Master's thesis
Master's Programme in Data Science

Pareto front learning in multi-objective Bayesian optimization

Tatu Linnala

April 10, 2025

Supervisor(s): Prof. Luigi Acerbi
Dr. Matthias Stosiek
Dr. Joakim Löfgren

Examiner(s): Prof. Luigi Acerbi
Dr. Matthias Stosiek

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Tatu Linnala

Työn nimi — Arbetets titel — Title

Pareto front learning in multi-objective Bayesian optimization

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	April 10, 2025	64

Tiivistelmä — Referat — Abstract

Bayesian optimization (BO) is a popular machine learning technique for optimizing expensive black-box functions. It is widely used in many fields of science and engineering, for example, in materials structure and composition optimization tasks. Many BO applications involve multiple competing objectives, requiring decision-makers to seek optimal trade-offs that form the so-called Pareto front. Multi-objective optimization algorithms aim to approximate this front.

A significant contribution of this work is to extend the Bayesian Optimization Structure Search (BOSS) code to support advanced multi-objective BO methods. The Hypervolume Improvement and three variants of the Expected Hypervolume Improvement acquisition function were implemented: an exact formulation for bi-objective cases and Monte Carlo-based approximations for scenarios with any number of objectives. Additionally, we propose adding an explicit exploration term to the Hypervolume Improvement acquisition function. We show that this can significantly boost the performance of the method in applications where exploration of the design space is required. Scalarization-based approaches and an extension of the single-objective ELCB acquisition function to multi-objective cases, which offer computationally cheaper solutions, were also included.

The methods are benchmarked on six test cases, including five synthetic functions and a real-world problem based on lignin extraction data. The results show that the hypervolume-based methods provide the most accurate Pareto front predictions, but cheaper methods may sometimes be sufficient. This highlights the trade-offs between computational cost and accuracy, emphasizing the importance of selecting methods based on application-specific needs. This thesis provides guidelines for choosing appropriate multi-objective BO settings, which can all be accessed within the extended BOSS code. Thus, the BOSS code now provides an accessible, comprehensive toolkit for multi-objective optimization tasks.

ACM Computing Classification System (CCS):
Applied computing → Operations research → Decision analysis → **Multi-criterion optimization and decision-making**
Computing methodologies → Machine learning → Machine learning approaches → Kernel methods → **Gaussian processes**

Avainsanat — Nyckelord — Keywords

Bayesian optimization, multi-objective optimization, machine learning, active learning, Pareto front

Säilytyspaikka — Förvaringsställe — Where deposited

HELDA – University of Helsinki Open Repository

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Optimization problems are ubiquitous in science and engineering, spanning diverse fields such as material design, drug discovery, and manufacturing. In the development of new medicines, for example, researchers wish to maximize the effectiveness of new candidate drugs while minimizing their harmful side effects and manufacturing costs. In materials discovery, there is an increasing demand for new advanced materials in areas such as battery technology and solar cells. This drives researchers to search for new material compositions that optimize key properties such as efficiency, environmental stability, and non-toxicity. A standard approach for studying material candidates is density functional theory (DFT). DFT can be costly and requires the minimization of complex energy landscapes. Efficiently solving such optimization tasks is vital because, for instance, new solar cell and battery materials are crucial to accelerating the green energy transformation.

Global optimization poses a particular challenge: a target function may have many local optima in an oftentimes large search space, which has to be explored efficiently to identify the global optimum. Here, computational methods have become essential for guiding and accelerating optimization efforts in many fields of science and engineering. However, a key challenge in many real-world optimization problems, such as the ones described above, is the high cost of objective function evaluations. For instance, the computational cost of DFT calculations and the time and resource investment of experimental laboratory work can be highly prohibitive. Combined with the vast design spaces typical of scientific optimization problems, this makes large-scale development efforts infeasible with traditional optimization techniques.

A common approach to expensive optimization tasks is Bayesian optimization (BO) [18], an active machine learning technique that aims to minimize the number of objective function evaluations while maximizing prediction accuracy. BO achieves this by utilizing a probabilistic surrogate model, which serves as an approximation of the unknown objective function, and an acquisition function, which is used to select the objective function samples. These two elements are combined into an iterative framework to optimize the objective function while respecting strict evaluation budgets. The role of the acquisition function is essential: it is used to select the next function sample at each iteration of the algorithm. The samples are chosen so that maximum utility is received from each new

sample while utilizing prior information gained from all the previous samples.

While BO has proven effective for single-objective optimization tasks, many practical applications inherently involve multiple objectives. For instance, in drug and materials development, researchers typically need to optimize multiple properties simultaneously. This presents a challenge, as different objectives can be conflicting, meaning that there is no single solution that optimizes all the objectives at once. For example, a highly effective drug is also likely to have more severe side effects, and a solar cell material with high efficiency might not be environmentally stable. Therefore, to efficiently address these demands, it is important to extend the single-objective BO framework to multi-objective optimization.

In developing a multi-objective optimization algorithm, the key idea is to consider trade-offs between objectives. This means that, instead of looking for a solution that optimizes all the objectives simultaneously, one should search for optimal trade-offs between the objectives. These trade-offs are non-dominated solutions in which none of the objectives can be improved without deteriorating another. The non-dominated solutions constitute the Pareto front in the objective space, and multi-objective optimization algorithms aim to provide an approximation of this front. Then, given a set of optimal trade-offs, the decision-maker may decide on the most desirable solution.

A critical challenge in multi-objective BO is the selection of new acquisitions in a way that accounts for the trade-offs between objectives. It is not enough that one objective is improved if it is done at the expense of another. This presents new challenges, as sophisticated acquisition strategies are needed to obtain a diverse set of non-dominated solutions. The strategies need to, for example, balance exploitation and exploration of the search space, be feasible in terms of computational cost and be flexible so that they can be applied to diverse problems.

This thesis therefore focuses on implementing and benchmarking acquisition strategies for multi-objective BO. The work presented here builds on the Bayesian Optimization Structure Search (BOSS) code, a general-purpose Python implementation of BO [28]. BOSS is under continuous development at the AI-based Materials Science group at the Technical University of Munich, the Computational Electronic Structure Theory group at Aalto University, and the Materials Informatics Laboratory at the University of Turku. It is available as `aalto-boss` on the PyPI repository. Prior to the work presented here, BOSS only supported single-objective BO. A major contribution of this thesis is an extension of BOSS that enables multi-objective BO.

In addition to implementing multi-objective acquisition strategies, it is also important to benchmark them. This is necessary to understand which methods should be used in practical applications. This optimal choice likely depends on the application in question. Potential factors to consider are, for example, prediction accuracy and compu-

tational cost. To this end, the implemented acquisition methods are benchmarked using five synthetic test functions and one constructed using real-world lignin extraction data [9]. These benchmarks demonstrate the efficacy of the implementation and provide insights into the key differences between the methods and their performance. Ultimately, the aim is to provide guidelines for choosing appropriate multi-objective BO settings, and the extended BOSS code offers an accessible toolkit for multi-objective optimization tasks.

1.1 Research objectives

This thesis focuses on implementing and benchmarking multi-objective acquisition strategies for BO. Acquisition methods are selected from existing literature and implemented into the BOSS package [28]. The implemented methods are benchmarked using test functions, most of which are synthetic and one of which is constructed using experimental lignin extraction data [9]. The extended BOSS code is intended to provide users with an easy-to-use toolkit for multi-objective optimization. Then, utilizing benchmarks, the aim is to gain an understanding of the implemented methods. These objectives and research questions are formulated as follows:

1. **Implementation of multi-objective methods**
   Implement multi-objective optimization methods in the Bayesian Optimization Structure Search (BOSS) package. Specifically, implement a multi-objective model class and a few multi-objective acquisition methods selected based on a literature review.

2. **Comparison of multi-objective methods**
   Why are possible differences in performance and behaviour observed across acquisition methods and how can they be explained? Utilizing benchmarks, the goal is to investigate and analyze the methods chosen for implementation.

3. **Guidelines for end-users**
   Given a new multi-objective optimization problem, how should a future user of BOSS choose the multi-objective acquisition method to use? The aim is to identify the settings to which the implemented methods are most suitable. Possible trade-offs between, for example, computational cost and prediction accuracy, are also investigated.

1.2 Thesis outline

In this thesis, Chapter 2 presents the theoretical framework behind multi-objective Bayesian optimization. This is done by first introducing standard, single-objective BO and then generalizing its principles to multi-objective problems. The multi-objective acquisition strategies implemented as a part of this thesis are introduced. Additionally, a brief review of other methods from the literature is provided. Chapter 3 describes the BOSS code, its current functionalities and the changes made to it as a part of this thesis. Additionally, code validation methods and benchmarks are described. Chapter 4 presents the conducted benchmarks and discusses the key results. Finally, Chapter 5 concludes the thesis.

2. Theory

This chapter provides the theoretical background of Bayesian optimization necessary to understand the thesis topic. Section 2.1 provides an overview of Bayesian optimization, while Section 2.2 formulates the multi-objective acquisition strategies implemented in this work. Finally, the chapter concludes with a brief review of related literature.

2.1 Bayesian optimization

Bayesian optimization (BO) is a sequential machine learning technique for the efficient optimization of black-box functions [18]. Throughout this work, the function to be optimized will be referred to as the *objective function*. In general, optimization may refer to the minimization or maximization of the objective function. These are equivalent because one can be transformed into the other by multiplying the objective function by minus one. Thus, this work explicitly considers only minimization problems.

The BO framework is iterative and consists of two essential components: a *surrogate model* and an *acquisition function*. The surrogate model serves as a computationally efficient approximation of the unknown objective function. Gaussian process models are often employed for this purpose. At each iteration of the BO algorithm, the acquisition function is used to acquire a new sample of the objective function. In a sequential setting, which is assumed here, samples are acquired one at a time. This new sample is used to update the surrogate model, gradually leading to a more accurate fit throughout the optimization. The underlying method of Gaussian process models, Gaussian process regression, is detailed in Section 2.1.1. The single-objective BO framework is described in more detail in Section 2.1.2, while the same principles are generalized to multi-objective scenarios in Section 2.1.3.

2.1.1 Gaussian process regression

Gaussian process regression (GPR) is a non-parametric regression technique [24, 1] used by many BO implementations. The goal of regression is to estimate an unknown, real-valued objective function $f(\boldsymbol{x})$ over a continuous domain $\mathcal{X} \subset \mathbb{R}^d$ using a finite set of

observed samples $\{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}_{i=1}^n$. The key idea in GPR is that no specific form of the objective function is assumed. Instead, GPR defines a distribution of possible functions that could explain the observed samples. This is achieved through the use of a Gaussian process (GP). By definition, a GP is a collection of random variables indexed by time or space [24]. In the context of this thesis, these random variables correspond to the values of the objective function at different locations in the search space. Any finite collection of these variables follows a multivariate normal distribution and thus defines a distribution of functions across the domain.

A Gaussian process is fully defined by its mean function and kernel (sometimes called covariance function) [24]. The mean function $m(\boldsymbol{x})$ and kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ of the unknown ground-truth function $f(\boldsymbol{x})$ are defined as the following expectations:

$$m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})], \tag{2.1}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))]. \tag{2.2}$$

Using these, the Gaussian process is written as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')). \tag{2.3}$$

The mean function $m(\boldsymbol{x})$ can be understood as the prediction of the model at $\boldsymbol{x}$. The kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ measures similarity, or correlation, between two input points $\boldsymbol{x}$ and $\boldsymbol{x}'$. Different kernel types capture different types of correlation and the choice of kernel depends on the application. In this work, the Radial Basis Function (RBF) kernel (also known as a squared exponential kernel) is used. The one-dimensional RBF kernel is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_k^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2l^2}\right). \tag{2.4}$$

Here, σ_k^2 is the kernel variance, l is the kernel lengthscale, and $||\cdot||$ denotes the L_2-norm. With the RBF kernel, the correlation between two inputs drops exponentially as the distance between them grows. It is a common choice of kernel due to its applicability to most optimization tasks. Other kernels are, for example, a linear kernel, a Matern kernel or a periodic kernel (which assumes a periodic shape of the ground-truth function).

One-dimensional kernels are combined to form product kernels:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \prod_{i=1}^{d} k_i(\boldsymbol{x}, \boldsymbol{x}'). \tag{2.5}$$

Here, d is the number of dimensions. Product kernels are used in problems with more than one feature, that is, when the search space is multi-dimensional. Note that when using product kernels, the individual kernel variance hyperparameters are multiplied into a single hyperparameter. In contrast, the lengthscales remain independent. Thus, a Gaussian

process regression model has $d + 1$ hyperparameters in total. The hyperparameters are easy to interpret: the kernel lengthscales indicate how rapidly the function value changes while moving in a specific direction in the search space, while the kernel variance reflects the overall value range of the function.

Given a set of input points $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \dots\}$ and their corresponding observed output values $\boldsymbol{y} = \{y_1, y_2, y_3, \dots\}$, the hyperparameters of the GPR model are optimized by maximizing the log marginal likelihood [24]:

$$\log p(\boldsymbol{y}|\mathbf{X}) = -\frac{1}{2}(\boldsymbol{y} - m(\mathbf{X}))^T(\mathbf{K} + \sigma_n^2\mathbb{I})^{-1}(\boldsymbol{y} - m(\mathbf{X})) \tag{2.6}$$

$$-\frac{1}{2}\log\det\left(\mathbf{K} + \sigma_n^2\mathbb{I}\right) - \frac{n}{2}\log(2\pi). \tag{2.7}$$

Here, n is the number of training points, $\mathbf{K}$ is the kernel matrix with elements $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The notation $m(\mathbf{X})$ represents a vector obtained by applying the mean function to every point in $\mathbf{X}$. The noise parameter σ_n^2 is also commonly be treated as a hyperparameter. However, the current BOSS implementation uses a fixed noise parameter set by the user. This parameter is not optimized.

Before fitting to any data, the GPR model has a constant mean function, which is typically set to zero. This prior model is illustrated on the left-hand side of Figure 2.1. The model is fitted to data by optimizing the hyperparameters. This results in a posterior model that captures the behaviour of the objective function, as illustrated on the right-hand side of Figure 2.1. After fitting the model, the mean $\mu(\boldsymbol{x})$ and variance $\sigma^2(\boldsymbol{x})$ predictions at a previously unobserved location $\boldsymbol{x}$ are computed using the following formulas:

$$\mu(\boldsymbol{x}) = \mathbf{K}(\mathbf{X}, \boldsymbol{x})^T[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2\mathbb{I}]^{-1}\mathbf{Y}, \tag{2.8}$$

$$\sigma^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \mathbf{K}(\mathbf{X}, \boldsymbol{x})^T[\mathbf{K}(\mathbf{X}, \mathbf{X} + \sigma_n^2\mathbb{I})]^{-1}\mathbf{K}(\mathbf{X}, \boldsymbol{x}). \tag{2.9}$$

2.1.2 Single-objective optimization

Single-objective Bayesian optimization (BO) is an active machine learning method for the efficient optimization of black-box functions [18]. The primary goal is to identify the global optimum (in this work, the minimum) of an unknown, scalar-valued objective function $f(\boldsymbol{x})$ over a bounded domain $\mathcal{X} \subset \mathbb{R}^d$. In many real-world optimization tasks, the objective function is expensive-to-evaluate, that is, acquiring samples from it is economically or computationally expensive. BO addresses this challenge by minimizing the number of function evaluations required to achieve good prediction accuracy. This efficiency is achieved through intelligent sampling strategies that maximize the information gained from each function evaluation.

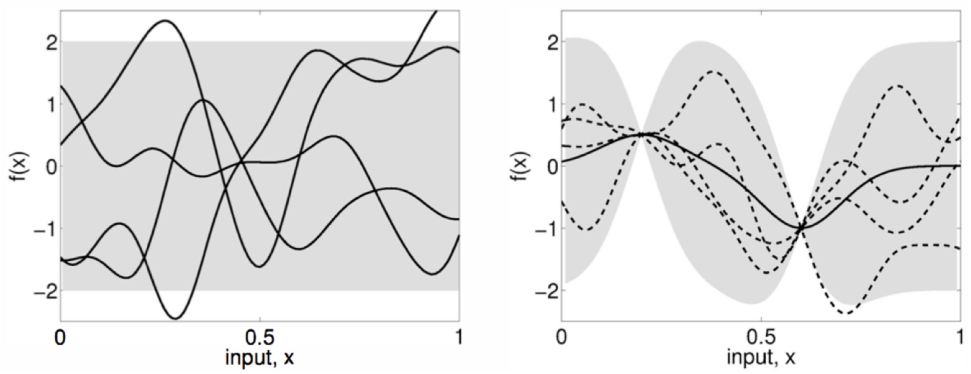


Figure 2.1: Left: four functions drawn at random from the prior GP (model before any data). Right: four functions drawn from the posterior GP (model fitted to data points) and the mean function. In both plots, the shaded regions represent uncertainty, which is lower near observations. Figure adapted from [24].

BO operates iteratively. In a typical sequential setting, a single function evaluation is performed at each iteration of the algorithm. Each new sample is used to update a surrogate model, which serves as a computationally efficient approximation of the objective function. Over successive iterations, the surrogate model provides increasingly accurate predictions of the function landscape, thereby improving the accuracy of the global optimum prediction. Many BO implementations, including BOSS, utilize GPs as surrogate models. This is because they offer a few key advantages:

- GPs are flexible models that do not assume any specific form of the objective function. This makes them suitable for modeling unknown and potentially complex functions.
- GPs are data-efficient models. This is important for BO applications where acquiring more data is typically expensive.
- GPs provide both a mean and a variance prediction for each input value. The mean prediction is commonly taken as the prediction of the function value itself, while the variance prediction provides an intuitive way to handle noise and uncertainty.

The acquisition function $a(\boldsymbol{x})$ incorporates the sampling strategy. At each iteration of the optimization process, this function can be used to assign a numerical fitness value to each input value. The acquisition function is then optimized (in our case, minimized) to determine the location of "optimal" fitness. The $\boldsymbol{x}$ that minimizes the acquisition function is where the objective function is next evaluated:

$$\boldsymbol{x}_{\text{next}} = \operatorname{argmin} a(\boldsymbol{x}). \tag{2.10}$$

There are many different acquisition functions. Many common acquisition functions consider both the mean and variance prediction of the surrogate model to balance

exploitation (sampling areas already known to be promising) and exploration (sampling areas where uncertainty is high). Two common acquisition functions, which are also used in this work, are the Expected Improvement (EI) and the Expected Lower Confidence Bound (ELCB). EI is defined as [18]:

$$\mathrm{EI}(\boldsymbol{x}) = \sigma(\boldsymbol{x}) h\left(\frac{\mu(\boldsymbol{x}) - y^*}{\sigma(\boldsymbol{x})}\right), \tag{2.11}$$

$$h(z) = \phi(z) + z\Phi(z). \tag{2.12}$$

Here, y^* is the best observed objective function value so far, and ϕ, Φ are the standard normal density and distribution functions, respectively. ELCB is defined as [4, 25]:

$$\mathrm{ELCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) - w\sqrt{\sigma^2(\boldsymbol{x})}. \tag{2.13}$$

Here, w is the exploration weight. Setting a suitable exploration weight can be challenging, and previous work has proposed to use [25]:

$$w = \sqrt{2\log_{10}\left(\frac{i^{d/2+2}\pi^2}{3\Delta}\right)}. \tag{2.14}$$

Here, i is the current iteration, d is the number of dimensions, and Δ is an additional parameter set to 0.1 in the BOSS implementation. With this definition of the exploration weight, the ELCB acquisition function initially exploits but turns more explorative as the optimization process continues.

The surrogate model and the acquisition function are combined to form the Bayesian optimization loop, which is depicted in Figure 2.2. In the initialization phase of the algorithm, a GP surrogate model is fitted to initial data points. After this, the optimization loop consists of the following steps:

1. The acquisition function $a(\boldsymbol{x})$ is minimized with respect to $\boldsymbol{x}$ to obtain $\boldsymbol{x}_{\mathrm{next}}$.
2. The objective function is evaluated at $\boldsymbol{x}_{\mathrm{next}}$ to obtain a new data point.
3. The surrogate model is updated with the new data point.

This loop is repeated until the acquisition budget is used or a convergence criterion is met. In this work, a fixed acquisition budget is used as the stopping criterion. After the criterion is met, the surrogate model is minimized to obtain a global minimum prediction μ_{min} and a global minimizer prediction $\boldsymbol{x}_{\mathrm{min}}$.

2.1.3 Multi-objective optimization

The framework for optimizing single-objective functions described in Section 2.1.2 is well studied, but not trivially generalized to scenarios with multiple objectives. This is because objectives can be conflicting, and finding a feature vector that minimizes all of

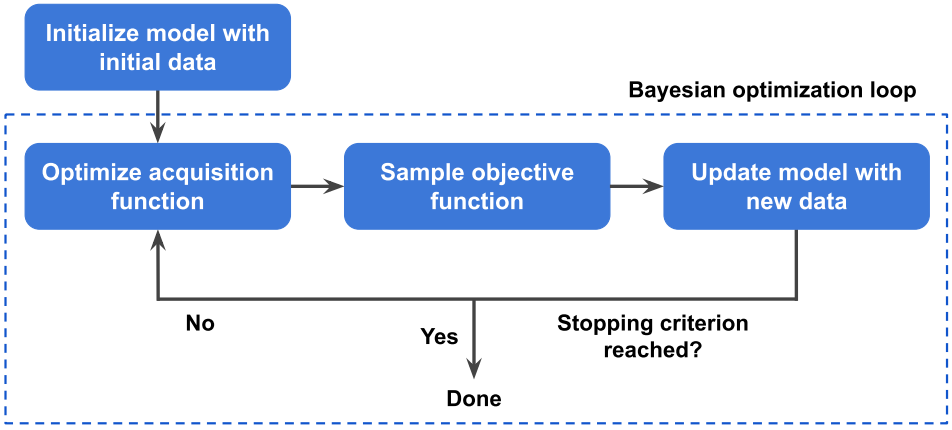


Figure 2.2: The general BO workflow. At the start, a surrogate model is initialized using initial data. In the Bayesian optimization loop, (1) the acquisition function is minimized to obtain $\boldsymbol{x}_{\text{next}}$, (2) the objective function is evaluated at $\boldsymbol{x}_{\text{next}}$ and (3) the surrogate model is updated using data from the new acquisition. The loop is repeated until the stopping criterion is met. The stopping criterion may be a convergence criterion or an acquisition budget.

them simultaneously might be impossible. This section generalizes Bayesian optimization to multi-objective cases by considering optimal trade-offs between the different objectives. The topic is approached by introducing the key concepts of the Pareto front and hypervolume.

The Pareto front

In a multi-objective optimization task, the goal is to minimize a vector-valued objective function $\boldsymbol{f}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}), \dots\}$ within a bounded domain $\mathcal{X} \subset \mathbb{R}^d$. The objectives $f_i(\boldsymbol{x})$ are assumed to be independent and potentially conflicting. Because of this, there may not exist a single solution that minimizes all the objectives simultaneously. Therefore, one must seek optimal trade-offs: solutions in which none of the objectives can be improved without deteriorating another. Solutions $\boldsymbol{x}^P$ that satisfy this condition are known as Pareto-optimal solutions. The corresponding points $\boldsymbol{f}(\boldsymbol{x}^P)$ in the objective space are non-dominated and, in a minimization task, they satisfy [33]:

$$f_i(\boldsymbol{x}^P) \leq f_i(\boldsymbol{x}) \quad \forall\, i, \tag{2.15}$$

$$f_i(\boldsymbol{x}^P) < f_i(\boldsymbol{x}) \quad \text{for some } i. \tag{2.16}$$

In words, a non-dominated solution $\boldsymbol{f}(\boldsymbol{x}^P)$ is less than or equal to any other solution $\boldsymbol{f}(\boldsymbol{x})$ in every dimension, and strictly less in at least one dimension. In the objective space, the points $\boldsymbol{f}(\boldsymbol{x}^P)$ make up the Pareto front. Thus, searching for Pareto-optimal solutions in the search space is equivalent to searching for the Pareto front in the objective space. In

practice, multi-objective optimization algorithms attempt to find an approximation for the Pareto front, and in turn, a set of Pareto-optimal solutions.

The true Pareto front is often an infinite set of points. Therefore, multi-objective optimization algorithms aim to identify a finite approximation $\mathcal{P}$ of the front [16, 19]. In this work, $\mathcal{P}$ is taken to be the set of acquisitions made by the optimization algorithm whose corresponding objective vectors are non-dominated. Rather than using the observed function values when filtering the dominated and non-dominated points, this work uses the means of the joint posterior distribution to estimate the true function values. This ensures better compatibility in applications where the observations may be corrupted by noise. In a noiseless scenario, if the noise hyperparameter of the surrogate model is fixed to zero, this approach is equivalent to using the observed values.

To illustrate the Pareto-optimal solutions and the Pareto front, consider the problem of minimizing the bi-objective function $\boldsymbol{f}(x) = \{f_1(x), f_2(x)\}$ over the domain $x \in [-3, 3]$. In this example, the independent objectives are defined as:

$$f_1(x) = \sqrt{1 + x^2}, \tag{2.17}$$

$$f_2(x) = 4 + 2\sqrt{1 + (x - 1)^2}. \tag{2.18}$$

These functions are shown on the left-hand side of Figure 2.3. Because in the region $x \in [0, 1]$ either increasing or decreasing x improves one objective and deteriorates the other, these x values are the Pareto-optimal solutions. The objective space is depicted on the right-hand side of Figure 2.3. All possible solutions form the feasible space, and solutions corresponding to $x \in [0, 1]$ map to the Pareto front, which represents Pareto-optimal trade-offs between the objectives.

The hypervolume indicator

The goal of a multi-objective optimization algorithm is to provide the decision-maker with a set of solutions that approximates the true Pareto front. To achieve this, it is necessary to quantify the quality of a point set as a Pareto front approximation and to establish preferences over different point sets. The hypervolume indicator is the most widely used set-quality indicator for this purpose. It is defined as the volume of the objective space dominated by a given point set $S \subset \mathbb{R}^d$ and bounded above by a reference point $\boldsymbol{r}_{\text{ref}} \in \mathbb{R}^d$ [13]. Mathematically, the hypervolume indicator is interpreted as the measure of the union of boxes:

$$\text{HV}(S) = \Lambda\left(\bigcup [\boldsymbol{s}, \boldsymbol{r}_{\text{ref}}]\right). \tag{2.19}$$

Here, $\Lambda(\cdot)$ denotes the Lebesgue measure and $[\boldsymbol{s}, \boldsymbol{r}_{\text{ref}}]$ is the hyperrectangle bounded below by $\boldsymbol{s} \in S$ and above by $\boldsymbol{r}_{\text{ref}}$. For the computation to be valid, the reference point $\boldsymbol{r}_{\text{ref}}$ must not dominate any point in S.

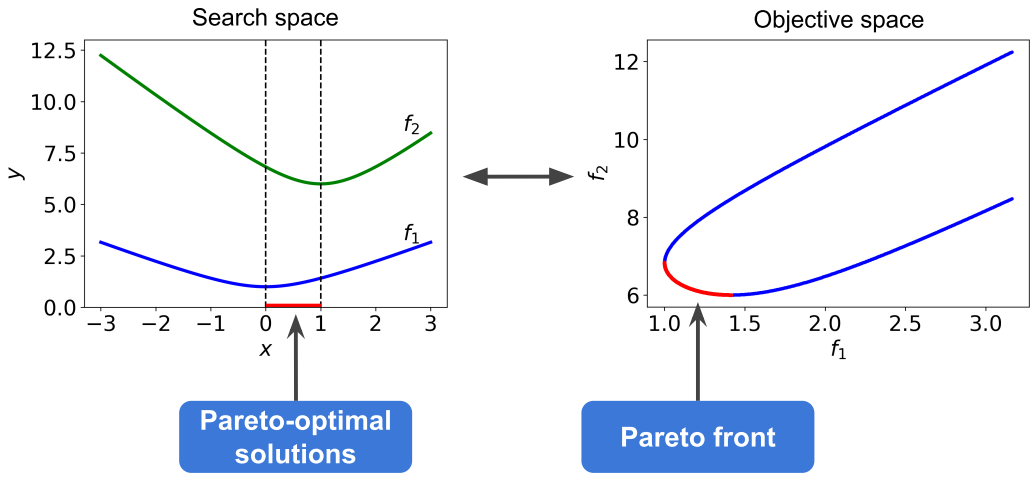


Figure 2.3: Functions $f_1(x) = \sqrt{1+x^2}$ and $f_2(x) = 4+2\sqrt{1+(x-1)^2}$ plotted in the search space (left) and the objective space (right). In the search space, the functions themselves are plotted as functions of the input variable x. In the objective space, the feasible space (blue curve) shows all possible combinations of the two objectives that can be obtained. The Pareto front (red curve) is a subset of the feasible space, and it corresponds to the set of Pareto-optimal solutions.

The hypervolume indicator implicitly considers three desirable characteristics of a Pareto front approximation: proximity to the true front, spread (how far apart the points are), and homogeneity (how evenly the points are distributed) [13]. A point set that performs well with respect to these criteria will yield a high hypervolume value. This property makes the hypervolume a particularly attractive objective for multi-objective optimization. In the absence of noise, any increase in hypervolume guarantees an improvement in the quality of the Pareto front approximation. Therefore, an optimal approximation can be obtained by maximizing its hypervolume.

Due to its utility in evaluating the quality of a point set as a Pareto front approximation, the hypervolume indicator can also be used to construct acquisition functions for multi-objective optimizers. Hypervolume-based acquisition functions select the next acquisition point so that the added hypervolume (or the expectation of this) to the current set of acquisitions is maximized. Acquisition functions based on the hypervolume indicator are further discussed in Section 2.2.3.

Despite its strengths, the hypervolume indicator comes with a high computational cost. The dependency of the computational cost on the number of objectives varies depending on the algorithm being used, but can be exponential in the worst case [13]. This can limit the practicality of the hypervolume indicator, especially in acquisition functions where the hypervolume improvement needs to be evaluated for a large number of candidate points. Additionally, in highly noisy scenarios, the hypervolume indicator may become unreliable, as noisy observations can appear to dominate regions of the true Pareto front, thereby distorting the quality assessment.

2.2 Acquisition strategies

This chapter presents the acquisition strategies for multi-objective Bayesian optimization that were implemented into the BOSS code as a part of this thesis. Section 2.2.1 introduces the baseline method, a straightforward generalization of single-objective acquisition functions to multi-objective applications. Section 2.2.2 introduces scalarization-based methods, in which the different objectives are combined into one using a transformation function. Section 2.2.3 introduces methods based on the hypervolume indicator. Finally, Section 2.2.4 provides a brief overview of alternative acquisition methods from the literature.

2.2.1 Baseline method

The standard ELCB acquisition function is used as a baseline method in this work. Although ELCB is typically suitable for only single-objective optimization tasks, it can be easily applied to multi-objective problems. This is done by selecting only one of the multiple objectives, fitting a GP surrogate model on that, and minimizing ELCB with respect to the resulting single-objective model. Throughout the BO run, the algorithm iterates over the objectives in order, starting from the first one. Effectively, the chosen acquisition locations will be points where the specific objective under consideration is believed to be promising without considering the others. In principle, this method can be used with any other acquisition function suitable for single-objective BO.

This method was chosen as the baseline because it is the simplest way to extend the widely used ELCB acquisition function to multi-objective scenarios without modifying the function itself. This offers a fair ground for comparison: more sophisticated methods specifically designed for multi-objective optimization should outperform this baseline to be worth consideration.

2.2.2 Scalarization methods

A straightforward approach to ensure compatibility between the single-objective BO framework and multi-objective objective functions is to combine the different objectives into a single one. This conversion makes multi-objective problems compatible with the pre-existing single-objective BO framework and allows the use of well-known acquisition methods.

Scalarization utilizes a transformation function g that operates on a vector and returns a scalar: $y = g(\boldsymbol{y}, \boldsymbol{\lambda})$. Here, $\boldsymbol{\lambda}$ is a vector of weights for which $\sum_i \lambda_i = 1$. Many different transformation functions are possible. For this work, two are considered. The

first is the WeightSum (WS) function, which takes the weighted sum of the objectives:

$$\mathrm{WS}(\boldsymbol{y}) = \sum_{i=1}^{n} \lambda_i y_i. \tag{2.20}$$

The second is the augmented Tchebycheff (AT) function:

$$\mathrm{AT}(\boldsymbol{y}) = \max(\mathrm{prod}(\boldsymbol{\lambda}, \boldsymbol{y})) + \beta \sum_{i=1}^{n} \lambda_i y_i. \tag{2.21}$$

Here, prod$(\cdot)$ denotes the element-wise product of two vectors and the max$(\cdot)$ selects the largest element of the resulting vector. The small constant β is set to 0.05. The augmented Tchebycheff function has the advantage with respect to the simpler alternative in Equation 2.20 that it can explore concave parts of the Pareto front [19, 26].

A common approach to scalarization, and the one used in this work, is ParEGO [19]. Using the selected transformation function and a random vector of weights, this approach transforms the vector-valued output data so that each input corresponds only to a scalar-valued output. A GP surrogate model is fitted to this data, and the standard Expected Improvement acquisition function is minimized with respect to the model. This yields a minimum, which is taken as the next acquisition point where the objective function is evaluated. By selecting a different weight vector at each BO iteration, an approximation of the Pareto front can be gradually built. Here, the weight vector is drawn at random from a uniform distribution at each BO iteration. In this work, the ParEGO approach is applied using the augmented Tchebycheff and the WeightSum transformation functions. To distinguish between these approaches, the former is henceforth referred to as Scal-AT and the latter as Scal-WS.

2.2.3 Hypervolume methods

The hypervolume metric introduced in Section 2.1.3 can be utilized to construct acquisition functions specific to multi-objective problems. Acquisition methods based on the hypervolume indicator select new acquisitions such that the volume of the objective space dominated by the entire set of acquisitions is increased. Due to the properties of the hypervolume indicator, this leads to a more accurate approximation of the true Pareto front. In the following, two methods based on the hypervolume indicator are introduced: the Hypervolume Improvement and the Expected Hypervolume Improvement with three different variants.

Hypervolume Improvement (HVI) is a well-known acquisition function based on the hypervolume indicator [10]. The HVI of a candidate point $\boldsymbol{x}$ and its corresponding objective values $\boldsymbol{y}(\boldsymbol{x})$, with respect to the current Pareto front approximation $\mathcal{P}$ (that is, the set of non-dominated acquisitions acquired during the optimization process so far), is

defined as:

$$\mathrm{HVI}(\mathcal{P}, \boldsymbol{y}) = \mathrm{HV}(\mathcal{P} \cup \boldsymbol{y}) - \mathrm{HV}(\mathcal{P}). \tag{2.22}$$

Because the true objective function values are generally unknown at unseen locations, the mean of the joint posterior distribution is used as an estimate for these values. Specifically, in practical calculations, $\boldsymbol{m}(\boldsymbol{x}) = \{m_1(\boldsymbol{x}), m_2(\boldsymbol{x}), \cdots\}$ replaces $\boldsymbol{y}(\boldsymbol{x})$ in the above equation. HVI indicates how much the hypervolume of the Pareto front approximation would improve if a new observation, obeying the joint posterior distribution, were added to it. It does not consider the posterior variance but aims to directly maximize the hypervolume of the Pareto front approximation. Consequently, HVI is expected to sample points that are known to be promising based on previous data, but may not adequately sample points where the uncertainty is high (if the mean prediction is not promising at those points).

To mitigate the potential over-exploitation tendency of HVI, we propose adding an element of exploration to the acquisition function. This can be achieved by introducing an explicit exploration term:

$$\mathrm{HVI}(\mathcal{P}, \boldsymbol{y}, \alpha) = \mathrm{HV}(\mathcal{P} \cup [\boldsymbol{y} - \alpha \boldsymbol{\sigma}^2(\boldsymbol{x})]) - \mathrm{HV}(\mathcal{P}) \tag{2.23}$$

Here, $\boldsymbol{\sigma}^2(\boldsymbol{x}) = \{\sigma_1^2(\boldsymbol{x}), \sigma_2^2(\boldsymbol{x}), \cdots\}$ represents a vector of variance values from the joint posterior distribution and α is the exploration weight. Effectively, adding the exploration term reduces the value of the acquisition function in areas of high uncertainty, making them more likely to be minima of the acquisition function. In the following parts of this section, the standard form of HVI is assumed (that is, the exploration weight is zero).

The Expected Hypervolume Improvement (EHVI) accounts for the uncertainty in the model predictions by computing the expectation of HVI. The EHVI of a candidate point $\boldsymbol{x}$ is defined as [32]:

$$\mathrm{EHVI}(\mathcal{P}, \boldsymbol{y}) = \int_{\mathbb{R}^d} \mathrm{HVI}(\mathcal{P}, \boldsymbol{y}) \cdot \mathrm{PDF}_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{y}) d\boldsymbol{y} \tag{2.24}$$

Here, $\mathrm{PDF}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is the multivariate normal distribution with a mean vector $\boldsymbol{\mu}$ and a standard deviation vector $\boldsymbol{\sigma}$. EHVI considers the uncertainty in the objective function predictions by integrating over the posterior variance. It therefore facilitates exploration more naturally than HVI. Evaluating the integral in Equation 2.24 is non-trivial, and analytic solutions can be difficult to obtain. As a part of this work, an analytic solution with gradients has been implemented only for bi-objective cases [32].

Numerical estimation methods can be used in cases with more than two objectives. In particular, Monte Carlo (MC) integration is often used [5]. This method involves drawing N sample points from the joint posterior distribution at the candidate point $\boldsymbol{x}$, computing the HVI individually for each of them, and then taking the mean. The

MC-based Expected Hypervolume Improvement (MC-EHVI) is defined as:

$$\text{MC-EHVI}(\mathcal{P}, \boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \text{HVI}(\mathcal{P}, S_i) \tag{2.25}$$

Here, S is a set of N samples drawn from the joint posterior distribution. This equation is equivalent to Equation 2.24 up to the Monte Carlo integration error. Using *i.i.d.* samples is the most straightforward approach, and with this, the estimation error scales as $1/\sqrt{N}$. However, the error can be reduced by using quasi-random samples [5], which is the approach used in this work. The accuracy of MC-EHVI depends on the number of MC samples drawn from the joint posterior distribution. This dependency is investigated in Section 4.4.3. Based on the results detailed later, the default number of samples is set to $N = 128$ in the BOSS implementation.

The analytic form of EHVI (Equation 2.24) and its Monte Carlo approximation (Equation 2.25) assume exact, noiseless observations of the objective function. However, this assumption is often not valid in practical applications, and noise in objective function evaluations should be accounted for. The MC-based approximation of EHVI can be generalized to noisy scenarios in a straightforward manner. In addition to drawing N samples from the joint posterior distribution at the candidate location $\boldsymbol{x}$, the approach also involves drawing N samples of the current Pareto front approximation. This is accomplished by drawing samples from the joint posterior distribution at all previously observed locations $\mathcal{X}_{\text{obs}} \subset \mathbb{R}^d$, resulting in a set of Pareto fronts denoted by $\mathcal{P}_S$. The noisy Expected Hypervolume Improvement (nMC-EHVI) is then defined [6] as:

$$\text{nMC-EHVI}(\mathcal{P}_S, \boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \text{HVI}(\mathcal{P}_i, S_i). \tag{2.26}$$

The equation is equivalent to Equation 2.25 in the noiseless case. While nMC-EHVI is expected to perform better in noisy scenarios compared to the analytic EHVI and MC-EHVI, it suffers from a significantly higher computational cost. This is because the algorithm must draw N samples of the Pareto front approximation from the joint posterior distribution and compute the hypervolume separately for each of them. In this work, the computational cost is reduced by fixing the set of Pareto front samples across repeated evaluations of the acquisition function [6].

The acquisition functions introduced in this section are based on the hypervolume indicator, which requires the specification of a reference point to compute the hypervolume of a point set. Importantly, the hypervolume is sensitive to the choice of this reference point [17]. A reference point placed too far from the true Pareto front tends to favour solutions at the extremes of the front. Conversely, if the reference point is too close, it favours less extreme points.

In this work, following previous studies [5, 32], a fixed, problem-specific reference point is always used. An alternative approach is to use a dynamic reference point strategy in which the reference point is selected separately for each BO iteration. In this strategy, the reference point is set to be slightly worse than the current observed Pareto front in each dimension, that is, $\boldsymbol{r}_{\text{ref}} = \boldsymbol{y}_{\text{nadir}} + 0.5 \cdot |\boldsymbol{y}_{\text{nadir}}|$, where $\boldsymbol{y}_{\text{nadir}}$ is the component-wise maximum of the current Pareto front approximation. While the dynamic reference point strategy can adapt to a Pareto front approximation that evolves across BO iterations, it may hinder exploration in cases where the true Pareto front is discontinuous. Specifically, if a candidate point is dominated by the current reference point, its hypervolume improvement is computed as zero, and it cannot be a minimizer of the acquisition function. As a result, such regions would remain unexplored.

In this section, the Hypervolume Improvement (HVI) and the Expected Hypervolume Improvement (EHVI) acquisition functions were introduced. HVI contains an exploration term that is set to zero unless noted otherwise. EHVI comes in three variants. The analytic and differentiable EHVI is applicable to bi-objective problems. Its MC-based approximations, MC-EHVI and nMC-EHVI, can be applied to arbitrary problems.

2.2.4 Related work

Extensive literature exists on additional acquisition strategies for multi-objective BO. While only a few methods were selected for this thesis, a broader overview (although not exhaustive) of related literature is provided here. For comparative studies, see [15], which compares the performances of conceptually different acquisition methods in material design problems, and [29], which considers the strengths and weaknesses of different methods from a more theoretical perspective.

The ParEGO approach [19] can be expanded in a way that incorporates user preferences [14]. In this interactive method, users provide feedback on the acquisitions by specifying constraints on the objectives. An alternative scalarization-based approach incorporates user preferences by enforcing a prior over the weight vector values $\boldsymbol{\lambda}$ and an application-specific transformation function [21]. This method can target specific sections of the Pareto front, which may be particularly useful in applications where parts of the front are known to be infeasible.

The single-objective Probability of Improvement acquisition function can be generalized to multi-objective scenarios using the hypervolume indicator [10]. This acquisition function quantifies the probability that a new acquisition lies in non-dominated space. A modification of this approach, intended to make it less exploitative, has also been developed [11]. SMS-EGO is an alternative method based on the hypervolume indicator, and it uses optimistic estimates of the objective function in a UCB fashion and selects the

next acquisition so that the hypervolume improvement is maximized [22]. For the EHVI acquisition function, an analytic form with gradients has been derived for problems with three objectives [31]. Other works have also derived gradients for EHVI in arbitrary cases using a re-parametrization trick and auto-differentiation [5]. A truncated version of EHVI has also been developed, which can be useful in cases where the objective function values are known to be within certain ranges. [30].

Conceptually different approaches are, for example, those based on an entropy measure. Predictive entropy search (PESMO) uses an acquisition function that selects the next samples by attempting to maximize the information gain in the search space [16]. Max-value entropy search (MESMO) is a more recent entropy-based acquisition function that attempts to maximize the information gain in the objective space [2]. MESMO claims to be more robust than PESMO, offering reduced computational cost and more accurate predictions.

In this work, only sequential optimization settings are considered. However, the MC-based EHVI acquisition function can also be extended to batch acquisition settings [5]. An alternative approach for batch acquisitions is the diversity-guided (DGEMO) approach, which optimizes for both hypervolume improvement and diversity of selected samples [20].

3. Computational implementation

This chapter describes the BOSS code and the changes made to it as a part of this thesis. The implementation of multi-objective models and acquisition methods consists of two parts: modifications made to the BOSS code and a separate benchmarking code used to validate and compare the acquisition methods. Section 3.1 first provides a general overview of the BOSS library, and subsequently describes the changes made to it. Section 3.2 describes the benchmarking code and the test functions used to benchmark the acquisition methods, as well as the metrics used to analyze their performance.

3.1 BOSS: Bayesian Optimization Structure Search

Bayesian Optimization Structure Search (BOSS) is a Python package for general-purpose Bayesian optimization [28]. The code is designed to be easily usable by researchers for applications in many fields of research. The code is under continuous development at the AI-based Materials Science group at the Technical University of Munich, the Computational Electronic Structure Theory group at Aalto University and the Materials Informatics Laboratory at the University of Turku. It is distributed as `aalto-boss` on the PyPi repository and can be easily used via the Python API. The open source code is hosted on GitLab: https://gitlab.com/cest-group/boss.

In the following sections, a short review of current BOSS functionalities and the new code developed for this work is provided. Additional information, such as installation instructions, documentation, and tutorials, can be found on the GitLab repository and the official BOSS website: https://sites.utu.fi/boss.

3.1.1 Current functionality

BOSS facilitates single-objective BO by fitting a GP surrogate model, enhancing the model through iterative, intelligent sampling of the objective function, and predicting its global optimum. The user interacts with BOSS mainly through keywords, which are used to set the optimization conditions. For example, the user must pass the objective function (defined as a Python function) using a specific keyword. Any other settings are also set

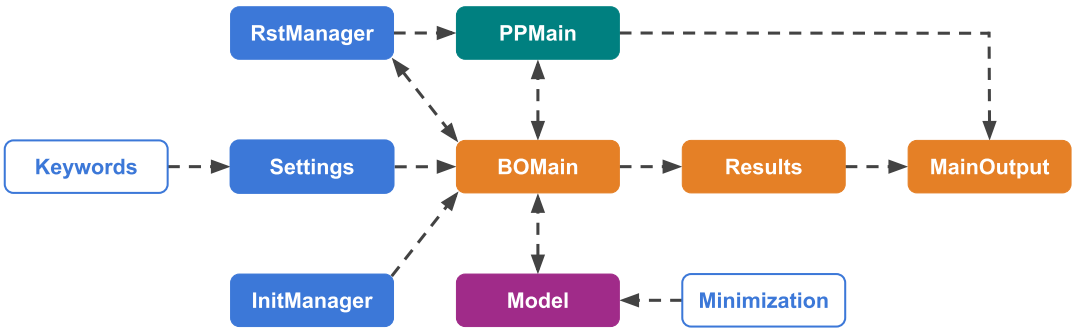


Figure 3.1: The most important modules and internal structure of BOSS.

by using their respective keywords.

BOSS tracks key performance metrics for monitoring the optimization, such as the predicted global optimum and kernel hyperparameters. These metrics are stored in an output file during the optimization. If interrupted, the optimization can be restarted using a restart file. BOSS also offers post-processing features to visualize the results of an optimization run. For example, the user can plot the evolution of the surrogate model and the key performance metrics throughout the optimization process.

The BOSS code consists of multiple modules, each with its own task in the code structure. The internal structure of BOSS is depicted in Figure 3.1. The most important modules are:

- `Keywords`: Contains all keywords that may be set by the user.
- `Settings`: Sets and contains all settings of the BO algorithm that determine the optimization conditions. The settings are set based on the keywords.
- `BOMain`: The main module that implements BO and contains the optimization loop.
- `Model`: Implements the surrogate models using Gaussian process models, along with functionalities such as making predictions and optimizing hyperparameters.
- `Minimization`: Contains utilities for minimizing surrogate models (for global minimum prediction) and acquisition functions (for determining the next acquisition).
- `PPMain`: Contains post-processing functionalities.
- `InitManager`: Initializes the initial data points using, for example, Sobol points.
- `RstManager`: Handles restarting the optimization run from a restart file.
- `Results`: Stores data, such as kernel hyperparameters, during the optimization process. After the optimization is finished, the results object can be used to access information about the run.

- `MainOutput`: Handles writing the output and restart files.

In BOSS, determining the next acquisition at each BO iteration is implemented using acquisition *functions* and acquisition *managers*. An acquisition function is a callable object that, given a feature vector, returns a scalar fitness value. This represents the mathematical function that is optimized at each iteration to determine the next sample. An acquisition manager serves as an interface between the acquisition function and the rest of the BOSS code. It handles calling and optimizing the actual acquisition function. For example, the standard single-objective ELCB and EI are implemented as acquisition functions. They are used through a manager that, when called, minimizes the function in question and returns the next acquisition.

3.1.2 New functionality

One of the main contributions of this work is the implementation of multi-objective optimization support in the BOSS code. This was achieved through a two-step process. The first step involved developing a multi-objective surrogate model class. The newly introduced `ListModel` class generalizes the existing single-objective surrogate model by maintaining a list of Gaussian process (GP) regression models, with each model corresponding to one of the objectives. These models are assumed to be independent but share the same input data (*i.e.*, observation locations). The use of this new model is different from the pre-existing models in several ways. First, during the initialization of the model, the output data needs to contain multiple objectives, each of which is assigned an independent GP model. At prediction time, the model returns outputs for all objectives simultaneously. Second, the (homoscedastic) noise variance can be specified separately for each objective by passing a list of values, as opposed to a single scalar.

The second major step was the implementation of multiple new acquisition functions designed specifically for multi-objective optimization. This was achieved by adding five new acquisition functions and two new acquisition managers. The new modules related to this are:

- `hvi` (acq. function): the basic Hypervolume Improvement acquisition function that seeks to directly maximize the hypervolume of the Pareto front approximation. Contains a parameter to control the exploration weight (defaults to zero).

- `ehvi` (acq. function): the exact EHVI acquisition function with gradients for problems with two objectives.

- `mcehvi` (acq. function): the MC-based EHVI acquisition function applicable to any number of objectives.

- `nmcehvi` (acq. function): the MC-based noisy EHVI acquisition function applicable to any number of objectives.
- `multielcb` (acq. manager): handles the application of the standard ELCB acquisition function to multi-objective cases. Cycles through the objectives in order, optimizing ELCB with respect to only one of them at a time.
- `multiobjective` (acq. manager): an acquisition manager that handles hypervolume-based acquisition functions. Contains methods that are shared between the HV-based acquisition functions to allow for easy implementation of additional functions.
- `scalarization` (acq. manager): an acquisition manager that facilitates the scalarization-based approach and minimizes the pre-existing Expected Improvement acquisition function over a surrogate model of the scalarized objectives.

With all the hypervolume-based acquisition functions, a reference point can be set for the hypervolume computations. If the reference point is not specified by the user, a dynamic reference point is used, as described earlier in Section 2.2.3. For gradient-free acquisition functions, BOSS automatically uses the finite-difference method to estimate the missing gradients. Specifically, the L-BFGS-B algorithm is used for function minimization. The `pygmo` library is used for hypervolume computations [3].

In addition to the two major additions described above, smaller modifications were made to the existing BOSS code to enable the support of multi-objective optimization. For example, it was necessary to modify some data structures to support multiple distinct kernels required by the new multi-objective model. New keywords were also added to handle the initialization of the new acquisition methods (mainly, setting the reference point and any parameters required by the new acquisition functions).

3.2 Code validation and benchmarks

A benchmarking code separate from the BOSS code was developed as a part of this thesis work. The code was written in Python and it was used to validate the new extensions of BOSS and to benchmark the implemented multi-objective acquisition strategies. In short, the benchmarking code executes repeated optimization runs using BOSS, tracks the results, performs post-processing tasks and generates visuals. The code, as well as instructions for installation and use, are available in a public GitLab repository: https://gitlab.com/tlinnala/mobo-bench.

The remaining parts of this Section will describe the methods of code validation. Section 3.2.1 describes the specific performance metrics used to evaluate the performance

of multi-objective optimizers. Furthermore, Section 3.2.2 introduces the six test functions chosen for this study. The actual benchmark results are presented and analyzed later in Chapter 4.

3.2.1 Performance metrics

In this work, several performance metrics are used to analyze the performance of different acquisition strategies. The utility of performance metrics is to provide information about the quality of the point set acting as a Pareto front approximation. Specifically, given a point set, the metrics aim to measure its hypervolume, inhomogeneity and mean distance to the true Pareto front. In addition to measuring the quality of the Pareto front predictions, the computational cost of acquisition methods is evaluated by measuring the acquisition time.

Hypervolume: The most important measure of the quality of a point set as a Pareto front approximation is its hypervolume. The hypervolume of a point set, as described in Section 2.1.3, is the volume (area, in the case of two objectives) in the objective space dominated by the set and bounded above by a reference point. In this work, the difference between the hypervolume of a point set $\mathcal{P}$ and the hypervolume of the true Pareto front $\mathcal{P}_T$ is used as a performance metric:

$$\Delta\text{HV} = \text{HV}(\mathcal{P}_T, \boldsymbol{r}_{\text{ref}}) - \text{HV}(\mathcal{P}, \boldsymbol{r}_{\text{ref}}) \tag{3.1}$$

Here, $\boldsymbol{r}_{\text{ref}}$ is the problem-specific reference point. The optimal value of ΔHV is zero. In a noiseless case, this is achieved when the predicted Pareto front corresponds perfectly to the true Pareto front. Additionally, in a noiseless case, $\Delta\text{HV} \geq 0$ always holds. This is because noiseless observations can never dominate a point on the true Pareto front. However, if observations are corrupted by noise, it is possible that $\Delta\text{HV} < 0$. This occurs if the predicted Pareto front is overly optimistic. Furthermore, in a noisy case, $\Delta\text{HV} = 0$ does not guarantee that the predicted Pareto front corresponds perfectly to the true front. This is because the predicted front may be overly optimistic in one region of the objective space and overly pessimistic in another. These effects could cancel out so that the hypervolume of the predicted front equals the hypervolume of the true front, even if the two fronts have different shapes.

Inhomogeneity: The second metric is inhomogeneity (S-metric, also called spacing) [27]. The S-metric describes the overall spread of a point set serving as a Pareto front approximation in the objective space. The S-metric of a point set is defined as:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \hat{d})^2}. \tag{3.2}$$

Here, d_i is the Euclidean distance between the i:th and the $i+1$:th sorted (by the first component) points in the set and $\hat{d}$ is the mean of these distances. A uniform point set gives a value of S that is close to zero, and when all the distances equal the mean distance (the distribution of points is perfectly uniform), $S = 0$. Hence, the S-metric can be used to analyze how homogeneously the points in the Pareto front approximation are spread across the objective space.

Distance: The next two performance metrics quantify explicitly how close the predicted Pareto-optimal solutions and Pareto front are to their ground-truth counterparts. Let $\mathbf{X}$ be a set of predicted Pareto-optimal solutions and $\mathbf{Y}$ their corresponding objective vectors. Similarly, let $\mathbf{A}$ be the true Pareto-optimal solutions and $\mathbf{B}$ their corresponding objective vectors. The mean distances in the search and objective spaces are defined as:

$$d_x = \frac{1}{|\mathbf{X}|} \sum_{\boldsymbol{x} \in \mathbf{X}} \min_{\boldsymbol{a} \in \mathbf{A}} ||\boldsymbol{x} - \boldsymbol{a}||, \tag{3.3}$$

$$d_y = \frac{1}{|\mathbf{Y}|} \sum_{\boldsymbol{y} \in \mathbf{Y}} \min_{\boldsymbol{b} \in \mathbf{B}} ||\boldsymbol{y} - \boldsymbol{b}||. \tag{3.4}$$

Here, $|\cdot|$ denotes the number of vectors in a set, and $||\cdot||$ is the L_2-norm of two vectors. In essence, d_x is the mean of the shortest distances from the predicted Pareto-optimal solutions to any true Pareto-optimal solution. Similarly, d_y is the mean of the shortest distances from the predicted Pareto front to any point on the true Pareto front. The optimal value of both metrics is zero, which is obtained if the predicted point sets correspond perfectly to their ground-truth counterparts. The metrics provide additional insight into how close the approximated Pareto front is to the true Pareto front in both spaces. To interpret the obtained values of these metrics meaningfully, they should be considered relative to the size of the input domain and the range of the true Pareto front of the test function in question.

Acquisition time: The last performance metric does not measure the quality of the Pareto front prediction but the computational resources required to obtain it. Specifically, acquisition time refers to the CPU wall-clock time taken by the algorithm to determine the next acquisition point at each iteration of the BO algorithm. This corresponds to the time required to optimize the acquisition function, that is, the first step of the BO optimization loop (see Figure 2.2). In the remainder of this work, the acquisition time is also referred to as *computational cost*.

In this work, the analysis of benchmarks focuses primarily on the ΔHV-metric due to its theoretical properties (that is, the hypervolume difference reaches zero when a perfect approximation of the true Pareto front is obtained). However, this metric may not adequately inform, for example, if the Pareto front has been sampled homogeneously. The S-metric is therefore used as a supporting metric to offer additional insight into the differences in sampling between acquisition methods. On the other hand, the S-metric

Function	Parameters	Objectives	Problem type	Reference point	Max HV
ZDT1	2	2	cont.	(2.0, 2.0)	3.6641
ZDT2	2	2	cont.	(2.0, 2.0)	3.3308
ZDT3	2	2	discont.	(2.0, 2.0)	4.8102
Branin-Currin	2	2	discont.	(40.0, 8.0)	206.74
Viennet	2	3	discont.	(20.0, 30.0, 1.0)	327.66
Lignin	3	2	discont.	(0.0, 0.0)	515.14

Table 3.1: The examined test functions. The problem type indicates whether the Pareto-optimal solutions in the search space consist of a single (continuous) or multiple (discontinuous) regions of optimal solutions. The reported maximum hypervolume is computed with respect to the given reference point.

may not be suitable for scenarios where the Pareto front is discontinuous. In such a case, the algorithm may only acquire samples from a single section of the Pareto front. This could result in a value of the S-metric that provides an overly positive picture of the performance. Additionally, the d_x and d_y metrics are used to analyze how far the predicted Pareto front is from the true one. Information provided by these two metrics is especially useful in noisy scenarios, where an overly optimistic Pareto front prediction may result in misleading values of ΔHV and S. In short, none of the metrics are perfect, but when analyzed together, they can be used to obtain a good understanding of the differences between different acquisition methods.

3.2.2 Test function suite

Six multi-objective test functions were implemented as a part of the benchmarking code to test the implementation and benchmark the different acquisition strategies. Five of the test functions are synthetic and from existing literature. However, one of them is constructed using experimental lignin extraction data [9]. The functions were selected based on their properties and assumed level of difficulty. Essentially, the difficulty of a multi-objective function depends on the shape of its objectives across the search space, the shape of the Pareto front, and the number of features and objectives. During function selection, the number of objectives was limited to three, since adequately benchmarking functions with more than three objectives was computationally too expensive, especially when using hypervolume-based methods. The reference point required for hypervolume computations was manually selected for each test function based on the known true Pareto fronts. The functions are summarized in Table 3.1. The input domains and the approximate true Pareto front ranges are listed in Table 3.2. The test functions are described in more detail below.

Function	Parameter domain	PF range
ZDT1	$[0, 1]^2$	$[0, 1]^2$
ZDT2	$[0, 1]^2$	$[0, 1]^2$
ZDT3	$[0, 1]^2$	$[0, 1] \times [-1, 1]$
Branin-Currin	$[0.001, 1.0]^2$	$[0, 35] \times [0, 7]$
Viennet	$[-3, 20] \times [-20, 3]$	$[0, 8] \times [14, 18] \times [0, 0.2]$
Lignin	$[250, 1000] \times [160, 195] \times [0.25, 2]$	$[-20, 0] \times [-50, 0]$

Table 3.2: The input parameter domains and approximate Pareto front (PF) ranges for each examined test function. The PF ranges indicate the range of values obtained by the ground-truth Pareto-optimal solutions in each dimension of the objective space.

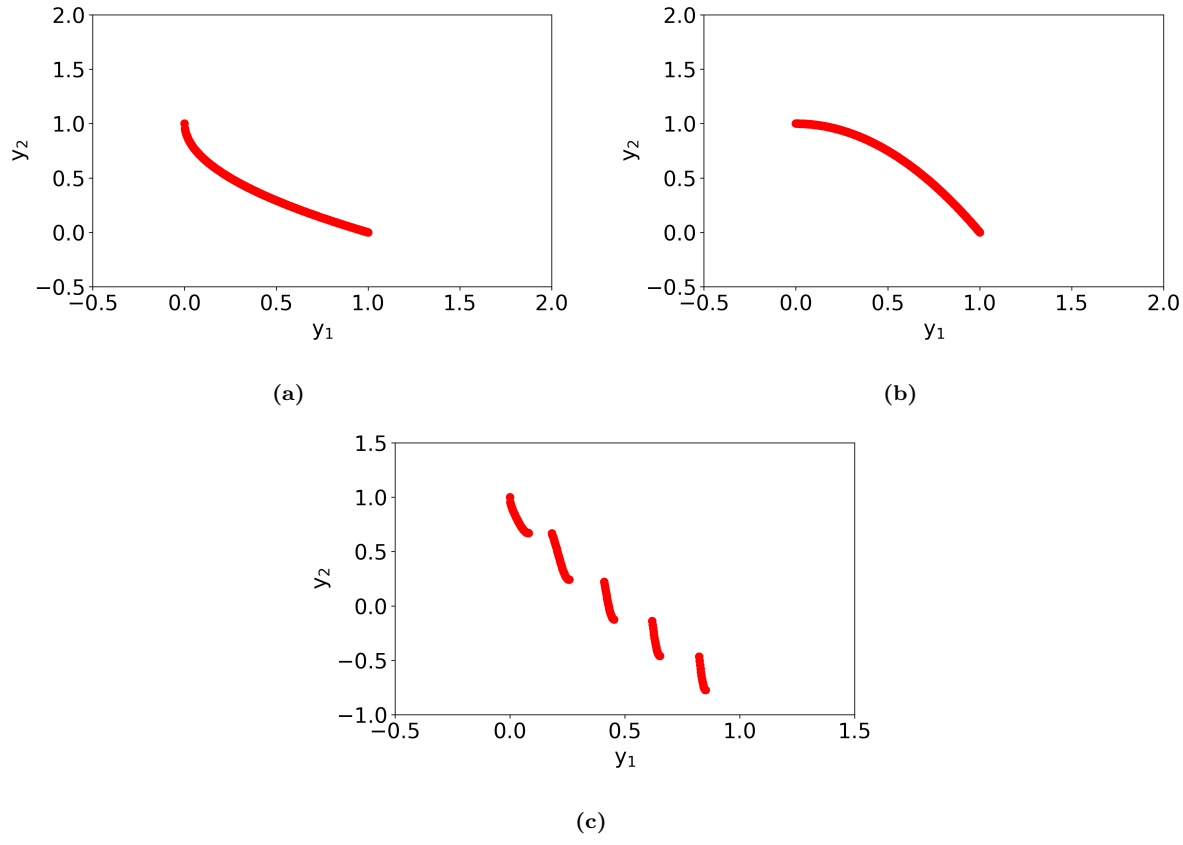
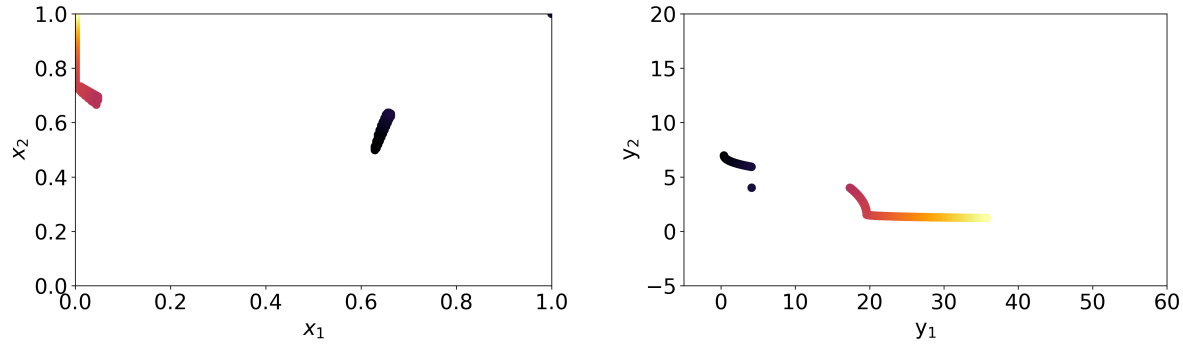


Figure 3.2: The true Pareto fronts of the a) ZDT1, b) ZDT2, c) ZDT3 in their respective objective spaces. In the case of ZDT3, the Pareto front consists of five disconnected sections.



(a) (b)

Figure 3.3: The true a) Pareto-optimal solutions in the search space and b) Pareto front in the objective space of the Branin-Currin function. The Pareto-optimal solutions exist in multiple disconnected regions and they correspond to disconnected sections of the Pareto front.

ZDT functions: The ZDT1, ZDT2 and ZDT3 functions are defined as [8]:

$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = x_1 \\ f_2(\boldsymbol{x}) = g(\boldsymbol{x})h(f_1(\boldsymbol{x}), g(\boldsymbol{x})) \end{cases} \tag{3.5}$$

$$g(\boldsymbol{x}) = 1 + \frac{9}{d-1}\sum_{i=2}^{d} x_i. \tag{3.6}$$

Here, d is the number of parameters which can be freely adjusted. In this work, $d = 2$. That is, these are bi-objective problems. The function g is the same for all functions in the ZDT family. However, the functions differ in how h is defined:

$$h_{\text{ZDT1}}(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}}, \tag{3.7}$$

$$h_{\text{ZDT2}}(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)^2, \tag{3.8}$$

$$h_{\text{ZDT3}}(f_1(\boldsymbol{x}), g(\boldsymbol{x})) = 1 - \sqrt{\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}} - \left(\frac{f_1(\boldsymbol{x})}{g(\boldsymbol{x})}\right)\sin(10\pi f_1(\boldsymbol{x})). \tag{3.9}$$

Using these definitions, the function f_1, which is the same for all ZDT functions, and the different variants of the f_2 function are plotted in Figure A.2 in Appendix A. The ZDT functions are relatively easy to optimize because all the Pareto-optimal solutions satisfy $x_2 = 0$. That is, once a single Pareto-optimal solution is found, the others can be found by changing only one of the parameters. Most real optimization problems are not structured like this, but rather, to find a new Pareto-optimal solution, many or all of the parameters need to be changed. Nevertheless, the ZDT functions serve as basic, easy-to-optimize synthetic test functions.

The different h functions result in varying types of Pareto fronts among the ZDT functions. ZDT1 has a convex Pareto front, while ZDT2 has a concave one. ZDT3 has a discontinuous Pareto front that consists of five distinct sections. The Pareto fronts are plotted in Figure 3.2. The diversity in Pareto front types was the reason these three functions were selected. In particular, including both a convex and a concave Pareto front in the test function suite allows one to investigate the effect of the transformation function when using a scalarization-based approach. Additionally, including a discontinuous Pareto front allows one to investigate possible differences in how the acquisition strategies explore the search space. Some methods may find the different sections of the Pareto front more effectively than others.

Branin-Currin: The Branin-Currin (BC) function is defined as [2]:

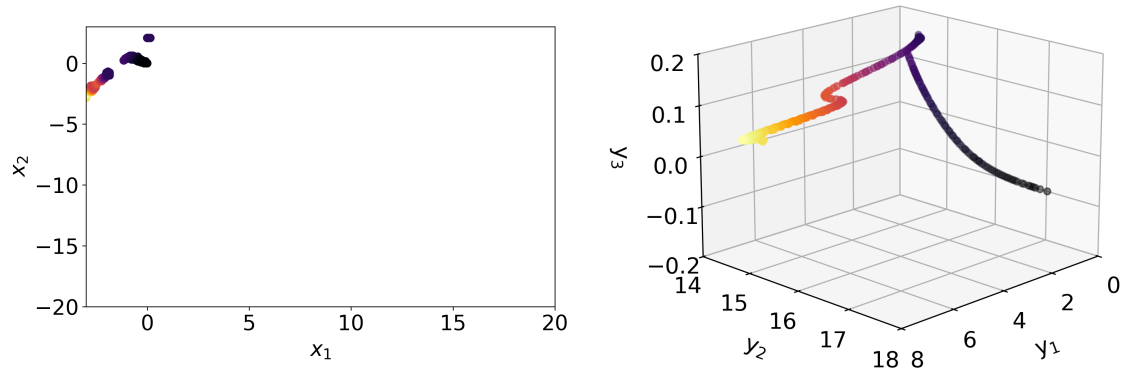
$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}') = \left(x_2' - \frac{5.1}{4\pi^2}(x_1')^2 + \frac{5}{\pi}x_1' - r\right)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1') + 10 \\ f_2(\boldsymbol{x}) = \left[1 - \exp\left(-\frac{1}{2x_2}\right)\right] \frac{2300x_1^3+1900x_1^2+2092x_1+60}{100x_1^3+500x_1^2+4x_1+20} \end{cases} \tag{3.10}$$

The two functions are plotted in Figure A.1 in Appendix A. Above, the input vector $\boldsymbol{x}' = \{x_1', x_2'\} = \{15x_1 - 5, 15x_2\}$ contains rescaled parameters and r is an additional parameter set to $r = 6$. The function has two design parameters and two objectives. The Pareto-optimal solutions and the Pareto front are plotted in Figure 3.3. It can be seen that Branin-Currin has multiple disconnected regions of Pareto-optimal solutions, each of which corresponds to a distinct section of the Pareto front. Due to these properties, Branin-Currin was chosen as a more difficult-to-optimize bi-objective problem.

Viennet: The Viennet function is defined as [23]:

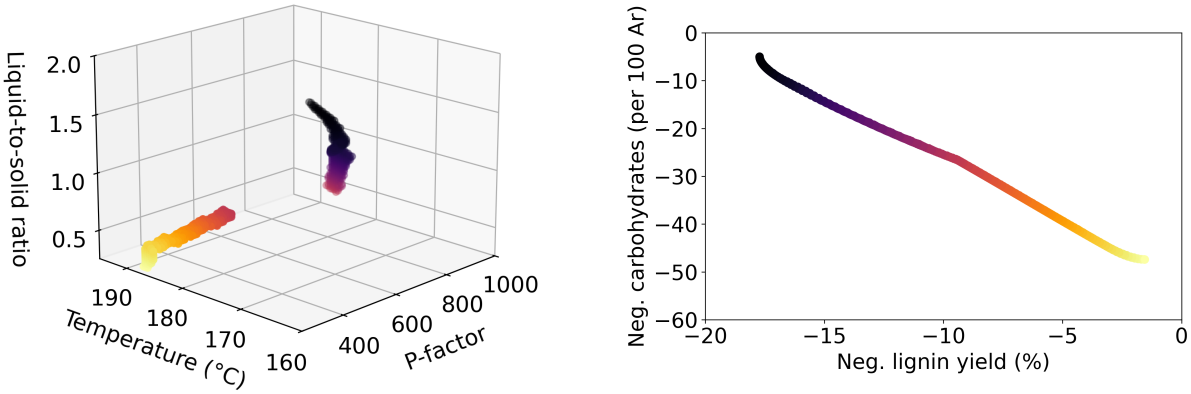
$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = \frac{x_1^2+x_2^2}{2} + \sin(x_1^2 + x_2^2) \\ f_2(\boldsymbol{x}) = \frac{(3x_1-2x_2+4)^2}{8} + \frac{(x_1-x_2+1)^2}{27} + 15 \\ f_3(\boldsymbol{x}) = \frac{1}{x_1^2+x_2^2+1} - 1.1\exp(-x_1^2 - x_2^2) \end{cases} \tag{3.11}$$

The three functions are plotted in Figure A.3 in Appendix A. Viennet has two parameters and three objectives. The Pareto-optimal solutions in the search space and the corresponding Pareto front in the objective space are plotted in Figure 3.4. The Pareto-optimal solutions exist in a few separated regions, but the Pareto front is connected. Because the regions of Pareto-optimal solutions are separated, a multi-objective optimizer might, in principle, only find some of these regions. This might not be observed in experiments, however, since the distances between the regions are small relative to the size of the search space. Primarily, Viennet was chosen as a synthetic problem because it has three objectives. Including a three-objective problem in the test function suite allows one to investigate how the performance of multi-objective BO methods depends on the number of objectives.



(a) (b)

Figure 3.4: The true a) Pareto-optimal solutions in the search space and b) Pareto front in the objective space of the Viennet function.



(a) (b)

Figure 3.5: The true a) Pareto-optimal solutions in the search space and b) Pareto front in the objective space of the Lignin function. The objective space view displays the negative lignin yield and carbohydrate content.

Lignin: The Lignin function is constructed from experimental lignin extraction data [9]. The data provided in the study contains a number of input features and multiple possible objectives to choose from. For this example, reaction severity (p-factor), temperature (temp) and reactor liquid-to-solid ratio (ls-ratio) were selected as the three design parameters. Acetone-extracted lignin yield (ace) and total carbohydrate content (carbs) were selected as the two objectives. Because the lignin yield and carbohydrate content are to be maximized, they are multiplied by minus one to transform the problem into a minimization task. The test function was constructed by fitting a BOSS multi-objective surrogate model on the experimental data points and treating this as the ground-truth objective. The benchmark code provides an interface with which this model can be sampled. The Pareto-optimal solutions and the Pareto front of the Lignin function are visualized in Figure 3.5. The Pareto front appears connected in the objective space. However, looking at the search space, it is revealed that the Pareto-optimal solutions exist in two distinct regions. In a case such as this, where these regions are far apart, a multi-objective optimizer may struggle to explore the entire Pareto front. This is because, in the search space, a jump from one region of Pareto-optimal solutions to the other is required to effectively discover the entire front.

4. Results

This chapter presents the results of this thesis work. Section 4.1 describes the setup of the experiments, including the implementation of noisy objective functions. Sections 4.2 and 4.3 describe in detail the results of the experiments conducted using noiseless and noisy objective functions, respectively. Section 4.4 describes additional experiments related to, for example, the computational cost of the acquisition strategies. Finally, Section 4.5 summarizes the findings in the form of a discussion and provides general guidelines for choosing the appropriate acquisition strategy.

4.1 Experiment setup

A benchmark test using a single test function is henceforth referred to as an *experiment*. Each experiment may involve benchmarking the same test function with multiple acquisition strategies, allowing for a direct comparison of their performance.

An experiment is conducted by optimizing a given test function multiple times under identical settings. Repeating BO runs is crucial for obtaining reliable statistics, as individual runs exhibit inherent variability, even with the same settings. This randomness arises from differences in the initial data points, potential noise in the objective function samples and randomness in acquisition function optimization (*e.g.*, initial point sampling for gradient descent). To account for this, relevant metrics are tracked across multiple BO runs, and their mean and standard deviation are computed as a function of the BO iteration. Each BO run was repeated 10 to 100 times, with a total of 50 to 150 iterations per run, depending on the complexity of the test function (*i.e.*, the number of design parameters and objectives). Generally, for the ZDT and Branin-Currin functions, 100 BO runs of 50 iterations each were performed, whereas for the Viennet and Lignin functions, 50 BO runs of 100 iterations each were performed.

In noisy experiments, homoscedastic Gaussian noise is added to the test function samples. The variance of the added noise is defined separately for each test function and objective, as different objectives may have vastly different value ranges, even for the same test function. To ensure that the impact of the added noise is approximately equal for every objective, the standard deviation of the noise is defined as a percentage of the value

range of the true Pareto front in that particular dimension. For example, suppose that a test function obtains Pareto-optimal values in the range $[a_i^P, b_i^P]$ in dimension i of the objective space. Then, an observed value of the i-th objective is $y_i = f_i + \beta$, where f_i is the ground-truth value and β is drawn from a normal distribution:

$$\beta \sim \mathcal{N}(\mu = 0, \sigma = c|b_i^P - a_i^P|). \tag{4.1}$$

Here, c is the experiment-specific noise level, μ is the mean, and σ is the standard deviation of the distribution. The terms $|b_i^P - a_i^P|$ were determined visually based on plots of the true Pareto fronts and can be found in Table A.1 in Appendix A. An exception is the Lignin function, for which the terms were set to match the true noise values observed in the lignin extraction study [9].

In all conducted experiments, the GP surrogate models use the Radial Basis Function (RBF) kernel. The number of initial points is always set to $2(d + 1)$, where d is the number of design parameters specific to each test function. Unless stated otherwise, these initial points are sampled from a scrambled Sobol sequence and uniformly cover the search space. In experiments involving noise, the noise parameter of each GP surrogate model is set to the true noise variance. If noise is not used, the noise parameters are set to 10^{-12}, which is the default value in BOSS.

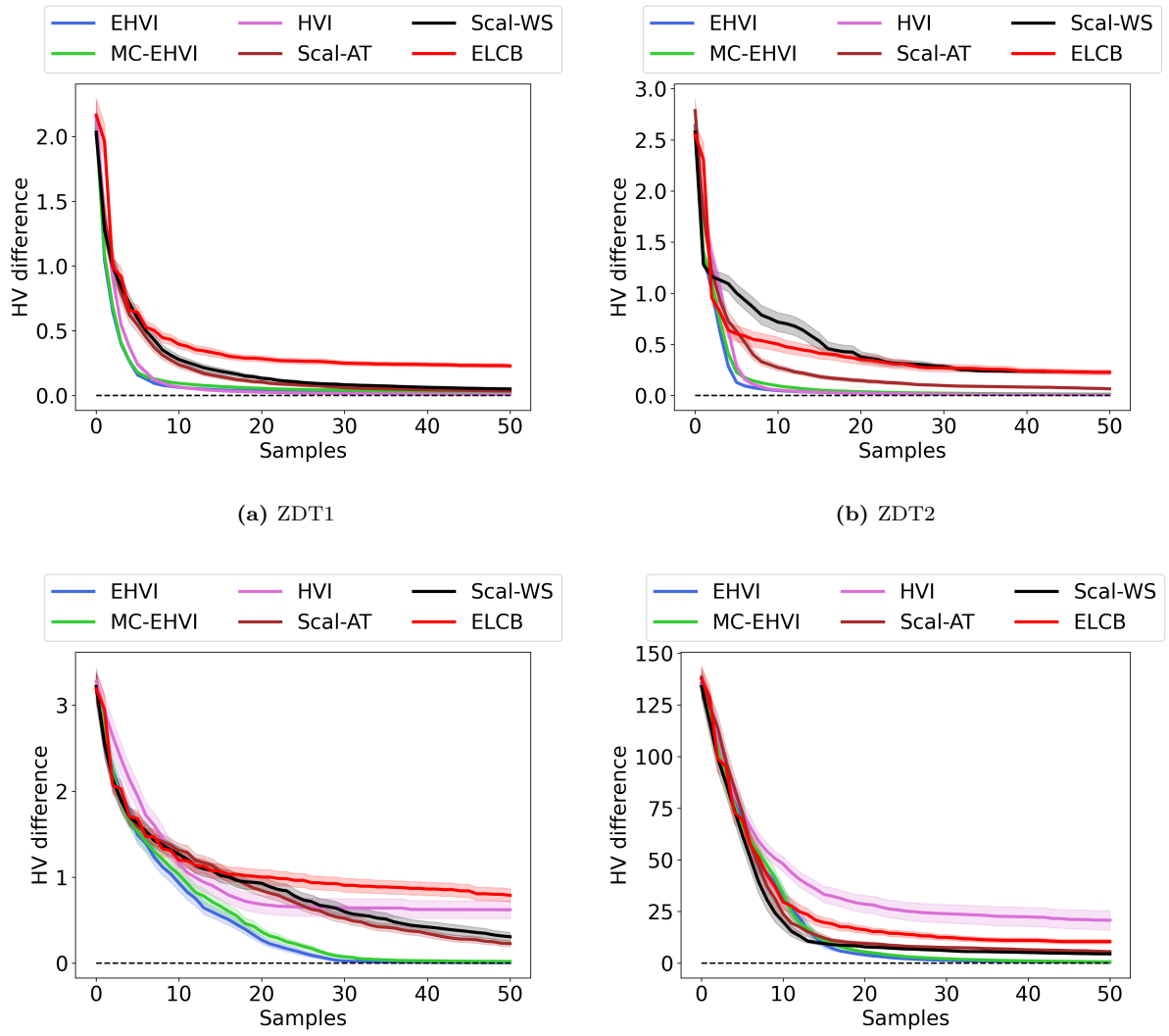
4.2 Noiseless experiments

In noiseless experiments, observations of the objective function are exact. Such experiments were conducted for all six test functions. In the following, the results of these experiments are analyzed one function at a time.

ZDT functions

Figures 4.1a, 4.1b, and 4.1c show the development of the hypervolume difference between the approximate and the true Pareto fronts for the ZDT family, plotted as functions of the BO iteration. Table 4.1 summarizes the performance metrics at the last optimization iteration for each acquisition strategy.

With all the ZDT functions, EHVI and MC-EHVI achieve excellent performance, as indicated by the achieved hypervolume. This is expected, since hypervolume-based methods are specifically designed to maximize the hypervolume of the predicted Pareto front. These methods also typically achieve an even sampling of the true Pareto front, as indicated by the S-metric values in Table 4.1. EHVI consistently outperforms MC-EHVI, although the differences are small. The similarity in performance serves to verify that the implementations of EHVI and MC-EHVI are correct. Furthermore, the results indicate



(a) ZDT1

(b) ZDT2

(c) ZDT3

(d) Branin-Currin

Figure 4.1: The hypervolume difference for a) ZDT1, b) ZDT2, c) ZDT3 and d) Branin-Currin in the noiseless experiments as a function of the BO iteration. The mean and two standard errors over 100 BO runs are reported.

Function	Metric	Algorithm					
		EHVI	MC-EHVI	HVI	Scal-AT	Scal-WS	ELCB
ZDT1	HV	3.6514 ±0.0009	3.6361 ±0.0051	**3.6558** ±0.0017	3.6282 ±0.0244	3.6131 ±0.0217	3.4352 ±0.0776
	S	0.0168 ±0.0105	0.0266 ±0.0141	**0.0119** ±0.0012	0.0731 ±0.0500	0.0655 ±0.0368	0.2394 ±0.0540
	d_x	**0.0005** ±0.0006	0.0020 ±0.0005	0.0012 ±0.0001	0.0027 ±0.0016	0.0025 ±0.0016	0.0088 ±0.0065
	d_y	**0.0014** ±0.0058	0.0068 ±0.0038	0.0019 ±0.0002	0.0160 ±0.0142	0.0137 ±0.0149	0.0649 ±0.0511
ZDT2	HV	3.3235 ±0.0002	3.3180 ±0.0007	**3.3235** ±0.0003	3.2655 ±0.0282	3.1037 ±0.0568	3.1034 ±0.1235
	S	0.0349 ±0.0289	0.0159 ±0.0043	**0.0158** ±0.0146	0.1065 ±0.0416	0.3344 ±0.1424	0.2413 ±0.0665
	d_x	**0.0009** ±0.0006	0.0014 ±0.0001	0.0013 ±0.0003	0.0033 ±0.0021	0.0031 ±0.0035	0.0037 ±0.0035
	d_y	0.0049 ±0.0051	0.0042 ±0.0009	**0.0027** ±0.0026	0.0216 ±0.0188	0.0222 ±0.0282	0.0314 ±0.0309
ZDT3	HV	**4.8041** ±0.0023	4.7897 ±0.0069	4.1886 ±0.5171	4.5816 ±0.1784	4.5026 ±0.2585	4.0211 ±0.3832
	S	0.0688 ±0.0290	0.0693 ±0.0142	**0.0479** ±0.0287	0.1522 ±0.0319	0.1653 ±0.0463	0.2452 ±0.0641
	d_x	0.0011 ±0.0012	0.0028 ±0.0008	**0.0010** ±0.0009	0.0043 ±0.0030	0.0079 ±0.0047	0.0340 ±0.0234
	d_y	0.0076 ±0.0103	**0.0075** ±0.0055	0.0079 ±0.0093	0.0230 ±0.0221	0.0432 ±0.0346	0.1561 ±0.0942
BC	HV	**206.32** ±0.0611	206.08 ±0.0990	185.97 ±23.252	201.12 ±2.1424	202.02 ±1.3431	196.30 ±5.3078
	S	2.3126 ±0.0910	2.4369 ±0.0872	**1.5558** ±0.9380	4.4737 ±0.4960	4.0453 ±0.3455	3.9475 ±0.3983
	d_x	**0.0013** ±0.0004	0.0020 ±0.0003	0.0131 ±0.0168	0.0040 ±0.0034	0.0023 ±0.0014	0.0017 ±0.0020
	d_y	**0.0349** ±0.0038	0.0460 ±0.0070	0.4470 ±0.7420	0.0863 ±0.0956	0.0597 ±0.0245	**0.0349** ±0.0675

Table 4.1: Performance metrics in the noiseless experiments with ZDT1, ZDT2, ZDT3 and Branin-Currin at the last optimization iteration. The mean and one standard deviation over 100 BO runs are reported. The best results are highlighted in bold.

that the number of MC samples (128) used by MC-EHVI is sufficient, at least in these two-dimensional, bi-objective experiments.

The performance of HVI varies across the ZDT functions. With ZDT1 and ZDT2, HVI achieves the best hypervolume of the predicted Pareto front, as seen in Table 4.1. This is because HVI aims to directly maximize the hypervolume of the Pareto front approximation by only considering the mean of the joint posterior distribution. This approach is rather exploitative but performs well in easy, noiseless problems such as ZDT1 and ZDT2. However, with ZDT3, HVI performs poorly compared to EHVI, MC-EHVI and the scalarization-based methods. As seen in Figure 4.1c, the hypervolume difference achieved by HVI converges to a suboptimal value at around the 20th iteration. Looking further at Table 4.1, HVI achieves a significantly lower hypervolume of the predicted Pareto front than the other methods (with the exception of ELCB). This is explained by the disconnected Pareto front of ZDT3. Due to its exploitative approach, HVI fails to sufficiently explore the search space and thus struggles to discover all the disconnected parts of the front. Additionally, there is high variability in the hypervolume achieved by HVI across BO runs. The number of Pareto front sections discovered by HVI varies depending on the configuration of initial points, leading to high variation in the achieved hypervolume.

The values of S, d_x, and d_y in Table 4.1 suggest, at first glance, good performance of HVI with the ZDT3 function. However, this is misleading, as these metrics fail to consider that HVI does not generally discover the entire Pareto front. Instead, HVI discovers only a subset of the true Pareto front and focuses on achieving a good sampling of that rather than exploring and discovering other parts of the front. This limitation can only be detected by inspecting the achieved hypervolume.

With ZDT1 and ZDT3, the two scalarization methods perform similarly in terms of the hypervolume of the predicted Pareto front, as seen in Figures 4.1a and 4.1c, and in terms of the other performance metrics, as seen in Table 4.1. However, with ZDT2, Scal-AT (ParEGO utilizing the augmented Tchebycheff transformation function) outperforms Scal-WS (ParEGO utilizing the WeightSum function) in both the hypervolume and the S-metric. This is likely because the Pareto front of ZDT2 is concave. The more complex form of the transformation function used by Scal-AT might aid the algorithm in exploring concave parts of the front, as noted in Section 2.2.2 and by [19, 26]. In practice, this results in a more uniform sampling of the true Pareto front. This is confirmed by the results in Table 4.1: Scal-AT achieves $S = 0.1065$, while Scal-WS achieves $S = 0.3344$.

With all the ZDT functions, the multi-objective ELCB implementation performs poorly in terms of the hypervolume of the predicted Pareto front. This is because ELCB does not consider the trade-offs between the objectives, but instead attempts to optimize only one of the objectives at each iteration. Consequently, ELCB only samples specific points on the true Pareto front. This results in poor hypervolume and spread of the predicted front, as seen in Table 4.1. This is confirmed by inspecting points sampled by ELCB during exemplary optimization runs. Supplementary Figure B.1 shows an exemplary BO run with the ZDT1 function. ELCB samples mostly the left and right edges of the search space (where $x_1 = 0.0$ or $x_1 = 1.0$). Comparing this with Figure A.2, it is clear that in these areas of the search space, either f_1 or f_2 of the ZDT1 function is minimized.

Branin-Currin function

Figure 4.1d shows the development of the hypervolume difference for Branin-Currin as a function of the BO iteration. Table 4.1 summarizes the performance metrics at the last optimization iteration. With this function, EHVI and MC-EHVI achieve the best performance. By consistently exploring the Pareto front, these methods achieve an excellent hypervolume of the predicted front. Their performances are also nearly identical, indicating that the number of MC samples used by MC-EHVI is sufficient.

The Pareto front of Branin-Currin is disconnected, which leads to poor performance of HVI in terms of the hypervolume difference. As in the previous experiments, HVI fails to explore the entire Pareto front. Additionally, the variation in the hypervolume

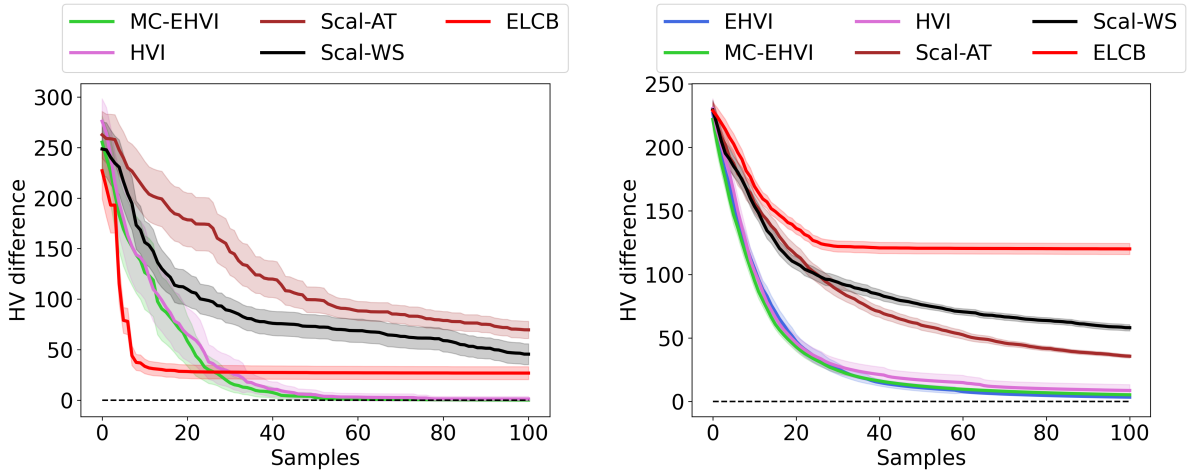
difference across BO runs is high when using HVI, as shown in Table 4.1. The extent to which HVI explores the Pareto front depends on the configuration of the initial points. Thus, HVI explores the front thoroughly in some runs but fails to do so in other runs. This leads to a high standard deviation of the hypervolume difference across trials. Further inspection of Table 4.1 reveals that HVI achieves the best S-metric value. However, this is misleading. In optimization runs where HVI discovers only a single section of the front, the S-metric gives an overly optimistic view of the performance.

The two scalarization methods perform very similarly. As shown in Figure 4.1d, Scal-AT and Scal-WS reduce the hypervolume difference faster than EHVI and MC-EHVI in the early stages of the optimization (before the 10th iteration). It seems that the scalarization-based methods find the first Pareto-optimal solutions faster than hypervolume-based methods. Scalarization-based methods convert the multi-objective problem into a single-objective one by assigning priorities to the objectives. Hence, each optimization iteration focuses on finding a single Pareto-optimal solution rather than improving the overall Pareto front prediction. However, after finding the first Pareto-optimal solutions, the scalarization-based approaches tend to resample around the same promising points. As such, the true Pareto front is not explored sufficiently. This results in a poorer hypervolume difference by the end of the optimization compared to the hypervolume-based methods.

Multi-objective ELCB performs poorly, though better than HVI. ELCB considers only one of the objectives at a time when determining the next acquisition without accounting for trade-offs between the objectives. As a result, ELCB fails to explore the Pareto front despite locating some solutions on it, and the hypervolume difference converges relatively quickly to a suboptimal value. However, inspecting Table 4.1, the low values of d_x and d_y indicate that the Pareto-optimal solutions discovered by ELCB are relatively close to the true Pareto front. On the other hand, the poor value of the S-metric indicates that these solutions are not uniformly spread across the objective space

Viennet function

Figure 4.2a shows the development of the hypervolume difference for the Viennet function as a function of the BO iteration. Table 4.2 presents the values of the performance metrics at the last optimization iteration. As before, the hypervolume-based methods achieve the best hypervolume difference. Additionally, MC-EHVI and HVI achieve similar performance. This is somewhat unexpected because the Pareto-optimal solutions of Viennet exist in multiple disconnected regions, as shown in Figure 3.4. However, these regions are relatively close together and HVI consistently discovers all of them despite its exploitative approach. EHVI was not tested, as Viennet is a three-objective function and



(a) Viennet **(b)** Lignin

Figure 4.2: The hypervolume difference for a) Viennet and b) Lignin in the noiseless experiments as a function of the BO iteration. The analytic EHVI cannot be used with Viennet because it has three objectives. The mean and two standard errors over 50 BO runs are reported.

Function	Metric	Algorithm					
		EHVI	MC-EHVI	HVI	Scal-AT	Scal-WS	ELCB
Viennet	HV	NA	**327.54** ±0.2146	326.50 ±8.9552	258.05 ±30.121	282.13 ±36.423	300.86 ±22.564
	S	NA	0.7273 ±0.1777	**0.4520** ±0.4254	2.6944 ±1.0043	2.6314 ±1.1247	1.2088 ±1.0858
	d_x	NA	0.0471 ±0.0209	**0.0410** ±0.1419	1.4010 ±0.5289	0.9807 ±0.6516	0.1464 ±0.1427
	d_y	NA	**0.0173** ±0.0053	0.0845 ±0.5118	4.4222 ±2.1853	3.1588 ±2.6156	0.3653 ±0.5156
Lignin	HV	**511.72** ±0.2091	509.80 ±0.2076	506.42 ±16.416	479.46 ±5.0719	457.03 ±8.7733	395.03 ±15.449
	S	**0.1471** ±0.0141	0.1854 ±0.0151	0.1552 ±0.0175	1.0083 ±0.1401	1.9461 ±0.2548	2.9926 ±0.4509
	d_x	**1.0037** ±0.1112	1.9874 ±0.1886	1.7417 ±0.1172	26.709 ±13.464	12.284 ±5.5962	13.262 ±5.3883
	d_y	**0.0316** ±0.0029	0.0488 ±0.0031	0.0519 ±0.0425	0.4651 ±0.0985	0.3364 ±0.1295	0.5367 ±0.1622

Table 4.2: Performance metrics in the noiseless experiments with Viennet and Lignin at the last optimization iteration. The analytic EHVI cannot be used with Viennet because it has three objectives. The mean and one standard deviation over 50 BO runs are reported. The best results are highlighted in bold.

the analytic form of EHVI was only implemented for bi-objective cases.

The two scalarization-based methods perform poorly in terms of the hypervolume difference. In this experiment, both Scal-AT and Scal-WS exhibit highly exploratory behaviour. Inspection of sample BO runs visualized in supplementary Figure B.2 confirms that these methods fail to sufficiently sample the Pareto front and instead focus on exploring the search space. These results are consistent with observations made in other works. It has been found that the ParEGO algorithm tends to prioritize reducing the error of the surrogate model rather than finding the Pareto front [7]. As a result, the ParEGO might require many BO iterations to discover the Pareto front effectively. This is also supported by the observation that, in the case of Viennet, the hypervolume difference achieved by either scalarization-based method has not converged within 100 BO iterations. Furthermore, Scal-WS appears to perform significantly better than Scal-AT. We hypothesize that the more complex form of Scal-AT may further exacerbate the inefficient sampling since the function landscape of the auxiliary GP model, over which the Expected Improvement acquisition function is optimized, becomes more complex. In such a case, more exploration of the search space is required to effectively reduce the error of the surrogate model.

ELCB initially achieves the fastest reduction in the hypervolume difference. However, after roughly ten iterations, the hypervolume difference converges to a suboptimal value. This suggests that ELCB finds the first Pareto-optimal solutions quickly but subsequently fails to explore the entire Pareto front. Supplementary Figure B.3 shows an exemplary BO run at the 20th and 100th iterations using ELCB. By the 20th iteration, ELCB has already found a section of the Pareto front, but then fails to discover new sections. Instead, ELCB repeatedly samples the already discovered section of the front or explores areas of the search space far from the true Pareto-optimal solutions. Neither of these approaches improves the Pareto front approximation or its hypervolume.

Lignin function

Figure 4.2b shows the development of the hypervolume difference for the Lignin function as a function of the BO iteration. Table 4.2 presents the values of the performance metrics at the last optimization iteration. Once again, the hypervolume-based methods achieve excellent performance in all metrics. EHVI and MC-EHVI yield very similar performance, although EHVI again has a slight advantage. The performance of HVI is also strong, with its achieved hypervolume only slightly smaller than that of EHVI and MC-EHVI. Although Lignin has two disconnected areas of Pareto-optimal solutions in the search space, HVI generally appears to discover both regions. This is likely because the initial points cover the search space uniformly and provide sufficient prior information

about both regions of Pareto-optimal solutions. However, the variation across BO runs with HVI is higher than with the other hypervolume-based methods. This indicates that the performance of HVI depends more heavily on the configuration of the initial data points.

The scalarization-based methods exhibit relatively poor performance. These methods fail to sample the Pareto front uniformly, as reflected by the S-metric values reported in Table 4.2. Additionally, the achieved values of d_x suggest that the predicted Pareto-optimal solutions are relatively far from the true Pareto-optimal solutions. There is also disparity between the two scalarization-based methods, as Scal-AT achieves a somewhat better hypervolume difference than Scal-WS. This is likely caused by the slightly concave Pareto front of the Lignin function. Due to this, Scal-WS samples mainly the ends of the front, while Scal-AT also samples the middle parts. This behaviour is also evident from the S-metric values achieved by the two methods and by Supplementary Figure B.4, which displays two exemplary BO runs comparing Scal-AT and Scal-WS.

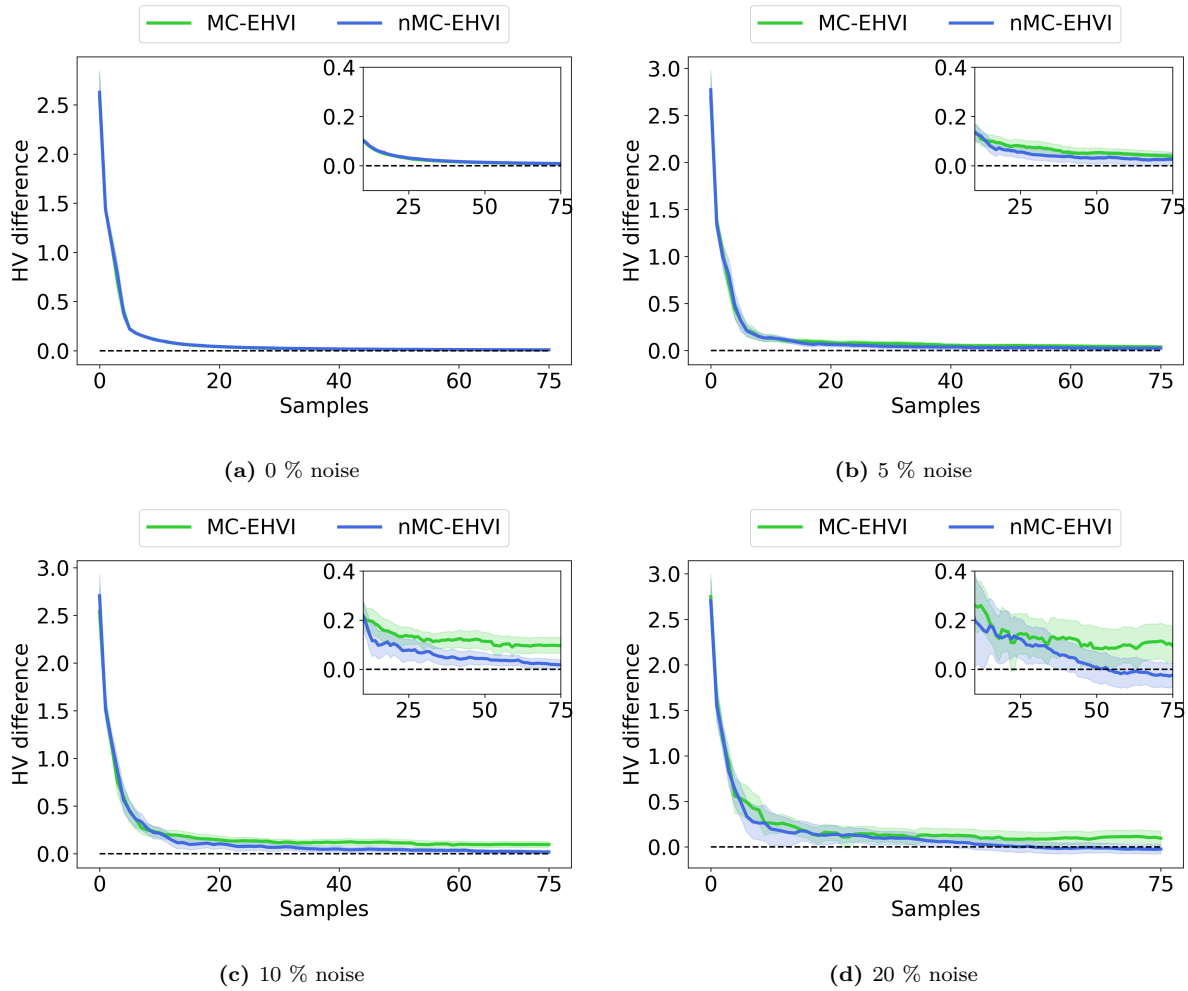
The performance of ELCB is poor, with behavior similar to that observed in earlier examples: the hypervolume of the predicted Pareto front quickly converges to a suboptimal value. As shown in Table 4.2, the achieved values S, d_x, and d_y are all relatively poor. Furthermore, the values of d_x and d_y suggest that the predicted Pareto-optimal solutions and the Pareto front are relatively far from their ground-truth counterparts. In short, ELCB struggles to effectively discover the true Pareto front.

4.3 Noisy experiments

One of the implemented multi-objective acquisition strategies, nMC-EHVI, is specifically designed for scenarios where the objective function samples are corrupted by noise. The performance of nMC-EHVI was tested and compared to that of MC-EHVI in two experiments using the ZDT2 and Lignin functions. In the following, the results of these experiments are analyzed.

ZDT2

The experiment with the noisy ZDT2 test function was repeated with multiple noise levels ranging from 0.00 to 0.20 (0 % to 20 % noise. The noise level defines the noise relative to the range of values in the true Pareto front, as described in Section 4.1. This experiment was repeated with multiple noise levels to identify how the relative performances of MC-EHVI and nMC-EHVI depend on the strength of the noise. In each experiment, 30 BO runs of 75 iterations each were conducted. The number of MC samples used by the acquisition functions was set to $N = 128$.



(a) 0 % noise

(b) 5 % noise

(c) 10 % noise

(d) 20 % noise

Figure 4.3: The difference in hypervolume for the noisy ZDT2 function with a) 0 %, b) 5 %, c) 10 % and d) 20 % noise. The mean and two standard errors over 30 BO runs are reported. The insets highlight the behaviour at the end of the optimization. With 0 % noise, the two curves overlap nearly perfectly.

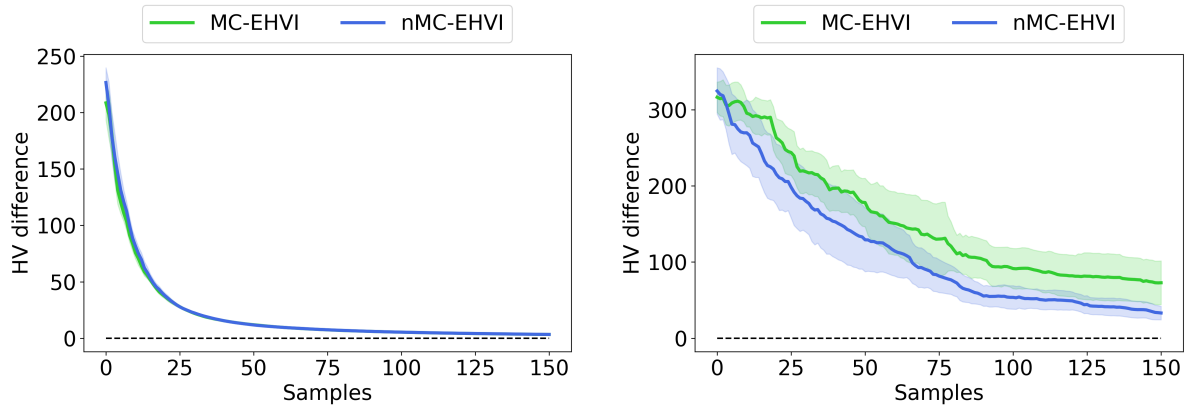
Algorithm	Metric	Noise level				
		0.00	0.05	0.10	0.15	0.20
nMC-EHVI	HV	3.3217 ±0.0006	**3.3042** ±0.0662	**3.3120** ±0.0549	**3.3520** ±0.0992	**3.3537** ±0.1365
	S	0.0173 ±0.0109	**0.0303** ±0.0100	**0.0441** ±0.0143	**0.0474** ±0.0098	**0.0693** ±0.0376
	d_x	0.0015 ±0.0003	0.0063 ±0.0045	0.0060 ±0.0043	0.0079 ±0.0060	**0.0054** ±0.0049
	d_y	0.0054 ±0.0025	0.0591 ±0.0294	0.0677 ±0.0383	0.0879 ±0.0541	**0.0836** ±0.0388
MC-EHVI	HV	**3.3229** ±0.0005	3.2928 ±0.0456	3.2337 ±0.0862	3.2051 ±0.1681	3.2351 ±0.2100
	S	**0.0157** ±0.0157	0.0572 ±0.0271	0.0722 ±0.0335	0.0827 ±0.0252	0.0973 ±0.0435
	d_x	**0.0014** ±0.0003	**0.0043** ±0.0030	**0.0046** ±0.0034	**0.0057** ±0.0037	0.0070 ±0.0058
	d_y	**0.0040** ±0.0024	**0.0389** ±0.0279	**0.0501** ±0.0311	**0.0749** ±0.0282	0.1011 ±0.0491

Table 4.3: Values of the performance metrics in the noisy experiments with ZDT2 at the last optimization iteration. The mean and one standard deviation over 30 BO runs are reported. The best results are highlighted in bold.

The evolution of the hypervolume differences in experiments with 0 %, 5 %, 10 %, and 20 % noise are shown as functions of the BO iteration in Figure 4.3. The values of all performance metrics at the final optimization iteration are detailed in Table 4.3. With no noise, nMC-EHVI and MC-EHVI perform identically. This was expected because the definition of nMC-EHVI (Equation 2.26) reduces to that of MC-EHVI (Equation 2.25) in a noiseless case. When noise is introduced, nMC-EHVI consistently outperforms MC-EHVI in terms of the hypervolume. At high noise levels, nMC-EHVI tends to overestimate the hypervolume: the hypervolume of the true Pareto front is 3.3308, while nMC-EHVI predicts a value of 3.3520 at 15 % noise and 3.3537 at 20 % noise. This suggests that the predicted Pareto front is overly optimistic. However, these values are still closer to the ground-truth than the values predicted by MC-EHVI at the corresponding noise levels. In terms of the S-metric, nMC-EHVI typically performs better, meaning that it achieves a more homogeneous sampling of the Pareto front. In terms of d_x and d_y, the differences are less clear. MC-EHVI achieves better d_x and d_y values at intermediate noise levels (5 % to 15 %), whereas nMC-EHVI achieves better values at the highest noise level (20 %). However, the standard deviations associated with the values are relatively high. A higher number of repeated BO runs would be desirable to obtain more precise results. This was not addressed here due to the high computational cost of additional BO runs.

Lignin

A noisy experiment was also conducted with the Lignin function. The noise was set to match the true noise observed in the lignin extraction study [9]. For comparison, the experiment was repeated with no noise. In these experiments, 10 BO runs of 150 iterations each were conducted. The number of MC samples was set to $N = 128$.



(a) 0 % noise

(b) 100 % noise

Figure 4.4: The difference in hypervolume for the noisy Lignin function with a) 0 % and b) 100 % noise. The mean and two standard errors over 10 BO runs are reported.

Algorithm	Metric	Noise level	
		0.00	1.00
nMC-EHVI	HV	**511.83** ±0.1081	**482.01** ±14.040
	S	0.1344 ±0.0095	**0.4596** ±0.0714
	d_x	**1.8555** ±0.0894	**17.797** ±16.063
	d_y	**0.0041** ±0.0018	**0.6005** ±0.2118
MC-EHVI	HV	**511.83** ±0.1621	442.34 ±45.348
	S	**0.1185** ±0.0083	1.0463 ±0.3025
	d_x	1.9020 ±0.0841	31.320 ±24.425
	d_y	0.0396 ±0.0023	0.7669 ±0.2873

Table 4.4: Values of the performance metrics in the noisy experiments with Lignin at the last optimization iteration. The mean and one standard deviation over 10 BO runs are reported. The best results are highlighted in bold.

Figure 4.4 shows the evolution of the hypervolume difference as a function of the BO iteration for the Lignin function with and without noise. Table 4.4 details the values of all performance metrics at the final optimization iteration. In the noiseless case, nMC-EHVI and MC-EHVI exhibit identical performance. This aligns with earlier results and indicates that the methods are equivalent in the noiseless case. In the presence of noise, nMC-EHVI outperforms MC-EHVI across all performance metrics. These results clearly demonstrate that nMC-EHVI outperforms MC-EHVI. Additionally, nMC-EHVI performs more consistently than MC-EHVI, as indicated by the lower standard deviation values associated with the achieved values of the performance metrics. This suggests that the performance of nMC-HEVI is less influenced by randomness, such as the selection of initial points and the effects of noise.

As shown in Figure 4.4, neither algorithm has fully converged within the budget of 150 iterations. Both methods would likely benefit from a higher number of BO iterations. Furthermore, a higher number of MC samples could potentially accelerate the convergence of nMC-EHVI, as the function evaluation would be more precise. However, these modifications were not explored due to the substantial computational cost of nMC-EHVI.

4.4 Additional experiments

In addition to the previously described experiments, further experiments were conducted to gain deeper insights into the acquisition methods under investigation. Given the expected variability in computational cost across the methods, the costs of selected methods are compared in Section 4.4.1. The exploration and exploitation tendencies of HVI are analyzed in Section 4.4.2. Furthermore, the performance of MC-EHVI as a function of the MC sample count is examined in Section 4.4.3.

4.4.1 Computational cost

The computational cost (acquisition time as a function of the BO iteration) is an important factor to consider when selecting the acquisition strategy for a new application, and it is expected to vary significantly across different methods. To assess the computational cost of an acquisition method, the acquisition computation time can be measured as a function of the BO iteration. The measured time includes the minimization of the acquisition function, that is, the time required to determine the next acquisition at each BO iteration.

The acquisition times were measured in an experiment using the noiseless Branin-Currin function. In this experiment, 100 BO runs each with 50 iterations were performed. The computational cost was measured for EHVI, MC-EHVI, HVI, Scal-AT, and ELCB.

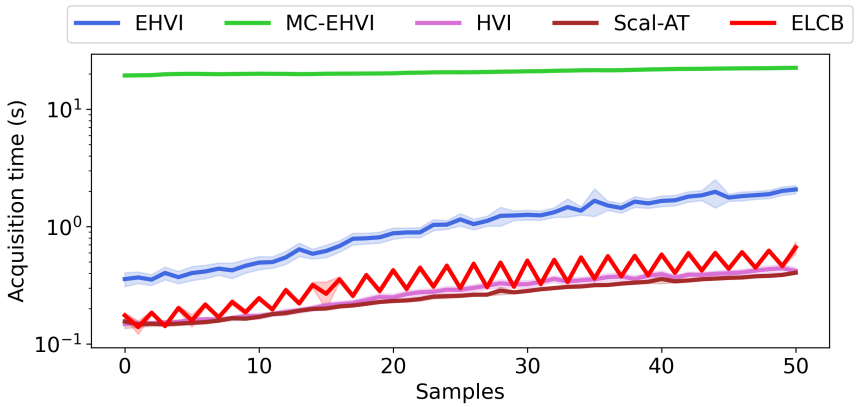


Figure 4.5: Acquisition computation time as a function of the BO iteration for different acquisition strategies. The experiment was conducted using the noiseless Branin-Currin function. The mean acquisition time in seconds and two standard errors over 50 BO runs are reported.

Only one scalarization method was tested, as their computational costs are expected to be very similar. The experiment used six initial points and $N = 128$ MC samples for MC-EHVI. CPU time was measured on 2x14 core Xeon E5-2680 v4 at 2.40 GHz. Computations were performed using 64-bit floating point precision.

Figure 4.5 shows the development of acquisition times across BO iterations. Table 4.5 shows the acquisition times for the first and last BO iterations. Ideally, one would wish to infer possible power laws followed by the acquisition times in the limit of many samples. In the regime of small sample counts relevant to this work, one cannot identify the asymptotic behaviours. Still, one can make observations that are relevant to the applications considered here.

The tested methods can be roughly divided into two categories based on their computational cost. EHVI and MC-EHVI represent the computationally expensive acquisition functions. MC-EHVI is far more expensive, requiring about 19 seconds to determine the next acquisition even at the first iteration, while EHVI requires about 0.4 seconds. Additionally, the computational cost of MC-EHVI grows quicker than the cost of EHVI as a function of the BO iteration. This is due to the expensive hypervolume computations associated with MC-EHVI, the cost of which increase as the size of the Pareto front approximation increases. These observations highlight the advantage of using the analytic form of EHVI, which can be minimized by utilizing gradient information. In practical applications, EHVI should be preferred over MC-EHVI whenever feasible due to the lighter computational cost and more precise computations. Nonetheless, the analytic EHVI remains more expensive than the most efficient methods.

Scal-AT, ELCB and HVI represent the category of computationally cheap solutions. At the first iteration, the cost of these methods is similar and roughly 1 % of the cost

Iteration	Algorithm				
	EHVI	MC-EHVI	HVI	Scal-AT	ELCB
1st	0.3576 ±0.0339	19.442 ±0.3155	0.1481 ±0.0035	0.1556 ±0.0139	0.1754 ±0.0068
50th	2.0760 ±0.1171	22.608 ±0.1119	0.4203 ±0.0200	0.4053 ±0.0075	0.6676 ±0.0545

Table 4.5: Acquisition times in seconds in the noiseless Branin-Currin experiment at the first and 50th BO iterations. Mean and two standard errors over 100 BO runs are reported.

of MC-EHVI and half of the cost of EHVI. The computational cost of Scal-AT, ELCB and HVI also appears to grow more slowly than the cost of MC-EHVI or EHVI as a function of the BO iteration. Furthermore, as seen in Figure 4.5, the ELCB acquisition function exhibits a distinct alternating pattern, where the acquisition time slightly varies depending on the objective being considered.

4.4.2 Exploitation tendency of HVI

In previous experiments, it was observed that the HVI acquisition function sometimes prioritizes exploitation over exploration. This means that HVI tends to sample locations in the search space where the objective function values are known to be promising, potentially neglecting other regions. To investigate this behaviour further, additional experiments were conducted using the Branin-Currin function with a modified experimental setup and the HVI acquisition function with a non-zero exploration weight.

The Pareto-optimal solutions of the Branin-Currin function (shown in Figure 3.3) are located in multiple disconnected regions of the search space. In the objective space, these regions correspond to distinct segments of the Pareto front. In the modified setup, initial points are constrained to satisfy $x_1 \leq 0.2$, meaning they only cover a limited portion of the search space. Consequently, the points encompass only one of the Pareto-optimal regions, initially leaving the model with no knowledge of the second region. Otherwise, this experiment follows the procedure described in Section 4.2. Using this setup, 50 BO runs with 50 iterations each were performed with each tested acquisition function. The goal is to determine whether the algorithm can eventually discover the second Pareto-optimal region.

Figure 4.6 shows the development of the hypervolume difference achieved by HVI across BO iterations for two exploration weights, $\alpha = 0$ and $\alpha = 1$. The exploration weight, defined in Equation 2.23, controls the balance between exploration and exploitation. When $\alpha = 0$, the acquisition function reduces to the standard HVI used in earlier experiments (Equation 2.22). For comparison, the test was also conducted using EHVI. With $\alpha = 0$, HVI converges to a suboptimal hypervolume difference, as it often fails to fully discover the Pareto front. In contrast, with $\alpha = 1$, the hypervolume difference

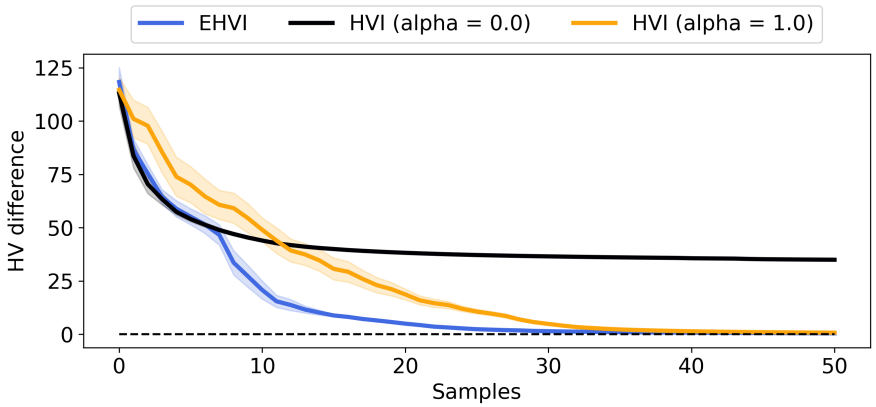


Figure 4.6: The difference in hypervolume between the approximate PF and the true PF for the Branin-Currin function using HVI with $\alpha = 0$ and $\alpha = 1$ and EHVI. In this experiment, the initial points do not cover the search space uniformly but are constrained to satisfy $x_1 \leq 0.2$. The mean and two standard errors over 50 BO runs are reported.

converges to zero, indicating that the entire Pareto front is discovered. This is confirmed by inspecting the predicted Pareto fronts from two exemplary BO runs, which are shown in supplementary Figures B.5 and B.6.

The convergence of the hypervolume difference is slower with HVI using $\alpha = 1$ than with EHVI. This suggests that this value of α is not optimal and that another value could lead to faster convergence. In essence, with $\alpha = 1$, HVI overemphasizes exploration and spends more iterations exploring than necessary.

This experiment shows that the standard HVI acquisition function is highly exploitative and may miss parts of a disconnected Pareto front if the initial points poorly cover the search space. Setting a non-zero exploration weight mitigates this issue. As an additional observation, EHVI performs well despite the restricted initial points, showing that it can balance exploitation and exploration without an explicit exploration term.

4.4.3 Number of MC samples

The performance of MC-EHVI is expected to approach that of the analytic EHVI as the number of MC samples, N, is increased. As an additional benchmark, the Branin-Currin experiment was repeated using MC-EHVI with varying values of N. This investigation served two purposes. First, if the performance of MC-EHVI approaches that of EHVI with sufficiently large N, it indicates that the methods are correctly implemented. Second, a suitable default value of N can be determined by inspecting how the performance of MC-EHVI, relative to that of EHVI, changes as a function of N.

Figure 4.7 compares the hypervolume difference between the true and predicted Pareto fronts using 2, 8, 32, 128, and 512 MC samples. The hypervolume difference is

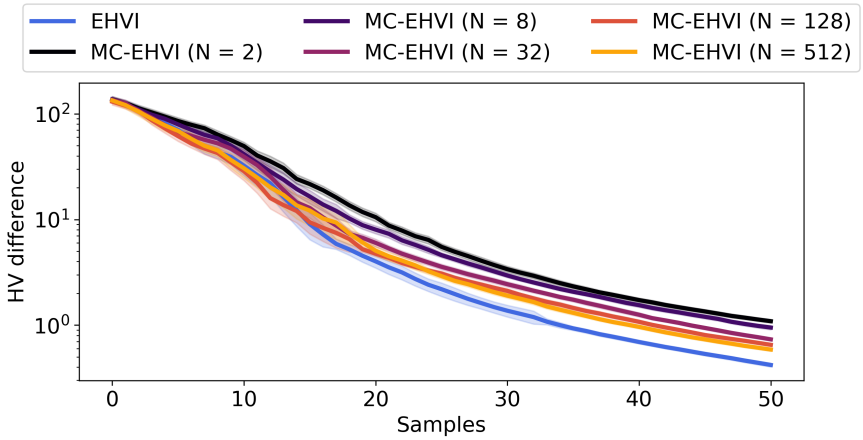


Figure 4.7: The difference in hypervolume between the approximate Pareto front and the true Pareto front for the Branin-Currin function using EHVI and MC-EHVI with 2, 8, 32, 128 and 512 MC samples. The mean and two standard errors over 50 BO runs are reported, except in the case of EHVI, for which 100 BO runs were performed.

plotted on a logarithmic scale to highlight the differences toward the end of the optimization runs. The results show that as N increases, the quality of the Pareto front prediction from MC-EHVI improves and gradually approaches that of EHVI. This confirms that the two methods are equivalent, up to the MC integration error, as discussed in Section 2.2.3. Since the error scales as $1/\sqrt{N}$ (for *i.i.d.* samples), larger values of N are needed to achieve diminishing reductions in error. In practical applications, MC-EHVI is not recommended when EHVI is available. The latter is exact and more efficient, as it avoids repeated hypervolume computations and utilizes analytic gradient information during function minimization.

Based on this experiment, the default number of MC samples used by the MC-based approximations of EHVI (MC-EHVI and nMC-EHVI) was set to 128. This value is likely to provide good performance in most applications while keeping the computational cost reasonable. To accommodate different use cases, the number of samples can be freely set by the user in the BOSS implementation.

4.5 Discussion

This section discusses the key findings of the thesis. The topic is approached by describing the advantages and limitations of the acquisition methods and considering their potential use cases. In addition, general guidelines for choosing the appropriate acquisition method for a new multi-objective optimization task are provided.

EHVI and MC-EHVI exhibited consistent performance across test functions and

generally outperformed the other acquisition methods. This was an expected result, as hypervolume-based methods are designed to perform the multi-objective optimization directly. In contrast, the other investigated methods resort to solving a simplified proxy problem. Furthermore, EHVI and MC-EHVI demonstrated the ability to balance exploration and exploitation, and these methods typically discovered the entire Pareto front even in cases where the true front was disconnected or the initial data did not cover the search space uniformly. Considering these results, EHVI and MC-EHVI are the most reliable acquisition methods that are likely to work well in most problems. However, EHVI should be preferred over MC-EHVI whenever possible, since it is exact and computationally more efficient.

EHVI and MC-EHVI are theoretically equivalent, given a sufficiently large number of Monte Carlo (MC) samples. However, achieving arbitrarily good estimation accuracy may not be possible in practice, as time and computational resources are always finite. In this work, we used 128 MC samples, which yielded strong performance in noiseless experiments with up to three objectives (which was the maximum number of objectives considered). For problems with more than three objectives, a higher sample count may be beneficial. Although the convergence rate of MC estimation does not depend on the number of objectives, its absolute error does. Thus, maintaining a desired level of accuracy in higher-dimensional settings requires more samples. A higher number of MC samples should also be considered in applications where computational cost is not a concern, such as settings where BO is integrated with experimental work. Since the error of MC integration scales as $1/\sqrt{N}$, where N is the number of samples, increasing the number of samples always improves estimation accuracy, in principle. However, arbitrarily high sample counts are not advisable as one eventually encounters diminishing returns in estimation accuracy, while the computational cost scales linearly.

The nMC-EHVI acquisition function consistently performed better than MC-EHVI in noisy problems when considering the hypervolume of the predicted Pareto front. This was expected and aligns well with the results of a previous study [6] (which did not consider other performance metrics). However, in the intermediate noise region (5–15 % noise) with the ZDT2 function, MC-EHVI achieved slightly better values of d_y and d_x. This indicates that the Pareto-optimal solutions predicted by MC-EHVI were closer to the ground-truth than those predicted by nMC-EHVI. However, achieved values of d_y and d_x were small compared to the size of the search space and the true Pareto front with both methods (compare Table 4.3 and Table 3.2). Additionally, considering the relatively high standard deviations associated with the values, the differences between the methods may not be significant. Nonetheless, this behaviour in the intermediate noise region is surprising and might be caused by some specific features of the ZDT2 function. Considering that the hypervolume difference achieved by nMC-EHVI was sometimes negative with the ZDT2

function, it is likely that this method tends to produce overly optimistic Pareto front predictions. The number of repeated BO runs was limited in noisy experiments, and the standard deviations of the obtained metric values were sometimes high. Hence, we suggest further experimentation with other test functions to gain a deeper understanding of the behaviour of these methods.

HVI performed as well as EHVI and MC-EHVI in some experiments. This was somewhat surprising, as the method is purely exploitative and does not consider uncertainty as EHVI and MC-EHVI do. Particularly, HVI appears to perform well in experiments where the true Pareto front is continuous in the search space and the initial points provide sufficient coverage of that space. However, in cases where the true Pareto front is not continuous or the initial points are not uniformly distributed, HVI may struggle to explore the search space and sometimes entirely miss sections of the Pareto front. This problem can be effectively mitigated by setting the exploration term to a non-zero value. This approach is attractive, as by using exploratory HVI, it is possible to achieve excellent performance typical of hypervolume-based methods while avoiding the relatively high computational cost of EHVI and MC-EHVI. However, the task of determining an optimal value for the exploration weight was not addressed here and remains an open question.

The performance of the scalarization-based ParEGO approach varied and sometimes depended on the transformation function. ParEGO performs better with the augmented Tchebycheff transformation function (Scal-AT) than with the WeightSum function (Scal-WS) in cases where the true Pareto front is concave. However, Scal-AT performed worse than Scal-WS with the Viennet function. It has been observed in the literature that ParEGO tends to choose new acquisitions to reduce the error of the surrogate model rather than to find the Pareto front. Extensions to ParEGO, which aim to mitigate this issue, have been proposed [7]. Since Scal-AT and Scal-WS appear to perform differently depending on the scenario, it is difficult to choose one over the other in a general application. Application-specific knowledge would be useful in determining which one to use. If the true Pareto front is expected to be concave, Scal-AT is a safer option as it is better able to explore the concave parts of the front.

The performance of ELCB was generally poor. This was expected because ELCB is not designed to handle multiple objectives. It does not consider the trade-offs between objectives and typically samples around the global optima of the individual objectives instead of exploring the Pareto front. Considering the low computational cost and easy implementation, ELCB could be used to quickly search for the first Pareto-optimal solution. However, the use of ELCB in multi-objective optimization tasks is generally not advised, as its performance is unreliable and the found solutions are not guaranteed to be Pareto-optimal.

The choice of acquisition strategy in multi-objective BO ultimately depends on the

goal of the decision-maker and the available computational resources. This decision often involves a trade-off between the accuracy of the Pareto front prediction and the associated computational cost. Scalarization-based methods, including ELCB, offer a low-cost option and may suffice if the goal is to identify any Pareto-optimal solution with limited computational resources. The cheaper methods would also be appropriate in applications where the computational overhead of BO would otherwise be an issue. However, these methods often fall short when a more complete approximation of the Pareto front is needed or when dealing with more difficult optimization tasks. In such cases, hypervolume-based methods, especially EHVI or MC-EHVI (depending on the number of objectives), should be preferred. These methods consistently produced the most accurate Pareto front predictions and achieved excellent performance in all tested scenarios. HVI serves as a middle ground: it can perform comparably to EHVI when the initial data covers the search space uniformly. However, its high exploitation tendency can cause problems in scenarios where the Pareto-optimal solutions exist in multiple disconnected regions of the search space. This issue can be addressed by introducing a non-zero exploration weight α, though determining the optimal value of α remains an open question.

5. Conclusion

This thesis investigated multi-objective Bayesian optimization (BO) methods, focusing on their implementation within the Bayesian Optimization Structure Search (BOSS) package. As a key contribution, the BOSS code was extended to support multi-objective optimization. Multiple multi-objective acquisition strategies were implemented, enabling their use for diverse optimization tasks. Furthermore, the acquisition strategies were benchmarked using six test functions, including five synthetic functions and one constructed from experimental lignin extraction data. The results prove the efficacy of the implementation and provide guidelines for the use of the methods.

The implemented multi-objective acquisition strategies included the Hypervolume Improvement (HVI) and three variants of the Expected Hypervolume Improvement (EHVI). The variants of EHVI included an exact formulation for bi-objective cases, and Monte Carlo-based approximations for noisy and noiseless cases with any number of objectives. As a new contribution to the field, we proposed adding an explicit exploration term to the HVI acquisition function to boost its exploration tendency. The exploration tendency of the resulting exploratory HVI acquisition function is determined by an exploration weight, which can be set according to application-specific needs. Additionally, the scalarization-based ParEGO approach and a generalization of the single-objective ELCB acquisition function to multi-objective cases were implemented.

We find that the selection of the acquisition strategy in multi-objective optimization is often a trade-off between computational cost and prediction accuracy. The results show that EHVI consistently offers high accuracy but is computationally expensive, especially if Monte Carlo approximations are used. On the other hand, scalarization-based approaches provide cheaper alternatives that might be sufficient for some applications. Adding the exploration term into HVI can greatly boost its performance in some applications, especially if the Pareto-optimal solutions exist in multiple disconnected regions. This makes exploratory HVI an attractive option, as it can achieve high prediction accuracy typical of hypervolume-based methods while remaining computationally cheap. However, a suitable value for the exploration term needs to be determined.

The computationally expensive hypervolume-based methods are most suitable when the cost of BO is negligible compared to other factors. A typical example is experimental

materials optimization, where running experiments dominates the time budget regardless of the computational overhead of BO [9]. In contrast, when working with high-dimensional search spaces and large datasets, the computational cost of BO can become a bottleneck. In this regime, faster scalarization-based methods are the better choice. An example of such a high-dimensional application is the design of new chemicals with generative AI [12].

There are several potential ways to mitigate the high computational cost of hypervolume methods. Future work could focus on implementing more efficient EHVI methods, especially for tasks that currently require gradient-free optimization of the acquisition function. Major reductions in computational cost can be achieved by implementing EHVI methods with gradients. This could be done by implementing an analytic and differentiable form of EHVI for tasks with more than two objectives [31]. Alternatively, auto-differentiation techniques could be used [5]. This may be a more attractive approach, as auto-differentiation approaches are easier to generalize to, for example, applications with noise, a high number of objectives or batch acquisitions. Regardless of the chosen approach, the modular implementation of multi-objective models and acquisition methods in the extended BOSS code ensures that adding new related functionalities is straightforward in the future.

This thesis focused on sequential optimization. Hence, multi-objective batch acquisitions were not investigated and are not currently supported by the BOSS code. In many practical applications, such as experimental materials synthesis, objective function evaluations are costly but can be conducted in parallel. Therefore, methods that enable multi-objective batch acquisitions are of interest and should be addressed in future work. To address this need, one interesting approach is the diversity-guided batch-selection method [20]. Concerning future development of the BOSS code, another interesting possibility is to further modularize the structure of multi-objective models. Users could be given the ability to combine different single-objective models into a multi-objective model. Then it would be possible to, for example, mix standard homoscedastic noise models with heteroscedastic noise models to accommodate diverse applications.

In summary, this thesis provides insights into the effectiveness of different acquisition strategies in multi-objective Bayesian optimization. The findings offer practical guidelines for BOSS users and the extended BOSS code offers an accessible, comprehensive toolkit for multi-objective optimization tasks. Overall, the work contributes to the broader development of efficient optimization techniques for multi-objective black-box functions.

Bibliography

[1] T. Beckers. An introduction to Gaussian process models, 2021.

[2] S. Belakaria, A. Deshwal, and J. R. Doppa. Max-value entropy search for multi-objective Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[3] F. Biscani and D. Izzo. A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53):2338, 2020.

[4] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *ArXiv*, abs/1012.2599, 2010.

[5] S. Daulton, M. Balandat, and E. Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[6] S. Daulton, M. Balandat, and E. Bakshy. Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2024. Curran Associates Inc.

[7] J. Davins-Valldaura, S. Moussaoui, G. Pita-Gil, and F. Plestan. ParEGO extensions for multi-objective optimization of expensive evaluation functions. *Journal of Global Optimization*, 67(1–2):79–96, 2017.

[8] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 1, pages 825–830, 2002.

[9] D. Diment, J. Löfgren, M. Alopaeus, M. Stosiek, M. Cho, C. Xu, M. Hummel, D. Rigo, P. Rinke, and M. Balakshin. Enhancing lignin-carbohydrate com-

plexes production and properties with machine learning. *ChemSusChem*, 2024. https://doi.org/10.1002/cssc.202401711.

[10] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[11] M. T. M. Emmerich, K. Yang, and A. H. Deutz. *Infill Criteria for Multiobjective Bayesian Optimization*, pages 3–16. Springer International Publishing, Cham, 2020. https://doi.org/10.1007/978-3-030-18764-471.

[12] R.-R. Griffiths and J. M. HernÃ¡ndez-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chem. Sci.*, 11:577–586, 2020.

[13] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. The hypervolume indicator: Computational problems and algorithms. *ACM Computing Surveys*, 54(6):1–42, 2021.

[14] J. Hakanen and J. D. Knowles. On using decision maker preferences with ParEGO. In *Evolutionary Multi-Criterion Optimization*, pages 282–297, 2017.

[15] K. Hanaoka. Comparison of conceptually different multi-objective Bayesian optimization methods for material design problems. *Materials Today Communications*, 31:103440, 2022.

[16] D. Hernandez-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams. Predictive entropy search for multi-objective Bayesian optimization. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48, pages 1492–1501, 2016.

[17] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima. How to specify a reference point in hypervolume calculation for fair performance comparison. *Evolutionary Computation*, 26(3):411–440, 2018.

[18] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

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

[20] M. Konakovic Lukovic, Y. Tian, and W. Matusik. Diversity-guided multi-objective Bayesian optimization with batch evaluations. In *Advances in Neural Information Processing Systems*, volume 33, pages 17708–17720, 2020.

[21] B. Paria, K. Kandasamy, and B. Póczos. A flexible framework for multi-objective Bayesian optimization using random scalarizations. In *Conference on Uncertainty in Artificial Intelligence*, 2018.

[22] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In *Parallel Problem Solving from Nature – PPSN X*, pages 784–794, 2008.

[23] C. F. R. Viennet and I. Marc. Multicriteria optimization using a genetic algorithm for determining a pareto set. *International Journal of Systems Science*, 27(2):255–260, 1996.

[24] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

[25] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

[26] R. Steuer and E. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, pages 326–344, 1983.

[27] A. Tharwat, E. Houssein, and M. Ahmed. MOGOA algorithm for constrained and unconstrained multi-objective optimization problems. *Applied Intelligence*, 48(1):2268–2283, 2018.

[28] M. Todorović, M. U. Gutmann, J. Corander, and P. Rinke. Bayesian inference of atomistic structure in functional materials. *Npj computational materials*, 5(1):35, 2019.

[29] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser. On expected-improvement criteria for model-based multi-objective optimization. In *Parallel Problem Solving from Nature, PPSN XI*, pages 718–727, 2010.

[30] K. Yang, A. Deutz, Z. Yang, T. Back, and M. Emmerich. Truncated expected hypervolume improvement: Exact computation and application. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4350–4357, 2016.

[31] K. Yang, M. Emmerich, and A. Deutz. Efficient computation of expected hypervolume improvement using box decomposition algorithms. *Journal of Global Optimization*, 75:3–34, 2019.

[32] K. Yang, M. Emmerich, A. Deutz, and T. Bäck. Multi-objective Bayesian global optimization using expected hypervolume improvement gradient. *Swarm and Evolutionary Computation*, 44:945–956, 2019.

[33] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Appendix A. Supplementary data

Noise values

Table A.1 displays the noise standard deviation values used for each test function and objective. The values were determined by eye based on plots of the true Pareto fronts of each test function. These values are used to generate noise proportional to the true Pareto fronts. An exception is the Lignin function, for which the values correspond to the true observed noise in the data [9].

Function	Noise std.
ZDT1	(1, 1)
ZDT2	(1, 1)
ZDT3	(1, 1)
BC	(40, 15)
Viennet	(400, 1350, 1)
Lignin	(1.666, 5.433)

Table A.1: The noise standard deviation values used for each test function and objective.

Objective functions

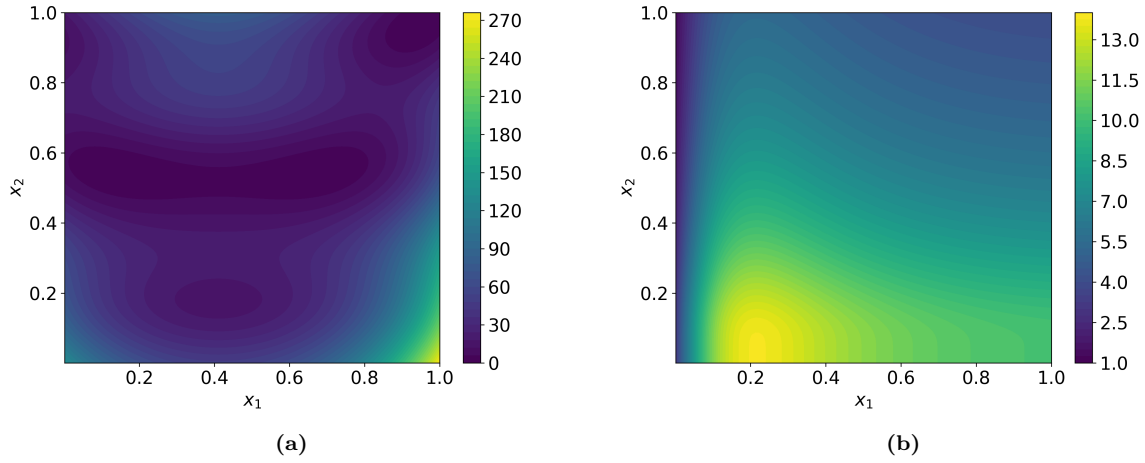


Figure A.1: The functions a) f_1 and b) f_2, which make up the Branin-Currin test function.

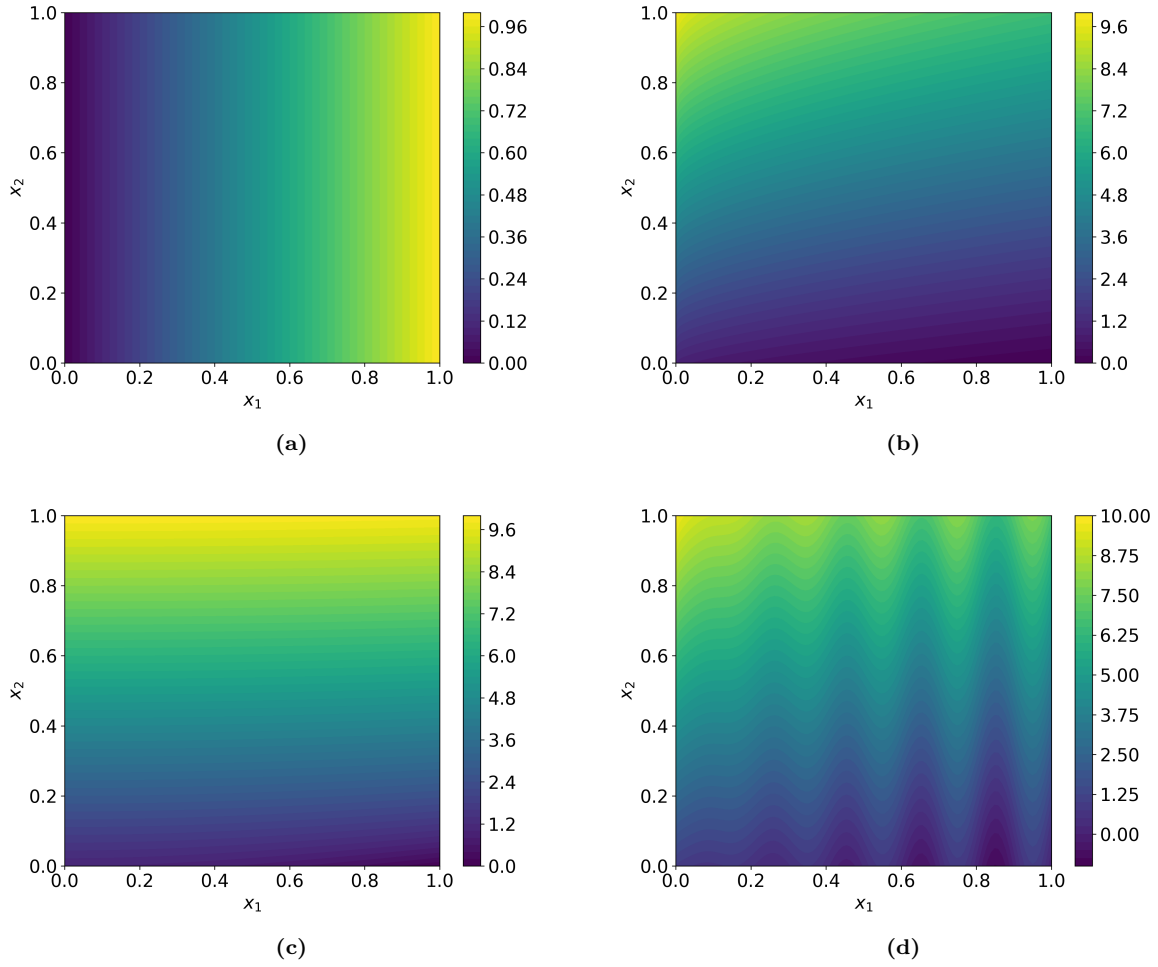


Figure A.2: The function a) f_1, which is shared among the ZDT functions, and the function f_2 for b) ZDT1, c) ZDT2 and d) ZDT3.

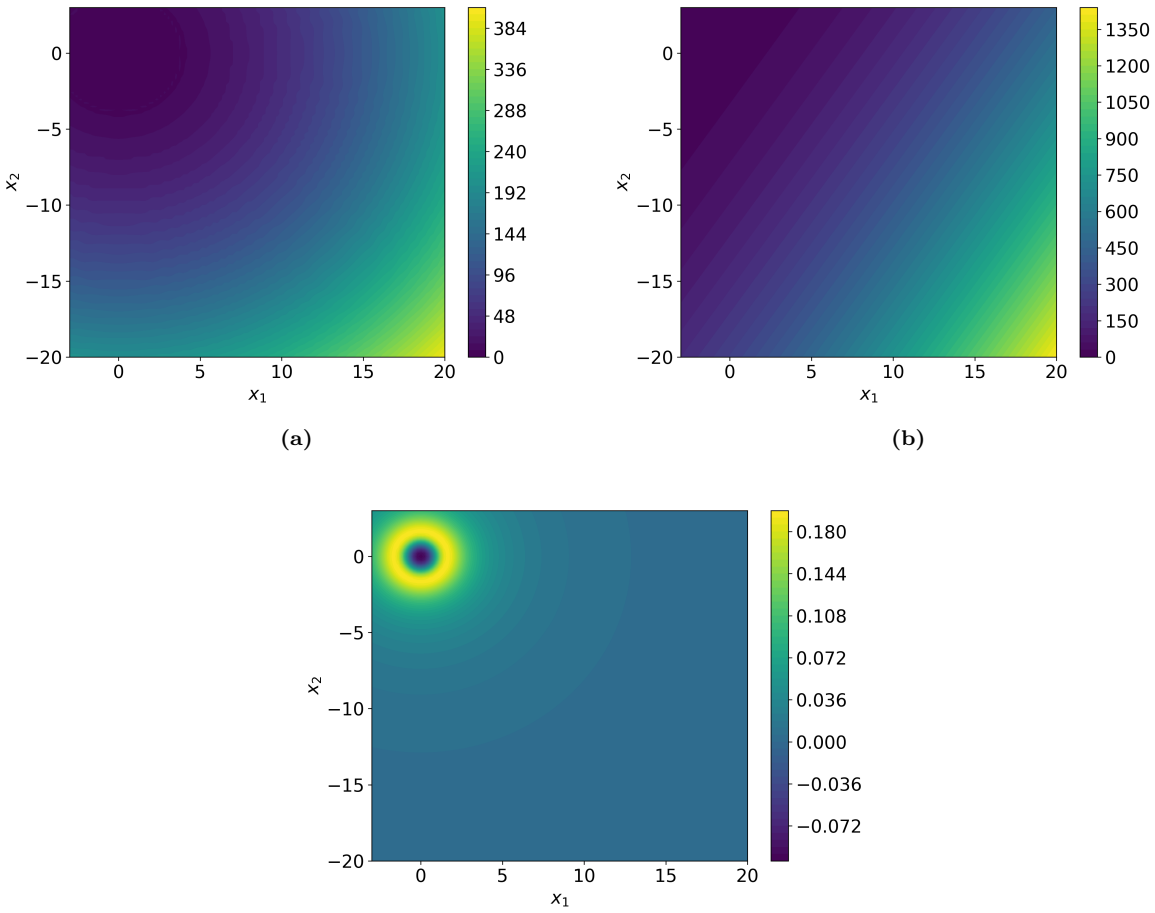
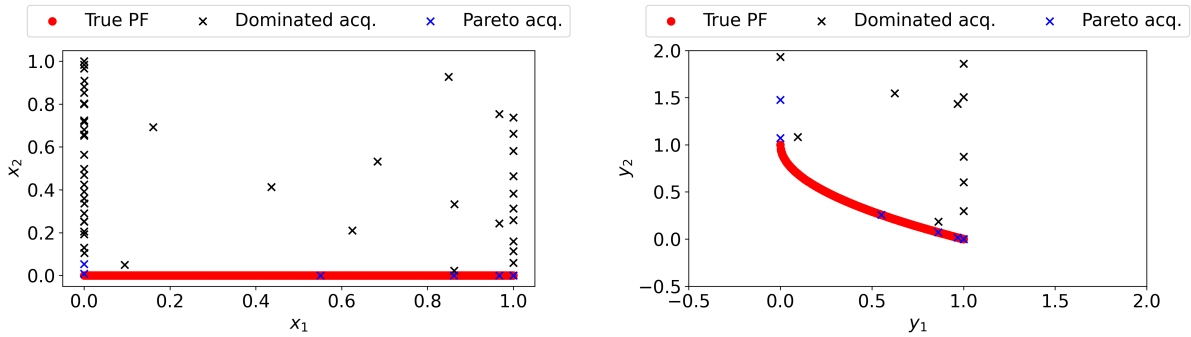


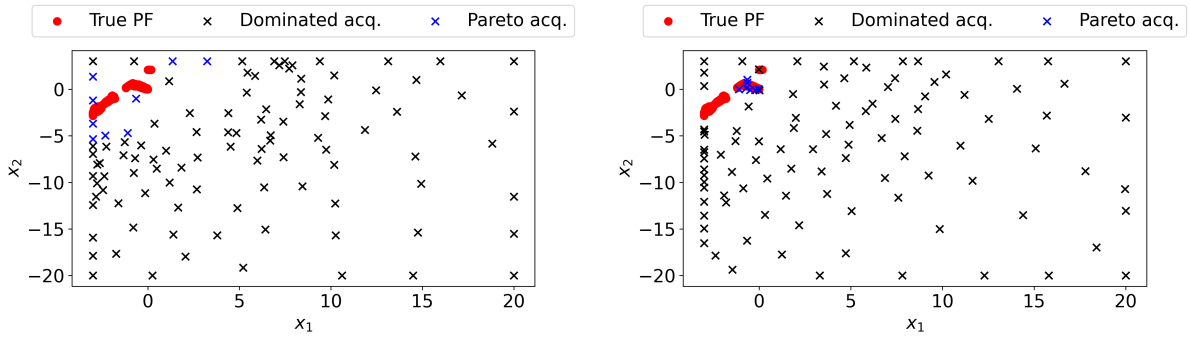
Figure A.3: The functions a) f_1, b) f_2 and c) f_3, which make up the Viennet test function.

Appendix B. Exemplary BO runs



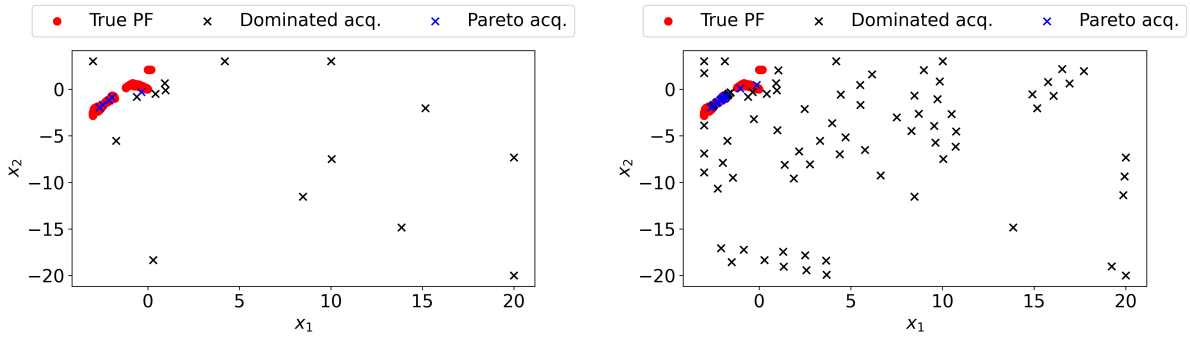
(a) **(b)**

Figure B.1: An exemplary BO run with the ZDT1 function at the 50th iteration using multi-objective ELCB. The a) search space and b) objective space, along with the dominated and non-dominated (Pareto) acquisitions, are shown.



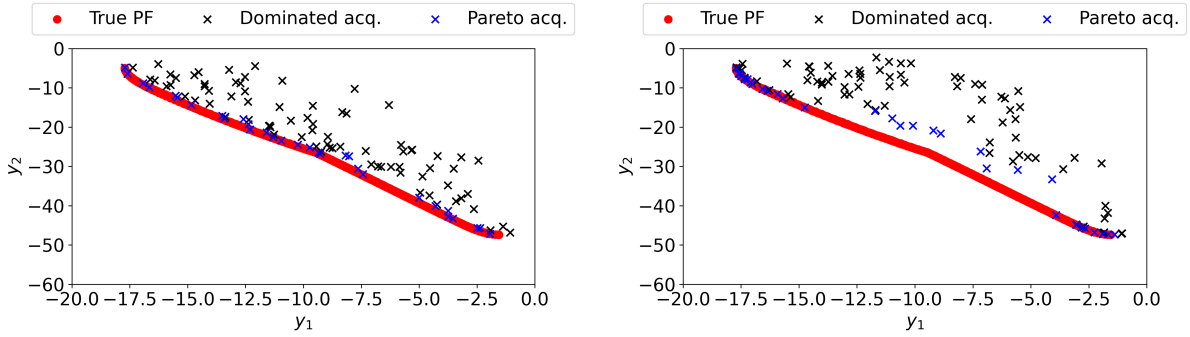
(a) **(b)**

Figure B.2: Two exemplary BO runs with the Viennet function at the 100th iteration using a) Scal-AT and b) Scal-WS. The objective spaces, along with the dominated and non-dominated (Pareto) acquisitions, are shown.



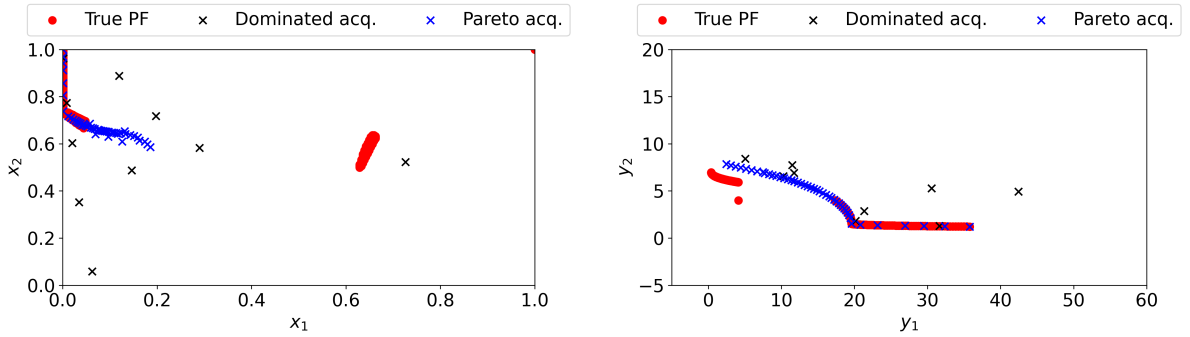
(a) (b)

Figure B.3: An exemplary BO run with the Viennet function at a) the 20th iteration and b) the 100th iteration using multi-objective ELCB. The search spaces, along with the dominated and non-dominated (Pareto) acquisitions, are shown.



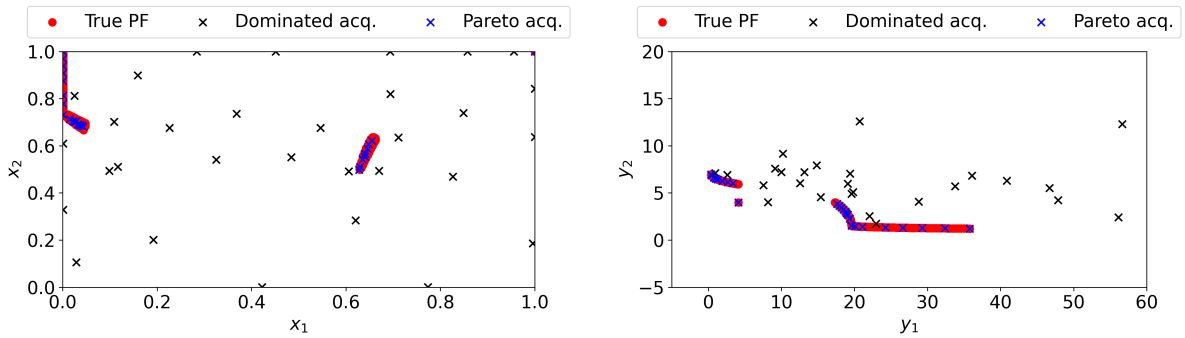
(a) (b)

Figure B.4: Two exemplary BO runs with the Lignin function at the 100th iteration using a) Scal-AT and b) Scal-WS. The objective spaces, along with the dominated and non-dominated (Pareto) acquisitions, are shown.



(a) (b)

Figure B.5: An exemplary BO run with the Branin-Currin function at the 50th iteration using HVI ($\alpha = 0$). The a) search space and b) objective space, along with the dominated and non-dominated (Pareto) acquisitions, are shown.



(a) **(b)**

Figure B.6: An exemplary BO run with the Branin-Currin function at the 50th iteration using HVI ($\alpha = 1$). The a) search space and b) objective space, along with the dominated and non-dominated (Pareto) acquisitions, are shown.