].

Conservative Stopping Estimation Previously proposed BO approaches for HPO consider a maximum number of epoch t_{max} at which the objective function will reach

its best value. However, in practice, different HPs do not need to necessarily run to the maximum number of epochs to reach their optimal value as the objective stops improving (reaches a plateau) Kaplan et al. [110]. Therefore, running them for longer epochs can be a waste of limited resource budget with diminishing returns. Existing work proposed early stopping of HPs based on their performance compared to previously evaluated data points [133, 39, 178] or based on the expected improvement per unit cost [146] which leads to the selection of very low number of epoch due to the high cost of t_{max}. We propose to define a conservative stopping $t_{\mathbf{x}}^{opt}$ for each HP $\mathbf{x}$ as the smallest number of epoch needed to reach the best function value at $\mathbf{x}$. Our approach enables the estimation of when a learner becomes ϵ-close to its asymptotic value. To the best of our knowledge, no previous work used the estimation of the function values at another location to reason about the HP selection and optimal early termination before reaching that epoch. The problem of estimating $t_{\mathbf{x}}^{opt}$ for a HP $\mathbf{x}$ based on the GP posterior is defined as below and efficiently solved using binary search.

$$t_{\mathbf{x}}^{opt} \leftarrow \arg min_{t \in [t_{min}, t_{max}]}\ t \tag{9.8}$$
$$s.t\ \mu(\mathbf{x}, t_{max}) - \mu(\mathbf{x}, t) \leq \epsilon$$

Cost Modeling The cost prediction is an important component in our algorithm. Therefore, it is important to have an accurate and informative model for the cost. We

propose to model the cost by an independent Gaussian process $\mathcal{GP}_c$ that captures two important characteristics: 1) The cost of the training of different HPs for the same number of iterations t can vary significantly. 2) The cost of training of a fixed HP $\mathbf{x}$ increases linearly with the number epochs t. We propose to use the product kernel $\kappa_c([\mathbf{x}, t], [\mathbf{x}', t']) = \kappa_{c_x}(\mathbf{x}, \mathbf{x}') \times \kappa_{c_t}(t, t')$, where κ_{c_x} is an RBF kernel over $\mathbf{x}$ and κ_{c_t} is a linear kernel over t. Note that previous work assumes the cost is same for different HP $\mathbf{x}$ and linear with respect to t. This might lead to an inaccurate estimation of the cost especially if some of the dimensions of $\mathbf{x}$ represent architectural variables (e.g number of layers, number of hidden nodes etc.)

Budget-Aware Planning For Iterative Learners (BAPI)

In this section, we describe the overall budget-aware non-myopic BO algorithm for HPO of iterative learners. The main idea is to use the reparametrized iterative greedy q-EI proposed in Wilson et al. [190] to approximate the optimal sequence of selections with respect to the available budget. q-EI will have an adaptive batch size with budget exhaustion as a stopping criteria. We propose to adaptively add inputs to the horizon based on their expected improvement at their conservative stopping iteration without normalizing the utility function by the cost during the optimization. The details of execution can be found in the Non-Myopic Optimization (NMO) function described in Algorithm 14. This function returns a set of inputs representing the optimal horizon Z^*.

Input Selection From Horizon Given the set of inputs Z^*, how to select the next input to evaluate? We propose to select the input with the highest immediate expected reward per unit cost *at its conservative stopping iteration*. We note here that this is *different* from optimizing the utility function per unit cost and the issue of selecting low non-informative number of iterations would not arise. In this case, the number of iterations is already fixed to an optimal high value for each input $\mathbf{x}^*$.

Early Termination After selecting the next candidate HP to evaluate, the function evaluation will return a y_t value after each epoch. Based on the function values of the initial p epochs, we can re-estimate the final performance of $\mathbf{x}$ and its new conservative stopping $t_{\mathbf{x}}^{opt_n}$. The algorithm makes a decision to continue model training with the current HP or early-stop it. If both 1. $\mu(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq y_{best}$, and 2. $\sigma(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq \tau\sigma(\mathbf{x}, t)$, then model training will be early stopped in epoch t, otherwise, the conditions will be verified again after running another set of p epochs or when it reaches the estimated $t_{\mathbf{x}}^{opt_n}$, whichever happens earlier. The first condition will recommend stopping the training if the predicted function value at $t_{\mathbf{x}}^{opt_n}$ will not be higher than the current best value achieved across all evaluated HP. The second condition recommends the early stopping only if the uncertainty of the model about the predicted function value at the estimated conservative stopping is no more than a factor $\tau \geq 1$ of the uncertainty of the model about the last evaluated epoch. In another word, condition 2 will prevent the early stopping if the model is not certain enough about its prediction of the

function value at $t_{\mathbf{x}}^{opt_n}$. Algorithm 15 summarizes evaluation with early termination.

Algorithm 12 BAPI

Input: $\mathcal{X}$; $f(\mathbf{x}, t)$;$c(\mathbf{x}, t)$; t_{max}; B_T
Output:$\mathbf{x}^*, t_{\mathbf{x}^*}^{opt}, f(\mathbf{x}^*, t_{\mathbf{x}^*}^{opt})$

Initialize with N_0 initial points
Fit the models: $\mathcal{GP}_f, GP_c$
$B_r \leftarrow B_T - \sum_{i=0}^{N_0} c(\mathbf{x}_i, t_{\mathbf{x}_i})$
while $B_r > 0$ **do**
  # Find the budget constrained horizon and their corresponding conservative stopping
  $Z^* : \{(\mathbf{x}_1, t_{\mathbf{x}_1}^{opt}) \cdots (\mathbf{x}_r, t_{\mathbf{x}_r}^{opt})\} \leftarrow NMO(GP_f, GP_c, B_r)$
  # Select one point for evaluation
  $\mathbf{x}, t_{\mathbf{x}}^{opt} \leftarrow \arg max_{Z^*} \frac{EI(\mathbf{x}_i, t_{\mathbf{x}_i}^{opt})}{\mu_c(\mathbf{x}_i, t_{\mathbf{x}_i}^{opt})}$
  $\mathbf{y}, \mathbf{y}_c \leftarrow Evaluate(f(\mathbf{x}, t_{\mathbf{x}}^{opt}))$
  Aggregate data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, \mathbf{y}, \mathbf{y}_c)\}$
  Update Models: $\mathcal{GP}_f, GP_c$
  $B_r \leftarrow B_r - c(\mathbf{x}, t_{\mathbf{x}})$
end while

Algorithm 13 Conservative Stopping Estimation

$ConservativeStopping(GP_f, \mathbf{x})$

$t_{\mathbf{x}}^{opt} \leftarrow \arg min_{t \in [t_{min}, t_{max}]} t$
  s.t $\mu(\mathbf{x}, t_{max}) - \mu(\mathbf{x}, t) < \epsilon$
Return $t_{\mathbf{x}}^{opt}$

Algorithm 14 Adaptive Horizon q-EI AFO

$NMO(GP_f, GP_c, B_r)$

$Z^* = \{\}$
while $B_r > 0$ **do**
  # Add $\mathbf{x}$ *based on the highest number of epochs*
  $\mathbf{x} \leftarrow \arg max_{\mathbf{x} \in \mathcal{X}} q\text{-EI}(\mathbf{x}, t_{max})$
  # Estimate the conservative stopping for $\mathbf{x}$
  $t_{\mathbf{x}}^{opt} \leftarrow ConservativeStopping(GP_f, \mathbf{x})$
  Deduct estimate cost at $t_{\mathbf{x}}^{opt}$ *from budget*
  $B_r \leftarrow B_r - \mu_c(\mathbf{x}, t_{\mathbf{x}}^{opt})$
  $Z^* = Z^* \cup \{(\mathbf{x}, t_{\mathbf{x}}^{opt}\}$
end while
Return Z^*

Data Points Selection From Learning Curve Iterative machine learning models evaluated with an input configuration $\mathbf{x}^*$ and a number of epochs t^* return a

Algorithm 15 Evaluate Function

$Evaluate(f(\mathbf{x}, t_{\mathbf{x}}^{opt}))$

$t \leftarrow p$
while $t \leq t_{max}$ and $Continue$ **do**
  $\mathbf{y} = f(\mathbf{x}, t)$; $\mathbf{y}_c = c(\mathbf{x}, t)$
  $t_{\mathbf{x}}^{opt_n} = ConservativeStopping(GP_f, \mathbf{x})$
  if $\mu(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq y_{best}$ and $\sigma(\mathbf{x}, t_{\mathbf{x}}^{opt_n}) \leq \tau\sigma(\mathbf{x}, t)$:
    $Continue \leftarrow False$
  else:
    $Continue \leftarrow True$
    $t \leftarrow min(t_{\mathbf{x}}^{opt_n}, t + p)$
end while
Return $\mathbf{y}, \mathbf{y}_c$

vector of t^* function values and a vector of t^* cost values associated with each iteration $t \leq t^*$. Most of existing work, do not utilize these data points and use only the function value at the last epoch. However, leveraging part of this data can help the learning of the monotonic shape of the objective function and result in a more accurate extrapolation. We select, from each curve, the points with the highest model uncertainty(variance) following the approach proposed in Nguyen et al. [146].

Practical Considerations Considering a perfect model of the function, querying a complete lookahead horizon in each iteration would be optimal. However, as pointed out by previous work on non-myopic BO [99, 131, 130, 197], the model is usually uncertain about long-term predictions. Consequently querying a long horizon can hurt the optimization by evaluating misleading points and causing a higher computational cost. Therefore, we follow previous work [13] and set a maximum horizon length as an additional stopping condition to the size of the horizon. Given this mitigation, we expect that the horizon adaptation to the budget to occur depending on the

remaining budget. Additionally, selected points would always be within the limits of the remaining budget.

Cost of a Restarted Hyperparameter In iterative learning, the optimization algorithm might select a configuration that was previously evaluated for a lower number of epochs. However, the cost will always be estimated with respect to the evaluation iteration. For accurate optimization, our algorithm handles this special case by assigning a cost that only reflects the additional epochs to be run. This is accounted for in the non-myopic optimization function, input selection, and budget deduction after function evaluation.

9.1.4 Experiments and Results

In this section, we first provide details about our experimental setup. Next, we evaluate the performance of BAPI approach and compare it to several state-of-the-art baselines.

Baselines. We evaluate state-of-the-art baselines, described in the related work: from cost-aware non-myopic BO literature `BMS-EI`[2] [13], from non-myopic BO BINOCULARS `(BINOC)` [99][3] and `MS-EI` [100][3], from general BO literature `EI` [106], from HPO for iterative learners literature `BOHB` [69][4] and HyperBand `(HB)` [133][4], from multi-fidelity BO for HPO literature `BOIL` [146][5]. Each baseline implementation uses

[2]github.com/RaulAstudillo06/BudgetedBO
[3]github.com/shalijiang/bo/tree/main/enbo
[4]github.com/automl/HpBandSter
[5]github.com/ntienvu/BOIL

settings recommended by the original authors and publicly available code. We also evaluated GLASSES[3] [77] and random search. However, both of them performed always poorly when compared to all other baselines. Therefore, for clarity of the figures, we do no report them. We note that previous work on non-myopic BO do not include HB and BOHB as baselines but given their competitive performance in iterative learning settings, we recommend they become standard in future work in this problem setting.

Experimental Setup All experiments were averaged over 10 runs with different random seeds. The code of our BAPI implementation is publicly available[6]. We considered several state-of-the-art HPO benchmarks: 1) Logistic regression with MNIST dataset; 2) Multi-layer perceptron with Olivetti dataset; 3) Multi-layer perceptron with Covtype dataset ; 4) Fully connected network with MNIST dataset with two different t_{max} setups 5) CNN on image dataset CIFAR10 with two different t_{max} setups; 6) CNN on SVHN dataset with two different t_{max} setups; 7) Resnet on CIFAR100 dataset; 8) A Dueling DQN (DDQN) agent in the CartPole-v0 environment; 9) An Advantage Actor Critic (A2C) agent in the Reacher-v2 environment; and 10) An Advantage Actor-Critic (A2C) agent in the InvertedPendulum-v2 environment. We report the validation error as the evaluation metric for consistency across datasets. We evaluate two different variants of our algorithm: BAPI-4 and BAPI-8, where the maximum horizon is set to 4 and 8 respectively. BAPI-8 was evaluated on two bench-

[6]github.com/belakaria/BAPI

marks (LR with MNIST and MLP with Olivetti) to demonstrate the effect of varying the maximum horizon on the performance. The uncertainty threshold τ is set to 2 for all experiments. The parameter p is set to 20% of the maximum number of epochs for all experiments except for CNN-SVHN, where it is set to 10% due to the high cost of each epoch. We select at most three data points from each learning curve.

Setting ϵ For experiments 1 to 7, ϵ is set to 0.01 (interpreted as at most 1% degradation in accuracy) except for CNN-SVHN, where it is set to 0.005 due the small variation in the validation error. In the case of a loss/reward function where ϵ cannot be easily set (e.g experiments 8 to 10), it is automatically set as the smallest degradation in the function value at t_{max} in the evaluated data: $\epsilon = min\{f(x, t_{max}) - f(x, t)\ \forall (x, t) \in D\}$. In general, ϵ is not required to be fixed. A strategy for updating it, that balances exploration and exploitation (e.g., a wider value and therefore earlier stopping in the beginning), can be set by the practitioner and interfaced with our code easily.

Logistic Regression with MNIST: We train the logistic regression classifier on the MNIST image dataset LeCun et al. [127]. The dataset consists of 70,000 images categorized into 10 classes. We use 80% for training and 10% for validation. We optimize the model over three hyperparameter: the learning rate $\in [10^{-6}, 1]$, the L_2 regularization $\in [0, 1]$ and the batch size $\in [20, 2000]$. We apply a log transformation to the learning rate and batch size. We set the maximum number of epochs to 100.

MLP with Olivetti and Covtype: We train a multi-layer perceptron with two fully connected layers on the Olivetti dataset Samaria and Harter [162] and Covtype dataset. We use 10% of the data for the validation set. We optimize four hyperparameters learning rate $\in [10^{-6}, 1]$, batch size $\in [8, 128]$ for Olivetti and $\in [32, 1024]$ for Covtype , the L_2 regularization $\in [10^{-7}, 10^{-3}]$ and the momentum $\in [0.1, 0.9]$. We apply a log transformation to the learning rate, the batch size and the L_2 regularization. We set the maximum number of epochs to 100. The experiment with Covtype dataset was run on Tesla V100 GPU machine and the experiment on Olivetti was run on a CPU machine with Intel(R) Core(TM) i9-7960X CPU 2.80GHz.

FCNET MNIST: We train a fully connected network with on the MNIST dataset. We use 50,000 images for the training set and 10,000 images for the validation set. We optimize six hyperparameters learning rate $\in [10^{-6}, 0.1]$, batch size $\in [32, 1024]$, , the L_2 regularization $\in [10^{-7}, 10^{-3}]$, the momentum $\in [0.1, 0.9]$, the number of hidden layers $\in [1, 4]$ and the size of hidden layers $\in [100, 1000]$ We apply a log transformation to the learning rate and the batch size. We set evaluate all algorithms on two different setups where in figure 9.3 we report the results with the maximum number of epochs set to $t_{max} = 25$ and in figure 9.4 we report the results with the maximum number of epochs set to $t_{max} = 50$. The total wallclock time budget is extended accordingly. These experiments were run on Tesla V100 GPU machine.

CNN with CIFAR10 and SVHN: We train a CNN model on two image datasets CIFAR10 [122] and the Street View House Numbers (SVHN) [145]. For CIFAR10 we use 40,000 image for the training set and 10,000 for the validations set. For SVHN 63,257 image for the training set and 10,000 for the validations set. We optimize six hyperparameters: the batch size$\in [32, 1024]$, the learning rate$\in [10^{-6}, 0.1]$, the momentum$\in [0.1, 0.9]$, the L_2 regularization$\in [10^{-7}, 10^{-3}]$, the number of convolutional filters$\in [32, 256]$, and the number of dense units $\in [64, 512]$. We apply a log transformation to the learning rate, the batch size. We set evaluate all algorithms on two different setups where in figure 9.2 we report the results with the maximum number of epochs set to $t_{max} = 25$ and in figure 9.4 we report the results with the maximum number of epochs set to $t_{max} = 50$. The total wallclock time budget is extended accordingly. These experiments were run on Tesla P100 GPU machine.

Resnet with CIFAR100: We train a ResNet model on a the image dataset CIFAR100 [122]. We employ 40,000 images for the training set and 10,000 for the validations set. We optimize six hyperparameters: the batch size $\in [32, 512]$, the learning rate $\in [1e-6, 1e-1]$, the momentum $\in [0.1, 0.9]$, the L_2 regularization$\in$ $[1e-7, 1e-3]$, the number of convolutional filters$\in [32, 256]$, and the number of layers $\in [10, 18]$. We report the results with the maximum number of epochs set to $t_{max} = 100$ in Figure 9.3. The total wall-clock time budget is extended accordingly. These experiments were run on a Tesla V100 GPU machine.

DQN CartPole: We train a Dueling DQN (DDQN) [188] agent in the CartPole-v0 environment. We employ the same setting proposed by Nguyen et al. [146]. We optimize two hyperparameters: the discount factor $\in [0.8, 1]$ and the learning rate for the model $\in [1e-6, 0.01]$. We vary the number of episodes from 200 to 500. We map the episodes into epochs with each three episodes equivalent to one epoch, resulting in a maximum number of epochs $t_{max} = 100$. We report the results in Figure 9.4. The total wall-clock time budget is extended accordingly. These experiments were run on a 1 core of a Xeon CPU machine.

A2C Reacher: We train a Advantage Actor Critic (A2C) [138] agent in the Reacher-v2 environment. We employ the same setting proposed by Nguyen et al. [146]. We optimize three hyperparameters: the discount factor $\in [0.8, 1]$, the learning rate for the actor $\in [1e-6, 0.01]$, and the learning rate for the critic $\in [1e-6, 0.01]$. We vary the number of episodes from 200 to 500. We map the episodes into epochs with each three episodes equivalent to one epoch, resulting in a maximum number of epochs $t_{max} = 100$. We report the results in Figure 9.4. The total wall-clock time budget is extended accordingly. These experiments were run on a 1 core of a Xeon CPU machine.

A2C Inverted Pendulum: We train a Advantage Actor Critic (A2C) [138] agent in the InvertedPendulum-v2 environment. We employ the same setting proposed by Nguyen et al. [146]. We optimize three hyperparameters: the discount factor $\in [0.8, 1]$,

the learning rate for the actor $\in [1e-6, 0.01]$, and the learning rate for the critic $\in [1e-6, 0.01]$. We vary the number of episodes from 700 to 1500. We map the episodes into epochs with each eight episodes equivalent to one epoch, resulting in a maximum number of epochs $t_{max} = 100$. We report the results in Figure 9.4. The total wall-clock time budget is extended accordingly. These experiments were run on a 1 core of a Xeon CPU machine.

Results and Discussion Figure 9.2 shows the results (best validation error as a function of wall-clock time) of all methods on four HPO tasks. We make the following observations. **1)** BAPI identifies better candidates with less total cost than the baselines due to its ability to plan selections while accounting for budget and early terminating non-promising candidates. **2)** A longer horizon for BAPI was tested on the relatively cheaper experiments LR MNIST and MLP Olivetti datasets and shows some performance degradation (LR MNIST) and some improvement (MLP Olivetti) suggesting that optimal horizon length is problem-dependent but clearly helpful in some cases. **3)** BMS-EI had an unstable performance and was not able to uncover good candidate in several experiments. We speculate that it is due to the approach being conservative about which points would satisfy the remaining horizon constraint. **4)** HB and BOHB can identify good candidates faster than most algorithms in the beginning, mainly because their strategy forces initial evaluations to be low-epoch trained runs. In the mid-range, their performance slows down, perhaps a consequence

of their exploitation behavior and reliability on successive halving which might limit their extrapolation ability and stop promising candidates very early. Similar analysis has been reported in previous work [39]. With longer search times, BOHB can catch up. However, both BOHB and HB performance degrades significantly in RL settings since successive halving cannot extrapolate accurately when the function might take a sigmoid or logit shape. **5)** BINOCULARS and MS-EI, are slower to uncover promising candidates due to spending more budget in evaluating all selected candidates to the maximum number of epochs. However, they both arrive at a competitive performance towards the end that can be attributed to their planning capabilities. **6)** BOIL is worse than most baselines across all benchmarks. BOIL selects the next candidates by weighting EI by the cost and might suffer from the cost miscalibration pathology. Similar observations were made in Astudillo et al. [13].

We compared our approach to a wide range of baselines on 13 different experiments. All the baselines had inconsistent performance across the different experiments while our algorithm performed fairly well across all of them. The gain was significant in the case of limited budget, which is the desired behavior since a planning approach is more needed when the budget is limited. The gain was less significant in experiments with higher budgets but our approach was still competitive. Therefore, in Table 9.1, we provide the average ranking of each algorithm over all experiments based on their final performance.

We report additional results of our BAPI approach and existing baselines. We report an additional variant of our algorithm that we name BAPI-4-L. We test the case where we do not add additional points from each curve but rather use only the last epoch. We notice that this variant performs competitively and sometimes better than adding additional points to the curve. This opens a discussion about the utility of leveraging additional data points from each curve especially while using the monotonic GP. It is important to note that previous methods built for HPO frequently suggest using additional points. This includes approaches proposed in the papers by Nguyen et al. [146], Dai et al. [39], and Wu et al. [193]. We plan to work on investigating this problem further to develop a sound theoretical understanding of this phenomenon.

In Figure 9.4, we test all algorithms on settings with an extended budget and a higher number of maximum epochs $t_{max} = 50$. We observe that given a sufficiently large budget, most of the baselines converge to statistically comparable results. We notice that HB, in most of the experiments, is able to reduce the validation error in the beginning but does not always converge to good results. However, BOHB performance was remarkably stronger with a higher number of maximum epochs. Increasing the maximum number of epoch enables BOHB to evaluate a larger number of configurations at a low budget and therefore we can see a significant drop in the validation error earlier than all baselines. The results in Figure 9.3 and Figure 9.4

show that BAPI-4 becomes less competitive when the total budget is significantly increased and the maximum number of epochs is higher, but also provides results suggesting that performance loss can be brought back by adjusting pruning behavior in BAPI-4-L.

We report additional results for reinforcement learning experiments optimized with RBF kernel over the number of epochs. Figure 9.4 shows the increasing discounted cumulative reward with discount factor 0.9 as suggested by [39]. The results show that BAPI-4 performs better or similar to the baselines. We observe that HB and BOHB performance degrades significantly with RL experiments most likely because they do not account for the possibility that the learning curve can be flat in the middle. We additionally notice that BAPI-4-L performance is competitive but worse than BAPI-4. One candidate reason for this behavior is due to the use of the RBF kernel, where adding intermediate points from the curve can be more crucial to avoid fluctuations.

Algorithm	BAPI	HB	BOHB	EI	MS-EI	BINOC	BOIL	BMS-EI
Average ranking	2.9 ±0.51	6.1±0.51	3.6±0.67	4.9±0.53	3.4±0.50	4.3±0.38	6.3±0.63	4.2±0.73

Table 9.1: Average ranking of BAPI and baseline methods across all experiments.

Ablation: GP without enforced monotonicity We provide an ablation study where we run our algorithm using a GP without enforced monotonicity to show the benefit of using a monotonic GP. The first figure illustrates differences in learning

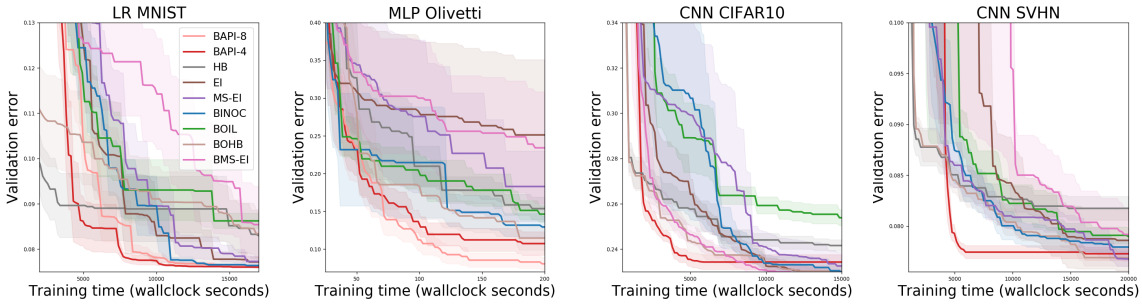

Figure 9.2: Results of validation error ± standard error for different baselines and our proposed approach on multiple iterative learners against training budget.

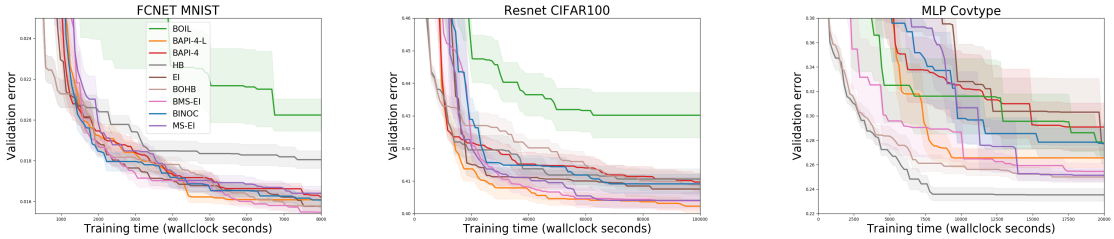

Figure 9.3: Results of validation error ± standard error for different baselines and our proposed approach on ResNet with $t_{max} = 100$, FCNET with $t_{max} = 25$ and MLP-Covtype with $t_{max} = 100$

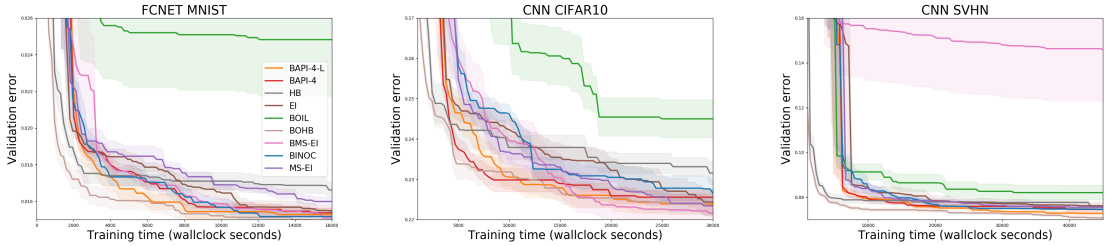

Figure 9.4: Results of validation error ± standard error for different baselines and our proposed approach on FCNET-MNIST, CNN-CIFAR10 and CNN-SVHN with $t_{max} = 50$

curve extrapolation between a **GP with enforced monotonicity and vanilla RBF**

GP. The RBF GP fluctuates further from evaluated points rendering extrapolation

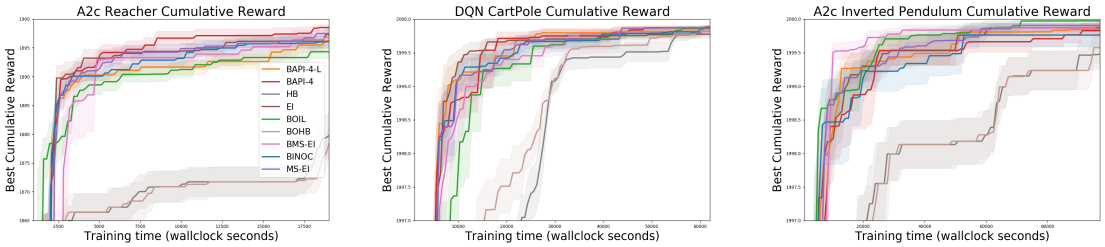

Figure 9.5: Results of Cumulative discounted reward ± standard error for different baselines and our proposed approach on A2C Reacher, DQN Cartpole, and A2C Inverted Pendulum

highly uncertain (as well as inaccurate). Basing an estimation of optimal stopping time on this vanilla model directly affects budgeted HP optimization performance as shown in the **ablation study** displayed in the second and third figures. These show the performance of BAPI with a Non-Monotonic GP (BAPI-4-NM) is inferior to BAPI with monotonic GP. However, BAPI-4-NM shows competitive performance that might be associated with its budget-aware planning strategy.

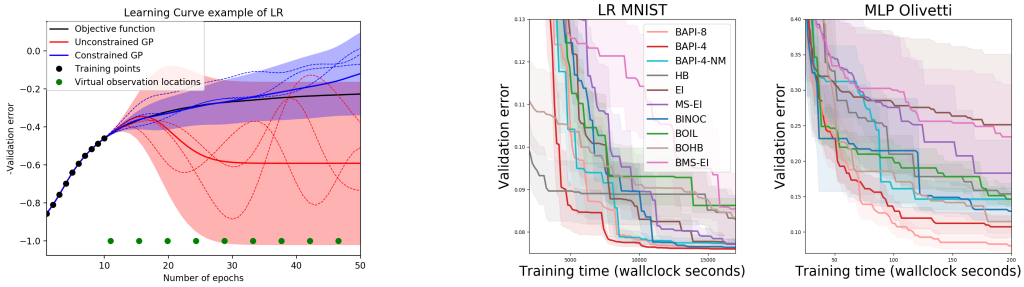

Figure 9.6: Ablation Study

9.2 Non-Myopic Multi-Objective Bayesian optimization

In this section, we provide an approach for non-myopic multi-objective Bayesian Optimization named BINOM: Batch-Informed NOnmyopic Multi-objective optimization. Similar to single-objective sequential decision-making problems, the multi-objective problem can be formulated as a multi-objective Markov Decision Process (MDP). The only solution that has been provided for this problem in the context of Bayesian optimization, is the reduction of the horizon to a length of one, where the problem is solved by evaluating the input that maximizes the utility function at the next iteration only. *To the best of our knowledge, there is no previous work on non-myopic Bayesian optimization in the multi-objective setting.*

Challenges: The key challenges to solve budget-aware MOO problem are:

- Defining a principled criteria for the look-ahead search in the presence of multiple conflicting objective functions.
- Recent solutions for the single-objective optimization problem rely on the Bellman equations. However, these equations do not hold when the output has several *conflicting* objective functions.

9.2.1 Problem Setup

Cost/Budget-agnostic Non-Myopic MOO problem: Consider the problem of sequentially optimizing several black-box objective functions $f_1 \cdots f_K$ over the input

space $\mathcal{X}$ where the evaluation of each candidate input $\mathbf{x} \in \mathcal{X}$ is expensive and the goal is to identify the optimal trade-off between the objective functions within a maximum number of evaluation queries T_{max}. Our problem can be formulated as

$$\max_{X \in P(\mathcal{X})} \max_{\mathbf{x} \in X} f_1(\mathbf{x}) \cdots f_K(\mathbf{x}) \qquad \text{with } |X| = T_{max} \tag{9.9}$$

where $P(\mathcal{X})$ denotes the power set of $\mathcal{X}$ and $X = \{\mathbf{x}_1 \cdots \mathbf{x}_{T_{max}}\}$ is the number of input evaluations when T_{max} is exhausted. The problem in Equation (9.9) is solved using a non-myopic policy, where at each iteration, the algorithm accounts for the sequence of inputs that can be evaluated within the total number of evaluations T_{max}, i.e., the horizon is *fixed* as h=T_{max}.

Cost/Budget-aware Non-Myopic MOO problem: Consider the problem where each input evaluation incurs a different cost c that is unknown before the evaluation. The objective is to identify the optimal trade-offs between the objective functions in a number of queries whose cumulative cost is bounded by a total budget B_T. Our problem can be formulated as

$$\max_{X \in P(\mathcal{X})} \max_{\mathbf{x} \in Z} f_1(\mathbf{x}) \cdots f_K(\mathbf{x}), \qquad \text{s.t. } \sum_{\mathbf{x} \in X} c(\mathbf{x}) \leq B_T \tag{9.10}$$

where $P(\mathcal{X})$ denotes the power set of $\mathcal{X}$ and $X = \{\mathbf{x}_1 \dots \mathbf{x}_h\}$ is the sequence of inputs evaluated until the budget B_T is exhausted.

The problem in Equation (9.10) is solved using a non-myopic policy, where at each iteration, the algorithm accounts for the sequence of inputs that can be evaluated within the remaining budget, i.e., the horizon h is *adaptive*.

9.2.2 BINOM: Batch-Informed Non-Myopic Multi-Objective Optimization

Intuition. Intuitively, there exists a link between the multi-step lookahead problem and batch Bayesian optimization [99, 77]. Batches of inputs are selected in each iteration of the algorithm and evaluated in parallel. When inputs are selected and evaluated sequentially (batch size is one), the selection is more informative because, at each iteration, the algorithm has full information about the selected data points since they are already evaluated. However, in the batch setting, the points within the same batch are selected without any knowledge about their evaluations. Ideally, a good batch selection algorithm relies on the ability of the batch criterion of predicting future steps of the algorithm and would select a batch of points that are similar to what would have been selected sequentially.

In a nutshell, batch BO methods aim to find a set of points in X where the functions should be evaluated in parallel rather than sequentially. So, intuitively, constructing a good batch is the same as computing a good approximation of the lookahead horizon X. Additionally, in the single-objective setting, the optimal batch expected utility is a lower bound of the optimal sequential expected utility as shown in Section 9.1.1 [99].

Using this intuition and by exploiting the analogy to single objective non-myopic approximation, we propose the use of a batch Bayesian optimization approach to approximate the horizon for non-myopic multi-objective Bayesian optimization. We propose a new approach named BINOM: Batch-Informed NOnmyopic Multi-objective optimization.

Cost-agnostic BINOM. To approximate the lookahead horizon, we optimize the batch expected hypervolume acquisition function, also known as $qEHVI$, via joint optimization with the reparameterization trick and Monte Carlo approximation [43]. We set the batch size q to the horizon length T_{max}. In order to select the next input for evaluation, we need to pick an input from the suggested horizon. We considered several options inspired by the solution suggested by Jiang et al. [99] in the single-objective setting:

- Selecting the input with highest expected immediate hypervolume improvement.
- Randomly selecting an input proportional to its expected immediate reward.

Cost/Budget-aware BINOM. To approximate the lookahead horizon, we optimize the batch expected hypervolume improvement acquisition function. In the context of budgeted non-myopic optimization, the horizon of remaining queries r is unknown and would depend on the remaining budget B_r and expected costs of horizon queries $\mathbf{z}_i, i \in \{1 \dots r\}$. An efficient method should allow the horizon to be adaptive to the budget. Therefore, the *joint* $qEHVI$ is not a suitable solution. $qEHVI$ [43]have

been shown to be a submodular acquisition function that guarantees a near-optimal optimization via a sequential greedy approach. This version of the acquisition function is amenable to a budget-adaptive horizon, where we can stop adding inputs to the batch based on the remaining budget. In order to select the next input for evaluation, we need to pick an input from the suggested horizon. We considered several options inspired by the solution suggested by Belakaria et al. [30] in the single-objective setting:

- Selecting the input with the highest expected immediate hypervolume improvement per unit resource cost
- Randomly selecting an input proportional to its expected immediate reward per unit resource cost.

Note: Any batch MOO approach can be applied to approximate the horizon. We consider $qEHVI$ due to its submodularity property that is useful in building an adaptive horizon for the budgeted/cost-aware setting [29, 118].

9.2.3 Experiments and Results

In this section, we first provide details about our experimental setup. Next, we evaluate the performance of BINOM approach in the cost-agnostic setting and compare it to state-of-the-art baselines.

Baselines. We compare BINOM with EHVI, ParEGO, and random selection base-

lines. We evaluate both types of input selection strategies proposed for cost-agnostic BINOM. BINOM-B refers to the selection of the input with the best immediate hypervolume improvement and BINOM-S refers to randomly selecting an input proportional to its expected immediate reward.

Experimental Setup. We evaluated all algorithms on two benchmarks. ZDT1 [204] is a synthetic problem with 5 dimensions and 2 objective functions. Vehicle crashworthiness design [180] is a real-world problem with 5 dimensions and 3 objective functions. All experiments are averaged over 25 runs with different random seeds. For practical consideration, as explained in section 9.1.3, we limit the horizon length to 4.

Results and Discussion. The results of our preliminary experiments on these two benchmark problems (Figure 9.7) showed that BINOM can be a competitive algorithm. BINOM-S outperformed all baselines and BINOM-B on the vehicle safety problem. On the ZDT1 problem, BINOM-S and BINOM-B had comparable performance and converged to the same results as EHVI. Based on our experimental results, the non-myopic optimization can provide a significant performance gain in the best-case scenario and does not deteriorate the performance in the worst-case scenario. In the future, we will conduct additional experiments to assess the effectiveness of the proposed algorithms.

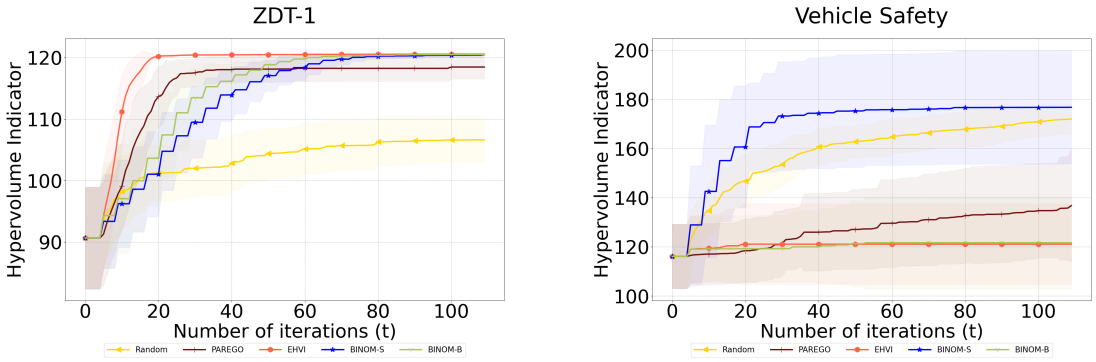

Figure 9.7: Results of BINOM and different baselines on synthetic and real-world problems. The PHV metric is shown as a function of the number of function evaluations.

9.3 Summary

This chapter considered the problem of non-myopic MOO where the optimization is constrained by a predefined budget/number of experiments. We addressed the budgeted non-myopic problem in the single objective setting and applied it to hyper-parameter optimization (HPO) for iterative learners. The proposed BAPI approach addressed gaps in prior work including modeling of structured responses (learning curves) and miscalibration between response and cost models leading to biased search. More importantly, our planning-based BAPI approach allows for non-myopic candidate selection over horizons adaptive to the budget. Combined with subset selection and early termination procedures, our experimental evaluation on a variety of HPO benchmarks demonstrated BAPI's efficacy over previous methods in finding high-performing candidates with less cost budget. We additionally proposed an extension of the non-myopic approach to the cost-agnostic and cost-aware multi-objective set-

tings with promising preliminary experimental results.

CHAPTER TEN

CONCLUSION AND FUTURE DIRECTIONS

In this dissertation, motivated by science and engineering applications, we addressed several challenging adaptive experimental design problems to solve a large class of multi-objective optimization (MOO) problem settings, where it is expensive to evaluate objective functions. We significantly pushed the frontiers of Bayesian optimization in terms of modeling and reasoning algorithms along with theory.

First, we developed a general MOO framework based on the principle of output space entropy search. The key idea is to select the sequence of experiments that maximize the information gained per unit resource cost about the optimal Pareto front. We appropriately instantiated this principle to solve a variety of MOO problems from the most basic single-fidelity setting and its constrained version to the multi-fidelity and continuous-fidelity settings. Second, we studied an uncertainty-aware search framework to address the basic MOO problem, handle different types of constraints, and select a batch of experiments for parallel evaluation. This reduction framework allows us to leverage prior work on acquisition functions for single-objective BO to solve MOO problems. It selects an acquisition function from a given library in each iteration using a multi-arm bandit strategy and solves a cheap MOO problem defined in terms of the acquisition functions (one for each unknown objective) to identify a set of promising candidates. Next, it selects a batch of diverse candidates for evaluation

from this set using Determinantal Point Processes. Finally, we addressed the MOO problem under budget constraints where a planning approach is necessary without violating the resource budget. We proposed a budgeted non-myopic approach for hyper-parameter optimization of iterative machine learning models and a generalization for MOO problems.

Our comprehensive experimental evaluation on synthetic benchmarks and challenging real-world engineering design problems showed that our MOO algorithms significantly improve resource efficiency over prior methods to uncover high-quality Pareto solutions.

10.1 Lessons Learned

In this section, we describe the most important lessons we learned from this work.

1. The output space entropy search framework is more accurate, scalable, and robust when compared to prior MOO methods including those based on input space entropy search. It is scalable because the number of output objectives is significantly smaller when compared to the number of input design variables. The Monte Carlo estimation of the entropy is robust with respect to the number of samples: works well even with a single sample!.

2. There is no single acquisition function that is universally better and consistently outperforms all others. Therefore, the choice of acquisition function to

select experiments in each iteration should depend on the problem at hand. By defining an appropriate multi-objective reward, we can employ multi-arm bandit strategy to dynamically select acquisition functions for multi-objective optimization problems.

3. Variable cost experimentation scenarios including multi-fidelity setting and hyper-parameter optimization of iterative machine learning models require special strategies to handle the cost vs. accuracy trade-off. Normalization of the acquisition function by the cost can lead to poor performance. Therefore, building specific methods to guide the search to select informative inputs is crucial.

4. Non-myopic Bayesian optimization strategies are effective and useful when the available resource budget is small. For large budgets, myopic BO can be effective and there is little to no gain from using expensive non-myopic reasoning procedures.

5. Leveraging side information from experiments in the modeling and reasoning process improves the resourc-efficiency of BO algorithms. This is especially important when there are limited resources.

10.2 Summary of Contributions

The main contribution of this work is the development and evaluation of a suite of novel reasoning algorithms for adaptive experimental design to solve a large class of

MOO problems (single-fidelity, constrained, multi-fidelity, and budget-aware). Our algorithms are based on the principles of information gain per unit resource cost and uncertainty reduction. We appropriately instantiate these principles to derive efficient algorithms for several MOO problem settings (many of them studied for the first time) as summarized below:

- We developed a multi-objective optimization approach based on the principle of output space entropy search named MESMO: Max-value Entropy Search for Multi-Objective Bayesian Optimization [17].

- We developed a multi-objective optimization approach based on the principle of uncertainty reduction named USEMO: Uncertainty-Aware Search framework for Multi-Objective Bayesian Optimization [22].

- We derived the first theoretical analysis to prove a sub-linear regret bound for multi-objective BO setting for both USeMO and MESMO algorithms [17, 22].

- We developed an approach (generalization of USEMO) to solve MOO problems by selecting a batch of diverse experiments for parallel evaluations to improve the resource efficiency to uncover Parto solutions with high quality and Pareto-front diversity [29].

- We developed a multi-arm bandit approach and proposed a suitable multi-objective reward function to adaptively select the suitable acquisition function

in each BO iteration to solve MOO problems and demonstrate its efficacy over using a static acquisition function. [29].

- We developed two approaches (extensions of MESMO and USEMO algorithms) to handle MOO problems with black-box constraints, which cannot be evaluated without performing experiments.
  - MESMOC: Max-value Entropy Search for Multi-Objective BO with Constraints [26, 20]
  - USEMOC: Uncertainty-Aware Search framework for Multi-Objective BO with Constraints [18]
- We developed the first multi-fidelity optimization approaches to solve MOO problems by appropriately leveraging the available side information.
  - MF-OSEMO algorithm to solve MOO problems in the discrete multi-fidelity setting, where experiments can vary in the amount of resources consumed and their evaluation accuracy [21].
  - iMOCA algorithm to solve MOO problems in the continuous-fidelity setting, where continuous function approximations result in a huge space of experiments with varying cost. We provide two qualitatively different approximations for iMOCA. [19, 26].

- We developed non-myopic optimization algorithms when the available resource budget is limited by viewing the problem from a planning perspective.
  - A budget-aware approach to solve hyper-parameter optimization problems for iterative machine learning algorithms with structured responses in the form of learning curves (cost and accuracy vs. number of epochs). We leverage the side-information in the form of the structure of objective functions through appropriate modeling and reasoning tools.
  - A non-myopic optimization approach to solve general MOO problems when the resource budget is limited.
- We applied the developed algorithms to diverse real-world problems in engineering and industrial domains in close collaboration with domain experts.

10.3 Future Work

In this section, we list some important future research directions within the scope of this dissertation.

- We need to apply and investigate our proposed algorithms for important scientific applications including biological sequence design [194] and molecule design [60] by handling the surrogate modeling challenges of combinatorial spaces, e.g., sets, sequences, tress, and graphs [65, 53, 150, 57, 59]

- An important challenge for adaptive experimental design algorithms is to handle high-dimensional input search spaces. There is very little work [45] on studying high-dimensional BO algorithms for solving MOO problems.

- Another interesting and fertile research direction is to explore the synergies between causal modeling and adaptive experimental design: 1) leverage available causal knowledge from the domain in both modeling and reasoning processes to further improve the resource-efficiency of uncovering high-quality Pareto solutions; and 2) apply the principles behind adaptive experimental design algorithms to learn causal models in a resource-efficient manner.

BIBLIOGRAPHY

[1] Majid Abdolshah, Alistair Shilton, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Multi-objective bayesian optimisation with preferences over objectives. *Advances in neural information processing systems*, 32, 2019.

[2] Christian Agrell. Gaussian processes with linear operator inequality constraints. *JMLR*, 2019.

[3] Alaleh Ahmadianshalchi, Syrine Belakaria, and Janardhan Rao Doppa. Preference-aware constrained multi-objective bayesian optimization. *NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*, 2022.

[4] 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.

[5] Christof Angermueller, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin Murphy, Lucy Colwell, and D Sculley. Population-based black-box optimization for biological sequence design. In *International Conference on Machine Learning*, pages 324–334. PMLR, 2020.

[6] Reinaldo B Arellano-Valle, JAVIER E CONTRERAS-REYES, and Marc G

Genton. Shannon entropy and mutual information for multivariate skew-elliptical distributions. *Scandinavian Journal of Statistics*, 40(1), 2013.

[7] Setareh Ariafar, Zelda Mariet, Dana Brooks, Jennifer Dy, and Jasper Snoek. Faster & more reliable tuning of neural networks: Bayesian optimization with importance sampling. In *AISTATS*, pages 3961–3969. PMLR, 2021.

[8] Atthaphon Ariyarit and et al. Multi-fidelity multi-objective efficient global optimization applied to airfoil design problems. *Applied Sciences*, 7(12), 2017.

[9] Aqeeb Iqbal Arka, Srinivasan Gopal, Janardhan Rao Doppa, Deukhyoun Heo, and Partha Pratim Pande. Making a case for partially connected 3d noc: NFIC versus TSV. *ACM J. Emerg. Technol. Comput. Syst.*, 16(4):41:1–41:17, 2020.

[10] Aqeeb Iqbal Arka, Biresh Kumar Joardar, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. ReGraphX: NoC-enabled 3D heterogeneous ReRAM architecture for training graph neural networks. In *DATE*, 2021.

[11] Aqeeb Iqbal Arka, Biresh Kumar Joardar, Ryan Gary Kim, Dae Hyun Kim, Janardhan Rao Doppa, and Partha Pratim Pande. HeM3D: Heterogeneous manycore architecture based on monolithic 3D vertical integration. *ACM Trans. Design Autom. Electr. Syst.*, 26(2):16:1–16:21, 2021.

[12] Raul Astudillo and Peter Frazier. Multi-attribute bayesian optimization with interactive preference learning. In *International Conference on Artificial Intelligence and Statistics*, pages 4496–4507. PMLR, 2020.

[13] Raul Astudillo, Daniel Jiang, Maximilian Balandat, Eytan Bakshy, and Peter Frazier. Multi-step budgeted Bayesian optimization with unknown evaluation costs. *NeurIPS*, 34, 2021.

[14] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT*, pages 41–53. Citeseer, 2010.

[15] Adelchi Azzalini. A class of distributions which includes the normal ones. *Scandinavian Journal of Statistics*, 1985.

[16] 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.

[17] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective Bayesian optimization. In *Conference on Neural Information Processing Systems*, pages 7823–7833, 2019.

[18] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Uncertainty-

aware search framework for multi-objective Bayesian optimization with constraints. In *ICML Workshop on Automated Machine Learning (AutoML)*, 2020.

[19] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Information-theoretic multi-objective bayesian optimization with continuous approximations. *NeurIPS Workshop on Machine Learning for Engineering Modeling, Simulation, and Design*, 2020.

[20] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective bayesian optimization with constraints. *NeurIPS Workshop on Machine Learning and the Physical Sciences*, 2020.

[21] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Multi-fidelity multi-objective Bayesian optimization: An output space entropy search approach. In *AAAI*, pages 10035–10043, 2020.

[22] Syrine Belakaria, Aryan Deshwal, Nitthilan Kannappan Jayakodi, and Janardhan Rao Doppa. Uncertainty-aware search framework for multi-objective Bayesian optimization. In *AAAI conference on artificial intelligence*, 2020.

[23] Syrine Belakaria, Derek Jackson, Yue Cao, Janardhan Rao Doppa, and Xiaonan Lu. Machine learning enabled fast multi-objective optimization for electrified aviation power system design. In *ECCE*, 2020.

[24] Syrine Belakaria*, Derek Jackson*, Yue Cao, Janardhan Rao Doppa, and Xiaonan Lu. Machine learning enabled fast multi-objective optimization for electrified aviation power system design. In *IEEE Energy Conversion Congress and Exposition (ECCE)*, 2020.

[25] Syrine Belakaria*, Zhiyuan Zhou*, Aryan Deshwal, Janardhan Rao Doppa, Partha Pande, and Deuk Heo. Design of multi-output switched-capacitor voltage regulator via machine learning. In *Proceedings of the Twenty-Third IEEE/ACM Design Automation and Test in Europe Conference (DATE)*, 2020.

[26] 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.

[27] Syrine Belakaria*, Derek Jackson*, Yue Cao, Janardhan Rao Doppa, and Xiaonan Lu. Machine learning enabled design automation and multi-objective optimization for electric transportation power systems. *IEEE Transactions on Transportation Electrification*, 2021.

[28] Syrine Belakaria, Rishit Sheth, Janardhan Rao Doppa, and Nicolò Fusi. Bayesian optimization over iterative learners with structured responses: A budget-aware planning approach. *CoRR*, abs/2206.12708, 2022. doi: 10.48550/arXiv.2206.12708. URL `https://doi.org/10.48550/arXiv.2206.12708`.

[29] Syrine Belakaria*, Alaleh Ahmadian*, and Janardhan Rao Doppa. Pareto front-diverse batch multi-objective bayesian optimization. In *Under review (denotes equal contributions)*, 2023.

[30] Syrine Belakaria, Janardhan Rao Doppa, Nicolo Fusi, and Rishit Sheth. Bayesian optimization over iterative learners with structured responses: A budget-aware planning approach. In *International Conference on Artificial Intelligence and Statistics*, pages 9076–9093. PMLR, 2023.

[31] Alexei Borodin. Determinantal point processes. *arXiv preprint arXiv:0911.1153*, 2009.

[32] Alexei Borodin and Grigori Olshanski. Harmonic analysis on the infinite-dimensional unitary group and determinantal point processes. *Annals of mathematics*, pages 1319–1422, 2005.

[33] Zdravko I Botev. The normal law under linear restrictions: simulation and estimation via minimax tilting. *Journal of the Royal Statistical Society*, 79(1): 125–148, 2017.

[34] Luis Ceze, Mark D. Hill, and Thomas F. Wenisch. Arch2030: A vision of computer architecture research over the next 15 years. *CoRR*, abs/1612.03182, 2016. URL `http://arxiv.org/abs/1612.03182`.

[35] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang-Ha Lee, and et al. Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE international symposium on workload characterization (IISWC)*, 2009.

[36] Wonje Choi, Karthi Duraisamy, Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. On-chip communication network for efficient training of deep convolutional networks on heterogeneous manycore systems. *IEEE Transactions on Computers (TC)*, 67 (5):672–686, 2018.

[37] Choi et al. On-chip communication network for efficient training of deep convolutional networks on heterogeneous manycore systems. *IEEE Transactions on Computers (TC)*, 67(5):672–686, 2018.

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

[39] Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. Bayesian optimization meets Bayesian optimal stopping. In *ICML*. PMLR, 2019.

[40] Sourav Das, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu

Chakrabarty. Monolithic 3d-enabled high performance and energy efficient network-on-chip. In *ICCD*, pages 233–240, 2017.

[41] Sourav Das, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. Design-space exploration and optimization of an energy-efficient and reliable 3D small-world network-on-chip. *IEEE TCAD*, 36(5), 2017.

[42] Das et al. Design-space exploration and optimization of an energy-efficient and reliable 3D small-world network-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 36(5):719–732, 2017.

[43] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *NeurIPS*, 33:9851–9864, 2020.

[44] 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, 2021.

[45] Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective bayesian optimization over high-dimensional search spaces. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022. URL `https://openreview.net/forum?id=r5IEvvIs9xq`.

[46] Kalyanmoy Deb and Aravind Srinivasan. Innovization: Innovating design principles through optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1629–1636, 2006.

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

[48] 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.

[49] 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.

[50] Aryan Deshwal and Janardhan Rao Doppa. Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8185–8200, 2021.

[51] Aryan Deshwal, Nitthilan Kannappan Jayakodi, Biresh Kumar Joardar, Janardhan Rao Doppa, and Partha Pratim Pande. MOOS: A multi-objective

design space exploration and optimization framework for NoC enabled many-core systems. *ACM TECS*, 2019.

[52] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Scalable combinatorial Bayesian optimization with tractable statistical models. *CoRR*, abs/2008.08177, 2020. URL `https://arxiv.org/abs/2008.08177`.

[53] Aryan Deshwal, Syrine Belakaria, Janardhan Rao Doppa, and Alan Fern. Optimizing discrete spaces via expensive evaluations: A learning to search framework. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 3773–3780. AAAI Press, 2020.

[54] Aryan Deshwal, Syrine Belakaria, Janardhan Rao Doppa, and Alan Fern. Optimizing discrete spaces via expensive evaluations: A learning to search framework. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[55] Aryan Deshwal, Syrine Belakaria, Ganapati Bhat, Janardhan Rao Doppa, and Partha Pratim Pande. Learning pareto-frontier resource management policies for heterogeneous socs: An information-theoretic approach. In *(DAC)*, 2021.

[56] Aryan Deshwal, Syrine Belakaria, Ganapati Bhat, Janardhan Rao Doppa, and Partha Pratim Pande. Learning pareto-frontier resource management policies for heterogeneous socs: An information-theoretic approach. In *2021*

58th ACM/IEEE Design Automation Conference (DAC), pages 607–612. IEEE, 2021.

[57] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Mercer features for efficient combinatorial bayesian optimization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 7210–7218. AAAI Press, 2021.

[58] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Bayesian optimization over hybrid spaces. In *ICML*, 2021.

[59] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Bayesian optimization over hybrid spaces. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 2632–2643. PMLR, 2021.

[60] Aryan Deshwal, Cory Simon, and Janardhan Rao Doppa. Bayesian optimization of nanoporous materials. *ChemRxiv*, 2021.

[61] Aryan Deshwal, Syrine Belakaria, Janardhan Rao Doppa, and Dae Hyun Kim. Bayesian optimization over permutation spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 2022.

[62] Aryan Deshwal, Sebastian Ament, Maximilian Balandat, Eytan Bakshy, Janardhan Rao Doppa, and David Eriksson. Bayesian optimization over high-dimensional combinatorial spaces via dictionary-based embeddings. *CoRR*,

abs/2303.01774, 2023. doi: 10.48550/arXiv.2303.01774. URL https://doi.org/10.48550/arXiv.2303.01774.

[63] Deshwal et al. MOOS: A multi-objective design space exploration and optimization framework for NoC enabled manycore systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):77:1–77:23, 2019.

[64] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, 2015.

[65] Janardhan Rao Doppa. Adaptive experimental design for optimizing combinatorial structures. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4940–4945, 2021.

[66] Janardhan Rao Doppa, Justinian Rosca, and Paul Bogdan. Autonomous design space exploration of computing systems for sustainability: Opportunities and challenges. *IEEE Design and Test*, 36(5):35–43, 2019.

[67] 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.

[68] M.T.M. Emmerich, K.C. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamod-

els. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006. doi: 10.1109/TEVC.2005.859463.

[69] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*, 2018.

[70] Paul Feliot, Julien Bect, and Emmanuel Vazquez. A Bayesian approach to constrained single-and multi-objective optimization. *Journal of Global Optimization*, 67(1-2):97–133, 2017.

[71] Daniel Fernández-Sánchez, Eduardo C Garrido-Merchán, and Daniel Hernández-Lobato. Max-value entropy search for multi-objective bayesian optimization with constraints. *arXiv preprint arXiv:2011.01150v1*, 2020.

[72] Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

[73] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[74] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *ICML*, volume 2014, pages 937–945, 2014.

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

[76] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. In *AISTATS*, pages 648–657. PMLR, 2016.

[77] Javier González, Michael Osborne, and Neil Lawrence. GLASSES: Relieving the myopia of Bayesian optimisation. In *AISTATS*. PMLR, 2016.

[78] Abhijith M Gopakumar, Prasanna V Balachandran, Dezhen Xue, James E Gubernatis, and Turab Lookman. Multi-objective optimization for materials discovery via adaptive design. *Scientific reports*, 8(1):3738, 2018.

[79] Ahsanul Habib, Hemant K Singh, and et al. A multiple surrogate assisted multi/many-objective multi-fidelity evolutionary algorithm. *Information Sciences*, 2019.

[80] Kyohei Hanaoka. Comparison of conceptually different multi-objective bayesian optimization methods for material design problems. *Materials Today Communications*, 31:103440, 2022.

[81] Javier E Hasbun. *Classical mechanics with MATLAB applications*. Jones & Bartlett Publishers, 2012.

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

[83] 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.

[84] 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.

[85] Matthew Hoffman, Eric Brochu, Nando De Freitas, et al. Portfolio allocation for bayesian optimization. In *UAI*, pages 327–336. Citeseer, 2011.

[86] Matthew Hoffman, Bobak Shahriari, and Nando Freitas. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *AISTATS*, pages 365–374. PMLR, 2014.

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

[88] Wookpyo Hong and et al. A dual-output step-down switched-capacitor voltage regulator with a flying capacitor crossing technique for enhanced power efficiency. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27 (12), 2019.

[89] Deng Huang, Theodore T Allen, William I Notz, and R Allen Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 2006.

[90] Carl Hvarfner, Frank Hutter, and Luigi Nardi. Joint entropy search for maximally-informed bayesian optimization. *arXiv preprint arXiv:2206.04771*, 2022.

[91] Moksh Jain, Sharath Chandra Raparthy, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. *arXiv preprint arXiv:2210.12765*, 2022.

[92] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*, pages 240–248. PMLR, 2016.

[93] Nitthilan Kanappan Jayakodi, Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Design and optimization of energy-accuracy tradeoff networks for mobile platforms via pretrained deep models. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(1):4:1–4:24, 2020.

[94] Nitthilan Kanappan Jayakodi, Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Design and optimization of energy-accuracy tradeoff networks for mobile platforms via pretrained deep models. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(1):1–24, 2020.

[95] Nitthilan Kanappan Jayakodi, Janardhan Rao Doppa, and Partha Pratim Pande. Petnet: Polycount and energy trade-off deep networks for producing 3d objects from images. In *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*, pages 1–6. IEEE, 2020.

[96] Nitthilan Kanappan Jayakodi, Janardhan Rao Doppa, and Partha Pratim Pande. SETGAN: scale and energy trade-off gans for image applications on mobile platforms. In *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020*, pages 23:1–23:9. IEEE, 2020.

[97] Nitthilan Kannappan Jayakodi, Anwesha Chatterjee, Wonje Choi, Janardhan Rao Doppa, and Partha Pratim Pande. Trading-off accuracy and energy of deep inference on embedded systems: A co-design approach. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 37(11):2881–2893, 2018.

[98] Nitthilan Kannappan Jayakodi, Janardhan Rao Doppa, and Partha Pratim Pande. A general hardware and software co-design framework for energy-

efficient edge AI. In *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2021, Munich, Germany, November 1-4, 2021*, pages 1–7. IEEE, 2021.

[99] Shali Jiang, Henry Chai, Javier Gonzalez, and Roman Garnett. BINOCULARS for efficient, nonmyopic sequential experimental design. In *ICML*. PMLR, 2020.

[100] Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. *NeurIPS*, 33, 2020.

[101] Biresh Kumar Joardar, Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems. *IEEE Transactions on Computers*, 68(6):852–866, 2018.

[102] Biresh Kumar Joardar, Aqeeb Iqbal Arka, Janardhan Rao Doppa, and Partha Pratim Pande. 3D++: Unlocking the next generation of high-performance and energy-efficient architectures using M3D integration. In *DATE*, 2021.

[103] Biresh Kumar Joardar, Janardhan Rao Doppa, Partha Pratim Pande, Hai Li, and Krishnendu Chakrabarty. AccuReD: High accuracy training of cnns on

ReRAM/GPU heterogeneous 3D architecture. *IEEE TCAD*, 40(5):971–984, 2021.

[104] Joardar et al. Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems. *IEEE Transactions on Computers*, 68(6):852–866, 2018.

[105] 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.

[106] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[107] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[108] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, and et al. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Conference on Neural Information Processing Systems*, 2016.

[109] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *ICML*. PMLR, 2017.

[110] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[111] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. *Advances in Neural Information Processing Systems*, 29, 2016.

[112] Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 2000.

[113] Ryan Gary Kim, Janardhan Rao Doppa, and Partha Pratim Pande. Machine learning for design space exploration and optimization of manycore systems. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, page 48. IEEE, 2018.

[114] Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. Machine learning and manycore systems design: A serendipitous symbiosis. *IEEE Computer*, 51(7):66–77, 2018.

[115] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with Bayesian neural networks. In *The International Conference on Learning Representations*, 2016.

[116] 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*, 2017.

[117] 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.

[118] Mina Konakovic Lukovic, Yunsheng Tian, and Wojciech Matusik. Diversity-guided multi-objective bayesian optimization with batch evaluations. *Advances in Neural Information Processing Systems*, 33:17708–17720, 2020.

[119] Spyridon G Kontogiannis, Jean Demange, T Kipouros, and et al. A comparison study of two multi-fidelity methods for aerodynamic optimization. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018.

[120] Jayesh H Kotecha and Petar M Djuric. Gibbs sampling approach for generation

of truncated multivariate Gaussian random variables. In *ICASSP*, volume 3, pages 1757–1760. IEEE, 1999.

[121] 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.

[122] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[123] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[124] Marucha Lalee, Jorge Nocedal, and Todd Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, 8(3):682–706, 1998.

[125] Rémi Lam, Douglas L Allaire, and et al. Multifidelity optimization using statistical surrogate modeling for non-hierarchical information sources. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2015.

[126] Remi R Lam, Karen E Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *NeurIPS*. Citeseer, 2016.

[127] 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.

[128] Dongjin Lee, Sourav Das, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. Performance and thermal tradeoffs for energy-efficient monolithic 3D network-on-chip. *ACM TODAES*, 23(5):60:1–60:25, 2018.

[129] Dongjin Lee, Sourav Das, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. Impact of electrostatic coupling on monolithic 3D-enabled network on chip. *ACM TODAES*, 24(6):62:1–62:22, 2019.

[130] Eric Lee, David Eriksson, David Bindel, Bolong Cheng, and Mike Mccourt. Efficient rollout strategies for Bayesian optimization. In *UAI*. PMLR, 2020.

[131] Eric Hans Lee, David Eriksson, Valerio Perrone, and Matthias Seeger. A non-myopic approach to cost-constrained Bayesian optimization. In *UAI*, pages 568–577. PMLR, 2021.

[132] Bing Li, Janardhan Rao Doppa, Partha Pratim Pande, Krishnendu Chakrabarty, Joe X. Qiu, and Hai (Helen) Li. 3d-reg: A 3d reram-based het-

erogeneous architecture for training deep neural networks. *ACM J. Emerg. Technol. Comput. Syst.*, 16(2):20:1–20:24, 2020.

[133] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 18(1):6765–6816, 2017.

[134] Xi Lin, Zhiyuan Yang, Xiaoyuan Zhang, and Qingfu Zhang. Pareto set learning for expensive multi-objective optimization. *arXiv preprint arXiv:2210.08495*, 2022.

[135] Zhiyun Lu, Chao-Kai Chiang, and Fei Sha. Hyper-parameter tuning under a budget constraint. *arXiv preprint arXiv:1902.00532*, 2019.

[136] Mark McLeod, Michael A Osborne, and Stephen J Roberts. Practical Bayesian optimization for variable cost objectives. *arXiv preprint arXiv:1703.04335*, 2017.

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

[138] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[139] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*. Springer, 1975.

[140] Henry B Moss, David S Leslie, and Paul Rayson. Mumbo: Multi-task max-value Bayesian optimization. *The European Conference on Machine Learnin*, 2020.

[141] Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Found. Trends Mach. Learn.*, 7(1):1–129, 2014.

[142] Shouvik Musavvir, Anwesha Chatterjee, Ryan Gary Kim, Dae Hyun Kim, Janardhan Rao Doppa, and Partha Pratim Pande. Power, performance, and thermal trade-offs in M3D-enabled manycore chips. In *DATE*, 2020.

[143] Gaurav Narang, Aryan Deshwal, Janardhan Rao Doppa, Partha Pratim Pande, Raid Ayoub, and Mike Kishinevsky. Dynamic power management in large manycore systems: A learning-to-search framework. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2023.

[144] Elvis Nava, Mojmir Mutny, and Andreas Krause. Diversified sampling for batched bayesian optimization with determinantal point processes. In *International Conference on Artificial Intelligence and Statistics*, pages 7031–7054. PMLR, 2022.

[145] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[146] Vu Nguyen, Sebastian Schulze, and Michael Osborne. Bayesian optimization for iterative learning. *NeurIPS*, 33, 2020.

[147] Aleksandar Nikolov. Randomized rounding for the largest simplex problem. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 861–870, 2015.

[148] J Nocedal and SJ Wright. Numerical optimization (springer, new york, 1999)., 2006.

[149] 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.

[150] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian Optimization using the Graph Cartesian Product. In *NeurIPS*, 2019.

[151] Changyong Oh, Roberto Bondesan, Efstratios Gavves, and Max Welling. Batch bayesian optimization on permutations using acquisition weighted kernels. *arXiv preprint arXiv:2102.13382*, 2021.

[152] 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.

[153] Michael A Osborne. *Bayesian Gaussian processes for sequential prediction, optimisation and quadrature*. PhD thesis, Oxford University, UK, 2010.

[154] 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. PMLR, 2020.

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

[156] Victor Picheny, David Ginsbourger, and et al. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 2013.

[157] 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.

[158] 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.

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

[160] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

[161] Jaakko Riihimäki and Aki Vehtari. Gaussian processes with monotonicity information. In *AISTATS*, pages 645–652. JMLR Workshop and Conference Proceedings, 2010.

[162] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *WACV*, pages 138–142. IEEE, 1994.

[163] Robin Schmucker, Michele Donini, Valerio Perrone, Muhammad Bilal Zafar, and Cédric Archambeau. Multi-objective multi-fidelity hyperparameter optimization with application to fairness. In *NeurIPS Workshop on Meta-Learning*, volume 2, 2020.

[164] Amar Shah and Zoubin Ghahramani. Pareto frontier learning with expensive correlated objectives. In *ICML*, pages 1919–1927. PMLR, 2016.

[165] 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, 2015.

[166] 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.

[167] Harsh Sharma, Sumit K. Mandal, Janardhan Rao Doppa, Ümit Y. Ogras, and Partha Pratim Pande. SWAP: A server-scale communication-aware chiplet-based manycore PIM accelerator. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD)*, 41(11):4145–4156, 2022.

[168] Manali Sharma and Mustafa Bilgic. Evidence-based uncertainty sampling for active learning. *Data Mining and Knowledge Discovery*, 31:164–202, 2017.

[169] Leshi Shu, Ping Jiang, Qi Zhou, Xinyu Shao, Jiexiang Hu, and Xiangzheng Meng. An on-line variable fidelity metamodel assisted multi-objective genetic algorithm for engineering design optimization. *Applied Soft Computing*, 66, 2018.

[170] 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.

[171] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[172] 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.

[173] Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity Bayesian optimization with Gaussian processes. *International Conference on Artificial Intelligence and Statistics*, 2019.

[174] 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.

[175] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test

functions and datasets. Retrieved January 21, 2020, from `http://www.sfu.ca/~ssurjano`, 2020.

[176] 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. PMLR, 2020.

[177] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. In *Conference on Neural Information Processing Systems*, 2013.

[178] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

[179] Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian optimization with max-value entropy search. *arXiv:1901.08275*, 2019.

[180] Ryoji Tanabe and Hisao Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, 89:106078, 2020.

[181] Thiago de P Vasconcelos, Daniel ARMA de Souza, César LC Mattos, and João PP Gomes. No-past-bo: Normalized portfolio allocation strategy for bayesian optimization. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 561–568. IEEE, 2019.

[182] Thiago de P Vasconcelos, Daniel Augusto RMA de Souza, Gustavo C de M Virgolino, César LC Mattos, and João PP Gomes. Self-tuning portfolio-based bayesian optimization. *Expert Systems with Applications*, 188:115847, 2022.

[183] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[184] Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016.

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

[186] 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.

[187] Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. In *International Conference on Machine Learning*, pages 3656–3664. PMLR, 2017.

[188] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

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

[190] James T Wilson, Frank Hutter, and Marc Peter Deisenroth. Maximizing acquisition functions for Bayesian optimization. *NeurIPS*, 2018.

[191] Jian Wu and Peter Frazier. Practical two-step lookahead Bayesian optimization. *NeurIPS*, 32, 2019.

[192] Jian Wu and Peter I Frazier. Continuous-fidelity Bayesian optimization with knowledge gradient. *NIPS Workshop on Bayesian Optimization*, 2018.

[193] Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *UAI*. PMLR, 2020.

[194] Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694, 2019.

[195] Xiaoxuan Yang, Syrine Belakaria, Biresh Kumar Joardar, Huanrui Yang, Janardhan Rao Doppa, Partha Pratim Pande, Krishnendu Chakrabarty, and Hai Helen Li. Multi-objective optimization of reram crossbars for robust DNN inferencing under stochastic noise. In *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2021, Munich, Germany, November 1-4, 2021*, pages 1–9. IEEE, 2021.

[196] Xiaoxuan Yang, Syrine Belakaria, Biresh Kumar Joardar, Huanrui Yang, Janardhan Rao Doppa, Partha Pratim Pande, Krishnendu Chakrabarty, and Hai Helen Li. Multi-objective optimization of reram crossbars for robust dnn inferencing under stochastic noise. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[197] Xubo Yue and Raed AL Kontar. Why non-myopic Bayesian optimization is promising and how far should we look-ahead? a study via rollout. In *AISTATS*. PMLR, 2020.

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

[199] Yehong Zhang, Trong Nghia Hoang, and et al. Information-based multi-fidelity Bayesian optimization. In *Conference on Neural Information Processing Systems Workshop on Bayesian Optimization*, 2017.

[200] Zhiyuan Zhou, Syrine Belakaria, Aryan Deshwal, Wookpyo Hong, Janardhan Rao Doppa, Partha Pratim Pande, and Deukhyoun Heo. Design of multi-output switched-capacitor voltage regulator via machine learning. In *DATE*, 2020.

[201] Jiandao Zhu, Yi-Jen Wang, and Matthew Collette. A multi-objective variable-fidelity optimization method for genetic algorithms. *Engineering Optimization*, 46(4):521–542, 2014.

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

[203] 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.

[204] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8 (2):173–195, 2000.

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

[206] 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. URL `http://www.spiral.net/software/pal.html`.

APPENDIX A

THEORETICAL ANALYSIS FOR MESMO

In this appendix, we provide a proof for **Theorem 1** described in Section 3.4.

Lemma 1 *(Lemma C.1 in MES [185]). Pick* $\delta \in (0,1)$ *and set* $\zeta_t = \left(2\log\left(\frac{\pi_t}{2\delta}\right)\right)^{1/2}$, *where* $\sum_{t=1}^{T}(\pi_t)^{-1} \leq 1, \pi_t > 0$. *Then, it holds that for each function* f_j, $Pr[\mu_{j,t-1}(\mathbf{x_t}) - f_j(\mathbf{x_t}) \leq \zeta_t \sigma_{j,t-1}(\mathbf{x}), \forall t \in [1,T]] \geq 1-\delta$. *Here* $\mu_{j,t-1}$ *and* $\sigma_{j,t-1}(\mathbf{x})$ *refers to the predictive mean and variance of* j^{th} *GP at iteration number* t.

Lemma 2 *(Lemma C.2 in MES [185]) If* $\mu_{j,t-1}(\mathbf{x_t}) - f_j(\mathbf{x_t}) \leq \zeta_t \sigma_{j,t-1}(\mathbf{x})$, *for each* $j \in [1, \cdots, K]$, *the quantity* $r_t^j = f_j(x^*) - f_j(x_t) \leq \left(v_t^j + \zeta_t\right)\sigma_{j,t-1}(\mathbf{x_t})$, *where* $v_t^j \doteq \min_{\mathbf{x}\in\mathfrak{X}} \frac{y_j^* - \mu_{j,t-1}(\mathbf{x})}{\sigma_{j,t-1}(\mathbf{x})}$ *and* $y_j^* \geq f_j(x^*) \forall t \in [1,T]$.

Let $f_j^\star$ be the true maximum of jth function i.e. $\max_x f_j(x) = f_j^\star$.

Since x^* is in the Pareto set and each point in the pareto set has value less than or equal to the function maximum, it implies that $f_j(x^*) \leq f_j^*$. Therefore,

$$r_t^j = f_j(x^*) - f_j(x_t) \leq f_j^\star - f_j(x_t)$$

Now, following from EST Lemma 3.3 [1]:

$$\begin{aligned}\max_x f_j(x) - f(x_t) &= f_j^\star - f_j(x_t) \\ &\leq y_j^* - f_j(x_t) \\ &\leq y_j^* - \mu_{j,t-1}(x_t) + \zeta_t \sigma_{j,t-1}(\mathbf{x_t}) \\ &= \left(v_t^j + \zeta_t\right) \sigma_{j,t-1}(x_t)\end{aligned}$$

In MES Lemma C.2, this is true whenever $\widehat{m}_t \geq \max_x f(x)$. Similarly, in MESMO this is true whenever $y_j^* \geq f_j^\star \geq f_j(x^*)$.

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 is i.i.d $\mathcal{N}(0, \sigma)$ and $w = Pr[(y_1^* > f_1(x^*)), \cdots, (y_K^* > f_K(x^*))]$. If $\mathbf{x}_t$ is the candidate input selected by MESMO at the t^{th} iteration according to 3.13 and $[y_1^*, \cdots, y_K^*]$ is drawn from P, then with probability at least $1 - \delta$, in $T' = \sum_{i=1}^{T} \log_w \frac{\delta}{2\pi_i}$ number of iterations*

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

[1] `https://lis.csail.mit.edu/pubs/wang-aistats16.pdf`

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.*

Proof: To use Lemma **2.** and following the same reasoning as MES, the goal here is to find out when is the condition $y_j^* \geq f_j^\star$ satisfied for each function i.e. we need to find the probability $Pr[(y_1^* > f_1^\star), \cdots, (y_K^* > f_K^\star)]$. However, since $f_j^\star \geq f_j(x^*)$ for each function f_j, we use the quantity $w = Pr[(y_1^* > f_1(x^*)), \cdots, (y_K^* > f_K(x^*))]$ in the proof below:

The result for each R^j can be derived from the fact that the corresponding expression for a single sample in Equation 3.13 $\left(\frac{\gamma_s^j(\mathbf{x})\phi(\gamma_s^j(\mathbf{x}))}{2\Phi(\gamma_s^j(\mathbf{x}))} - \ln \Phi(\gamma_s^j(\mathbf{x}))\right)$ is equivalent to EST (optimization as estimation strategy) [186]. This fact is proven as Lemma **3.1** in MES (Max-value entropy search) [185]. Therefore, theoretical results from MES can be leveraged for each R^j provided $y_j^* > f_j(x^*)$ for all $j \in \{1, \cdots, K\}$.

Since $[y_1^*, \cdots, y_K^*]$ is drawn from P, the probability that there exists atleast one vector $[y_1^*, \cdots, y_K^*]$ in k_i iterations that satisfies $[(y_1^* > f_1(x^*)), \cdots, (y_K^* > f_K(x^*))]$ is

given by:

$$\Rightarrow w + (1-w)w + (1-w)^2 w \cdots + (1-w)^{k_i - 1} w \tag{A.2}$$

$$= w \cdot \left(\frac{1-(1-w)^{k_i}}{1-(1-w)} \right) \tag{A.3}$$

$$= 1 - (1-w)^{k_i} \tag{A.4}$$

$$\geq (1-(1-w))^{k_i} \quad \text{since } w \in (0,1) \tag{A.5}$$

$$\geq w^{k_i} \tag{A.6}$$

Suppose $T' = \sum_{i=1}^{T} k_i$ be the total number of iterations (function evaluations). Following Theorem 3.2 from MES [185], splitting the total number of iterations into T parts, where each part has k_i iterations, there exists at least one iteration t_i in each of the T parts with probability $1 - \sum_{i=1}^{T} w^{k_i}$ such that $[(y_1^* > f_1(x^*)), \cdots, (y_K^* > f_K(x^*))]$.

Let $\sum_{i=1}^{T} w^{k_i} = \frac{\delta}{2}$ and setting $k_i = \log_w \frac{\delta}{2\pi_i}$ for any $\sum_{i=1}^{T} \frac{1}{\pi_i} = 1$. A standard choice for π_i is $\pi_i = \pi^2 i^2/6$. Using this transformation of variables, the probability that there exists sampled functions such that $[(y_1^* > f_1(x^*)), \cdots, (y_K^* > f_K(x^*))]$ is atleast $1 - \delta/2, \forall i \in [1, T]$.

By lemma 1 and 2,

$$r_{t_i}^j = \left(v_{t_i}^j + \zeta_{t_i} \right) \sigma_{j, t_i - 1}(\mathbf{x_{t_i}}) \tag{A.7}$$

From Lemma C.3 in MES[185], $\sum_{i=1}^{T} \sigma^2_{j,t_i-1}(\mathbf{x}_{t_i}) \leq \frac{2}{log(1+\sigma^{-2})}\gamma^j_T$, where γ^j_T is the maximum information gain about function f_j and is an important theoretical quantity related to regret bounds in bayesian optimization literature[174]. By Cauchy-Schwarz inequality, $\sum_{i=1}^{T} \sigma_{j,t_i-1}(\mathbf{x}_{t_i}) \leq \sqrt{T\sum_{i=1}^{T} \sigma^2_{j,t_i-1}(\mathbf{x}_{t_i})} \leq \sqrt{2T\gamma^j_T/\log(1+\sigma^{-2})}$. Therefore, with probability $1-\delta$,

$$R^j(\mathbf{x}^*) = \sum_{i=1}^{T} r^j_{t_i} \leq \left(v^j_{t^*} + \zeta_T\right)\sqrt{\frac{2T\gamma^j_T}{log(1+\sigma^{-2})}} \tag{A.8}$$

Consequently,

$$R(x^*) = \sqrt{\sum_{j=1}^{K}\left(\left(v^j_{t^*} + \zeta_T\right)^2\left(\frac{2T\gamma^j_T}{log(1+\sigma^{-2})}\right)\right)} \tag{A.9}$$

APPENDIX B

THEORETICAL ANALYSIS FOR USEMO

B.1 Theoretical Analysis

This section provides the proof of Theorem 1 which depends on Lemma 1. For completeness, the GP-LCB acquisition function uses the following definition of $LCB_{i,t}(x)$ for function F_i at any iteration t.

$$LCB_{i,t}(x) = \mu_{i,t-1}(x) - \beta_t^{1/2}\sigma_{i,t-1}(x) \tag{B.1}$$

Lemma 1 Given $\delta \in (0,1)$ and $\beta_t = 2\log(|\mathcal{X}|\pi^2 t^2/6\delta)$, the following holds with probability $1-\delta$:

$$|F_i(x) - \mu_{i,t-1}(x)| \leq \beta_t^{1/2}\sigma_{i,t-1}(x) \tag{B.2}$$

$$\text{for all } 1 \leq i \leq k, x \in \mathcal{X}, \text{for all } t \geq 1 \tag{B.3}$$

Proof. Since F_i is modeled by a GP, $F_i(x) \sim \mathcal{N}(\mu_{i,t-1}(x), \sigma_{i,t-1}(x))$. According to Lemma 5.1 from [174], the below inequality holds:

$$\Pr\{|F_i(x) - \mu_{i,t-1}(x)| > \beta_t^{1/2}\sigma_{i,t-1}(x)\} \leq e^{-\beta/2} \tag{B.4}$$

Applying the union bound,

$$|F_i(x) - \mu_{i,t-1}(x)| \leq \beta_t^{1/2} \sigma_{i,t-1}(x) \tag{B.5}$$

holds with probability $1 - \delta$. The lemma holds by choosing $e^{-\beta/2}|\mathcal{X}| = 6\,\delta/\pi^2 t^2$ as suggested in [174].

Theorem 1 If $\mathcal{X}_t$ be the Pareto-set generated by the cheap multi-objective optimization at t-th iteration, then the following holds with probability $1 - \delta$,

$$R(x^*) \leq \sqrt{\sum_{i=1}^{k} C T_{max} \beta_{T_{max}} \gamma^i_{T_{max}}} \tag{B.6}$$

where C is a constant and $\gamma^i_{T_{max}}$ is the maximum information gain about F_i after T_{max} iterations.

Proof. For the sake of completeness, the cheap multi-objective optimization problem for GP-LCB becomes

$$\min_{x \in \mathcal{X}} \left(LCB_{1,t}(x), LCB_{2,t}(x), \cdots, LCB_{k,t}(x)\right) \tag{B.7}$$

Assuming optimality of $\mathcal{X}_t$, either there exists a $x_t \in \mathcal{X}_t$ such that

$$LCB_{i,t}(x_t) \leq LCB_{i,t}(x^*), \forall i \in \{1, \cdots, k\} \tag{B.8}$$

or x^* is in the optimal Pareto set $\mathcal{X}_t$ generated by cheap MO solver (i.e., $x_t = x^*$).

Now, using Lemma 1 for any function F_j,

$$LCB_{j,t}(x_t) \leq LCB_{j,t}(x^*) \leq F_j(x^*) \tag{B.9}$$

Therefore,

$$R_j(x^*) = F_j(x_t) - F_j(x^*) \leq F_j(x_t) - LCB_{j,t}(x_t) \tag{B.10}$$

$$R_j(x^*) \leq F_j(x_t) - \mu_{i,t-1}(x_t) + \beta^{1/2}\sigma_{i,t-1}(x_t) \tag{B.11}$$

$$R_j(x^*) \leq 2\beta^{1/2}\sigma_{i,t-1}(x_t) \tag{B.12}$$

Inequality (B.12) is similar to the result of Lemma 5.2 from [174] in the single-objective BO case. Since j is arbitrary, this is true for each function F_j, for all $j \in \{1, 2, \cdots, k\}$.

Further, using Lemma 5.4 from [174],

$$R_j(x^*) \leq \sqrt{CT_{max}\beta_{T_{max}}\gamma^j_{T_{max}}} \text{ with probability} \geq 1-\delta \tag{B.13}$$

Consequently, the bounds for $R(x^*)$ becomes

$$R(x^*) \leq \sqrt{\sum_{i=1}^{k} CT_{max}\beta_{T_{max}}\gamma^i_{T_{max}}} \tag{B.14}$$

The quantity $\gamma^i_{T_{max}}$ is employed in many theoretical studies of GP-based optimization including the well-known work of Srinivasan et al., [174].

APPENDIX C

APPENDIX FOR IMOCA

C.1 Full derivation of iMOCA's acquisition function

Our goal is to derive a full approximation for iMOCA algorithm. In this appendix, we provide the technical details of the extended-skew Gaussian approximation (iMOCA-E) for the computation of the information gain per unit cost.

C.2 Additional Experiments and Results

C.2.1 Description of Synthetic Benchmarks

In what follows, we provide complete details of the synthetic benchmarks employed in this paper. Since our algorithm is designed for maximization settings, we provide the benchmarks in their maximization form.

1) Branin, Currin experiment

In this experiment, we construct a multi-objective problem using a combination of existing single-objective optimization benchmarks [109]. It has two functions with two dimensions (K=2 and d=2).

Branin function: We use the following function where $\mathcal{C}(z) = 0.05 + z^{6.5}$

$$g(\mathbf{x}, z) = -\left(a(x_2 - b(z)x_1^2 + c(z)x_1 - r)^2 + s(1 - t(z))cos(x_1) + s\right)$$

where $a = 1$, $b(z) = 5.1/(4\pi^2) - 0.01(1-z)$, $c(z) = 5/\pi - 0.1(1-z)$, $r = 6$, $s = 10$ and $t(z) = 1/(8\pi) + 0.05(1-z)$.

Currin exponential function: We use $\mathcal{C}(z) = 0.1 + z^2$

$$g(\mathbf{x}, z) = -\left(1 - 0.1(1-z)\exp\left(\frac{-1}{2x_2}\right)\right)\left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right).$$

2) Ackley, Rosen, Sphere experiment

In this experiment, we construct a multi-objective problem using a combination of existing single-objective optimization benchmarks [192]. It has three functions with five dimensions (K=3 and d=5). For all functions, we employed $\mathcal{C}(z) = 0.05 + z^{6.5}$

Ackley function

$$g(\mathbf{x}, z) = -\left(-20\exp\left[-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right] - \exp\left[\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right] + e + 20\right) - 0.01(1-z)$$

Rosenbrock function:

$$g(\mathbf{x}, z) = -\sum_{i=1}^{d-1}\left[100\left(x_{i+1} - x_i^2 + 0.01(1-z)\right)^2 + (1-x_i)^2\right]$$

Sphere function:

$$g(\mathbf{x}, z) = -\sum_{i=1}^{d} x_i^2 - 0.01(1 - z)$$

3) DTLZ1 experiment

In this experiment, we solve a problem from the general multi-objective optimization benchmarks [79]. We have six functions with five dimensions (K=6 and d=5) with a discrete fidelity setting. Each function has three fidelities in which z takes three values from $\{0.2, 0.6, 1\}$ with z^*=1. The cost of evaluating each fidelity function is $\mathcal{C}(z)$=$\{0.01, 0.1, 1\}$

$$g_j(\mathbf{x}, z) = f_j(\mathbf{x}) - e(\mathbf{x}, z)$$

$f_1(\mathbf{x}) = -(1 + r)0.5\Pi_{i=1}^{5} x_i$

$f_j(\mathbf{x}) = -(1 + r)0.5(1 - x_{6-j+1})\Pi_{i=1}^{6-j} x_i$ with $j = 2 \ldots 5$

$f_6(\mathbf{x}) = -(1 + r)0.5(1 - x_1)$

$r = 100[d + \sum_{i=1}^{d}((x_i - 0.5)^2) - cos(10\pi(x_i - 0.5))]$

$e(\mathbf{x}, z) = \sum_{i=1}^{d} \alpha(z) cos(10\pi\alpha(z)x_i + 0.5\pi\alpha(z) + \pi)$ with $\alpha(z) = 1 - z$

4) QV experiment

In this experiment, we solve a problem from the general multi-objective optimization benchmarks [169]. We have two functions with eight dimensions (K=2 and d=8) with a discrete fidelity setting.

Function 1 has only one fidelity which is the highest fidelity

$$f_1(\mathbf{x}) = -(\frac{1}{d}\sum_{i=1}^{d}(x_i^2 - 20\pi x_i + 10))^{\frac{1}{4}}$$

Function 2 has two fidelities with cost $\{0.1, 1\}$ respectively and the following expressions:

High fidelity: $f_2(\mathbf{x}, High) = -(\frac{1}{d}\sum_{i=1}^{d}((x_i - 1.5)^2 - 20\pi(x_i - 1.5) + 10))^{\frac{1}{4}}$

Low fidelity: $f_2(\mathbf{x}, Low) = -(\frac{1}{d}((\sum_{i=1}^{d}(\alpha[\mathbf{i}](x_i - 1.5)^2 - 20\pi(x_i - 1.5) + 10))^{\frac{1}{4}}$

with α=$[0.9, 1.1, 0.9, 1.1, 0.9, 1.1, 0.9, 1.1]$

C.2.2 Additional information about experimental setup

Experimental setup for our proposed algorithms:

- The hyper-parameters are estimated after every five function evaluations (BO iterations) for MESMO and MESMOC. For iMOCA and MF-OSEMO, the number of evaluations would be higher due to the low cost of lower fidelities. Therefore, the hyper-parameters are estimated every twenty iterations.

- During the computation of Pareto front samples, we solve a cheap MO optimization problem over sampled functions using NSGA-II. We use Platypus[1] library for the implementation. For NSGA-II, the most important parameter is the number of function calls. We experimented with several values. We noticed that increasing this number does not result in any performance improvement for our algorithms. Therefore, we fixed it to 1500 for all our experiments.

Parameters used for NSAG-II and MOEAD as constrained baselines:

- Since we allow only 200 evaluations for MESMOC and PESMOC, we also set the number of functions evaluations for NSGA-II and MOEAD to 200. We leave any other parameter to the default value provided by the Platypus library.

Computational resources

- We performed all experiments on a machine with the following configuration: Intel i7-7700K CPU @ 4.20GHz with 8 cores and 32 GB memory.

[1] platypus.readthedocs.io/en/latest/getting-started.html#installing-platypus

APPENDIX D

APPENDIX FOR BAPI

D.1 Details of the monotonic Gaussian process

The posterior predictive distribution of the monotonic GP is $\mathbf{f}^*|Y, C$ which is the distribution of $\mathbf{f}^* = f(\mathbf{z}_*)$ for some new inputs $\mathbf{z}_* = [\mathbf{x}_*, t_*]$, conditioned on the observed data Y and the constraint C defined as $a(Z^v) \leq \mathcal{L}f(Z^v) \leq b(Z^v)$. The final derivation of the predictive distribution is defined as follow:

$$\mathbf{f}^*|Y, C \sim \mathcal{N}(\mu^* + A(\mathbf{C} - \mathcal{L}\mu^v) + B(Y - \mu), \Sigma) \tag{D.1}$$

$$\mathbf{C} = \widetilde{C}|Y, C \sim \mathcal{TN}(\mathcal{L}\mu^v + A_1(Y - \mu), B_1, a(Z^v), b(Z^v)) \tag{D.2}$$

where $\mathcal{TN}(\cdot, \cdot, a, b)$ is the truncated Gaussian $\mathcal{N}(\cdot, \cdot)$ conditioned on the hyper-rectangle $[a_1, b_1] \times \cdots \times [a_k, b_k]$, $\mu^v = m(Z^v), \mu^* = m(\mathbf{z}_*), \mu = m(Z)$. The matricies A, B, A_1, B_1

and Σ are defined as follow:

$$A_1 = (\mathcal{L}\kappa_{Z^v,X})(\kappa_{Z,Z} + \sigma^2 I)^{-1} \tag{D.3}$$

$$A_2 = \kappa_{\mathbf{z}_*,Z}(\kappa_{Z,Z} + \sigma^2 I)^{-1} \tag{D.4}$$

$$B_1 = \mathcal{L}\kappa_{Z^v,Z^v}\mathcal{L}^T + \sigma_v^2 I - A_1\kappa_{Z,Z^v}\mathcal{L}^T \tag{D.5}$$

$$B_2 = \kappa_{\mathbf{z}_*,\mathbf{z}_*} - A_2\kappa_{Z,\mathbf{z}_*} \tag{D.6}$$

$$B_3 = \kappa_{\mathbf{z}_*,Z^v}\mathcal{L}^T - A_2\kappa_{Z,Z^v}\mathcal{L}^T \tag{D.7}$$

$$A = B_3 B_1^{-1} \tag{D.8}$$

$$B = A_2 - AA_1 \tag{D.9}$$

$$\Sigma = B_2 - AB_3^T \tag{D.10}$$

Additionally, the probability that the unconstrained version of $\mathbf{C}$ falls within the constraint region, $p(C|Y)$, is defined as follow:

$$p(C|Y) = p\left(a(Z^v) \leq \mathcal{N}(\mathcal{L}\mu^v + A_1(Y - \mu), B_1) \leq b(Z^v)\right) \tag{D.11}$$

and the unconstrained predictive distribution is

$$\mathbf{f}^*|Y \sim \mathcal{N}(\mu^* + A_2(Y - \mu), B_2).$$

Sampling from the posterior distribution with constraints has been a challeng-

ing task in previous work [161]. However, Agrell [2] proposed to use a new method based on simulation via minimax tilting proposed by Botev [33]. This sampling approach was proposed for high-dimensional exact sampling and was shown to efficient and fast compared to previous approaches like rejection sampling and Gibb sampling [120].

For more details about the efficient posterior computation of the monotonic GP we refer the reader to [2].

kernels derivatives

The computation of the posterior of the monotonic Gaussian process requires the definition of derivatives of the kernel function. In this work we consider monotonicity with respect to one dimension t. Therefore, kernel derivatives would be defined as follow

$$\frac{\partial}{\partial t}\kappa([\mathbf{x},t],[\mathbf{x}',t']) = \kappa_x(x,x') \times \frac{\partial}{\partial t}\kappa_t(t,t') \tag{D.12}$$

$$\frac{\partial}{\partial t\partial t'}\kappa([\mathbf{x},t],[\mathbf{x}',t']) = \kappa_x(x,x') \times \frac{\partial}{\partial t\partial t'}\kappa_t(t,t') \tag{D.13}$$

In this work t is a single dimensional variables. However, for sake of generality , we provide the kernel derivatives for the general case where t can be multi-dimensional. We define d_t as the t. In our experiments, we focus mainly on cases where the kernel

over dimension t is an ED kernel. However, Our proposed method is not restrictive. In cases where the learning curve is not exponentially decaying, an RBF kernel with monotonicity over dimension t can be used. We provide the derivatives for both kernels

Exponential Decay Kernel

$$\kappa_t(\mathbf{t}, \mathbf{t}') = w + (\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha} \tag{D.14}$$

$$\frac{\partial}{\partial t'_j}\kappa_t(\mathbf{t}, \mathbf{t}') = -\frac{\alpha}{\beta_j}(\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha-1} \tag{D.15}$$

$$\frac{\partial}{\partial t_j \partial t'_j}\kappa_t(\mathbf{t}, \mathbf{t}') = \frac{\alpha(\alpha+1)}{\beta_j^2}(\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha-2} \tag{D.16}$$

$$\frac{\partial}{\partial t_i \partial t'_j}\kappa_t(\mathbf{t}, \mathbf{t}') = \frac{\alpha(\alpha+1)}{\beta_i \beta_j}(\frac{\mathbf{t}}{\boldsymbol{\beta}} + \frac{\mathbf{t}'}{\boldsymbol{\beta}} + 1)^{-\alpha-2} \tag{D.17}$$

Radial basis function Kernel

$$\kappa_t(\mathbf{t}, \mathbf{t}') = exp(\frac{-1}{2}\sum_{i=1}^{d_t}\frac{(t_i - t'_i)^2}{l_i}) \tag{D.18}$$

$$\frac{\partial}{\partial t'_j}\kappa_t(\mathbf{t}, \mathbf{t}') = \frac{t_j - t'_j}{l_j^2}\kappa_t(\mathbf{t}, \mathbf{t}') \tag{D.19}$$

$$\frac{\partial}{\partial t_j \partial t'_j}\kappa_t(\mathbf{t}, \mathbf{t}') = \frac{1}{l_j^2}(1 - \frac{t_j - t'_j}{l_j^2})\kappa_t(\mathbf{t}, \mathbf{t}') \tag{D.20}$$

$$\frac{\partial}{\partial t_i \partial t'_j}\kappa_t(\mathbf{t}, \mathbf{t}') = -\frac{t_j - t'_j}{l_j^2}\frac{t_i - t'_i}{l_i^2}\kappa_t(\mathbf{t}, \mathbf{t}') \tag{D.21}$$