New Paradigms for Bayesian Optimization: Harnessing Physics & High-Throughput Experiments

by

Leonardo D. González

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Chemical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2024

Date of final oral examination: 8/13/2024

The dissertation is approved by the following members of the Final Oral Committee:
Victor M. Zavala, Professor, Chemical and Biological Engineering
Reid C. Van Lehn, Associate Professor, Chemistry
Styliani Avraamidou, Assistant Professor, Chemical and Biological Engineering
Lav Thyagarajan, Chief Engineer, Eaton

To my mother and brother, thank you for being there every step of this journey.

ACKNOWLEDGMENTS

After a long and incredible journey, I can finally say that I have finished the 21st grade. Pursuing a doctoral degree was not a path I envisioned myself going down up until about five years ago. However, it has been an incredibly rewarding and fulfilling experience that I will always remember. That being said, this path was not without its trials, and a major part of the reason I have gotten to this point is because of the wonderful friends, mentors, and family who have supported me throughout this journey

Words cannot fully describe the debt of gratitude I have to my dearest mother. Without her, I would not be here (literally!), and it is largely thanks to her tireless self-sacrifice and devotion for her children that I was able to have the opportunities that I did. From her I learned to be strong, disciplined, and unyielding and to value compassion, honesty, and selflessness, all values that she has always embodied. She has constantly pushed me to excel, to dream big and never doubted my capacity to succeed, even when I did. It is not an exaggeration to say that no one has ever believed in me as much as she has. Mom, thank you for everything; my success is your success and I will forever be grateful.

To my brother, Ramses, thank you for keeping me grounded, reminding me to enjoy life, and for teaching me to not be afraid of failure. When we were kids, and even now sometimes, he was the one always trying and pushing me to try new things. His encouragement to follow my curiosity and venture into the unknown is a big part of the reason why I felt I could come down this path in the first place. I am also thankful we got the opportunity to be able to go through the graduate school journey together. It was blessing to have someone who could understand my experiences at the same level as him. I also deeply appreciate the regular phone calls and the game nights; they helped keep me sane and gave me something to look forward to.

I am grateful for the opportunity I received to attend the Texas Academy of Mathematics and Science (TAMS), which is where my journey in higher education started. Being a part of this program gave me the opportunity to learn at a level and pace I had never experienced before and gave me the confidence to believe I could successfully pursue a STEM career. It was here that I met the late Dr. John Ed Allen who, in addition to being one of the best teachers I have had, encouraged me to explore my interest in mathematics. Additionally, my two years at TAMS allowed me to be a part of a tight-knit community

of peers with whom I was able to share countless experiences. This allowed me to forge lasting friendships with Ryan Aven and Wyn Bell, who have provided me with endless encouragement and support no matter the time or distance that separate us.

I am fortunate to have received my undergraduate education at the University of Pennsylvania and Texas A&M University. The academic rigor and reputation of these institutions equipped me with the know-how to be successful beyond undergrad and provided me with the employment and graduate school opportunities I received after graduating. My undergraduate experience would not have been the same without the incredible professors who instructed me. In particular, I would like to acknowledge Dr. Dennis DeTurck and Dr. Perla Balbuena, whose mentorship instilled me with the confidence to seek to solve difficult problems. Dr. Balbuena's willingness to welcome me into her research lab provided me with an invaluable experience that I never imagined I would have. This, along with her repeated assertions that I ought to seriously consider a postgraduate education, were highly encouraging and motivated my decision to pursue a graduate degree.

I have deeply cherished my time at UW-Madison. The landscape of Madison is a stark contrast to the environment I grew up in in Southeast Texas. Perhaps this is a large part of the reason I have so fully embraced the natural beauty and four seasons I have experienced here. However, as beautiful as Madison is, it has been my advisor, Dr. Victor Zavala, who has been truly instrumental in making my graduate school journey the memorable experience it has been. During my time in his lab, Victor has given me the opportunity to work on an incredibly wide breadth of projects, which has allowed me to learn about topics I had never thought about before. This has greatly expanded my perspective of the world, and I have come to appreciate the "systems viewpoint" he frequently talks about. Through his excellent mentorship, he has helped me develop into an effective researcher and engineer by fostering my curiosity, critical thinking, and communication skills. I am also thankful for all the professional development opportunities he gave me, whether it was sending me to several conferences or letting me take two summers off for an internship. These opportunities provided me with the network and visibility I needed to forge my path after completing my degree. More than anything, however, I am grateful for who Victor is as a person and the compassion and respect with which he treats his students. I never felt that he saw me as just a graduate student, but rather, that he was genuinely invested in my success for the sake of seeing me succeed. I appreciate the joy and sincerity with which he celebrates the accomplishments of all of us; it is this genuineness that has allowed the Zavalab to feel like a second family. Victor, I will miss you and forever be grateful for having had the opportunity to be a part of your group.

I would like to extend my gratitude to all the members of the Zavalab, whose friendship and support made me look forward to coming to the office. In particular, I would like to acknowledge Qiugang Lu and Ranjeet Kumar for their mentorship and guidance with my first publication. Thank you to my cohort-mates, Amy Qin, Dilara Göreke, Jiaze

Ma, Lisa Je, and Bo-Xun Wang for their support and camaraderie during first semester classes, prelims, and preparing for my final defense. I will always be grateful for the guidance given to me by Aurora Mungiá-López as well as our regular conversations. I appreciate the support Brenda Cansino Loeza and Joe Flory have given me during the final months of my Ph.D. as I attempted to finalize my work for various projects. I would also like to express my gratitude towards Philip Tominac, Jordan Jalving, Apoorva Sampat, Yicheng Hu, Sungho Shin, Joshua Pulsipher, Ricky Shao, Alexander Smith, Shengli Jiang, and Weiqi Zhang for the warmth and openness with which they received me when I first joined the lab. David Cole, Jaron Thompson, Blake Lopez, Elvis Umaña, Daniel Laky, Ugochukwu Ikegwu, Yoel Cortés-Peña, and Brandon Flores, I will miss our office conversations and group outings. It has been a privilege being able to work and learn alongside you all.

I would like to say thank you the members of my dissertation committee: Dr. Victor Zavala, Dr. Reid Van Lehn, Dr. Styliani Avraamidou, and Dr. Lav Thyagarajan for giving of their time and expertise to make this journey possible. I would also like to acknowledge Dr. George Huber and Dr. Ophelia Venturelli for serving on my preliminary exam and fourth-year progress meeting committees. The feedback and guidance they have all provided have been an essential part in the development of my research plan. Additionally, they have regularly encouraged me to consider the broader impacts of my work and to strive to be an effective communicator.

I am deeply grateful for the incredible mentorship I received during my internships at John Deere and Cargill. Dr. Lav Thyagarajan and Abram Haich gave me the opportunity to work on the very topic that inspired me to pursue a graduate degree. During my time at Deere, they devoted themselves to guiding me through the process of developing a machine learning application for use in an industrial setting. The excitement with which they received me and the enthusiasm they had for the work we did were an integral part of the success we achieved that summer. While at Cargill, I was able to work alongside Dr. Ashwin Chemburkar, who introduced me to the world of custom process models and gave me the opportunity to experience a whole new side of chemical engineering. His tireless diligence and support allowed me to thrive, and it is largely thanks to him that I received the opportunity to return to Cargill. I would also like to acknowledge Dr. Chris Tyler and Dr. Jesse Pikturna, both of who were always willing to answer any question I had and provided constant encouragement.

I am extremely fortunate to have an incredible group of friends who helped fill this journey with wonderful experiences. Thank you to Jacob Tamayo and JC Davila for our Monday Smash Ultimate nights that allowed us to unwind and stay connected. I will miss the summer bike rides that I would regularly go on with Atharva Kelkar and Kevin Sánchez-Rivera; these adventures allowed me to explore a great deal of the city and its surroundings. I will always cherish the memories of the countless Chili's and MOOYAH lunches and dinners with Edgard Lebrón-Rodríguez. The conversation topics during

these meals never failed to entertain, and I appreciate Edgard's understanding and humor. I am thankful to Lisa Je and her enthusiasm for setting up dog park dates as well as the affection and care she has shown for my pets. I will always remember the random adventures Amy Qin, Dilara Göreke, and I went on together, especially when we drove to Milwaukee to buy furniture and somehow managed to fit it all in my Corolla. I would also like to extend my gratitude to Kelly Burton and the GERS community for being incredibly supportive and providing me with a safe space where I could share my experiences and listen to those of others.

Finally, I wish to acknowledge my wonderful dogs, Darby and Mona Lisa, who have been a constant source of joy and comfort during my time here. Their presence has made the hard days easier and the good days better. I would also like to express my immense gratitude and love for my incredible partner, Becca; my time here would not have been the same without her. I greatly appreciate the loving encouragement and infinite patience she was shown during this journey—I know it's not been easy! Thank you for always making me feel appreciated and cared for and for never failing to remind me that I am not alone.

Leonardo D. González
Madison, WI
August 2024

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This dissertation explores integrating available physics knowledge with the Bayesian optimization (BO) framework to develop a new set of paradigms that improve performance and enable functionalities like parallel or high-throughput experimenting. BO is a data-driven paradigm that uses input/output data to construct a surrogate model of the underlying performance function, f, of a black-box system, which it uses to navigate a design space in search of an optimum. While several other black-box optimization strategies exist, BO sets itself apart by using a *probabilistic* surrogate model that estimates prediction uncertainty in addition to performance. This feature allows for the consideration of informational value when selecting a new sample point, which pushes the algorithm to explore from several distinct regions of the design space. Consequently, BO is an especially powerful global optimizer that has been successfully applied to a wide array of problems including model predictive control, materials design, and process engineering.

While the standard formulation of BO (S-BO) is generally effective and highly generalizable, it exhibits two major drawbacks: (i) structural knowledge (e.g., physics and component interconnections) is often available for many systems, and S-BO does not allow for the consideration of this information; and (ii) the S-BO algorithm is inherently sequential (i.e., proposes one experiment at a time), which makes it unable to exploit high-throughput experiment capabilities. This represents a missed opportunity, as these resources can significantly enhance algorithm speed and performance. Various works have proposed modifications to the BO framework to overcome these shortcomings. However, these solutions can be can be limited by tractability and performance issues.

We address these challenges by developing a new set of paradigms that expand the capabilities of the BO framework in a computationally efficient and robust manner. Using these methods we are able to take advantage of low-fidelity approximations (referred to as reference models) built from simplified representations or empirical correlations to improve algorithm speed and sampling efficiency. We are also able to efficiently exploit system connectivity as well as scenarios where the system is a mixture of interconnected white-box and black-box components. This is achieved by shifting to a composite function representation of f, whose statistical moments are estimated using an adaptive linearization scheme. Finally, we leverage the coarse system trends provided by the reference model as well as the partially separable structure of different system components, which we determine by analyzing system connectivity, to develop an innovative pair of parallel BO methods that allow for system-specific partitioning of the design space.

Chapter 1

INTRODUCTION

In this chapter, we detail the background, motivation, and objectives of this dissertation. We introduce the concept of black-box optimization and provide an overview of the various methods for solving this class of problems, with a focus on Bayesian optimization (BO). We discuss the features of BO that have made it an especially effective optimization paradigm, and explain some of its shortfalls. Recent advances aimed at addressing these shortfalls are presented and used to establish the existing challenges that our work seeks to address. Finally, we outline the structure of this dissertation and summarize the content of its constituent chapters.

1.1 Black-Box Optimization

In this work, we consider general unconstrained optimization problems of the form:

$$\min_x f(x) \tag{1.1a}$$

$$\text{s.t. } x \in X \tag{1.1b}$$

where $f : X \to \mathbb{R}$ is a scalar performance function; x is a set of inputs; and $X \subseteq \mathbb{R}^{d_x}$ is the design space (of dimension d_x). Problems of this form are prevalent in diverse science and engineering applications, where a key goal is often to identify the configuration that maximizes the performance of a system (negative cost) or minimizes operation or

production costs [6]. Various approaches exist for solving (1.1), but by and large the most commonly used are derivative-based methods; this is largely due to their efficiency as well as the convergence guarantees they provide [7]. Consequently, significant work has been done on developing improved gradient-based optimizers, resulting in a rich variety of methods, including SQP, L-BFGS, and IPOPT [8, 9, 10]. While gradient-based optimizers have been successfully applied to a wide array of problems in almost every discipline, obtaining the required derivative information is often not possible in many real-world applications (e.g., process engineering, synthetic biology, machine learning, etc.). These applications usually involve systems that exhibit a significant level of complexity that make it near-impossible to derive closed-form representations that map system inputs to outputs; in other words, the system is a black-box where f can only be evaluated by sampling at an x of interest. Estimating the gradients numerically is also usually not possible as sampling is often expensive and the obtained data can be corrupted by noise. In these scenarios, solving (1.1) requires the use derivative-free methods, also referred to as black-box optimization algorithms.

Black-box optimization algorithms can broadly be separated into two categories: direct search and model-based methods [11]. Direct search methods include some of the earliest and most widely used black-box optimizers such as simplex methods, pattern searches, and evolutionary algorithms [12, 13, 14]. As their name implies, these algorithms navigate the design space by sampling the system at several points surrounding the current iterate to orient their search. Once a more favorable point is found, the algorithm moves towards it and then proceeds to search around that point. Direct search methods are quite flexible and do not require that the performance function be smooth or even continuous [15]. When compared to gradient-based methods, they are also generally conceptually simpler and easier to implement [16]. Additionally, these algorithms easily lend themselves to parallelization as the required function evaluations can be done independently or various instances of the algorithm can be initialized at separate points in the design space [17]. However, the sampling requirements can cause these algorithms

to be inefficient, resulting in slow convergence and scalability issues as the number of required function evaluations can increase significantly with the dimensionality of X. These methods can also be quite sensitive to the initialization point, and a poor initial guess can result in the algorithm getting trapped in a local optimum and can also significantly slow down the algorithm by further increasing the number of required function queries [18].

Model-based black-box optimization algorithms use the input/output data obtained from sampling the system to build a surrogate model of the performance function that is used to guide the search for the minimum [11]. A new sample point is selected based on a predefined criterion (e.g., minimizing the model value or maximizing model accuracy), and the surrogate model is iteratively refined as new data becomes available, improving its accuracy. These methods tend to be more complex than direct search, as they require training the surrogate with the generated data. Additionally, model selection and tuning are dependent on the specific system being analyzed, often requiring the use of heuristics or domain knowledge [19]. However, surrogates provide a significantly better understanding of global system trends than direct search. As a result, model-based algorithms are generally able to sample across a wider range of the design space rather than focusing on a few local regions [20]. Additionally, because the surrogate provides estimates for the value of f, these methods usually require significantly less function evaluations, making them more computationally efficient [21]. Furthermore, while model selection and tuning can be intensive tasks, they provide an opportunity to incorporate available information on the structure or behavior of the performance function. There exists a diverse selection of model-based algorithms such as Response Surface Methodology (RSM), radial basis function (RBF) methods, and kernel machines [22, 23, 24]. In recent years, Bayesian optimization has emerged as arguably the most popular optimization paradigm of this class due to its efficiency, flexibility, and capability to measure both information (exploration) and performance (exploitation) value when selecting new sample points via the use of a probabilistic surrogate model. A detailed description of the BO framework is provided in Chapter 2.

1.2 Current State of Bayesian Optimization

Bayesian optimization was initially developed and formalized in the 1960s and 70s by Kushner, Saltenis, and Mockus [25, 26, 27]. However, it was the rise of deep learning in the last decade that propelled BO to its current status, as it proved to be an especially effective method for tuning the various hyperparameters of complex machine learning models [28]. Today, BO solutions exist for tackling complex problems in various fields such as polymers, reaction engineering, and robotics [29, 30, 31]. Despite the success that the algorithm has had, there still remain various significant challenges that need to be addressed such as scalability, multi-objective optimization, constrained optimization, transfer learning or the incorporation of domain knowledge (e.g., physics), and parallelization [32, 33]. The methods presented in this work focus on addressing the last two elements.

1.2.1 *Existing strategies for incorporating physics knowledge in BO*

While the hyperparameter tuning problems that popularized BO are akin to true black-boxes, substantial domain knowledge is often available when dealing with physical systems (e.g., simplified physics models, simulations, empirical correlations, etc.). It stands to reason that the algorithm can benefit from this knowledge, as it can provide insight on coarse system trends and highlight potentially promising regions of the design space, thereby reducing the number of samples required to locate a solution. Recent efforts to exploit this type of system knowledge largely fall into two classes: multi-fidelity BO (MFBO) and composite function BO.

MFBO initially runs BO on a low-level representation of the system that is cheaper to sample. After the algorithm identifies the optimal regions of the design space using this model, a higher fidelity model is used to continue the search within these regions and further reduce the design space; this process continues until the real system is used

to obtain the final samples [34]. As a result, MFBO can significantly reduce the number of times that the real system is queried since the search is restricted to a much smaller space by the time the real system is used. This allows for a better use of the high-fidelity sample budget as it reduces the probability that the selected points are highly sub-optimal [35]. However, MFBO paradigms are quite complex as they require generating and managing multiple surrogate models across the various fidelity levels and integrating the information they provide. Additionally, determining the right level of fidelity as well as the point at which to switch to a higher fidelity model often relies on trial-and-error and expert knowledge [36]. These additional requirements can significantly increase the computational overhead of the algorithm, thereby reducing the net benefit obtained from using the low-fidelity models.

Composite function BO is a simple but powerful strategy that exploits system structure by representing f as a known composition of a vector-valued intermediate function, $y(x)$ [37]. The intermediate function measures the value of various internal system components that impact its performance (e.g., production rates, resource consumption, product purities). The elements of the intermediate function can either be modeled using closed-form representations (if available) or treated as black-boxes and estimated using a surrogate model. Additionally, the model of any particular element, y_i, can be defined such that it only depends on a subset of the variables in x or utilizes some other element, y_j, as in input. As a result, composite function BO allows for a high degree of customization in the construction of the intermediate function and the corresponding surrogate models, which can be leveraged to significantly improve the performance of the algorithm and increase its functionality[38]. While composite functions provide an intuitive and effective manner for representing the system, the implementation of this approach is not as straightforward. BO measures information value in the form of the prediction uncertainty of the surrogate model. Because $y(x)$ is not the performance metric being optimized, shifting the surrogate modeling focus to $y(x)$ introduces complexities due to the need to accurately propagate the uncertainty estimates from the intermediate predictions

to f. Existing composite function BO frameworks handle this issue by either numerically estimating the probability density of f via sampling methods like Monte Carlo or by solving an auxiliary problem over an augmented space where the range of y is constrained by set of upper and lower confidence bounds determined by the surrogate model [39, 40]. These approaches suffer from significant scalability issues as the problem size increases, making them rather computationally inefficient.

1.2.2 *Existing strategies for enabling parallelized BO*

Standard Bayesian optimization (S-BO) is a sequential algorithm that only proposes one new sample point at a time. A key strength of this features is that it enables a closed-loop design strategy, where the selection of the next sample point is fully informed by previous results [32]. However, such an approach is incompatible with high-throughput experiment (HTE) platforms (e.g., liquid-handling robots, parallel computing) that are capable of running multiple experiments (querying from the system at several points) in tandem. As these platforms have increasingly become more widely-available, the need to develop algorithms capable of designing batches rather than sequences of experiments has motivated the development of parallel BO strategies. Some of the more widely-used methods that have been devised include batch Bayesian optimization (q-BO), fantasy sampling, and Thompson sampling [41, 42, 43]. Because parallel BO presents several challenges, such as maintaining the capability to consider exploration and exploitation, algorithm efficiency, and potential asynchronicity considerations, the existing paradigms tend to be rather complex. As a result, they can be difficult to implement and often introduce an additional layer of hyperparameters that must be tuned[44, 45, 46]. Additionally, many of these methods are limited in the degree of parallelization that they can achieve, may lack mechanisms to prevent the suggestion of redundant experiments, and can encounter scalability issues at high dimensions [47, 48].

1.3 Research Objectives

The overarching theme of this work is the development of a novel set of strategies for harnessing available structural system knowledge, largely rooted in physics and sparse system interconnectivity, in a BO setting to improve algorithm performance and enable compatibility with HTE platforms. Our aim is to address the performance, scalability, and complexity challenges observed in existing methods while also providing a set of paradigms that are easy to implement and allow for a high level of user customization. To accomplish this goal, we seek to achieve the following:

- Avoid the challenges associated with working with several models of varying fidelity levels by incorporating available system knowledge directly into the BO framework via a single low-fidelity model.
- Harness knowledge of fundamental physical principles (e.g., mass and energy balances) along with sparse system interconnectivity and available white-box models to better specify the elements of y in composite function BO and their dependencies.
- Leverage the customization of the intermediates in composite function BO to shift the surrogate modeling task to a set of less complex and easier-to-learn variables.
- Reduce the computational cost of propagating the uncertainty estimates from y to f in composite function BO.
- Provide the BO algorithm with parallelization capabilities by utilizing the level sets of the performance function (which can be approximated with a low-fidelity model) to decompose the design space into a set of unique partitions with no overlap (user-defined or uniformly spaced) that can be explored individually and in parallel.
- Parallelize BO by exploiting the partially separable structure of the system, identified using system interconnectivity information or an available low-fidelity model

(or both), to split it into a set of modules that are optimized along a subset of the total inputs separately and in tandem.

- Package the developed methods into open-source software tools, ensuring they are readily usable in a variety of applications.

1.4 Dissertation Outline

This dissertation consists of two parts. Part I is composed of Chapters 2 through 4 which deal with sequential BO and introduce two paradigms that enable the use of various forms of available physical knowledge by the algorithm. Part II consists of Chapter 5 and focuses on the use of physics-based information to extend BO to the parallel setting. Chapter 6 provides a summary of the key findings of this work and suggestions for future research directions, concluding this dissertation. An outline of each chapter is included below.

Chapter 2 presents the various elements of Bayesian optimization. We discuss how Bayes' Theorem is leveraged to construct a sequential sampling strategy that, when paired with a goal-oriented decision making mechanism, leads to the BO paradigm. We formally outline the standard Bayesian optimization (S-BO) framework, and highlight its two core components: the surrogate model and the acquisition function. Next, an introduction to Gaussian processes (GP), the most common surrogate model choice in BO, is provided along with a brief overview on some outstanding research questions in the field of GPs. This chapter then concludes with a discussion on acquisition functions (AFs), which serve as the decision-making mechanism of the algorithm, and provides a summary of commonly used AFs.

Chapter 3 introduces the Reference-Based BO (Ref-BO) algorithm. Ref-BO incorporates physics knowledge via a low-fidelity representation of the system (which we refer to as the reference model) that is directly integrated into the optimization framework, eliminating the need to manage models of varying fidelity levels. This approach provides

the algorithm with a prior understanding of coarse system trends that enables faster identification of a solution and results in a more robust performance. We demonstrate the effectiveness of Ref-BO using a using a case study based on an MPC tuning problem at a real-life HVAC plant.

Chapter 4 explores the incorporation of physics knowledge based on the connectivity of system components. This is accomplished by using a composite representation of the performance function that is optimized using the Bayesian Optimization of Interconnected Systems (BOIS) algorithm, which we present in this chapter. This paradigm facilitates the use of composite functions in a BO setting by employing an adaptive linearization scheme that allows for the derivation of a set of closed-form expressions for the mean and variance of f. Additionally, this framework allows for better specification of the surrogate models, which improves their quality and facilitates training. We illustrate the advantages that BOIS provides over S-BO and existing composite function BO algorithms via case studies.

Chapter 5 leverages system knowledge to partition the design space and enable the compatibility of Bayesian optimization with HTE platforms. This is accomplished via the level-set BO (LS-BO) and variable-partitioning BO (VP-BO) algorithms, which are presented in this chapter. These algorithms expand the utility of the reference model by using it generate system-specific partitions that enable a more efficient division of HTE resources. LS-BO and VP-BO are straightforward to implement, can eliminate the occurrence of redundant sampling, and are capable of achieving high degrees of parallelization. We use a reactor network case study to benchmark the effectiveness of these algorithms against S-BO as well as other state-of-art parallel BO strategies.

Chapter 6 summarizes the key findings of this dissertation. We conclude with an outline of potential future directions of research and discusses relevant work that can be applied to or further motivates the continued development of the BO frameworks we have presented.

Part I

INCORPORATING PHYSICS KNOWLEDGE

Chapter 2

BAYESIAN OPTIMIZATION

Portions of this chapter are adapted with permission from González and Zavala from a working paper with a preprint available at https://doi.org/10.48550/arXiv.2311.11254

In this chapter, we present the Bayesian optimization framework. We begin with a discussion of Bayes' Theorem, which is the central idea behind BO, and explain how this concept is harnessed to develop an optimization strategy. We then provide a detailed overview of the two key components of the algorithm: (i) the probabilistic surrogate model and (ii) the acquisition function (AF). Our discussion of the surrogate model will focus on the Gaussian process (GP), as this is the most commonly used surrogate in BO. Similarly, while a wide array of AFs can be found in the literature, we will focus on the Probability of Improvement (PI), Expected Improvement (EI), and Lower Confidence Bound (LCB) functions, as these are the most common choices. Note that all of the results presented in this work were obtained using the LCB AF. Despite the fact that the exact implementation of the algorithm can differ significantly between different BO paradigms, these key elements are common across all variations of Bayesian optimization. Thus, the introduction to these concepts provided in this chapter is meant to facilitate the discussion of the various methods presented in this work.

2.1 Formulation of Bayesian Optimization

Bayes' Theorem calculates the conditional probability of an event happening given the occurrence of a correlated event [49]. Consider an unknown performance function, $f : X \to \mathbb{R}$, that is defined over some design space or domain, $X \subseteq \mathbb{R}^{d_x}$, Bayes' Theorem can be leveraged to estimate the value of f at some point of interest, $x \in X$, based on previous observations, $\mathcal{D}$, as follows:

$$p(\hat{f}|\mathcal{D}) = \frac{p(\mathcal{D}|\hat{f}) \cdot p(\hat{f})}{p(\mathcal{D})} \tag{2.1}$$

Here, $p(\hat{f})$ is referred to as the prior distribution and gives the probability that $f(x)$ takes on some value $\hat{f}$; $p(\mathcal{D})$ serves as a normalization factor that ensures the total probability sums to 1. The likelihood function, $p(\mathcal{D}|\hat{f})$, measures the likelihood that the observations in $\mathcal{D}$ would occur if $\hat{f}$ were the true value of $f(x)$. The posterior, $p(\hat{f}|\mathcal{D})$, denotes the updated belief in the function being equal to $\hat{f}$ at x based on the data in $\mathcal{D}$. Extending the posterior across all potential outcomes of $f(x)$ allows for the construction of a probability distribution of function values. This distribution provides an estimate of the most probable outcome of sampling f at x, while also quantifying the uncertainty of this prediction. Applying this method throughout the domain of f provides a distribution of function approximations as shown in Figure 2.1.

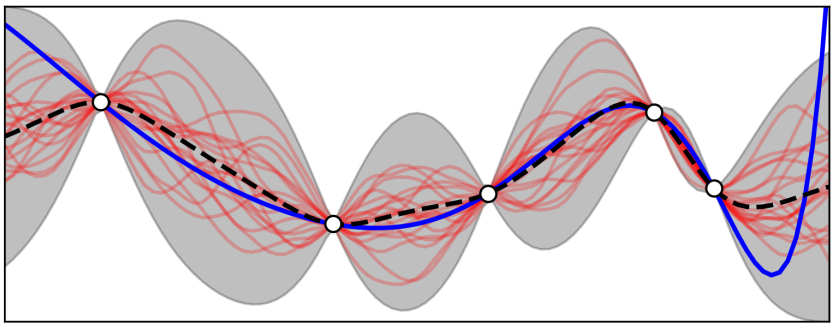

Figure 2.1: Distribution of f (blue line) calculated from 5 data points with the most likely outcome predicted shown in black and uncertainty estimates represented by the surrounding grey envelope; samples drawn from the distribution are shown in red.

A powerful feature of Bayes' Theorem is that it allows for recursive refinement of the posterior as more information becomes available. This is done by updating $p(\hat{f})$ to $p(\hat{f}|\mathcal{D})$ and using this new prior and an updated likelihood function to construct a new posterior distribution. As a result, the estimates of f can be continuously improved with additional data. By combining this capability with a decision-making mechanism that selects a new sampling location based on the current prediction and uncertainty values, it is possible to iteratively generate a sequence of datapoints that are used to refine the posterior distribution, which is then used to choose a new sample point. This process is known as Bayesian optimzation [50, 28].

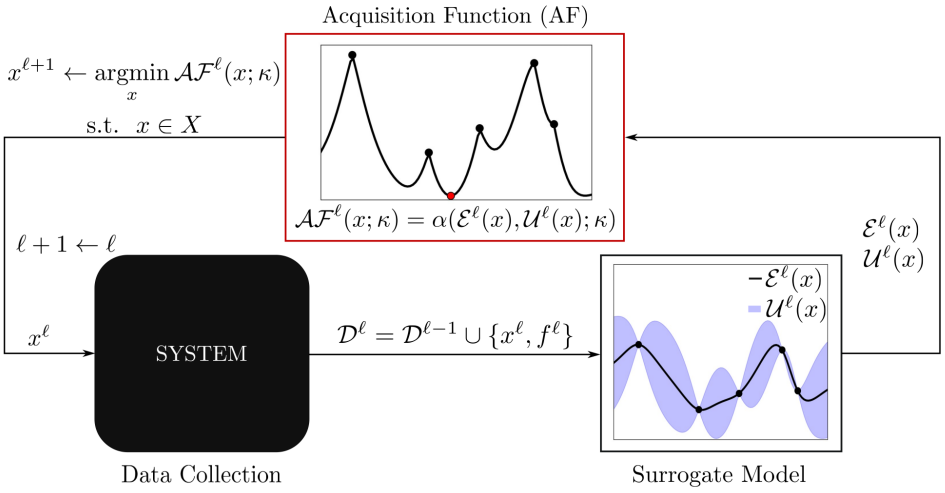

Figure 2.2: Workflow of the S-BO framework. Using dataset $\mathcal{D}^{\ell}$, S-BO builds a surrogate model that estimates f. The performance and uncertainty estimates calculated by the model are passed into an acquisition function that is optimized to suggest a new sampling point $x^{\ell+1}$. The system is sampled at this point and the collected data is appended to the dataset to retrain the model.

The standard Bayesian optimization algorithm (S-BO) is initialized using a set of input/output observations of size ℓ, $\mathcal{D}^{\ell} = \{x_{\mathcal{K}}, f_{\mathcal{K}}\}$, where $\mathcal{K} = \{1, ..., \ell\}$. The posterior distribution of f is constructed using a stochastic surrogate model, M_f^{ℓ}, usually a Gaussian process, that is conditioned on the data. The model prediction/performance and uncertainty estimates (e.g., the mean and variance of the distribution respectively) are then passed into what is known as an acquisition function (AF), which calculates the util-

ity of some point of interest based on its performance (exploitation) and informational value (exploration). AFs can generally be expressed as functions of the form:

$$\mathcal{AF}^{\ell}(x;\kappa) = \alpha(\mathcal{E}^{\ell}(x), \mathcal{U}^{\ell}(x);\kappa) \tag{2.2}$$

where $\mathcal{E}^{\ell}(x)$ and $\mathcal{U}^{\ell}(x)$ are the current performance and uncertainty estimates respectively; $\kappa \in \mathbb{R}_{+}$ is a hyperparameter, commonly referred to as the exploration weight, that determines the importance placed on the model uncertainty. Larger values of κ will make the algorithm more explorative while smaller values result in more exploitative behavior. Specific examples of α are provided in 2.3. The next sample point, $x^{\ell+1}$, is determined by solving the AF optimization problem:

$$x^{\ell+1} = \underset{x}{\operatorname{argmin}}\, \mathcal{AF}^{\ell}(x;\kappa) \tag{2.3a}$$

$$\text{s.t.} \quad x \in X \tag{2.3b}$$

After, taking a sample at $x^{\ell+1}$, the dataset is updated and the model is retrained. This process is repeated until a satisfactory solution is found or the data collection budget is exhausted. The pseudocode for S-BO is presented in Algorithm 1 and Figure 2.2 provides an illustrative summary of this workflow.

Algorithm 1: Standard Bayesian Optimization (S-BO)

Given κ, L, and $\mathcal{D}^{\ell}$;
Train M_f^{ℓ} using initial dataset $\mathcal{D}^{\ell}$ and obtain $\mathcal{AF}^{\ell}$;
for $\ell = 1, 2, ..., L$ **do**
  Compute $x^{\ell+1} \leftarrow \operatorname{argmin}_x \mathcal{AF}^{\ell}(x;\kappa)$ s.t. $x \in X$;
  Sample system at $x^{\ell+1}$ to obtain $f^{\ell+1}$;
  Update dataset $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^{\ell} \cup \{x^{\ell+1}, f^{\ell+1}\}$;
  Train model using $\mathcal{D}^{\ell+1}$ to obtain $M_f^{\ell+1}$ and $\mathcal{AF}^{\ell+1}$;
end

2.2 The Gaussian Process

The Gaussian Process (GP) is a stochastic regression model that can be thought of as an extension of the multi-variate normal distribution to the function space [51]. GPs are non-parametric and are instead fully specified by a pair of mean and covariance (kernel) functions:

$$\mathcal{GP}_f(x) \sim \mathcal{N}(m_f(x), k_f(x, x')) \tag{2.4}$$

Here, the mean function, $m_f(x)$, estimates the expected value of f at a point of interest x. The kernel function, $k_f(x, x')$, determines the similarity between any two points in the domain of f and provides a measure of how the value of $f(x)$ correlates with the value of $f(x')$. There exists a wide variety of kernel functions, and selection is often motivated by identifying a kernel that exhibits properties (e.g., smoothness, periodicity, continuity, etc.) that can accurately approximate f. The Mátern kernel [52] has emerged as a common default option due to its ability to model functions of any degree of smoothness (see Figure 2.3). It is the kernel of choice throughout this work and is defined as:

$$k_f\left(x, x'\right) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\sqrt{2\nu}d\left(x, x'\right)\right)^{\nu} K_\nu\left(\sqrt{2\nu}d\left(x, x'\right)\right) \tag{2.5}$$

where Γ is the gamma function and K_ν is a modified Bessel function. The smoothness of the function is controlled by ν, which is usually set to either 1.5 (function is once-differentiable), 2.5 (function is twice-differentiable) or ∞ (function is infinitely-differentiable); note that in the latter case, the kernel may be referred to as the radial basis function (RBF) kernel. The function $d(\cdot,\cdot) = \sqrt{(x - x')^T \Theta^{-2} (x - x')}$ is a scaled Euclidean distance function. Here $\Theta \in \mathbb{R}^{d_x \times d_x}$ is a diagonal matrix whose entries are the kernel length scales, $\theta_1, ..., \theta_{d_x}$. The length scales are also referred to as the kernel hyperparemeters and determine the impact of each dimension of x on the model predictions.

In GP regression, a prior distribution of f of the form (2.4) is constructed by setting $m_f(x) = \mathbf{0}$ and initializing θ based on domain knowledge or hueristics. Using the obser-

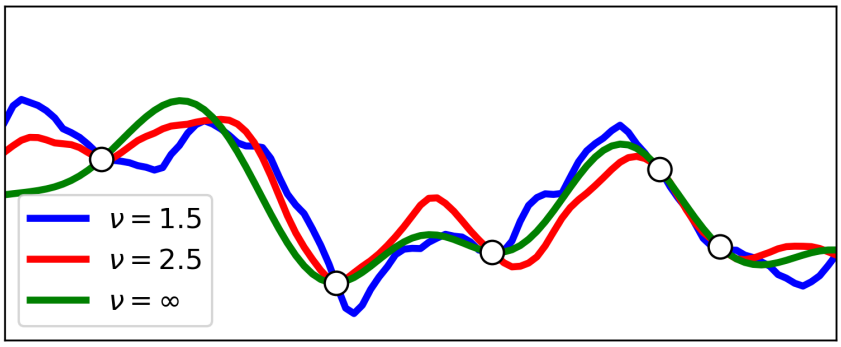

Figure 2.3: Samples drawn from GPs trained on five datapoints using the Mátern kernel at varying smoothness values

vations in $\mathcal{D}^\ell$, the kernel hyperparameters are then updated by solving the log marginal likelihood problem:

$$\theta^* = \underset{\theta}{\text{argmax}} -\frac{1}{2} f_{\mathcal{K}}^T \mathbf{K}(x_{\mathcal{K}}, x_{\mathcal{K}})^{-1} f_{\mathcal{K}} - \frac{1}{2} \log |\mathbf{K}(x_{\mathcal{K}}, x_{\mathcal{K}})| - \frac{\ell}{2} \log(2\pi) \tag{2.6}$$

where $\theta = [\theta_1, ..., \theta_{d_x}]^T$. The output data are assumed to follow a joint multivariate normal distribution of the form $f(x_{\mathcal{K}}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(x_{\mathcal{K}}, x_{\mathcal{K}}))$, where $\mathbf{0} \in \mathbb{R}^\ell$ and $\mathbf{K}(x_{\mathcal{K}}, x_{\mathcal{K}}) \in \mathbb{R}^{\ell \times \ell}$ is calculated such that $\mathbf{K}_{ij} = k_f(x_i, x_j)$. By conditioning the prior on the observed data, the posterior distribution, $\mathcal{GP}_f^\ell$ (which is also Gaussian), at a set of new points $\mathcal{X}$ can be determined:

$$\begin{bmatrix} f(x_{\mathcal{K}}) \\ f(\mathcal{X}) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(x_{\mathcal{K}}, x_{\mathcal{K}}) & \mathbf{K}(x_{\mathcal{K}}, \mathcal{X}) \\ \mathbf{K}(\mathcal{X}, x_{\mathcal{K}}) & \mathbf{K}(\mathcal{X}, \mathcal{X}) \end{bmatrix}\right) \tag{2.7}$$

From this result, the posterior mean and covariance can be calculated:

$$m_f^\ell(\mathcal{X}) = \mathbf{K}(\mathcal{X}, x_{\mathcal{K}})^T \mathbf{K}(x_{\mathcal{K}}, x_{\mathcal{K}})^{-1} f_{\mathcal{K}} \tag{2.8a}$$

$$\Sigma_f^\ell(\mathcal{X}) = \mathbf{K}(\mathcal{X}, \mathcal{X}) - \mathbf{K}(\mathcal{X}, x_{\mathcal{K}})^T \mathbf{K}(x_{\mathcal{K}}, x_{\mathcal{K}})^{-1} \mathbf{K}(x_{\mathcal{K}}, \mathcal{X}) \tag{2.8b}$$

The formulation presented above assumes that the observations in $\mathcal{D}^\ell$ are noise-free (i.e., the system is perfectly observable). In practice, however, data is often corrupted by

noise, which can be modeled as $f(x) = z(x) + \epsilon$, where $z(x)$ is the true observation and ϵ is a random error. If ϵ follows a normal distribution, $\mathcal{N} \sim (0, \sigma_\epsilon^2)$, (2.7) can be modified to account for the noise in the data:

$$\begin{bmatrix} f(x_\mathcal{K}) \\ f(\mathcal{X}) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(x_\mathcal{K}, x_\mathcal{K}) + \sigma_\epsilon^2 \mathbf{I} & \mathbf{K}(x_\mathcal{K}, \mathcal{X}) \\ \mathbf{K}(\mathcal{X}, x_\mathcal{K}) & \mathbf{K}(\mathcal{X}, \mathcal{X}) \end{bmatrix}\right) \tag{2.9}$$

The calculations of the posterior mean and covariance are then updated as follows:

$$m_f^\ell(\mathcal{X}) = \mathbf{K}(\mathcal{X}, x_\mathcal{K})^T \left[\mathbf{K}(x_\mathcal{K}, x_\mathcal{K}) + \sigma_\epsilon^2 \mathbf{I}\right]^{-1} f_\mathcal{K} \tag{2.10a}$$

$$\Sigma_f^\ell(\mathcal{X}) = \mathbf{K}(\mathcal{X}, \mathcal{X}) - \mathbf{K}(\mathcal{X}, x_\mathcal{K})^T \left[\mathbf{K}(x_\mathcal{K}, x_\mathcal{K}) + \sigma_\epsilon^2 \mathbf{I}\right]^{-1} \mathbf{K}(x_\mathcal{K}, \mathcal{X}) \tag{2.10b}$$

Due to their non-parametric nature, GPs provide an incredibly flexible framework that can be used to approximate a wide variety of functions. They also provide a straightforward and tractable approach to estimating model uncertainty, a feature which is of great use in the context of BO. Due to the small number of hyperparameters that have to be tuned compared to larger parametric models like neural networks, GPs are able to obtain very accurate estimates of the underlying performance function using relatively few datapoints. As a result, Gaussian processes have become the model of choice in applications involving complex physical systems or rigorous simulations where the amount of data that can be generated is limited [50].

Despite their utility, GPs have several limitations. For example, the flexibility in modeling is accompanied by the need to ensure that the selected kernel is appropriate, which often requires domain knowledge or the use of advanced model selection techniques [28]. Additionally, by assuming that f is normally distributed, a GP surrogate also assumes that f is symmetric. In reality, the range of f can be bounded, especially when dealing with physical quantities, making this assumption inaccurate. This can be especially problematic when dealing with points at are at or near the feasibility limit, as it can result in the GP model making infeasible predictions.

In terms of computational efficiency, the calculations required to train and sample from GPs are rather intensive. The inversion of the $\mathbf{K}$ seen in (2.6)(2.8), and (2.10) scales on the order of $\mathcal{O}(\ell^3)$ and has a memory cost of $\mathcal{O}(\ell^2)$ [53]. This effectively limits the size of datasets that can be used with GP models and can also make repetitive sampling computationally expensive. GPs are also known to scale poorly at high dimensions of x due to the increased distance between points sample points, which make it difficult to determine any meaningful correlations [54]. As a result, most of the applications that utilize Gaussian process surrogates have generally been limited to 20 or fewer dimensions.

Due to these challenges, GPs remain an active field of research and recent advancements have produced several novel tools aimed at addressing these issues such as sparse GPs, variational inference, warped GPs, and automatic relevance determination [55, 56, 57, 58]. Many of these methods, however, have their own set of challenges, such as reduced approximation accuracy, complex implementation, and limited availability of software libraries when compared to the support available for standard GPs. While not the central focus of this work, potential solutions for addressing some of the limitations of standard GPs are presented as they become relevant in subsequent chapters.

2.3 Acquisition Functions

Acquisition functions serve as the decision-making mechanism of the BO algorithm. They determine the value of sampling at a particular point based on the performance (exploitation) and uncertainty (exploration) estimates calculated by the surrogate model (the mean and variance when using a GP). This consideration of exploration as well as exploitation is a key feature of BO. It is what allows the AF to direct the collection of samples across various regions of the design space, making the algorithm more adept at locating the global solution. As previously mentioned, these two elements are balanced via the exploration weight, κ, which determines the emphasis placed on the model uncertainty. Lower values of κ result in repeated exploitation of areas that have already been identified as

optimal, while larger values drive the algorithm to explore new regions that exhibit high model uncertainty. Selection of a new sample point is done by optimizing the acquisition function, as shown in (2.3), with the use of gradient-based methods. Due to their central role in establishing the sampling strategy of the algorithm, efforts aimed at adding functionalities to the BO algorithm or adapting it to a particular application have largely centered around modifying existing AFs or crafting new ones [59, 60, 61]. Many of these functions are either derived from or modeled after three foundational acquisition functions: the probability of improvement (PI), the expected improvement (EI), and the lower confidence bound (LCB) functions.

Probability of improvement

The probability of improvement function was one of the first AFs developed for BO, it was introduced alongside Algorithm 1 by Harold Kushner [25]. PI simply measures the probability that sampling from some point x results in an improvement from the current best observed value, f^*. Mathematically, this AF can be expressed as

$$\mathcal{AF}^{\ell}_{\mathrm{PI}}(x;\kappa) = -\mathbb{P}(f(x) \leq f^*) \tag{2.11}$$

Recalling that $f(x)$ is distributed according to $f(x) \sim \mathcal{N}(m^{\ell}_f(x), (\sigma^{\ell}_f(x))^2)$ when a GP surrogate model is used, the following substitution can be made:

$$z(x) = \frac{f(x) - m^{\ell}_f(x)}{\sigma^{\ell}_f(x)} \tag{2.12}$$

where $z(x) \sim \mathcal{N}(0,1)$. Applying this substitution to (2.11), results in:

$$\mathcal{AF}^{\ell}_{\mathrm{PI}}(x;\kappa) = -\mathbb{P}\left(z(x) \leq \frac{f^* - m^{\ell}_f(x)}{\sigma^{\ell}_f(x)}\right) \tag{2.13}$$

Because z a normal random variable, (2.13) can be calculated using the normal cumulative distribution function, Φ, resulting in the following definition for the PI AF:

$$\mathcal{AF}^{\ell}_{\mathrm{PI}}(x;\kappa) = -\Phi\left(\frac{f^* - m^{\ell}_f(x) - \kappa}{\sigma^{\ell}_f(x)}\right) \tag{2.14}$$

where κ is included to promote exploration by essentially increasing the threshold for what is considered an improvement. This pushes the AF to prioritize points with larger improvements, even when they have higher uncertainty. Note that the negative sign in front of the expression is included to ensure consistency as the AF is *minimized* to select a new sample point.

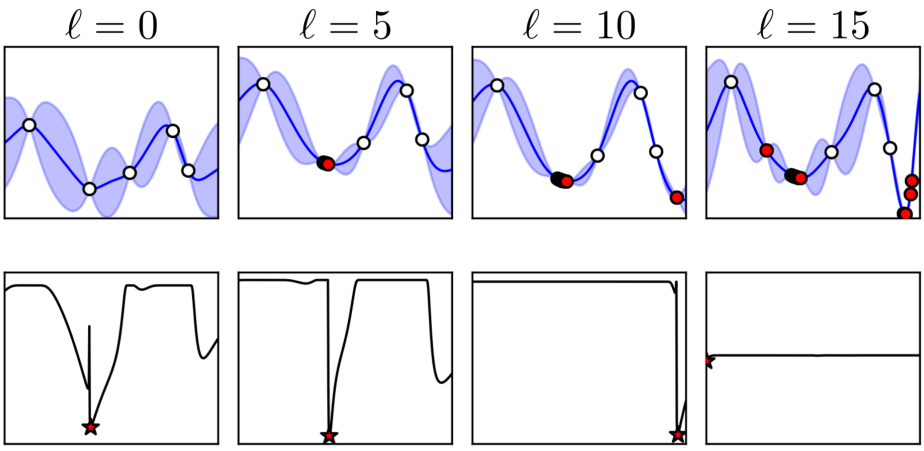

Figure 2.4: Snapshots of the current surrogate model (top) and the PI AF (bottom) at $\ell = 0$, 5, 10, and 15 with $\kappa = 0.01$. The mean is shown in blue and the uncertainty estimates are represented by the surrounding light blue envelope; seed points are shown in white and the optimum of the PI AF is marked by the red star. Note that the algorithm spends nearly 10 iterations exploring the region near the best seed point

The PI acquisition function is an intuitive method for selecting new sample points: it assigns more value to points that have a high probability of improving the current result. When a GP surrogate is used, it is easy to implement and computationally inexpensive. Additionally, the function is smooth and its analytical gradient can be computed, which can be leveraged to facilitate the calculation of (2.3). However, this AF also exhibits

various significant drawbacks that have limited its use. The function can be multi-modal (see Figure 2.4), and can be quite flat in certain regions of the design space. This makes the function especially sensitive to the initialization point, and multi-start strategies are required to ensure that the global solution is found. Numerical instability can become an issue when the value of $\sigma_f^\ell(x) << 1$, and it might be necessary to add a small constant to the denominator to ensure that the solver is able to converge.

The inability to consider the magnitude of a potential improvement makes the PI AF prone to sampling from points with a high probability of a low improvement as opposed to those with lower probability of a high improvement. This causes the algorithm to heavily over-exploit the area near f^* (see Figure 2.4), significantly increasing the probability that regions where a better solution could potentially exist are left unexplored [62]. While κ is intended to ameliorate this issue, due to the formulation of the AF, it can be difficult to tune and the function is quite sensitive to the selected value. As a result, it can be quite easy to make the algorithm overly-exploitative or overly-explorative.

Expected improvement

The expected improvement function was first introduced in [26] as a means for considering the amount of potential improvement when selecting a new point. The work of Jones [63] and Schonlau [41] popularized the use of the EI, and it is currently the most widely studied AF and the one most closely associated with BO. As its name suggests, the function measures the expectation of improvement where improvement is defined as:

$$I(x) = \min\{f(x) - f^*, 0\} \tag{2.15}$$

The EI AF is then:

$$\mathcal{AF}_{\text{EI}}^\ell(x;\kappa) = \mathbb{E}[I(x)] \tag{2.16}$$

When a GP surrogate model is used, f can be expressed in terms of a standard normal variable, $z(x) \sim \mathcal{N}(0,1)$:

$$f(x) = m_f^\ell(x) + \sigma_f^\ell(x) \cdot z \tag{2.17}$$

Applying this substitution and using the fact that (2.15) is nonzero only when $z < \frac{f^* - m_f^\ell(x)}{\sigma_f^\ell(x)}$, (2.16) can be expressed as:

$$\mathcal{AF}_{\text{EI}}^\ell(x;\kappa) = \int_{-\infty}^{z_0} (m_f^\ell(x) + \sigma_f^\ell(x) \cdot z - f^*)\phi(z)dz \tag{2.18}$$

where ϕ is the normal probability density function and $z_0 = \frac{f^* - m_f^\ell(x)}{\sigma_f^\ell(x)}$. This integral can be rearranged as follows:

$$\mathcal{AF}_{\text{EI}}^\ell(x;\kappa) = (m_f^\ell(x) - f^*) \int_{-\infty}^{z_0} \phi(z)dz + \sigma_f^\ell(x) \int_{-\infty}^{z_0} z\phi(z)dz \tag{2.19}$$

Using the properties of the normal distribution, this can be evaluated as:

$$\mathcal{AF}_{\text{EI}}^\ell(x;\kappa) = -(f^* - m_f^\ell(x) - \kappa)\Phi\left(\frac{f^* - m_f^\ell(x) - \kappa}{\sigma_f^\ell(x)}\right) - \sigma_f^\ell(x)\phi\left(\frac{f^* - m_f^\ell(x) - \kappa}{\sigma_f^\ell(x)}\right) \tag{2.20}$$

where, as with the PI AF, κ is included to promote exploration by increasing the threshold for what is considered an improvement. Note that in the limiting case when $\sigma_f^\ell(x) \to 0$ (i.e., at a sampled point), $\mathcal{AF}_{\text{EI}}^\ell(x;\kappa) \to 0$ as Φ tends to 0 as its argument approaches $-\infty$ and ϕ will similarly approach 0. This is the expected behavior as there is not an obtainable improvement from re-sampling at an already explored point.

As with the PI AF, the expected improvement is also smooth and has a gradient that can be determined analytically (when a GP surrogate is used). These are properties which, again, can be exploited during the optimization of the AF. Unlike the PI, however, the EI function explicitly rewards sampling at locations where the uncertainty is high. This results in a significantly more balanced sampling pattern where high uncertainty regions are more likely to be explored, especially if the potential improvement is high. As

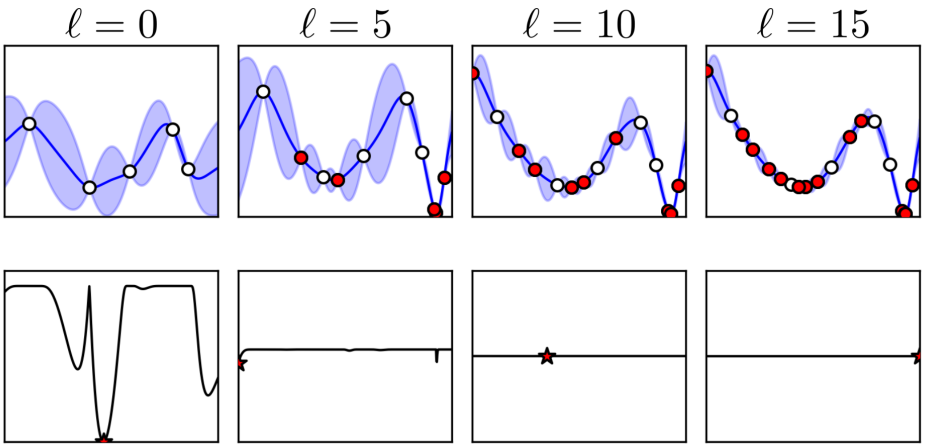

Figure 2.5: Snapshots of the current surrogate model (top) and the EI AF (bottom) at $\ell = 0$, 5, 10, and 15 with $\kappa = 0.01$. The mean is shown in blue and the uncertainty estimates are represented by the surrounding light blue envelope; seed points are shown in white and the optimum of the EI AF is marked by the red star. Note that range of the EI function quickly decreases after the two optima have been located, resulting in more exploratory sampling at later iterations

a result, this function is generally able to outperform the PI AF as shown in Figure 2.5. Use of the expected improvement is also arguably more intuitive than simply relying on the probability of improvement. Repeatedly sampling in the same region even if there is a high chance of improvement can quickly lead to diminishing returns. Moving, instead, to sample from an area with potentially high reward, even if the risk (uncertainty) is also high, can result in either a potentially substantial improvement or provide strong confirmation that the current best observed point is the optimal solution. This is arguably a better use of the sample budget.

While the EI AF is able to outperform the PI AF, is also shares various structural features that can result in it encountering similar issues. For example, the function is also highly multi-modal and can be quite flat along large portions of the design space, meaning that it is also quite sensitive to the initialization point. Additionally, numerical stability can again become an issue in the cases where $\sigma_f^\ell(x)$ is very small. The EI AF exhibits a similar sensitivity to the value of κ as the PI, meaning that it can likewise be challenging to tune this parameter. The range of the expected improvement function also

decreases as the algorithm progresses, albeit at a much faster rate compared to the probability of improvement AF. This can cause the EI AF to adopt an overly-explorative strategy as the values of potential improvements drop and the uncertainty starts to become the primary driver for sample point selection. Figure 2.5 provides a clear illustration of this issue: the majority of observations away from the global solution are collected *after* it has already been identified. This is done despite the fact that, as seen in the panel for $\ell = 5$ of the same figure, there does not appear to be a significant probability that sampling at these points will result in an improvement from the current f^*. However, it can clearly be observed that the selected points are those where the uncertainty is the highest. In addition to resulting in a rather inefficient use of samples, this behavior is also counter-intuitive, as one would expect the bulk of the exploration-oriented sampling to be done during the initial iterates to build the surrogate model.

Lower confidence bound

The lower confidence bound function was also introduced by Kushner alongside the PI AF [25]. Unlike the EI and PI functions, the LCB AF deals with the performance and uncertainty estimates directly and can generally be expressed as:

$$\mathcal{AF}^{\ell}_{\text{LCB}}(x;\kappa) = \mathcal{E}^{\ell}(x) - \kappa\mathcal{U}^{\ell}(x) \tag{2.21}$$

In the case where a GP surrogate is used then $\mathcal{E}^{\ell}(x) = m^{\ell}_f(x)$ and $\mathcal{U}^{\ell}(x) = \sigma^{\ell}_f(x)$. This allows for (2.21) to be reformulated as:

$$\mathcal{AF}^{\ell}_{\text{LCB}}(x;\kappa) = m^{\ell}_f(x) - \kappa\sigma^{\ell}_f(x) \tag{2.22}$$

which is commonly referred to as the GP-LCB [64]. This function can be interpreted to represent a quantile, specified by κ, of the estimated normal distribution of $f(x)$. For example, when $\kappa = 1.96$, the value of $\mathcal{AF}^{\ell}_{\text{LCB}}(x;\kappa)$ is value that is estimated to be better than all but 2.5% of the potential outcomes of $f(x)$, in other words $\Pr(f(x) < \mathcal{AF}^{\ell}_{\text{LCB}}(x;\kappa =$

$1.96)) = 0.025$. This allows for the selection of the exploration weight to be significantly more grounded in theory and statistical principles when compared to the PI and EI functions, which often rely on heuristics or empirical methods for tuning κ. Additionally, the more direct impact of the exploration weight on $\mathcal{AF}^{\ell}_{\text{LCB}}(x;\kappa)$ facilitates dynamic tuning where κ is adjusted as the optimization routine progresses [64]. This is due to the fact gradual changes in this parameter will not drastically affect the behavior of the acquisition function, which is not always the case when adjusting κ in the EI and PI AFs.

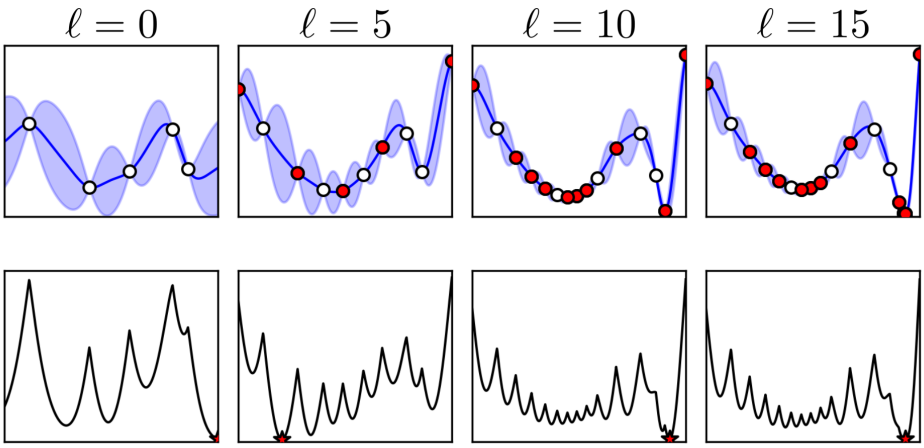

Figure 2.6: Snapshots of the current surrogate model (top) and the LCB AF (bottom) at $\ell = 0$, 5, 10, and 15 with $\kappa = 4.5$. The mean is shown in blue and the uncertainty estimates are represented by the surrounding light blue envelope; seed points are shown in white and the optimum of the LCB AF is marked by the red star. Note that, while the function is multi-modal, it lacks the flat regions observed in the PI and EI AFs and begins to follow the trend of $m^{\ell}_{f}(x)$ toward the final samples

In addition to its straightforward form and intuitive interpretation, the LCB AF offers various notable advantages over the probability of improvement and expected improvement acquisition functions. While this work focuses on the use of GP models, (2.21) can easily extend to various different classes of surrogates and make use of different performance and uncertainty metrics, whereas closed-form expressions for EI and PI functions can be difficult to derive for non-GP surrogates. As a result, the function is more flexible and it is easier to modify. Additionally, while the function is also multi-modal, it lacks the flat regions observed in the EI and PI AFs. In fact, range of the LCB AF does

not diminish and instead gradually begins to approximate the mean of the generated surrogate model. This results in the function promoting exploration during the initial iterations before eventually settling into the region around the located solution as shown in Figure 2.6. Finally, existing works have established guidelines for setting and adjusting the value of κ that guarantee convergence to a global optimum given enough iterations when using the lower confidence bound function [64, 65]. While the properties of the EI AF have also been widely studied, similar theoretical convergence guarantees have not yet been established [66]. This set of benefits motivated our selection of the LCB function as the acquisition function of choice for our work. Note that references to the acquisition function and the use of $\mathcal{AF}^{\ell}(x;\kappa)$ in subsequent chapters refer to the LCB function unless stated otherwise.

2.4 Summary

In this chapter we provided a tutorial introduction to the Baysian optimization framework. We began by discussing how BO is formulated from Bayes' Theorem through sequential data-driven updates of a prior distribution and likelihood function. We provided a detailed overview of Guassian processes, the most commonly used BO surrogate, and explained how they are constructed from kernel functions and updated using current data. We highlighted some of the advantages of using GP models, namely their flexibility and accuracy, and discussed some of their limitations, including scalability, prediction feasibility, and model tuning. This was followed by the introduction of the acquisition function. Here, we presented the three major acquisition functions that are generally used in Bayesian optimization, including their formulation for the GP surrogate, and discussed their strengths and drawbacks. This chapter introduces the core elements of the work that is presented in subsequent chapters, facilitating its introduction and discussion.

Chapter 3

BAYESIAN OPTIMIZATION WITH REFERENCE MODELS

3.1 Introduction

Model predictive control (MPC) is widely used in industrial systems due to its ability to handle diverse types of constraints, multivariable models, and operational objectives. The performance of MPC depends strongly on the controller formulation. Examples of typical tuning parameters that influence performance include the prediction and control horizon, weights of individual states or cost objectives, input rate constraints, and constraint back-off terms [67, 68]. Complex and non-intuitive dependencies are typically observed between the tuning parameters of the MPC controller and its closed-loop performance. As such, conducting MPC tuning by trial-and-error or by heuristics might require a significant number of closed-loop simulations. This poses a problem because a single closed-loop simulation might require the solution of hundreds to thousands of optimization problems. For instance, one is often interested in evaluating the performance of MPC over an entire year of operation and/or over different operational scenarios.

Self-tuning methods cast the MPC tuning problem as an optimization problem in

which the tuning parameters are used to maximize the closed-loop performance. However, in general, neither an explicit model relating the effects of tuning parameters to the closed-loop performance nor derivative information are available [11]. Thus, derivative-free methods that treat the optimization problem as a black-box have been utilized to address the problem [69]. Simple algorithms studied include sampling- or direct search-based approaches [70]; other algorithms such as genetic algorithms and particle swarm optimization have also been previously proposed. Well-known issues encountered with these techniques include slow progress (requiring many simulations) and lack of convergence guarantees. Moreover, these approaches are sensitive to the initial guess. An excellent review of MPC tuning methods can be found in [71]. Recently, more efficient derivative-free algorithms have been used for tuning MPC controllers [72].

Bayesian optimization (BO) is a powerful technique for optimizing computationally-intensive black-box functions [69]. BO has been widely used for hyper-parameter tuning of deep learning models [73], for design of experiments [74], and for conducting reinforcement learning tasks [75]. BO can also be adapted to accommodate a mixture of continuous and discrete decision variables [69] and has been shown to be effective at reducing the number of objective function evaluations [73]. In BO, probabilistic surrogate models (e.g., kriging models) are built from the function evaluation data, and these models can approximate the behavior of the actual objective function [25]. Specifically, the surrogate model provides information over both the predicted function value (via the posterior mean) and the associated prediction uncertainty (via the posterior variance). Subsequent sampling points are selected by satisfying the exploration and exploitation trade-off [76]. Exploration aims to evaluate the objective at points in the decision space with the goal of improving the accuracy of the surrogate model of the objective, while exploitation aims to use the surrogate model to identify decisions that minimize (or maximize) the objective function.

Black-box optimization methods such as BO are built specifically to handle problems that lack a model. As such, they traditionally do not incorporate any preexisting sys-

tem information and rely solely on data. However, many systems do have information available (of different forms) that could potentially be useful for improving the optimization process. Recent work has shown that leveraging this prior knowledge can improve the performance of BO. The most commonly used approach for this integration has been multi-fidelity BO (MFBO). MFBO explores X using a low-fidelity representation of $f(x)$ to identify promising regions and then searches through this reduced space using a higher fidelity model. This process is repeated iteratively, allowing the algorithm to gradually zero in on an optimal region and reducing the number of experiments that have to be performed with the real system [77, 78, 79]. The end result is a more efficient use of the high-fidelity sample budget, as suboptimal points are unlikely to be selected at these levels. However, the implementation of this paradigm is quite complex, as it requires integrating the information provided by the surrogate models generated at each fidelity level. Obtaining the models at the various fidelities and ensuring that they provide the right balance of information and computational efficiency can also be difficult. Additionally, by restricting the search space, there is a risk that the region containing the global solution could be missed due to model error.

Reference (prior) models, low-fidelity representations able to capture general system trends, provide a more straightforward approach for incorporating available information. By using this initial approximation, an algorithm can identify potential areas of interest from the beginning, thereby reducing the number of iterations (and simulations) required for convergence. Additionally, most systems must obey specified constraints. Without a system model, it is unrealistic to know if these constraints will be satisfied prior to evaluating the function. However, a reference model can approximate regions where constraint violations might occur, keeping the algorithm from needlessly exploring these areas [80, 81, 82]. The use of reference models has been shown to improve the performance of optimization algorithms in various applications, including process simulation and design [83], oil-field operations [84], petroleum extraction [85], and aerospace design [86]. Recent studies also indicate that the incorporation of prior knowledge in the

form of a reference model can enhance the performance of machine learning (neural net) models. For example, neural nets have been trained to solve partial differential equations while being provided with prior information from a reference model on the physical laws that the solutions must obey [87]. Other examples include the design of experiments where prior expert knowledge is supplied [88], the integration of mechanistic models with BO-inspired methods for cell engineering [89], and the combination of simplified physics models with Bayesian analysis to correct systematic bias [90]. Note that, unlike in MFBO, when a reference model is used, it is not modified after it has been loaded into the algorithm. Additionally, the algorithm always samples from the real system and does not use the low-fidelity representation to restrict sampling to any one region of the design space.

In this chapter, we study the use of reference models in BO for tuning MPC controllers. Our work is motivated by a real MPC application to central HVAC plants. The operating cost of HVAC plants is strongly affected by disturbances that cannot be forecasted perfectly (demands of electrical power and hot and cold water). Errors in disturbance forecasts result in frequent constraint violations of storage levels (overfilling or drying-up) that ultimately translate into decreased economic performance. Adding back-off terms to the storage levels has been shown to provide an effective strategy to deal with these issues [1]. This approach resembles constraint back-off approaches recently explored in the MPC literature [91]. Unfortunately, tuning these back-off terms requires extensive simulations; every year-long closed-loop simulation requires solving over 8,700 optimization problems and is time-consuming (a single simulation requires 2 hours of wall-clock time). BO approaches have recently been used for tuning MPC [92, 93, 94] and other control architectures [95, 96] and for performing goal-oriented learning of dynamical systems [97, 98]. To fully utilize the prior knowledge (or data) of the system, we propose incorporating a reference model into the standard BO algorithm, so as to accelerate the optimization speed and reduce the computational complexity. A unique feature of our work is the way in which we construct the reference model. Specifically, we build such a

model by using data collected from low-fidelity, closed-loop MPC simulations (use a formulation with a short prediction horizon). Our results indicate that BO with a reference model can find optimal back-off terms in 3 iterations with a reference model (20 hours including the time of training a reference model). For comparison, standard BO requires 14 iterations (28 hours); thus, our approach reduces the search time by 28%.

Our work also seeks to provide insights into why the use of reference models aids the BO search. We show that, by using a reference model, the BO algorithm goal is shifted from learning the objective function to learning the residual/error model between the reference model and the objective function (which is a much easier task). We also show that the use of a reference model focuses the BO search to small regions of the parameter space. Our results highlight that there exist multiple ways in which reference models can be incorporated. Most studies tend to incorporate reference models that arise from physics; here, we show that any type of approximate model can be used. In our context, using a low-fidelity MPC formulation provides a highly accurate reference model because this preserves the overall structure of the objective function (obtained with a high-fidelity MPC formulation). In other words, our approach can be seen as a coarsening strategy and this can be broadly applicable to other problem classes (e.g., by coarsening discretization meshes).

3.2 Bayesian Optimization with Reference Models

Incorporating a reference model to the BO algorithm provides an approach for introducing preexisting (prior) knowledge about the system into the search process. Specifically, the use of a reference model allows the algorithm to be initialized with an approximation of the objective. This has the effect of highlighting promising regions in which the solution might be located. BO can then focus sampling in these regions from the start and not spend unnecessary (expensive) simulations building up a surrogate model from scratch. Additionally, a reference model that is nonconvex can push the algorithm to explore var-

ious local minima, potentially improving the quality of the solution. The reference model can be obtained through various means; traditional approaches use physics-based models or simple empirical correlations. In the context of MPC, we show that one can construct a high-quality reference model by using a simple coarsening technique that simplifies the MPC formulation by reducing the prediction horizon. We will see that this provides a reference model that captures the overall structure of the objective function.

We have selected to incorporate the reference model by having the BO algorithm learn the model error instead of learning the objective function directly. To this end, (1.1) can be reformulated as:

$$\min_{x} \quad g(x) + \varepsilon(x) \tag{3.1a}$$

$$\text{s.t.} \quad x \in X \tag{3.1b}$$

where $g(\cdot)$ is the reference model and $\varepsilon(\cdot) = f(\cdot) - g(\cdot)$ is the residual or error model. The reference model is deterministic and less expensive to evaluate compared to the true objective function $f(\cdot)$. During the BO optimization routine, the reference model is fixed and unaffected by new data collected by the algorithm. The form of the residual model ε is not known (because f is not known), and thus a surrogate needs to be built from experimental data. Given a set of data $\mathcal{D}_\varepsilon^\ell = \{x_\mathcal{K}, \varepsilon_\mathcal{K}\}$, $\mathcal{K} = \{1, ..., \ell\}$, with $\varepsilon_\mathcal{K} = \{f(x_\mathcal{K}) - g(x_\mathcal{K})\}$, we construct a GP model for the residual, $\mathcal{GP}_\varepsilon^\ell$. The prediction of the residual at a new point x is then a Gaussian distribution with:

$$m_\varepsilon^\ell(x) = \mathbf{K}(x, x_\mathcal{K})^T \mathbf{K}(x_\mathcal{K}, x_\mathcal{K})^{-1} \varepsilon_\mathcal{K} \tag{3.2a}$$

$$\sigma_\varepsilon^\ell(x) = \mathbf{K}(x, x) - \mathbf{K}(x, x_\mathcal{K})^T \mathbf{K}(x_\mathcal{K}, x_\mathcal{K})^{-1} \mathbf{K}(x_\mathcal{K}, x) \tag{3.2b}$$

Recall that the GP prior is distributed according to $\mathcal{N}(\mathbf{0}, k_\varepsilon(x, x')$. If we assume that the reference model matches the objective in unsampled regions, then having an error model that is drawn from this distribution is significantly more reasonable than using it

to estimate the objective function.

Because $g(x)$ is assumed to be deterministic, it can be said to be drawn from $\mathcal{N}(g(x), \mathbf{0})$. Recall that $f(\cdot) = g(\cdot) + \varepsilon(\cdot)$ and thus, for some x, we can estimate the probability density of f as follows:

$$g(x) + \varepsilon(x) \sim \mathcal{N}(g(x), \mathbf{0}) + \mathcal{N}(m_\varepsilon^\ell(x), (\sigma_\varepsilon^\ell(x))^2) \tag{3.3a}$$

$$\mathcal{N}(g(x), \mathbf{0}) + \mathcal{N}(m_\varepsilon^\ell(x), (\sigma_\varepsilon^\ell(x))^2) = \mathcal{N}(g(x) + m_\varepsilon^\ell(x), (\sigma_\varepsilon^\ell(x))^2) \tag{3.3b}$$

$$f(x) \sim \mathcal{N}(g(x) + m_\varepsilon^\ell(x), (\sigma_\varepsilon^\ell(x))^2) \tag{3.3c}$$

Note that the LCB AF presented in equation (2.22) is expressed in terms of the objective function. Our previous derivations indicate that, when a reference model is used, we need to modify the acquisition function as:

$$\mathcal{AF}_\varepsilon^\ell(x; \kappa) = (g(x) + m_\varepsilon^\ell(x)) - \kappa\sigma_\varepsilon^\ell(x) \tag{3.4}$$

Algorithm 2 summarizes the framework for our proposed method. Overall, this is similar to the BO framework presented in Algorithm 1 with the following exceptions: every iteration requires the calculation of three quantities rather than one; the surrogate model is trained on the residuals rather than the performance function; and the modified AF in (3.4) is utilized for finding the subsequent sampling point.

Algorithm 2: Reference-Based Bayesian Optimization (Ref-BO)

Given reference model g, κ, L, and $\mathcal{D}_\varepsilon^\ell$;
Train GP $\mathcal{GP}_\varepsilon^\ell$ using $\mathcal{D}_\varepsilon^\ell$ and obtain $\mathcal{AF}_\varepsilon^\ell$;
for $\ell = 1, 2, ..., L$ **do**
  Compute $x^{\ell+1} \leftarrow \text{argmin}_x \, \mathcal{AF}_\varepsilon^\ell(x; \kappa)$ s.t. $x \in X$;
  Sample system at $x^{\ell+1}$ to obtain $f^{\ell+1}$ and residual $\varepsilon^{\ell+1}$;
  Update dataset $\mathcal{D}_\varepsilon^{\ell+1} \leftarrow \mathcal{D}_\varepsilon^\ell \cup \{x^{\ell+1}, \varepsilon^{\ell+1}\}$;
  Train GP using $\mathcal{D}_\varepsilon^{\ell+1}$ to obtain $\mathcal{GP}_\varepsilon^{\ell+1}$ and $\mathcal{AF}_\varepsilon^{\ell+1}$;
end

Remark 1. In general, the computational complexity of Algorithm 2 is less than Al-

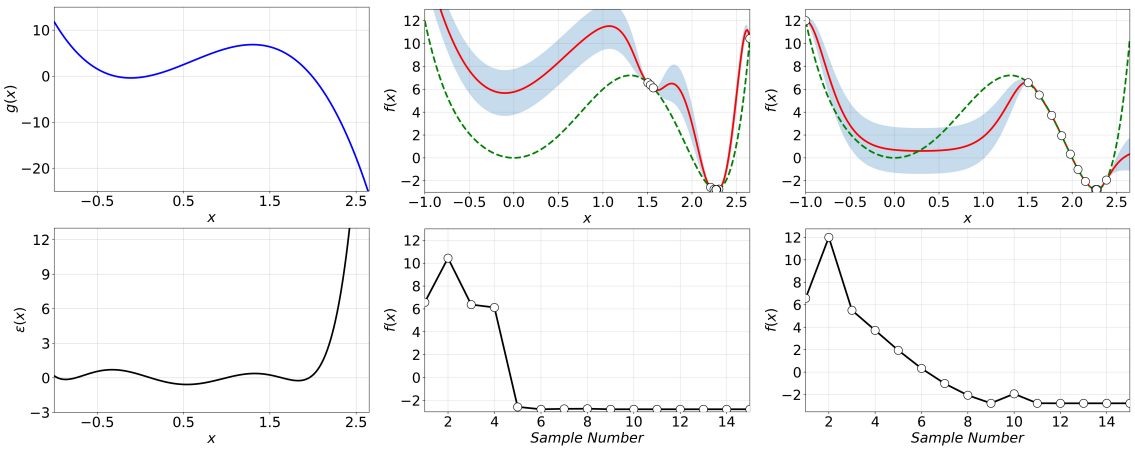

Figure 3.1: Left column: Reference model incorporated into the BO algorithm (top) and its residual function $\varepsilon(\cdot)$ (bottom); Middle column: Results of Bayesian optimization (top) with a reference model $g(\cdot)$ and evaluated function values (bottom) over 15 iteration showing the convergence. Right column: Results of traditional Bayesian optimization (top) and evaluated function values (bottom) over 15 iterations showing the convergence. Green dashed line: true objective function; Red line: posterior mean of GP model; Blue shaded area: posterior variance of GP model.

gorithm 1 for the following reasons. First, for Algorithm 1, S-BO requires quite a few exploration steps at the beginning to learn the general shape of the black-box function before it dives into exploiting local regions to find the optimum. In contrast, Algorithm 2, is able to leverage the information contained in $g(x)$ to identify and sample from potentially promising regions from the start, even as it develops the initial Gaussian process model for the residual during the first few iterations. In this way, the presence of the reference model reduces the computational time of BO by cutting back the required number of iterations. Second, the computational complexity for establishing the reference model is not a bottleneck. If a rough physics model is available, we can directly use it as the reference model. Otherwise, we can simply perform low-fidelity simulations without much computational cost to construct a coarse model to serve as $g(x)$. Thus, the additional effort required to obtain the reference model is not significant enough to negate the benefits it provides to the BO algorithm. As a result, the total computational time of our method shall be less than that of the standard BO approach.

Remark 2. The differences between our developed Algorithm 2 and standard GP-based BO are the following. The GP model is with respect to the residual function, $\varepsilon(\cdot)$, instead of the original function. However, the involved procedures for deriving the posterior distribution of ε follows those of standard GP modeling. Moreover, for our approach, constructing the AF is not based on the posterior distribution of the GP model of the error function, ε. Instead, it is based on the posterior distribution of the original function, f, as shown in (3.3). In addition, as the reference model is obtained explicitly in advance, the evaluations of the acquisition function at x can be easily acquired. Thus, traditional nonlinear optimization techniques used for solving (2.3) in S-BO, such as L-BFGS and DIRECT, can be implemented as well for optimizing (3.4).

Figure 3.1 demonstrates the benefits of applying BO with reference models using a simple example. The objective function and the reference model are given by:

$$f(x) = x^6 - 3x^5 + 8x^2 \tag{3.5a}$$

$$g(x) = -4.94x^3 + 8.90x^2 + 1.93x - 0.09 \tag{3.5b}$$

where $g(\cdot)$ was obtained by performing a linear regression using 5 random sample points, with the lowest order polynomial that yields an acceptable fit. Algorithms 1 and 2 were then implemented in Python 3.7 using Scikit-learn's `gaussian_process` module to construct the surrogate models and the AF was minimized using the L-BFGS-B method; both algorithms were initialized with the same random seed. Figure 3.1 illustrates that after 5 iterations, Algorithm 2 identifies and converges to the solution. Algorithm 1 approaches the solution slowly, sampling extensively along the way and taking 11 iterations to converge. Note also that the shape of $\varepsilon(\cdot)$ is much simpler than the shape of the objective (shown in the green dashed line). As a result, it is easier to learn the residual function $\varepsilon(\cdot)$ than the objective function $f(\cdot)$. This implies that Algorithm 2 is able to learn the residual model using fewer samples than Algorithm 1 requires for learning the objective.

3.3 Case Study: MPC Tuning for HVAC Plants

Thermal energy storage (TES) for chilled/hot water is used to shift energy loads of an HVAC plant to off-peak hours in order to reduce electricity costs and to mitigate peak demands [99]. Energy demands and prices are difficult to forecast and errors often result in violations of TES capacity limits (overfilling or drying up of water tanks). A strategy to mitigate these violations consists of using a reserved buffer (by adding a back-off term to the storage constraints). Typically, these back-off terms are selected by manual search, which requires repeated simulations of the closed-loop system. This approach is time-consuming as it involves year-long simulations. In this case study, we leverage the MPC formulation proposed in [1] and build a BO framework for tuning TES back-off terms. A reference model is introduced to BO to facilitate the optimization. In the HVAC plant, a chiller subplant produces chilled water and a heat recovery (HR) chiller subplant produces both chilled and hot water; a hot water generator produces hot water; cooling towers are used to decrease the temperature of water purchased from the market; a dump heat exchanger (dump HX) rejects heat from the hot water; and storage tanks (one for chilled water and one for hot water) are used as the TES. The MPC controller seeks to determine hourly operating loads for each unit in such a way that the HVAC plant satisfies the demands of chilled and hot water from multiple buildings of a university campus. The objective of the MPC is to minimize the total cost of the utilities (electricity, water, and natural gas) purchased from the market. Electricity is charged based on time-varying prices, while water and natural gas usage are charged at constant prices.

The HVAC plant cost includes the following items: (i) electricity required for the equipment operation and charged based on hourly time-varying prices, π_t^e, (ii) water required to make up for evaporative losses of water in the cooling towers and purchased at a fixed price π_t^w = \$0.009/gal, (iii) natural gas required for the operation of the hot water generator to satisfy the campus heating load and purchased at a fixed price of π_t^{ng} = \$0.018/kWh, and (iv) the peak electrical demand charges for each month charged at a

high rate of π^D = \$4.5/kW.

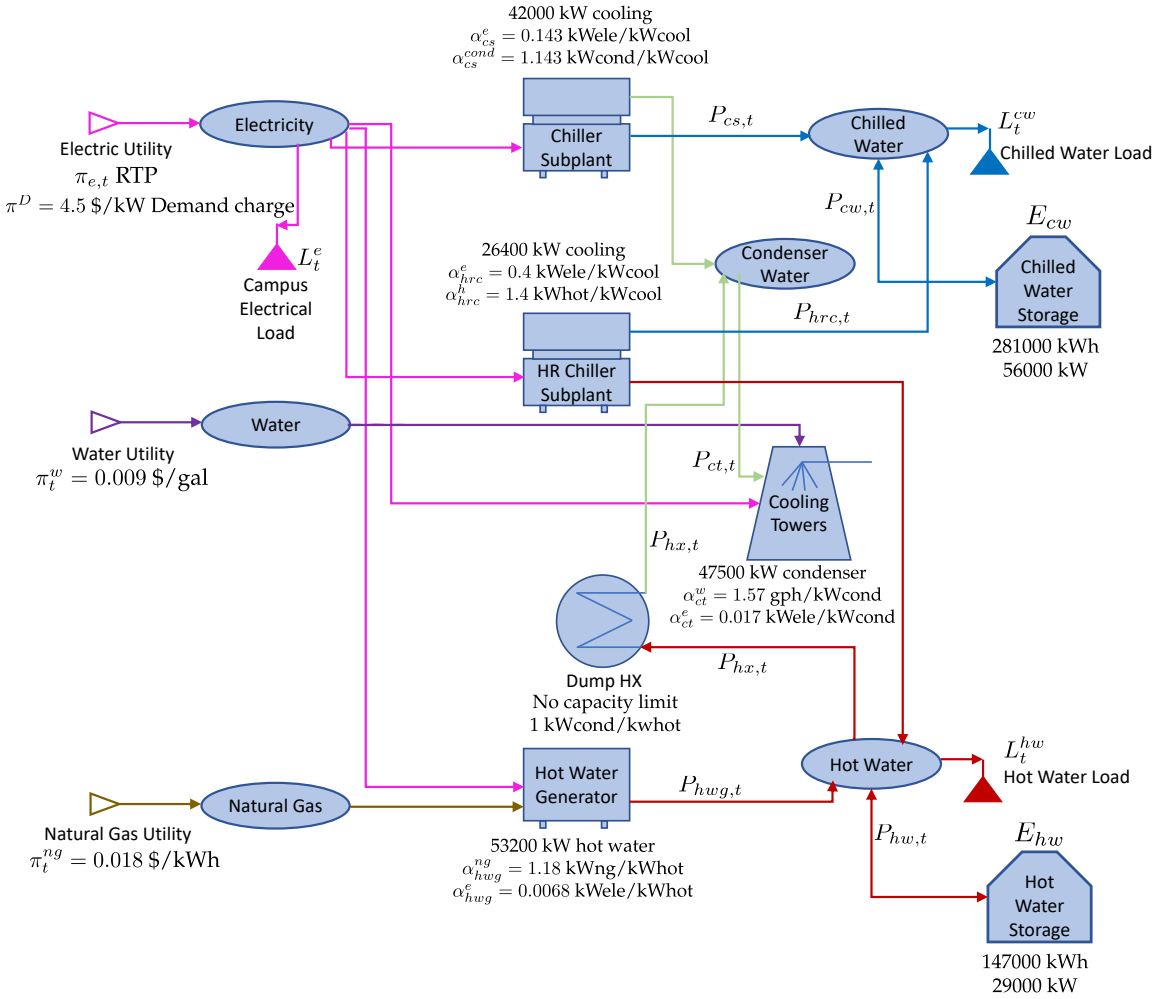

Figure 3.2: Schematic energy flow diagram of central HVAC plant (reproduced with permission) [1].

Figure 3.2 shows the energy flows between all units of the HVAC plant and interactions with loads and utilities. As illustrated in Figure 3.2, the amounts of electricity, water, and natural gas consumed by the units depend on their operating loads. The chiller and HR chiller subplants use α^e_{cs} and α^e_{hrc} kW of electricity for the production of 1 kW of chilled water, respectively; the hot water generator requires α^e_{hwg} kW of electricity and α^{ng}_{hwg} kW of natural gas for the production of 1 kW of hot water; and the cooling towers require α^e_{ct} kW of electricity and α^w_{ct} utility water for 1 kW of condenser water input. For

the chilled water load of the campus (L_t^{cw}), chilled water is produced by the chiller ($P_{cs,t}$), the HR chiller subplants ($P_{hrc,t}$), and the discharge from chilled water storage ($P_{cw,t}$). For the hot water load of the campus (L_t^{hw}), hot water is produced by the HR chiller subplant ($\alpha_{hrc}^h P_{hrc,t}$), the hot water generator ($P_{hwg,t}$), and the discharge from the hot water storage ($P_{hw,t}$). The excess hot water ($P_{hx,t}$) in the system is recycled by cooling it and producing condenser water in the dump HX, and the cooling towers use evaporative cooling to reduce the temperature of this condenser water along with the condenser water produced by the chiller and the HR chiller subplants (total $P_{ct,t}$ condenser water).

In the MPC formulation, the operating loads of all units of the HVAC plant are the manipulated variables, while the states include the state of charge (SOC) of the chilled water and hot water storage tanks (TES) and carryover quantities (e.g., peak electrical demand, unmet or overmet production of chilled/hot water). Multiple time-varying disturbances are present in this system; these include the campus electrical load (L_t^e), chilled water load (L_t^{cw}), hot water load (L_t^{hw}), and electricity prices (π_t^e). The MPC uses forecasts for these disturbances over a prediction horizon $\mathcal{T}$ to determine the control action for the next immediate hour. The horizon is shifted by one hour to update disturbance forecasts and to obtain the next control action. This procedure is repeated for an entire year to obtain the closed-loop policy and associated cost. The optimization problem solved at each time t is:

$$\min \sum_{k\in\mathcal{T}} \sum_{j=\{e,w,ng\}} \hat{\pi}^j_k r^j_k + \frac{\pi^D}{\sigma_t} R_{t+1}$$
$$+ \sum_{k\in\mathcal{T}} \sum_{j\in\{cw,hw\}} \rho_j (ul_{j,k} + ol_{j,k}). \tag{3.6a}$$

$$\text{s.t. } r^e_k = \sum_{j\in\{cs,hrc,hwg,ct\}} \alpha^e_j P_{j,k} + \hat{L}^e_k,\ k \in \mathcal{T} \tag{3.6b}$$

$$r^j_k = \alpha^j_{i_j} P_{i_j,k},\ j \in \{w, ng\}, k \in \mathcal{T},\ i_w = ct, i_{ng} = hwg \tag{3.6c}$$

$$P_{ct,k} = \alpha^{cond}_{cs} P_{cs,k} + P_{hx,k},\ k \in \mathcal{T} \tag{3.6d}$$

$$P_{cs,k} + P_{hrc,k} + P_{cw,k} + S^{un}_{cw,k} - S^{ov}_{cw,k} = \hat{L}^{cw}_k,\ k \in \mathcal{T} \tag{3.6e}$$

$$\alpha^h_{hrc} P_{hrc,k} + P_{hwg,k} - P_{hx,k} + P_{hw,k}$$
$$+ S^{un}_{hw,k} - S^{ov}_{hw,k} = \hat{L}^{hw}_k,\ k \in \mathcal{T} \tag{3.6f}$$

$$E_{j,k+1} = E_{j,k} - P_{j,k},\ j \in \{cw, hw\}, k \in \mathcal{T} \tag{3.6g}$$

$$ul_{j,k+1} = ul_{j,k} - S^{un}_{j,k},\ j \in \{cw, hw\},\ k \in \mathcal{T} \tag{3.6h}$$

$$ol_{j,k+1} = ol_{j,k} - S^{ov}_{j,k},\ m \in \{un, ov\}, j \in \{cw, hw\},\ k \in \mathcal{T} \tag{3.6i}$$

$$R_{t+1} \geq r^e_k \tag{3.6j}$$

$$R_{t+1} \geq R_t \tag{3.6k}$$

$$\underline{E}_{j,k} \leq E_{j,k} \leq \overline{E}_{j,k},\ j \in \{cw, hw\}, k \in \mathcal{T} \tag{3.6l}$$

$$\underline{P}_j \leq P_{j,k} \leq \overline{P}_j,\ j \in \{cs, hrc, hwg, ct, hx, cw, hw\},\ k \in \mathcal{T} \tag{3.6m}$$

$$S^m_{j,k} \geq 0,\ m \in \{un, ov\}, j \in \{cw, hw\}, k \in \mathcal{T} \tag{3.6n}$$

$$ul_{j,k} \geq 0,\ j \in \{cw, hw\}, k \in \mathcal{T} \tag{3.6o}$$

$$ol_{j,k} \geq 0,\ j \in \{cw, hw\}, k \in \mathcal{T} \tag{3.6p}$$

Here, the residual demands of electricity, water, and natural gas that need to be purchased from the market are given by the constraints (3.6b)-(3.6c). Constraints (3.6d)-(3.6f) are the energy balance equations for the condenser water. The sufficient chilled and hot water production is maintained by imposing constraints (3.6e) and (3.6f). For maintaining

feasibility in case of under-production or over-production of chilled water or hot water, the slack variables $S^{un}_{j,k}$ and $S^{ov}_{j,k}$, $j \in \{cw, hw\}$ are added in the constraints (3.6e) and (3.6f). The state variables $ul_{j,k}$ and $ol_{j,k}$, $j \in \{cw, hw\}$ carry over the under-production or over-production (slack variables) of chilled and hot water in constraints (3.6h) and (3.6i) and these state variables are penalized in the objective function. The dynamics of the SOC for chilled and hot water TES are given by constraints (3.6g). Constraint (3.6j) computes the peak demand over the horizon and constraint (3.6k) carries over the peak demand to the next time step in the closed-loop.

The actual realizations of the loads (disturbances) might induce constraint violations when they deviate from the forecasts. To account for such violations, bounds on the chilled and hot water TES in (3.6l) are modified to include a buffer capacity (the back-off term), $\beta_j \in [0, 0.5], j \in \{cw, hw\}$. In closed-loop formulation, the bounds on $E_{j,k}$ for $j \in \{cw, hw\}$ in constraints (3.6l) are updated as:

If $\beta_j \overline{E}_j \leq E_{j,t+1} \leq (1-\beta_j)\overline{E}_j$, set $\underline{E}_{j,t+1} = \beta_j \overline{E}_j$, $\overline{E}_{j,t+1} = (1-\beta_j)\overline{E}_j$.

If $(1-\beta_j)\overline{E}_j \leq E_{j,t+1} \leq \overline{E}_j$, set $\underline{E}_{j,t+1} = \beta_j \overline{E}_j$, $\overline{E}_{j,t+1} = E_{j,t+1}$.

If $0 \leq E_{j,t+1} \leq \beta_j \overline{E}_j$, set $\underline{E}_{j,t+1} = E_{j,t+1}$, $\overline{E}_{j,t+1} = (1-\beta_j)\overline{E}_j$.

If $E_{j,t+1} \geq \overline{E}_j$,set $E_{j,t+1} = \overline{E}_j$, $\underline{E}_{j,t+1} = \beta_j \overline{E}_j$, $\overline{E}_{j,t+1} = \overline{E}_j$, and update $ol_{j,k+1} = ol_{j,k+1} + (E_{j,t+1} - \overline{E}_j)$.

If $E_{j,t+1} \leq 0$, set $E_{j,t+1} = 0$, $\underline{E}_{j,t+1} = 0$, $\overline{E}_{j,t+1} = (1-\beta_j)\overline{E}_j$, and update $ul_{j,k+1} = ul_{j,k+1} - E_{j,t+1}$.

The above updates to the storage bounds ensure that the storage is set to the maximum or minimum capacity if the storage at $t+1$ overflows or dries up when implementing the MPC action; otherwise, the fractional buffer capacity is implemented. These corrections result in lost economic performance and inefficient use of storage. We perform closed-loop MPC simulations for the central HVAC plant with the formulation described above in order to develop a BO framework for tuning the back-off terms for the chilled water and hot water TES.

The back-off term $\boldsymbol{\beta} = [\beta_{cw}, \beta_{hw}]^T$ is introduced to reserve a fraction of the maximum

capacity as a buffer to account for the uncertainty associated with the predicted disturbances. Appropriate determination of the β values is critical for optimizing the closed-loop performance. Selecting values that are too large will induce an overly conservative strategy that prevents the storage tanks from being fully utilized to reduce economic costs. On the other hand, if the β terms are too small, the number of constraint violations may increase dramatically as the size of the buffer will not be enough to safeguard against unforeseen disturbances, leading to an economic penalty. For this study, our tuning objective is the annual closed-loop cost (denoted as $f(\cdot)$), which is a function of the back-off terms $x = \{\beta_{cw}, \beta_{hw}\}$. Because a closed-form representation of f is not available, a full year-long closed-loop simulation of the HVAC plant has to be performed to evaluate the value of the objective at any x of interest. Additionally, the back-off terms affect the closed-loop MPC performance in a non-intuitive way. As a result, finding the optimal back-off values requires repeatedly simulating the closed-loop system at any selected x.

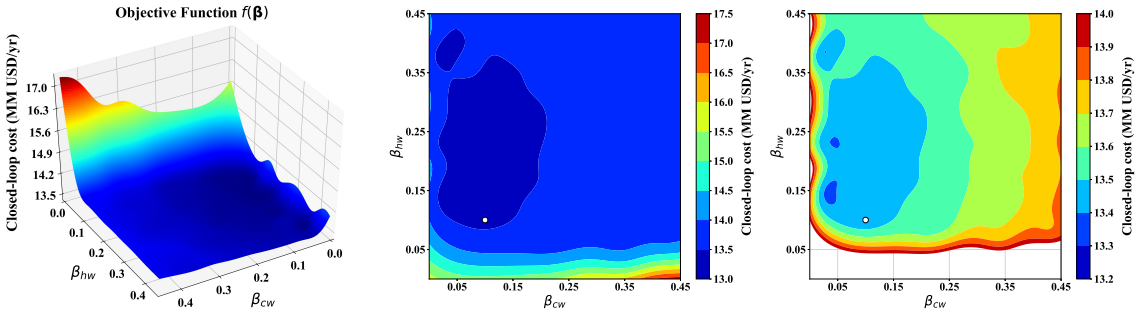

Figure 3.3: Left: 3D surface of annual closed-loop cost over back-off terms; Middle: 2D contour plot of annual closed-loop cost with the baseline cost (shown by a white marker) in [1]; Right: Refined view of closed-loop cost where the vertical scale is adjusted. These plots were obtained by interpolating simulation results under 10×10 mesh grids.

The prediction horizon of the MPC is set to 168 hours (1 week) to reflect the weekly periodicity of loads and electricity prices. The optimization problem solved at each hour is a linear program with 168,450 variables and 143,750 constraints [1]. The problems were implemented in Julia 0.6.4 and were solved with Gurobi 8.1 on a computing server with 188 GB RAM, 32-core Intel Xeon 2.30 GHz CPU. On average, each MPC problem requires

about one second to solve but simulating closed-loop behavior over an entire year requires about 2 hours of wall-clock time (each year-long simulation requires solving more than 8,700 optimization problems). Given the complexity of the underlying tuning problem, it is apparent that manual or grid search methods can easily become prohibitively expensive due to the possibly large number of trials and the resultant enormous time consumed.

Figure 3.3 shows the closed-loop cost at different combinations of back-off values for a given disturbance realization. To generate this surface, we conducted 100 simulations (obtained by using a coarse grid discretization with 10 points for each back-off term). One can see that the surface is non-convex with a couple of local minima (the global minimum is near $\beta_{cw} = 0.05$, $\beta_{hw} = 0.15$). We used the back-off term values $\beta_{cw} = \beta_{hw} = 0.1$ reported in [1], which were delicately selected based on engineering expertise, as a baseline. The year-long closed-loop cost for the baseline is 13.44 MM USD (million USD). Note that the baseline parameters may not be optimal as they were not obtained via rigorous optimization procedures. From Figure 3.3 we can see that the closed-loop cost is highly sensitive to the back-off terms; specifically, this can easily reach levels of more than 17 MM USD. The large costs illustrate that operating HVAC facilities is quite expensive, and thus, cutting down costs is essential.

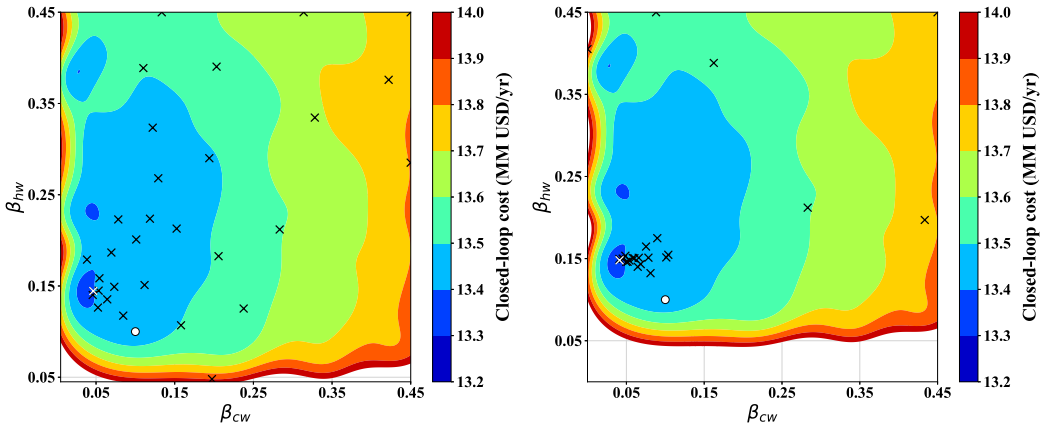

Figure 3.4: Sequence of sampling locations in the 2D parameter plane for S-BO (left) and Ref-BO (right). The white circle marker shows the baseline parameters.

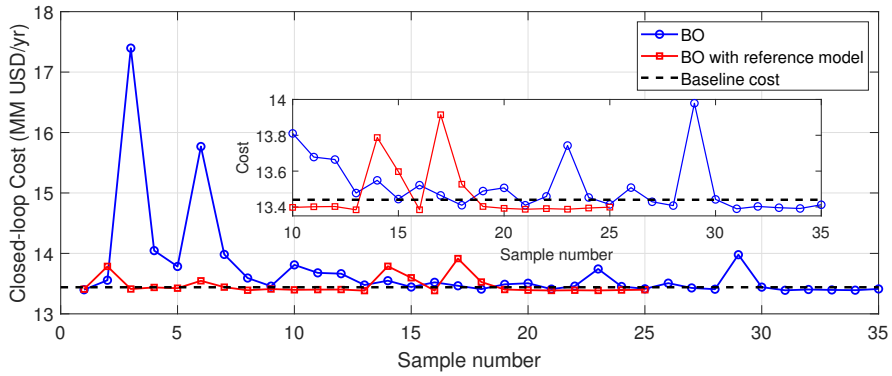

Figure 3.5: Closed-loop cost over iterations for S-BO and Ref-BO against the baseline cost. A microscopic view of the comparisons between costs from these methods is shown inside the figure.

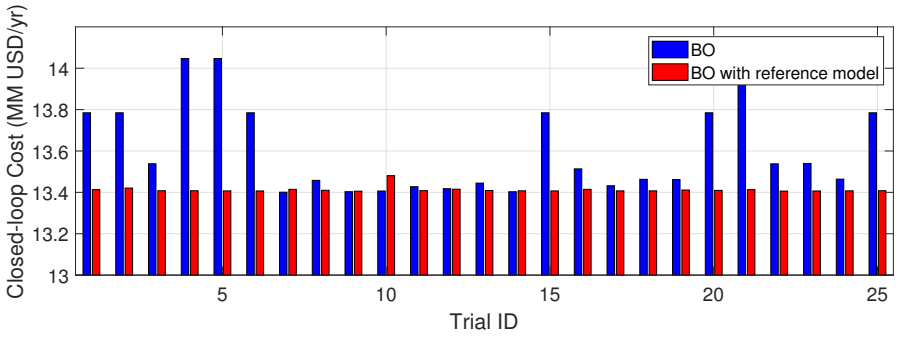

Figure 3.6: Closed-loop cost for S-BO and Ref-BO under different initialization points. Each trial corresponds to one initialization point on a 5×5 grid of the 2D parameter space.

We used Algorithm 2 to determine the optimal back-off terms for the MPC controller. First, we constructed a reference model from *reduced-horizon simulations* of the original MPC. Instead of maintaining the prediction horizon at 168 hours, we simulated the closed-loop MPC with a 24-hour horizon across 21 different combinations of back-off parameter values. Reducing the horizon in MPC simulation has been shown to significantly reduce the computational cost (by more than 50%) without significantly sacrificing performance. The 21 parameter combinations were determined sequentially using S-BO initialized at the point $\beta_{cw} = \beta_{hw} = 0.45$ and with an emphasis on *exploration* of the space

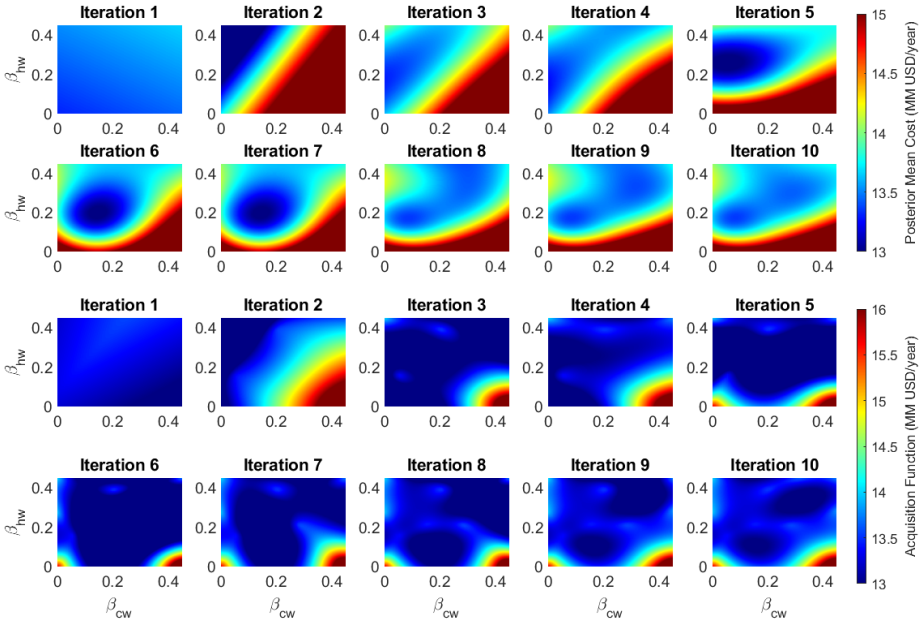

Figure 3.7: Posterior mean and AF from S-BO for the first 10 iterations. Top two rows: Posterior mean of GP model in each iteration. Bottom two rows: Acquisition function in each iteration.

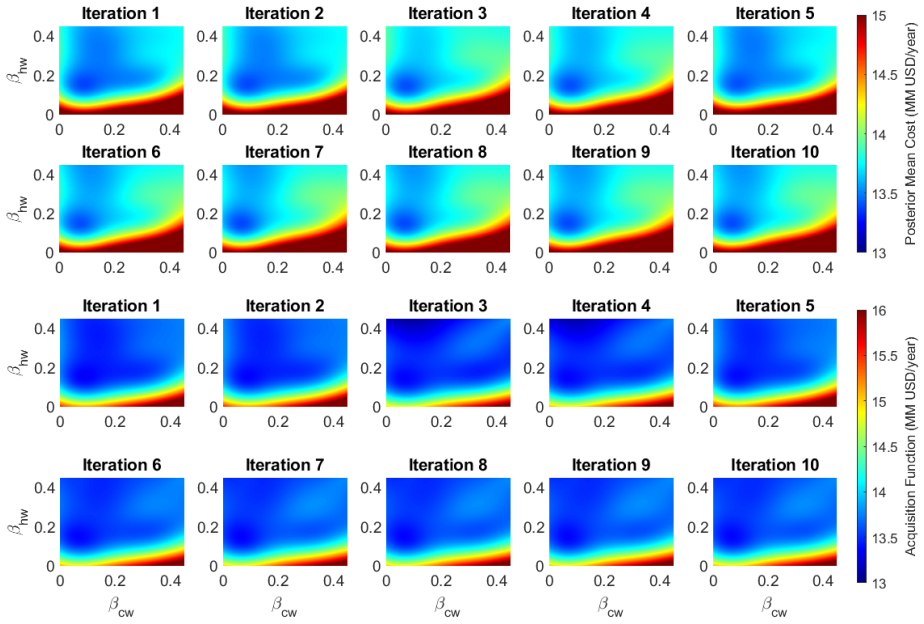

Figure 3.8: Posterior mean and AF from Ref-BO for the first 10 iterations. Top two rows: Posterior mean of GP model in each iteration. Bottom two rows: Acquisition function in each iteration.

(high value of κ). The predictive mean of a GP trained on the 21 data samples from the coarse simulations was used as the reference model. The resultant reference model can assist BO by providing prior information on the regions where the optimum may potentially reside. Thus, one can avoid the unnecessary exploration of regions that are far from the optimum. Algorithm 2 was programmed in Python 3.8.3 using Scikit-learn [100] with a `Matern` kernel function ($l = 1, \nu = 5/2, \sigma^2 = 1\text{e}{-6}$) for the GP model and the LCB AF ($\kappa = 0.5$). The optimizer selected for finding the minimum of the AF was the bounded limited-memory BFGS (L-BFGS-B) algorithm from Scipy [101]. Note that the operating cost is on the scale of 13-17 MM USD, as previously mentioned. In addition, as shown in Figure 3.3, the cost surface is largely flat over a large region of the parameter space. Thus, if the cost values are not normalized, the small predictive variance (due to the flatness of the surface) will become almost negligible compared with the scale of the operation cost. As a result, the sampling locations selected by BO will not move significantly during the search. This can significantly impact the performance of both BO methods. Therefore, prior to performing the GP modeling, the operating cost values are normalized. This can also help satisfy the zero mean assumption for GP modeling. In addition, to improve the performance of the surrogate model to the data, we re-trained the GP hyperparameters at each iteration. The sequence of sampling points selected by Ref-BO is shown in the right plot of Figure 3.4. For comparison, the left plot of Figure 3.4 illustrates the scatter of sampling locations returned by S-BO. We observe that the distribution of sampling locations is more dispersed in standard BO in our proposed method. Additionally, the bulk of the points sampled by Ref-BO are concentrated in the neighborhood of the global minimum. This observation validates the efficiency of Ref-BO at discovering the global solution.

Figure 3.5 demonstrates the progressive closed-loop operation costs over iterations for S-BO and Ref-BO. In this figure, the black dashed line represents the baseline cost evaluated at the expert-selected back-off parameters. The red line shows the yearly operation cost under the *full-horizon simulation* from using the underlying back-off parameters suggested at each iteration of our BO paradigm. Both methods were initialized at the same

point, the best performing of the 21 samples used to construct the reference model (as indicated by the reference model). By using the same initial point for both BO methods, we can compare their convergence performance on a fair basis. It is clear that Ref-BO converges after only 3 iterations (despite the slight exploratory behavior between iterations 14-18). In contrast, standard BO requires significantly more exploration and does not converge until after 14 iterations. In total, our BO framework required about 20 hours of wall-clock time; this time includes the execution of the 21 low-fidelity simulations for establishing the reference model and the 3 high-fidelity closed-loop simulations. For comparison, standard BO requires more than 28 hours (14 full-scale simulations) of wall-clock time to finding the solution. Our approach, thus, results in an 8 hour reduction in computational time (over 28%) when compared to S-BO. As stated in Remark 1, this is because the additional computational cost of developing the reference model using the low-fidelity system approximation (i.e., reduced-horizon simulations) is less than the computational savings that the reference model provides to the algorithm. The cause of the oscillations observed for both BO methods towards the later iterations in Figure 3.5 is likely the flatness of the cost surface, as observed in Figure 3.3. It is expected that the posterior mean of the GP model is also flat over a large region (cf. Figures 3.7-3.8). Thus, the AF value is extremely sensitive to minor perturbations in the posterior variance (e.g., due to disturbances) and the starting point of the optimizer when minimizing the AF. Note that the reason we choose a relatively small κ value is that the use of a reference model can guide the search directly towards the globally optimal region, and, in contrast to S-BO, exploration of a wide range of the parameter space becomes unnecessary. Besides, as shown in Figure 3.5, the initial point for both BO methods is already close to the solution. Selection of a small κ value, thus, enables a higher preference for the greedy search for the optimum. For the purpose of comparison, we also select the same κ value for S-BO. The distinct behaviors of these two methods clearly show the advantage of our method in reducing the required iterations. In practice, for standard BO, there exist some guidelines for determining the κ value. A common heuristic is a dynamic strategy where

κ is initially chosen to be large to encourage exploration and then gradually reduced to a small value to allow for exploitation [102].

From the microscopic (zoomed in) graph in Figure 3.5, we can see that both BO approaches yield superior back-off parameters than the baseline value. Specifically, using the parameters suggested by Ref-BO, ($\beta_{cw} = 0.0412$, $\beta_{hw} = 0.1480$), and S-BO, ($\beta_{cw} = 0.0465$, $\beta_{hw} = 0.1445$), the optimal operation costs are 13.3849 million USD and 13.3897 million USD respectively. Both costs are less than the baseline cost (13.4385 million USD), with a margin of 53,600 USD and 44,800 USD in annual savings. The relatively small improvement in cost is mainly attributed to the flat surface shown in Figure 3.3 and the fact that baseline parameters were carefully chosen by experts [1]. However, this case study only involves 2 parameters that can be easily tuned using domain insights. For actual MPC tuning problems, the number of tuning parameters can be so large that trial-and-error, grid search, and heuristics-guided approaches become computationally expensive and impractical. In contrast, Ref-BO scales well with the number of parameters. Thus, we anticipate that our approach will outperform traditional methods for MPC tuning in practice.

Figure 3.6 shows the closed-loop operation cost using the optimal tuning parameters determined by standard BO (blue) and our method (red) at 25 different initialization points for the optimization algorithm. Each trial corresponds to one initialization point selected from a 5×5 grid of the 2-D parameter space. Ref-BO is clearly able to outperform (lower cost) S-BO consistently across different starting locations. Standard BO is significantly more sensitive to the initial point and can be easily trapped at a local optimum if a bad initial value is chosen. In contrast, our method is robust to the initialization point, as evidenced by the fact that the optima it locates are essentially on the same level across all trials. This striking feature comes from the strong guiding effect provided by the reference model. Even if the initial point is far from the solution, the presence of the reference model can still force the search to move towards the global optimum. This is another advantage of our approach over standard BO.

The posterior mean heat maps of the GP model in each iteration and the corresponding AF values for S-BO and Ref-BO are shown in Figures 3.7 and 3.8, respectively. The top two rows of Figure 3.7 show that after 7 iterations the posterior mean surface remains relatively constant, indicating the convergence of the BO algorithm. The bottom two rows in Figure 3.7 present the AF at each iteration. Interestingly, after several iterations, the AF function demonstrates two apparent local minima. Thus, standard BO will likely converge to a local minimum (instead of global optimum) if the exploration/exploitation tradeoff is not appropriately balanced. The top two rows of Figure 3.8 show that, when using Ref-BO, the posterior mean stabilizes rapidly within a couple of iterations, remaining consistent thereafter. The developed AF function also converges quickly and exhibits only one minima towards the later iterations. Therefore, the presence of a reference model provides valuable guidance for the algorithm to select subsequent sampling locations and reduces the likelihood of converging to a local solution. In summary, our results indicate that incorporating a reference model into the BO framework can result in significant reductions in computational cost when compared to standard BO. The discovered global optimal tuning parameters can also improve the annual operating cost beyond the baseline parameters currently employed in the literature.

3.4 Conclusions

We presented a novel BO framework that incorporates a reference model for tuning MPC controllers. The tuning objective is treated as a black-box function of the controller parameters. This work is motivated by the observation that evaluating closed-loop performance can be computationally expensive and thus manual or grid search approaches are time-consuming. Moreover, existing knowledge about the underlying process is valuable for guiding the initial search in BO. To this end, we combine BO with a reference model (generated using prior knowledge or coarse system identification) to efficiently solve this complex MPC tuning problem. Specifically, we studied the optimization of the back-

off terms for the thermal energy storage of an HVAC plant. Our results show that this reference model-guided BO approach can efficiently discover the global solution in 3 iterations, whereas the standard BO algorithm requires 14 iterations, resulting in a reduction of 8 wall-clock hours of simulation (more than a 28% reduction in computational time). These observations clearly demonstrate the necessity and advantage of incorporating a reference model into the BO framework. Moreover, the proposed BO framework can easily be extended to the tuning of a large number of MPC parameters without significantly increasing the computation complexity, unlike traditional methods which rely on heuristics, grid search, or trial-and-error. As part of our future work, we are interested in exploring performance with a larger set of tuning parameters that capture different types of behavior and different types of functions to accelerate the search. Another important topic for our future study is to investigate how coarse the reference model can be to provide sufficient guidance for Ref-BO.

Chapter 4

BAYESIAN OPTIMIZATION OF INTERCONNECTED SYSTEMS

This chapter is adapted with permission from González and Zavala from a working paper and a preprint that is available at https://doi.org/10.48550/arXiv.2311.11254.

4.1 Introduction

A key element of engineering design is the identification of system configurations that maximize performance and reduce cost [103]. However, this task can often be challenging due to incomplete physical knowledge or the high complexity of experiments and available physical models. As a result, it is necessary to devise efficient optimization strategies the are capable of mitigating system complexity while also reducing the amount of model or experimental data required to find a solution. This need has motivated the development of a class of techiques known as black-box optimization algorithms [11]. These methods forgo the need for a closed-form representation of the system and instead treat it as a black-box that is sampled to generate input/output data that is then used to guide the search for a solution. Various strategies have been developed to improve the quality of the search such as response surface methodology [104], particle swarm optimization [105], and genetic algorithms [106]. In the context of engineering design, where system queries are often expensive and uncertainty in predicted performance is impor-

tant, Bayesian optimization (BO) [76] has emerged as one of the most effective black-box optimization paradigms.

While not a new algorithm, Bayesian optimization has, in recent years, become a widely-used tool for solving challenging design problems across a wide array of disciplines such as materials engineering [88], aerospace engineering [107], control [93], and synthetic biology [108]. It is a flexible algorithm, capable of accommodating both continuous and discrete design variables [69], handling problem constraints [109], and identifying failure regions [110]. Arguably its most powerful feature, however, is BO's methodology for selecting new sample points. BO uses the collected input/output data to train a probabilistic surrogate model, typically a Gaussian process (GP), that estimates not only the predicted system performance but also the uncertainty of the predictions. These estimates are passed into an acquisition function (AF), which serves as the decision-making mechanism of the algorithm, that assigns value to sample points based on both their information gain and expected performance. The AF can be tuned to place greater importance on sampling from regions with high predicted performance (exploitation) or high model uncertainty (exploration). This consideration of informational value in addition to performance enables BO to efficiently sample from several distinct regions of the design space making it an especially powerful global optimizer [53].

While the black-box assumption makes BO highly generalizable (only an interface for providing inputs and collecting outputs is needed), there is often some form of structural system knowledge available (e.g., physics or sparse interconnectivity). For example, when dealing with a complex physical system (e.g., a chemical process), several components might be well-modeled and understood, while others might not. In other words, the system is actually a *composition* of various white-box (i.e., an analytical representation is available) and black-box elements as shown in Figure 4.1. Furthermore, the fundamental principles governing the behavior of the black-box elements (e.g., conservation laws, equilibrium, value constraints) are, at least qualitatively, understood. Additionally, sparse connectivity, which provides information on how different components affect each other,

is also often known. As a result, the system of interest is usually not truly a black-box but rather a “grey-box" that is partially observable with a known structure [111]. Previous work done using several different optimization frameworks has demonstrated that exploiting this knowledge as opposed to relying on a purely black-box strategy can significantly improve the optimization routine [112, 113, 114]. Thus, it is reasonable to conclude that BO can similarly benefit by switching from a black-box to a grey-box representation of the system.

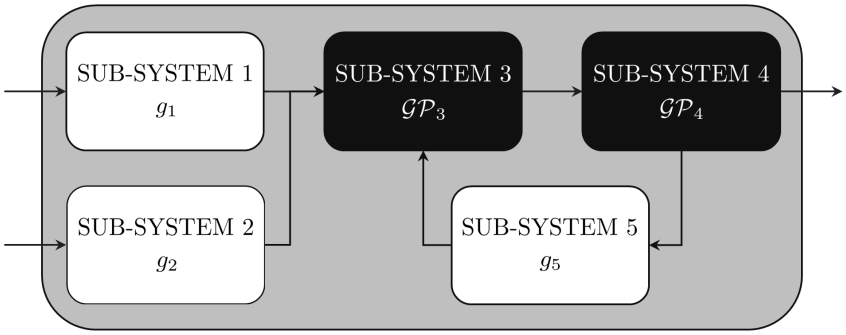

Figure 4.1: Grey-box systems often exhibit a known structure where the connectivity between different elements is understood. Not every component is always a black-box that requires a surrogate model as a closed-form representation might be available for various components.

Various methods have been developed that allow for the consideration of grey-box sytems in BO. Most of these approaches involve the use of a low-fidelity approximation of the system that is cheaper to evaluate. This simplified representation is fed to the algorithm, allowing it to learn coarse system trends and to identify potentially promising regions of the design space from the start. The approximation can be obtained through a variety of means (e.g., simplified physical models, empirical correlations, or lower-fidelity simulations) and can either be gradually refined [78, 77, 79] or remain unchanged as the algorithm progresses [115].

More recently, the work in [37] has led to a push towards developing BO frameworks that represent a system as a composite function, $f(x, y(x))$, where x are the system inputs, f is a known scalar function, and y is an unknown vector-valued function that describes

the behavior of internal system components. This shifts the modeling task from estimating the performance function directly to estimating the values of y, which serve as inputs to $f(x, y(x))$. This can result in derivative information for f becoming available, allowing for a clearer understanding of the effects of x and y on system performance [116]. Additionally, such a representation allows for explicit consideration of the white-box and black-box sections of the system, which can enable a reduction in the dimensionality of the surrogate models and allows for the modeling task to be redistributed to a simpler set of intermediate functions when f is complex [40]. This approach also readily lends itself to the inclusion of constraints, as these are often dependent on internal variables which can be captured by y [38, 117]. As a result, composite functions allow for a more complete representation of a system, especially in the context of engineering design. For example, in chemical process design the cost equations for equipment, material streams, and utilities are often readily available and it is the parameters these equations rely on (e.g., flowrates and heat duties) that are unknown. Furthermore, traditional unit operations (e.g., heat exchangers, distillation columns, compressors) have significantly more developed and mature modeling libraries available than those that tend to be more niche (e.g., bioreactors, non-equilibrium separators, solids-handling). It therefore makes sense to construct a composite function where the outer function, f, measures the price of the system based on the known cost equations, while its inputs, y, are mass and energy flows that are estimated via either mechanistic or data-driven models. Constraints can then be incorporated using values estimated for y to ensure that data-driven models obey fundamental physical laws and to enforce more traditional requirements, such as product specifications, waste generation, utility consumption, and equipment sizing, which are often important in process design.

While setting up a composite function optimization problem might be intuitive, implementing it in a BO setting is not a trivial task. As previously stated, one of the main advantages of BO is the inclusion of uncertainty estimates in the surrogate model, which allows for greater exploration of the design space when compared to a deterministic

model [112]. However, when using a composite function, the GP models generated are of y not f. Given that f is the performance metric that needs to be optimized, it is necessary propagate the predicted uncertainty from $y(x)$ to $f(x, y(x))$ (i.e., the density of f or desired summarizing statistics must be determined). A Gaussian density for $y(x)$ is directly obtained from the GP model. As a result, when f is a linear model, we can make use of the closure of Gaussian random variables under linear operations to generate the density of $f(x, y(x))$ (which is also a Gaussian). When f is nonlinear, however, a closed-form solution is not readily available, and alternative methods must be used to calculate the density. This problem has traditionally been solved numerically using sampling methods like Monte Carlo [37, 118, 39, 38], however, this approach can quickly become very computationally intensive. An alternative method proposed in [40] avoids the need to explicitly generate a probability density for f by utilizing an optimism-driven algorithm that solves an auxiliary problem, which is defined over an augmented space, allowing for the optimization to be carried out with respect to x and y. The trained GP models are used to construct a set of lower and upper confidence bound functions that are incorporated into the auxiliary problem as constraints. This specifies a range from which values for y can be selected based on the performance and uncertainty estimates of the GPs. While allows this approach eliminates the need to explicitly calculate the statistical moments of f, it also increases the size and complexity of the optimization task. This can significantly lengthen the computational time required to find a solution, especially when y is high-dimensional.

The increased functionality of composite functions coupled with the high computational intensity of exisiting methods motivates the need to develop more efficient paradigms for composite function BO. To this end, we propose the Bayesian Optimization of Interconnected Systems (BOIS) framework, a novel method that facilitates the use of composite functions via adaptive linearizations of $f(x, y(x))$ in the neighborhood of a $y(x)$ of interest (see Figure 4.2) [119]. This allows us to construct local Laplace approximations that can be used to generate closed-form expressions for the mean and uncertainty of f. In this

chapter we provide a detailed introduction to and analysis of the BOIS framework. Using a pair of complex case studies, we provide evidence of the performance and efficiency improvements this algorithm provides over standard BO as well as the composite function BO paradigms presented in [37] and [40]. Additionally, we introduce functionalities that allow us to handle feasibility considerations for the intermediate functions. We also exploit the ability of this algorithm to make use of available white-box models, which provides a significant degree of flexibility in the selection of the intermediate functions. This allows us to reduce the number of intermediates that must be modeled and the dimensions of the corresponding input spaces of these models. As a result, we are able to develop black-box models that enable system-wide optimization in a more scalable and efficient manner than existing methods.

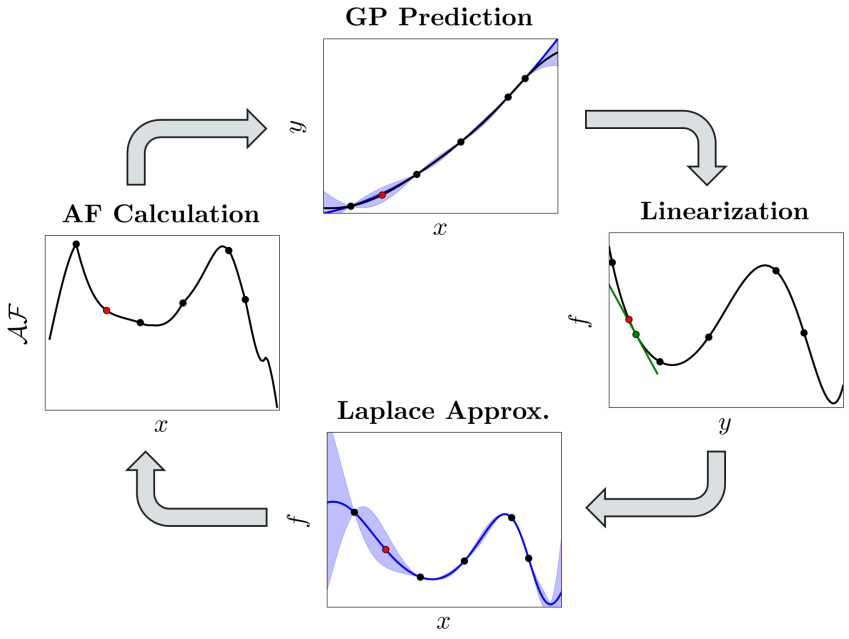

Figure 4.2: Illustrative representation of the adaptive linearization scheme employed by BOIS. At a point x of interest (red marker), a GP model estimates the value of intermediate y. A local Laplace approximation is then constructed by linearizing f around a neighboring point (green marker). The summarizing statistics are passed into an acquisition function that determines the value of sampling at the selected point. This process is repeated until the optimum of the AF is found.

4.2 Bayesian optimization with composite functions

The use of a composite function objective in a BO setting was introduced in [37]. In this context, (1.1) is recast as:

$$\min_{x} \quad f(x, y(x)) \tag{4.1a}$$

$$\text{s.t.} \quad x \in X \tag{4.1b}$$

Here, f is now a known composite function with $f : X \times Y \to \mathbb{R}$, and $y : X \to \mathbb{R}^{d_y}$ is a black-box vector-valued function with range $Y \subseteq \mathbb{R}^{d_y}$ that captures the unknown intermediate elements of the system. Note that y can be set up so that any element, y_i, is only dependent on a subset of the inputs in x, or to have a nested structure where y_i is also a function of another element in y, y_j [38, 40, 61]. This feature makes this approach especially adept at representing complex network systems where inputs often enter at different sections (e.g., material and energy inputs) and several of the elements in y are interdependent (e.g., inter-unit streams, yields, recycle loops). As f is now a known function, the formulation in (4.1) shifts the modeling task from estimating the performance function to estimating the intermediate functions. In this work, we model $y(x)$ by using an independent single-output GP for each of the black-box elements. While multi-output GP models that can consider the correlation between outputs exist [120, 121], these generally exhibit a higher computational complexity than single-output GPs and have a greater number of hyperparameters, making them more difficult to train. Additionally, we can make use of the nested structure of y to capture the correlation between relevant y_i's indirectly.

The shift from black-box to a known composite function results in a loss of the direct performance and uncertainty estimates for f that are available in S-BO. Instead, these must be calculated or somehow inferred from the statistical moments of y calculated by the GP models. In the case where f is a linear transformation of y of the form $f = a^T y + b$,

this can be done by making use of the closure of normal random variables under linear operations:

$$m_f^\ell(x) = a^T m_y^\ell(x) + b \tag{4.2a}$$

$$\sigma_f^\ell(x) = \sqrt{a^T \Sigma_y^\ell(x) a} \tag{4.2b}$$

where $m_y^\ell(x) \in \mathbb{R}^{d_y}$ and $\Sigma_y^\ell(x) \in \mathbb{R}^{d_y \times d_y}$ are the mean and variance of y. However, in the more general case where f is a nonlinear transformation, this property can no longer be used, and closed-form expressions for $m_f^\ell(x)$ and $\sigma_f^\ell(x)$ are not readily available. Composite function BO paradigms have typically addressed this issue by using some variation of Monte Carlo sampling that allows for these values to be estimated numerically [118, 39, 38]. An alternative approach was presented in [40] wherein the GP models of $y(x)$ are used to construct a set of upper and lower confidence bound functions that enable the optimization problem to be cast onto a augmented space, $X \times \hat{Y}^\ell$, where $\hat{Y}^\ell$ is the range of the intermediate functions estimated by the confidence bounds. The specific details of both of these methods are presented in 4.2.1 and 4.2.2.

4.2.1 *Monte Carlo-driven composite function Bayesian optimization*

Given the GP models of the intermediate functions, $\mathcal{GP}_y^\ell$, trained on a dataset $\mathcal{D}_y^\ell = \{x_\mathcal{K}, y_\mathcal{K}\}$, $\mathcal{K} = \{1, ..., \ell\}$, Monte Carlo sampling estimates the mean and variance of the performance function at some point x of interest by drawing S samples from the distribution of y generated by $\mathcal{GP}_y^\ell(x)$. These samples are then passed into $f(x, y(x))$, generating a range of outcomes that allow for the numerical estimation of $m_f^\ell(x)$ and $\sigma_f^\ell(x)$:

$$\hat{m}_f^\ell(x) = \frac{1}{S} \sum_{s=1}^{S} f(x, m_y^\ell(x) + A_y^\ell(x) z_s) \tag{4.3a}$$

$$\hat{\sigma}_f^\ell(x) = \frac{1}{S-1} \sqrt{\sum_{s=1}^{S} \left(f(x, m_y^\ell(x) + A_y^\ell(x) z_s) - \hat{m}_f^\ell(x) \right)^2} \tag{4.3b}$$

Here, $A_y^\ell(x) \in \mathbb{R}^{d_y \times d_y}$ is the Cholesky factor of the GP covariance ($A_y^\ell (A_y^\ell)^T = \Sigma_y^\ell$) and $z_s \in \mathbb{R}^{d_y}$ is a random vector drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. These estimates are used to construct the lower confidence bound for composite functions (LCB-CF) AF, $\mathcal{AF}^\ell_{\text{LCB-CF}}$, as outlined in Algorithm 3. As in S-BO the AF is then optimized to select a select a new sampling point. The resulting data is appended to $\mathcal{D}_y^\ell$ and the GP models are retrained. The framework for this paradigm (which we refer to as MC-BO) is summarized in Algorithm 4. Note that it is quite similar to S-BO, with the main differences being the shift to modeling the intermediate functions and the use of the LCB-CF.

Algorithm 3: Lower Confidence Bound for Composite Functions (LCB-CF)

Given x, $\mathcal{GP}_y^\ell$, S, and κ;
$m_y^\ell(x), \Sigma_y^\ell(x) \leftarrow \mathcal{GP}_y^\ell(x)$;
Calculate the Cholesky decomposition of $\Sigma_y^\ell(x)$ to determine the Cholesky factor $A_y^\ell(x)$;
for $s = 1, 2, ..., S$ **do**
  Draw sample z_s from $\mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $m_{f,s}^\ell(x) \leftarrow f(x, m_y^\ell(x) + A_y^\ell(x) z_s)$;
end
$\hat{m}_f^\ell(x) \leftarrow \frac{1}{S} \sum_{s=1}^S m_{f,s}^\ell(x)$;
$\hat{\sigma}_f^\ell(x) \leftarrow \frac{1}{S-1} \sqrt{\sum_{s=1}^S \left(m_{f,s}^\ell(x) - \hat{m}_f^\ell(x) \right)^2}$;
return $\hat{m}_f^\ell(x) - \kappa \cdot \hat{\sigma}_f^\ell(x)$

Algorithm 4: Monte Carlo-Driven Composite Function Bayesian Optimization (MC-BO)

Given κ, L, and $\mathcal{D}_y^\ell$;
Train $\mathcal{GP}_y^\ell$ using initial dataset $\mathcal{D}_y^\ell$ and obtain $\mathcal{AF}^\ell_{\text{LCB-CF}}$;
for $\ell = 1, 2, ..., L$ **do**
  Compute $x^{\ell+1} \leftarrow \text{argmin}_x \, \mathcal{AF}^\ell_{\text{LCB-CF}}(x; \kappa)$ s.t. $x \in X$;
  Sample system at $x^{\ell+1}$ to obtain $y^{\ell+1}$;
  Update dataset $\mathcal{D}_y^{\ell+1} \leftarrow \mathcal{D}_y^\ell \cup \{x^{\ell+1}, y^{\ell+1}\}$;
  Train GP using $\mathcal{D}_y^{\ell+1}$ to obtain $\mathcal{GP}_y^{\ell+1}$ and $\mathcal{AF}^{\ell+1}_{\text{LCB-CF}}$;
end

While Monte Carlo provides a convenient manner for calculating the density of f, it

is a computationally intensive method for doing so. Accurately estimating $m_f^\ell(x)$ and $\sigma_f^\ell(x)$ in regions of the design space with high model uncertainty or where $f(x, y(x))$ exhibits high sensitivity to variations in $y(x)$ can require a significant number of samples (on the order of 10^3 or more). Given that the cost of drawing a sample from a GP scales as $\mathcal{O}(S\ell^3)$, generating the samples necessary in these instances can require a significant amount of computational time [53]. Additionally, even though f is a known function and is significantly cheaper to evaluate than the system, at large values of S the cost of repeatedly calculating the value of $f(x, y(x))$ can also become nontrivial. This issue is compounded by the fact that (4.3) must be recalculated at every point of interest.

4.2.2 *Optimism-driven composite function BO*

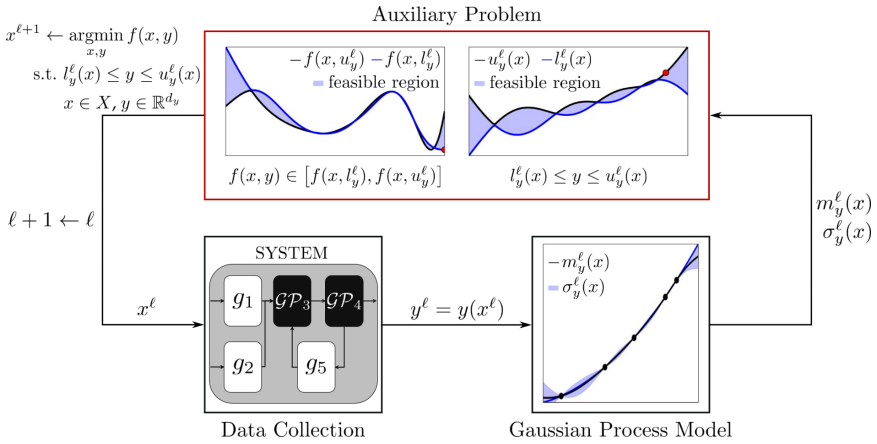

Figure 4.3: Workfow of the OP-BO algorithm. Mean and uncertainty estimates from $\mathcal{GP}_y^\ell$ are used to create a confidence interval bounded by $l_y^\ell(x)$ and $u_y^\ell(x)$ that constrains the possible values of y. These are incorporated into an auxiliary problem that is optimized to select a new sample point $x^{\ell+1}$. The resulting data is then appended to the dataset $\mathcal{D}_y^\ell$ and the GP models are retrained

As previously stated, when f is formulated as a composite function, its closed-form representation is known. As a result, its derivatives can be calculated, making it possible to determine the optimal values of x and y using gradient-based methods [116]. However, the solution might be infeasible as the proposed values of y might be inconsistent with the relationships imposed by the intermediate functions. Typically, this issue is handled

by the use of equality constraints that are designed to ensure feasibility. This requires that the closed-form representation of y be available, which is not the case in a composite function BO setting. However, the behavior of the intermediate functions can be estimated using $\mathcal{GP}_y^\ell$. The simplest approach for setting up the required constraints would then be to use the means of the GP models, but this discounts the information provided by the uncertainty estimates. The optimism-driven composite function BO algorithm (which we refer to as OP-BO) proposes an alternative approach wherein the values of y are instead restricted to a *confidence interval* that is specified by the GP models [40]. This is done via a set of upper and lower confidence bound functions that are incorporated into the problem as inequality constraints and are of the form:

$$l_y^\ell(x) = \max\{m_y^\ell(x) - \kappa \cdot \sigma_y^\ell(x), \hat{l}_y\} \tag{4.4a}$$

$$u_y^\ell(x) = \min\{m_y^\ell(x) + \kappa \cdot \sigma_y^\ell(x), \hat{u}_y\} \tag{4.4b}$$

where $\hat{l}_y \in \mathbb{R}^{d_y}$ and $\hat{u}_y \in \mathbb{R}^{d_y}$ are the lower and upper feasibility bounds of y respectively; κ determines the size of the confidence interval and thereby sets the emphasis placed on exploration similar to (2.2). This allows OP-BO to construct and solve an auxiliary problem of (4.1) of the form:

$$\min_{x,y} \quad f(x,y) \tag{4.5a}$$

$$\text{s.t.} \quad l_y^\ell(x) - y \leq 0 \tag{4.5b}$$

$$y - u_y^\ell(x) \leq 0 \tag{4.5c}$$

$$x \in X, y \in \mathbb{R}^{d_y} \tag{4.5d}$$

This problem is solved at every iteration of the algorithm to determine the next sampling point $x^{\ell+1}$, essentially filling the role of the AF in OP-BO. After sampling at this point, $\{x^{\ell+1}, y^{\ell+1}\}$ is appended to the current dataset and the GP models are re-trained allowing for $l_y^\ell(x)$ and $u_y^\ell(x)$ to be updated. The workflow for the OP-BO algorithm is detailed in

Algorithm 5 and Figure 4.3 provides an illustrative summary.

Algorithm 5: Optimism-Driven Composite Function Bayesian Optimization (OP-BO)

Given κ, L, and $\mathcal{D}_y^\ell$;
Train $\mathcal{GP}_y^\ell$ using initial dataset $\mathcal{D}_y^\ell$ and obtain l_y^ℓ and u_y^ℓ;
for $\ell = 1, 2, ..., L$ **do**
  Compute $x^{\ell+1}$ by solving

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ \text{s.t.} \quad & l_y^\ell(x) - y \leq 0 \\ & y - u_y^\ell(x) \leq 0 \\ & x \in X, y \in \mathbb{R}^{d_y} \end{aligned}$$

  Sample system at $x^{\ell+1}$ to obtain $y^{\ell+1}$;
  Update dataset $\mathcal{D}_y^{\ell+1} \leftarrow \mathcal{D}_y^\ell \cup \{x^{\ell+1}, y^{\ell+1}\}$;
  Train GP using $\mathcal{D}_y^{\ell+1}$ to obtain $\mathcal{GP}_y^{\ell+1}$, $l_y^{\ell+1}$, and $u_y^{\ell+1}$;
end

Unlike the AFs used in S-BO and MC-BO, (4.5) does not require the estimation of the probability density of f. Thus, OP-BO does not need to rely on the use of sampling methods like Monte Carlo. While this approach might seem more efficient, it should be noted that the dimensions of search space in the auxiliary problem ($\mathbb{R}^{d_x} + \mathbb{R}^{d_y}$) are greater than in the original ($\mathbb{R}^{d_x}$). As a result, problems that have a significant number of intermediate functions (large value of d_y) can lead to scenarios where (4.5) potentially requires a significant amount of computational time to solve. Thus, there is likely a point at which the cost of optimizing the lower dimensional $\mathcal{AF}_{\text{LCB-CF}}^\ell$ is lower than the cost of solving the high-dimensional auxiliary problem. Note that in these scenarios the computational intensity of both MC-BO and OP-BO would likely be high, further highlighting the need to develop more efficient paradigms for composite function BO.

4.3 The BOIS Approach

While MC-BO and OP-BO provide two very distinct methodologies for handling composite functions in a BO setting, both paradigms are motivated by the same fundamental challenge: the lack of closed-form expressions for $m_f^\ell(x)$ and $\sigma_f^\ell(x)$. As previously stated, it is generally not possible to obtain these when f is a nonlinear mapping of y. However, we can make use of the availability of the derivatives of f and the closure of normal random variables under linear transformations to formulate a solution that allows us to overcome this issue.

Consider the case where f is a once-differentiable mapping with respect to y, in this scenario, it is possible to conduct a linearization of f at the current iterate (as is done in standard optimization algorithms such as Newton's method). For the purpose of our discussion, we choose to represent f as:

$$f(x, y) = g(x) + h(x, y) \tag{4.7}$$

Using a first-order Taylor series expansion, we linearize f with respect to y around some reference point y_0:

$$f(x, y) \approx g(x) + h(x, y_0) + J^T(y - y_0) \tag{4.8}$$

where

$$J = \nabla_y h(x, y_0) \tag{4.9a}$$

$$= \nabla_y f(x, y_0) \tag{4.9b}$$

At some point of interest, x, we can calculate $m_y^\ell(x)$ and $\Sigma_y^\ell(x)$ using $\mathcal{GP}_y^\ell$ and define the

following:

$$\Delta_{\text{lo}} = \max\{0, \hat{l}_y - m_y^\ell(x)\} \tag{4.10a}$$

$$\Delta_{\text{hi}} = \min\{0, \hat{u}_y - m_y^\ell(x)\} \tag{4.10b}$$

$$\hat{y}^\ell = m_y^\ell(x) + \Delta_{\text{lo}} + \Delta_{\text{hi}} \tag{4.10c}$$

$$\hat{\Sigma}^\ell = \Sigma_y^\ell(x^{\ell+1}) \tag{4.10d}$$

where (4.10a) and (4.10b) ensure that $\hat{y}^\ell$ is within any specified feasibility bounds of y. Note that the rationale behind these two calculations is that if $m_y^\ell < \hat{l}_y$ or $m_y^\ell > \hat{u}_y$, then it is reasonable to interpret this as an indication that the true value of $y(x)$ is likely near the closest bound. If we then select a reference point, $\hat{y}_0^\ell$, in the ϵ-neighborhood of $\hat{y}^\ell$, where $|\hat{y}_0^\ell - \hat{y}^\ell| \leq \epsilon$, we can approximate f at x as:

$$f(x, y(x)) \approx g(x) + h(x, \hat{y_0}^\ell) + J^T(y(x) - \hat{y}_0^\ell) \tag{4.11a}$$

Note that in this context, g contains only the white-box elements of f that have no dependency on y and, therefore, $g(x)$ is a *deterministic* variable. Combining this approximation of the performance function with (4.2), we are now able to derive a set of closed-form expressions that estimate the mean and uncertainty of f:

$$m_f^\ell(x) = J^T\hat{y}^\ell + g(x) + h(x, \hat{y}_0^\ell) - J^T\hat{y}_0^\ell \tag{4.12a}$$

$$\sigma_f^\ell(x) = \left(J^T\hat{\Sigma}^\ell J\right)^{\frac{1}{2}} \tag{4.12b}$$

Using this expression, we are now able to construct the Bayesian Optimization of Interconnected Systems (BOIS) paradigm for composite function BO.

Similar to the previously discussed methods, the BOIS framework is initialized by using a dataset $\mathcal{D}_y^\ell$ to train a set of GP models of the intermediate functions $\mathcal{GP}_y^\ell$. Given some point of interest, x, and the corresponding mean and uncertainty estimates for

$y(x)$, Equations (4.8)-(4.12) are used to construct the lower confidence bound for BOIS acquisition function (LCB-BOIS, denoted as $\mathcal{AF}^{\ell}_{\text{BOIS}}$), and this AF is then optimized to select a new sampling point, $x^{\ell+1}$. After sampling, the obtained datapoint, $\{x^{\ell+1}, y^{\ell+1}\}$, is appended to the dataset and the GP models are retrained. A summary of this procedure is presented in Algorithm 6 and Figure 4.4 provides a visual representation. Note that this framework is quite similar to MC-BO with the LCB-CF AF being replaced with LCB-BOIS.

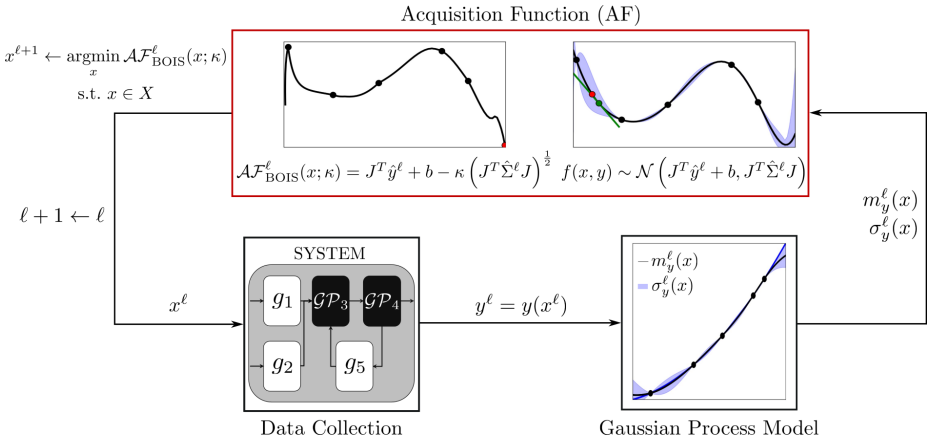

Figure 4.4: Workflow of the BOIS algorithm, note that $b = g(x) + h(x, \hat{y}^{\ell}_0) - J^T\hat{y}^{\ell}_0$. A set of GP surrogate models of y is trained using $\mathcal{D}^{\ell}_y$. The mean and variance estimates calculated by the GPs are passed into $\mathcal{AF}^{\ell}_{\text{BOIS}}$ which generates a local Laplace approximation for the density of f. This AF is then optimized to obtain a new sample point, $x^{\ell+1}$. The system is then sampled at this point and the collected data is appended to the dataset and used to retrain the GP models.

Algorithm 6: Bayesian Optimization of Interconnected Systems (BOIS)

Given κ, ϵ, L, and $\mathcal{D}^{\ell}_y$;
Train $\mathcal{GP}^{\ell}_y$ using initial dataset $\mathcal{D}^{\ell}_y$ and obtain $\mathcal{AF}^{\ell}_{\text{BOIS}}$;
for $\ell = 1, 2, ..., L$ **do**
  Compute $x^{\ell+1} \leftarrow \text{argmin}_x\, \mathcal{AF}^{\ell}_{\text{BOIS}}(x;\kappa)$ s.t. $x \in X$;
  Sample system at $x^{\ell+1}$ to obtain $y^{\ell+1}$;
  Update dataset $\mathcal{D}^{\ell+1}_y \leftarrow \mathcal{D}^{\ell}_y \cup \{x^{\ell+1}, y^{\ell+1}\}$;
  Train GP using $\mathcal{D}^{\ell+1}_y$ to obtain $\mathcal{GP}^{\ell+1}_y$ and $\mathcal{AF}^{\ell+1}_{\text{BOIS}}$;
end

Unlike MC-BO and OP-BO which are agnostic to the nature of the density of f, the BOIS framework implicitly assumes that f is normally distributed in the neighborhood of

the iterate $\hat{y}_0^\ell$. In other words, at any x of interest, BOIS passes the mean and uncertainty estimates calculated by $\mathcal{GP}_y^\ell$ into (4.11) to construct a local Laplace approximation of the performance function. As this approximation is then also Gaussian, it is also possible to obtain expressions for the probabilities and quantiles (to construct different types of AFs). Because $f(x, y(x))$ is likely not a normally distributed random variable, the Laplace approximation will result in a worse fit as the distance between $\hat{y}^\ell$ and $\hat{y}_0^\ell$ grows, similar to how (4.8) becomes less accurate. Thus, it is important to choose a value for ϵ that provides an accurate estimate of the density of f at $\hat{y}^\ell$ and to not extrapolate the fit beyond this point. However, it is important to note that the linearization is updated in an adaptive manner (by linearizing around the current iterate).

By deriving closed-from approximations for $m_f^\ell(x)$ and $\sigma_f^\ell(x)$, BOIS is able to reduce the number of function calls to $\mathcal{GP}_y^\ell$ and f significantly when compared to MC-BO. At a given point x, BOIS only has to sample from the GPs once to obtain the estimates for $\hat{y}^\ell$ and $\hat{\Sigma}^\ell$ and the performance function is similarly only evaluated once to calculate $f(x, \hat{y}_0)$; recall that this is done tens to thousands of times in MC-BO. While BOIS does have to compute (4.9), this is also only done once per x. Additionally, calculating function gradients has been shown to have a computational cost similar to that of evaluating the function itself when methods like automatic differentiation are used [122], [123]. As a result, the computational cost of calculating $\mathcal{AF}_{\text{BOIS}}^\ell(x)$ can be significantly lower than that of calculating $\mathcal{AF}_{\text{LCB-CF}}^\ell(x)$. If we perform a similar comparison between BOIS and OP-BO, we observe that, like BOIS, OP-BO only samples from the GP models and evaluates the performance function once when setting up the auxiliary problem. However, the auxiliary problem is a constrained problem that is optimized over a higher dimensional space than the LCB-BOIS AF. As a result, OP-BO likely requires more computational time obtain a new sample point than BOIS, especially when y is high-dimensional.

4.4 Numerical Experiments

We tested and compared the performances of S-BO, MC-BO, OP-BO and BOIS using various numerical experiments. Our aim is to demonstrate that BOIS can perform as well or better than the existing methods while being less computationally intensive than MC-BO and OP-BO. In this section, we present the results from two case studies. The first study focuses on the performance of a simulated chemical process, and the second examines the design of a photobioreactor (b-PBR) in a nutrient recovery process. In both systems, closed-form models are available for some of the process units, making them excellent candidates for benchmarking the composite function BO algorithms. A detailed overview of the systems used in each case study, along with the corresponding process unit models, can be found in Appendix A.1. Note that MC-BO used $S = 100$ samples to calculate $\hat{m}_f^\ell(x)$ and $\hat{\sigma}_f^\ell(x)$, and the value of ϵ in BOIS was set to $\hat{y}^\ell \times 10^{-3}$ The data and code needed to reproduce the results can be found at `https://github.com/zavalab/bayesianopt`. All algorithms were implemented in Python 3.11 and used the `gaussian_process` module from Scikit-learn [100] for GP modeling and `minimize` from Scipy [101] to optimize the acquisition functions; the gradient of f in (4.9) was evaluated using `approx_fprime`, also from Scipy.

4.4.1 *Optimization of a chemical process*

Consider the following chemical process: reagents A and B are compressed, heated and then fed into a reactor where they form product C. The reactor effluent is sent to a separator, where C is recovered as a liquid. Note that B is essentially non-condensable, while small amounts of A can be present in the liquid phase. The vapor stream exiting the separator is largely composed of unreacted reagents. A fraction of this stream is recycled and fed back to the reactor after being heated and compressed, while the remainder is purged. The demand for C is capped at a specified value, $\bar{F}$, and any excess product

generated cannot be sold. Our goal is to determine the operating temperatures and pressures of the reactor and separator as well as the recycle fraction that will minimize the operating cost of the process, which we define as:

$$f_1(x, y(x)) = \sum_{j\in\{A,B\}} w_{j0} F_j + F_S \sum_{i\in\{A,B,C\}} w_i \psi_i + w_3 \left(\frac{\psi_C F_S - \bar{F}}{\bar{F}} \right)^2 \tag{4.13a}$$

$$f_2(x, y(x)) = \sum_{h=1}^{5} w_h \dot{Q}_h, + w_e \sum_{k=1}^{3} \dot{W}_k \tag{4.13b}$$

$$f(x, y(x)) = f_1(x, y(x)) + f_2(x, y(x)) \tag{4.13c}$$

Here, F_j denotes the molar flowrate of A or B into the process; ψ_i is the mol fraction of A, B, or C in the product stream, which exits the process at rate F_S. The heating and cooling requirements of the heaters, reactor, and separator are denoted as $\dot{Q}_h$, and $\dot{W}_k$ is the power load of the k^{th} compressor. The costs of reagents and heat and power utilities are w_j, w_h and w_e respectively, while w_i refers to the value of species i in the product stream. The demand cap is enforced via a quadratic penalty term that incurs an additional cost, scaled by w_3, when the process operates at value of F_S that is different than $\bar{F}$.

The design space was defined as the box domain $X = [673, 250, 288, 140, 0.5] \times [973, 450, 338, 170, 0.9]$ with the optimal solution located at $x = (844, 346, 288, 170, 0.9)$. The performance of the algorithms was measured across 25 trials. During each trial, all of the algorithms ran for 100 iterations and were initialized using the same two points drawn from a uniform distribution of X. The reactor and separator were treated as black-boxes, and the compressors and heaters were assumed to be white-box elements. We defined the intermediate functions as the purge to feed ratio of B, η_B, the product to purge ratio of A, η_A, the purge to product ratio of C, η_C, and the utility requirements of the reactor and separator, $\dot{Q}_4$ and $\dot{Q}_5$ respectively. By combining these with the white-box models for the compressors and heaters, we were able to fully specify the system using only five intermediates. For comparison, if we had chosen to model the elements in (4.13) directly, we would have had 8 black-box functions, and we would not have been able to

use the white-box models for the recycle compressor and heater. Additionally, by nesting some of the selected functions within each other, we were able to reduce the number of inputs used by the GP models of most of the intermediates. The specific details on the construction of the GP models for these can be found in Appendix B.1.1.

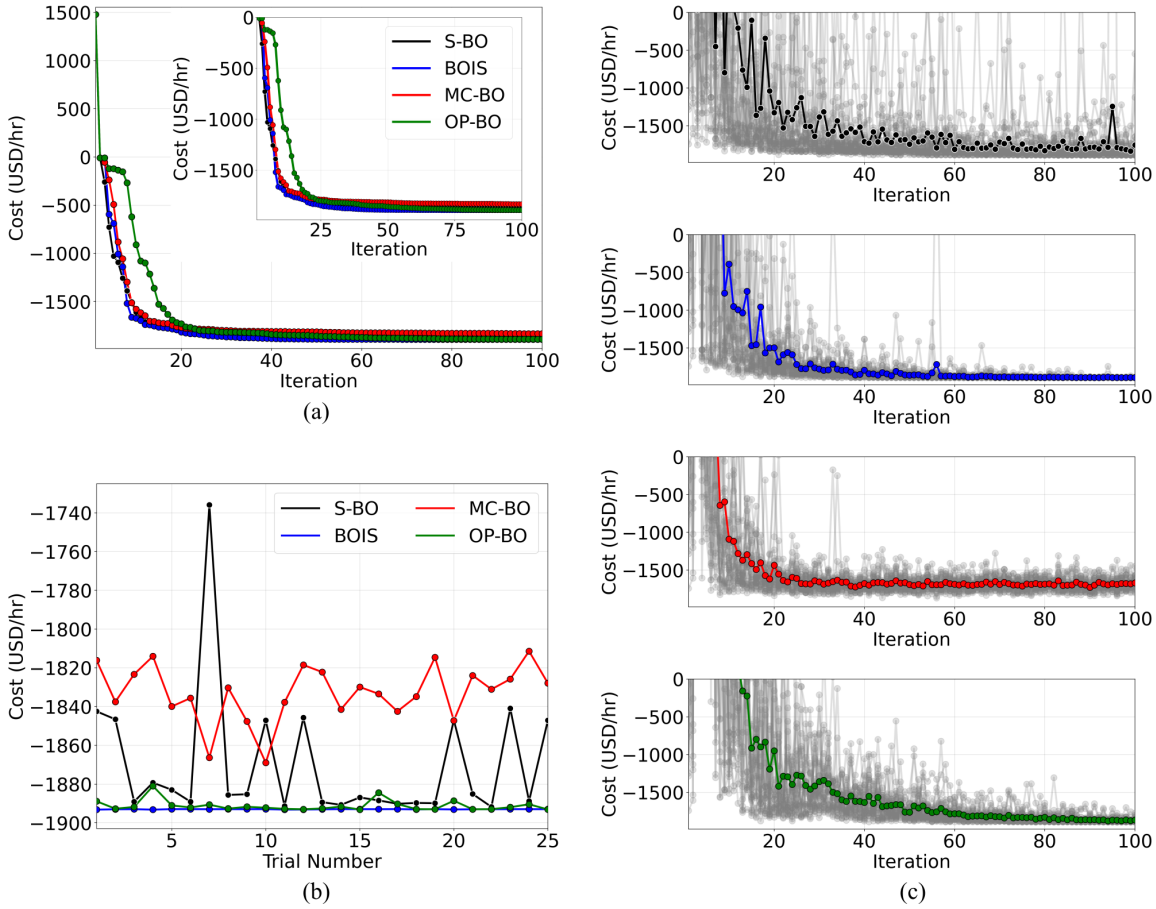

Figure 4.5: Performance comparison of the tested algorithms for the chemical process optimization problem based on (a) the best solution at the current iterate, (b) the best solution located by each algorithm during each trial, and (c) the distribution of the sampling behavior across the 125 runs for each of the tested methods with the average behavior shown in color.

The results shown in Figure 4.5 summarize the performance of the tested algorithms across the 25 runs. We observed that BOIS outperformed the other methods. On average, it beat out S-BO and MC-BO by 1.2% and 3.3% respectively in terms of solution value. While OP-BO returned a similarly-valued solution, it required approximately 30 more

iterations to find it. BOIS was also remarkably robust, it consistently arrived at the global optimum regardless of where it was initialized. Again, OP-BO performed similarly, it located a solution within 1% of the global optimum at every trial. S-BO and MC-BO exhibited significantly more variability with S-BO appearing to be especially sensitive to the initial guess. Note that MC-BO was unable to find the global solution.

The comparatively worse performances of S-BO and MC-BO were likely due to the fact that, as shown on the right side of Figure 4.5, neither of these algorithms appeared to converge to a solution within 100 iterations. S-BO, especially, continued to sample from highly sub-optimal regions late into each trial. This indicates that the algorithm struggled to learn the flow penalty, and provides a clear demonstration of the advantages of employing a composite representation of f. The sampling behavior of the composite BO algorithms was significantly less variable as they were provided with a representation of the performance function that includes the flow penalty, allowing them to effectively identify the regions of the design space that minimize its value. S-BO, meanwhile, did not have access to this information and could only learn it by sampling, which is clearly an ineffective method.

In the case of MC-BO, we surmise that its behavior was likely the result of the need for a greater number of samples. We observed that the performance function was sensitive to changes in the value of the intermediate functions. At a given x with $S = 100$, different evaluations of the LCB-MCBO AF could return values that differed by over 10%. This made the AF optimization step more likely to recommend a sub-optimal sampling point as it actually calculated a range of utility values rather than a single, replicable value as is the case in S-BO, OP-BO, and BOIS. While a solution to this problem would be to increase the value of S, this would also increase the computational cost of the algorithm. This highlights the advantage of utilizing the more specialized methods employed by OP-BO and BOIS to obtain a closed-form representation of the acquisition function/auxiliary problem over the more general Monte Carlo estimation approach.

We used the difference between the total execution time and total system sampling

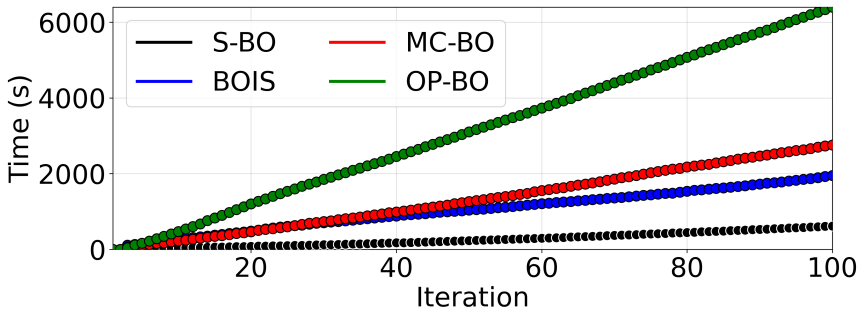

Figure 4.6: Computational intensity of the tested algorithms for solving the chemical process optimization problem measured as the difference between the total execution time and total system sampling time.

time of each algorithm as a measurement of its computational intensity. This metric is largely dominated by AF or auxiliary problem optimization step, which is the distinguishing feature between MC-BO, OP-BO, and BOIS. From the results shown in Figure 4.6 we can observe that, unsurprisingly, S-BO was the least computationally intensive of the algorithms tested, clocking in at an average time of 619 seconds. Given that it directly models the performance function, S-BO only needs to sample from its GP model to obtain the required mean and uncertainty estimates; this makes the evaluation of its AF faster. Additionally, due to fact that it contains several white-box elements that need to be evaluated, calculating $f(x, y)$ does require more computational time than sampling from $\mathcal{GP}_f^\ell$. However, as discussed above, this streamlined approach comes at the cost of decreased performance. Among the three composite function BO algorithms, BOIS, with an average time of 1952 seconds, was 41% faster than MC-BO (2758 seconds) and 3.3 times faster than OP-BO (6398 seconds). This result, coupled with the strong performance of BOIS, demonstrates that not only is the LCB-BOIS AF able to effectively direct the search for the optimal solution, but it also requires significantly less computational resources to optimize. While OP-BO achieved a similar level of performance, its efficiency was significantly hampered by the need to optimize the auxiliary problem over x and y. As these are both five dimensional, solving (4.5) involves navigating a ten-dimensional space, significantly increasing the computational time required to solve this problem com-

pared to the AF optimization step of any of the other algorithms. MC-BO, while 2.3 times faster than OP-BO, was ultimately unable to match BOIS in either performance or speed. In summary, these observations affirm the claim that BOIS is an effective paradigm for composite function BO as it was clearly able to match or surpass the existing methods in terms of both performance and computational efficiency.

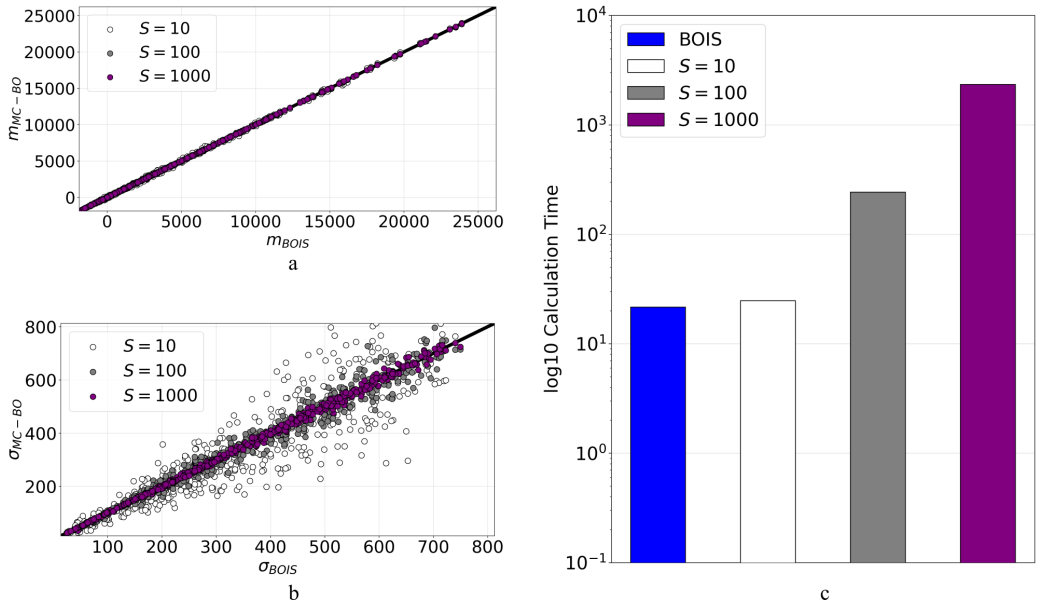

Figure 4.7: Parity plots of the estimates of $m_f^\ell(x)$ (a) and $\sigma_f^\ell(x)$ (b) for (4.13) and $\log_{10}$ of the time required to generate the estimates (c) at 500 points in X using BOIS with $\epsilon = \hat{y} \times 10^{-3}$ and MC-BO with samples sizes $S = 10$, 10^2, and 10^3; the same trained GP model of $y(x)$ was used by both algorithms.

To confirm that BOIS provides accurate estimates of mean and uncertainty of f, we compared the values calculated by BOIS for $m_f^\ell(x)$ and $\sigma_f^\ell(x)$ with those obtained from MC-BO. We know that as we increase the number of samples, (4.3) will return values closer to the true moments of f. Using a trained GP model of $y(x)$, we calculated $\hat{m}_f^\ell(x)$ and $\hat{\sigma}_f^\ell(x)$ at 500 randomly selected points in X using 10, 100, and 1000 samples. If the values calculated by BOIS in (4.12) are accurate, the difference between these values and those returned by MC-BO should decrease as S increases. The results presented on the left side of Figure 4.7 demonstrate that this was precisely the case. While the estimates for $m_f^\ell(x)$ remained fairly constant across the various values of S, the estimates

for $\sigma_f^\ell(x)$ were significantly more dynamic. Remarkably, we observed that BOIS estimated the uncertainty of f with the same degree of accuracy as MC-BO with 1000 samples. We also observed that the amount of time required to generate these estimates, shown on the the right side of Figure 4.7, was two orders of magnitude lower when we used BOIS than when we used MC-BO with $S = 1000$. These results demonstrate that the adaptive linearization scheme employed by BOIS is not only fast but also accurate, further emphasizing that this method provides BOIS with a significant advantage over algorithms that rely on sampling-based estimation techniques.

4.4.2 *Design of a photobioreactor*

Nutrient management is a key challenge facing the agriculture sector as current practices are unsustainable. Processes that allow for nutrient recycling offer a potential solution to this issue. One such process involves the production of a cyanobacteria (CB) biofertilizer from animal waste. At the center of this operation is a bag photobioreactor (b-PBR) in which CB is grown. Due to the novelty of this application, this unit has not been widely studied and must be designed experimentally. However, computational methods like BO can aid in the identification of reactor settings that optimize overall process performance.

In this case study, we considered the deployment of a biofertilizer production facility coupled with biogas generation using the waste produced at a hypothetical 1000 animal unit dairy farm. We measured performance using the minimum selling price (MSP) that the biofertilizer must be sold at to achieve a 15% discounted return on investment (DROI) over a 10 year project lifetime. The b-PBRs were modeled as black-boxes while the remaining units (anaerobic digester, biogas purification and CB harvesting section) were treated as white-boxes that make use of existing models (see A.1.2). Our goal was to identify the settings for three key reactor parameters: surface area to volume ratio (m^{-1}), batch time (days), and CB nutrient density (mass fraction), that minimize the MSP.

We defined the design space as the box domain $X = [11.5, 22.5, 0.013] \times [19.2, 37.5, 0.154]$.

Note that the MSP function is a highly multi-modal function within this domain and the global solution is located at $[15.4, 30, 0.056]$. Given that the b-PBR is the only black-box in the system, it did not make sense to include all of the elements of the MSP function (material flows and unit sizes) in $y(x)$ as this unnecessarily increases its dimensionality. Thus, we instead opted to use two intermediate functions that enabled the full specification of the reactor: the total reactor volume, and the final CB titer (see B.1.2). In addition to reducing the size of $y(x)$, this selection allowed for the development of a b-PBR surrogate model that is highly refined in the regions near the optimum. The performance of each algorithm was measured across 125 trials, each initialized with a different pair of points selected from a $5 \times 5 \times 5$ grid of the design space. During each trial, all of the algorithms ran for 50 iterations and used the same set of initialization points.

The performance profiles shown in Figure 4.8 illustrate that BOIS outperformed S-BO and MC-BO by and average 5.4% and 1.6% respectively. While OP-BO was able to locate the same optimum as BOIS, we observed that, on average, it took 10 additional iterations to find this point. BOIS was also the only algorithm that appeared to consistently converge by the end of each trial. While BOIS explored extensively during the first half of each trial, after approximately 40 iterations it tended to switch to exploiting the region near the optimum. Meanwhile, S-BO, MC-BO, and OP-BO continued to sample from sub-optimal regions, even at the end of each trial. From this, we can conclude that BOIS was able to navigate the design space more efficiently and could differentiate between optimal and sub-optimal regions more quickly than the other methods. Note that this increase in speed did not come at the expense of a decreased solution value as BOIS did not get trapped in a local optimum during any of the trials.

In terms of robustness, we observed that OP-BO and BOIS were able to arrive at the same solution regardless of where they were initialized. MC-BO exhibited some sensitivity, with best solution values located in each trial varying between -1.1% to 2.0% from the average minimum value. As was seen in the previous case study, this stems from the fact that the value returned by the LCB-CF AF at a given x varies between evaluations.

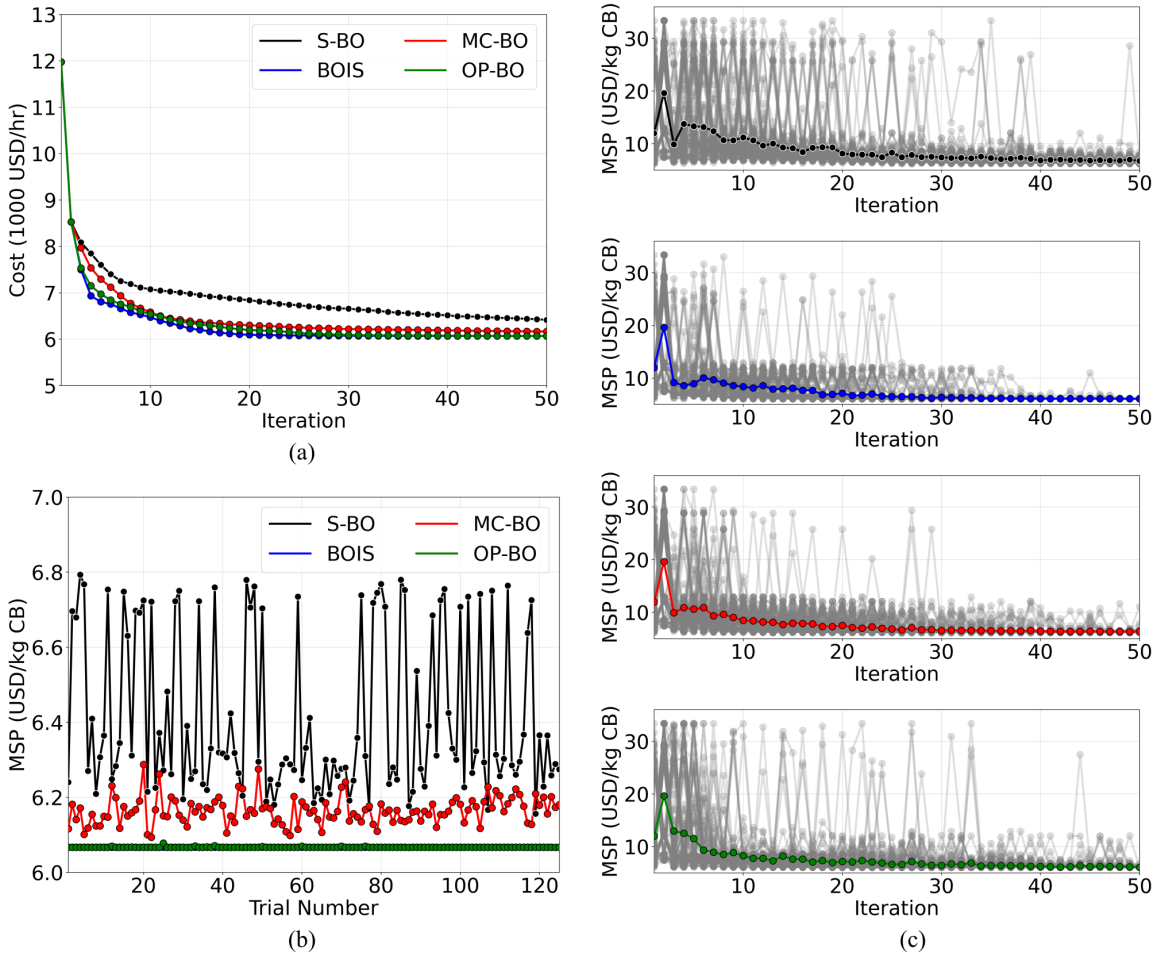

Figure 4.8: Performance comparison of the tested algorithms for the photobioreactor design problem based on (a) the best solution at the current iterate, (b) the best solution located by each algorithm during each trial, and (c) the distribution of the sampling behavior across the 125 runs for each of the tested methods with the average behavior shown in color.

This can make if difficult to ascertain the true utility value of sampling at x, increasing the chance of selecting a sub-optimal sample point. However, it is worth noting that the variability observed for MC-BO was significantly less than what was observed for S-BO, which returned values across an almost 10% range (-4.0% to 5.9%) around the average minimum value. This extreme sensitivity was likely due to the fact that the MSP is a difficult function to learn as it is highly non-smooth. As a result, S-BO was unable to construct a good surrogate model of the performance function, causing it to struggle to navigate the design space. Meanwhile, the composite function BO algorithms were tasked with learning functions that are comparatively much simpler and were thus better able to predict the behavior of the performance function. This demonstrates that, in addition to the structural system knowledge it provides, the ability to to shift the learning task from a complex function to a set of simpler and easier-to-learn intermediate functions is a key advantage of using composite function BO over S-BO; this feature is part of the reason why MC-BO, OP-BO and BOIS all outperformed S-BO.

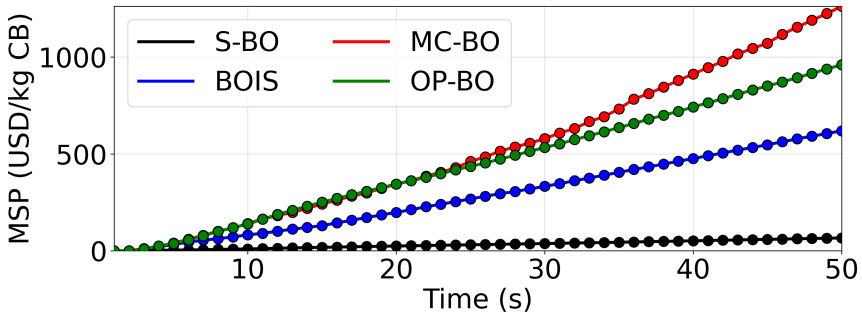

Figure 4.9: Computational intensity of the tested algorithms for solving the photobioreactor design problem measured as the difference between the total execution time and total system sampling time.

Figure 4.9 illustrates the computational time (sampling time not included) required by the algorithms to complete one trial. With an average time of 66 seconds, S-BO was the least intensive algorithm, beating out its closest competitor by almost an order of magnitude. However, its poor performance indicates that, despite being a fast-running algorithm, it is not the best choice for solving the problem. From among the remaining

algorithms, BOIS was the fastest method. At 619 seconds, it outpaced OP-BO by 55% and MC-BO by 104%. Note that in this case study, OP-BO was faster than MC-BO, which is the opposite of what was observed in 4.4.1. This highlights the fact that the comparative intensity of these two methods is variable. At low values of d_x and d_y MC-BO is the more computationally expensive algorithm. However, as the dimensions of x and y increase, the time required to solve the auxiliary problem over the larger space increases to the point that the computational intensity of OP-BO becomes greater. Meanwhile, because BOIS utilizes a set of closed-form expressions to estimate $m_f^\ell(x)$ and $\sigma_f^\ell(x)$, it always requires fewer operations to evaluate its AF than MC-BO and this AF is always optimized over a smaller space than the auxiliary problem in OP-BO. As a result, BOIS is able to maintain a consistent speed advantage over both MC-BO and OP-BO and appears to be a more scalable method for composite function BO.

Using the same approach as in chemical process optimization study, we estimated the values $m_f^\ell(x)$ and $\sigma_f^\ell(x)$ at 500 randomly selected points using BOIS and MC-BO. From these estimates, which are shown in Figure 4.10, we can conclude that BOIS was once again able to generate highly accurate estimates of the statistical moments of the performance function. These results also proved that the adaptive linearization scheme is able to accurately estimate the behavior of a complex function like the MSP without necessarily requiring additional computational time. In fact, the relative differences in the generation times shown on the right side of Figure 4.10 were fairly similar to what was observed in the previous case study.

If we specifically look at the spread of the estimates for $\sigma_f^\ell(x)$ calculated by BOIS vs those calculated by MC-BO when $S = 1000$, we observe that there appears to be a slight bias in the direction that the data points deviate from the center line. We believe that this is likely due to the fact that the intermediate functions are not actually symmetric as is assumed by the GP models (i.e., CB titer and reactor volume cannot be negative). This issue becomes especially poignant when $m_y^\ell(x)$ is near the feasibility bounds of any one of the intermediate functions, as the distribution of $y(x)$ spans values that are not

permissible. We attempted to mitigate this problem by clipping the value of $m_y^\ell(x)$ to the corresponding upper or lower bound when it was outside of its allowable range as shown in (4.10). This provides BOIS with a workable solution as it ensures that only feasible values of $\hat{y}^\ell$ and $\hat{y}_0^\ell$ are passed into $f(x, y)$. However, because MC-BO samples from the distribution to select the values of y, it can still select infeasible values. As a result, we had to clip sampled values of y to $\hat{l}_y$ or $\hat{u}_y$ when they were outside of their permissible range. This caused the calculated density of f to not be symmetric and thus different from the density calculated by BOIS. While this indicates that the Laplace approximation does not provide an accurate representation of the density of f near the bounds of the intermediate functions, we should note the the numerical results we presented indicate that this was not an issue. This demonstrates that, despite assuming an incorrect shape, BOIS was still able to generate performance and informational value estimates that were at least consistent with those of the true underlying distribution. Additionally, it should be noted that at points where $m_y^\ell(x)$ was not near a boundary, which was the case for the majority of those selected, the approximation was still remarkably accurate.

4.5 Conclusions and Future Work

We presented the BOIS framework, a method designed to facilitate the use of composite functions $f(x, y(x))$ in a BO setting. Composite performance functions offer an intuitive way for exploiting structural knowledge (in the form of physics or sparse interconnections) and enable the integration of available white-box models. Additionally, this approach provides significant flexibility in selecting the black-box elements and setting up the corresponding surrogate models (i.e., the inputs can be customized). This can enable a reduction in the dimensionality of $y(x)$ as well as in the inputs of the corresponding GP models, which improves the scalability of the algorithm. Additionally, we can specifically opt to consider intermediates of interest in order to develop surrogate models for these that are highly-refined in regions around any explored optima.

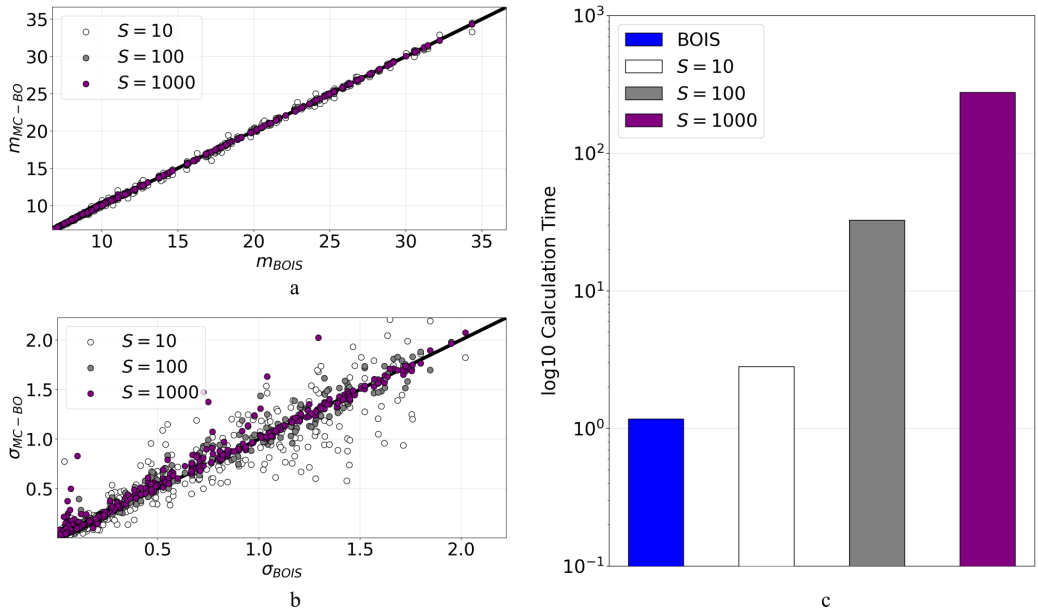

Figure 4.10: Parity plots of the estimates of $m_f^\ell(x)$ (a) and $\sigma_f^\ell(x)$ (b) for the MSP function and $\log_{10}$ of the time required to generate the estimates (c) at 500 points in X using BOIS with $\epsilon = \hat{y} \times 10^{-3}$ and MC-BO with samples sizes $S = 10$, 10^2, and 10^3; the same trained GP model of $y(x)$ was used by both algorithms.

We benchmarked the performance of BOIS against standard Bayesian optimization and two existing composite function BO algorithms (MC-BO and OP-BO) using two case studies centered around chemical engineering optimization and design problems. Our results showed that BOIS significantly outperformed S-BO and was able to match or beat the performances of MC-BO and OP-BO while being significantly less computationally intensive. We also demonstrated that the values of the statistical moments of f estimated by the adaptive linearization scheme we propose are generally very accurate and require significantly less time to compute compared to Monte Carlo estimates of comparable accuracy. However, we did observe a reduction in the accuracy of these predictions in regions near the feasibility limits of the intermediate functions. This is due to the symmetry assumption made by the GPs causing a significant portion of the calculated distribution of $y(x)$ to span non-permissible values in these regions. It should be noted, though, that BO is not limited to using GPs to construct the surrogate model; any probabilistic model can be used. Therefore, we would like to explore the use of alternatives such as

warped GPs [57], RNNs [124], and reference models [115] as potential solutions to this issue. Additionally, we are also interested in investigating the performance of BOIS in high-dimensional systems and in developing alternative types of AFs that can extend the functionality of the algorithm, such as enabling parallelization.

Part II

FROM SEQUENCES TO BATCHES

Chapter 5

NEW PARADIGMS FOR EXPLOITING PARALLEL EXPERIMENTS IN BAYESIAN OPTIMIZATION

This chapter is adapted with permission from González and Zavala. *Computers and Chemical Engineering* 170 (2023): 108110.

5.1 Introduction

The use of high-throughput experimental (HTE) platforms is accelerating scientific discovery in diverse fields such as catalysis [125], pharmaceuticals [126], synthetic biology [127], and chemical engineering [128]. Such platforms permit large numbers of experiments to be executed in parallel, sometimes automatically; this enables the exploration of wider design spaces, reduces time to discovery, and can potentially decrease the use of resources. However, due to the large number of design variables involved, determining optimal conditions manually is often infeasible. As a result, HTE platforms rely on the use of design of experiments (DoE) algorithms, which aim to systematically explore the design space.

Screening is a simple DoE approach in which experiments are performed at points on a discretized grid of the design space [129]; this approach is intuitive but does not scale well with the number of design variables and can ultimately lead to significant waste of

resources (conduct experiments that do not provide significant information). The central aim of advanced DoE approaches is to maximize the value provided by each experiment and ultimately reduce the number of experiments and resources used (e.g., experiment time). The value of an experiment is usually measured either by information content (e.g., reduces model uncertainty) or if it results in a desirable outcome (e.g., improves an economic objective) [104]. A widely used DoE approach that aims to tackle this problem is response surface methodology or RSM [23]. This approach is generally sample-efficient (requires few experiments) but uses second-degree polynomial surrogate models that can fail to accurately capture system trends. In addition, parameters used in the RSM surrogate model are subject to epistemic uncertainty and this uncertainty is not resolved via further experiments [62] (i.e., RSM is an open-loop DoE technique).

Another powerful approach for DoE that aims to maximize value of experiments is Bayesian experimental design [130]. Recently, the machine learning (ML) community has been using variants of this paradigm to conduct closed-loop experimental design [131]. One of the most effective variations of this paradigm is the Bayesian optimization (BO) algorithm [132]; BO has been shown to be sample-efficient and scalable (requires minimal experiments and can explore large design spaces) [73]. BO is widely used in applications such as experimental design, hyper-parameter tuning, and reinforcement learning. Of particular interest is the flexibility of the BO paradigm; it is capable of accommodating both continuous and discrete (e.g., categorical) design variables as well as constraints (which help encode domain knowledge and restrict the design space) [69]. Additionally, BO uses probabilistic surrogate models (e.g. Gaussian process models) which greatly facilitates the quantification of uncertainty and information in different regions of the design space [28]. This feature is particularly useful in guiding experiments where information gain can be as important as performance. BO can also be tuned to emphasize exploration (sampling from regions with high uncertainty) over exploitation (sampling from regions with high economic performance) [76]. This trade-off is achieved by tuning the so-called acquisition function (AF), which is a composite function that captures

uncertainty and performance.

A fundamental caveat of BO is that it is inherently a sequential algorithm (samples a single point in the design space at each iteration), limiting its ability to exploit HTE platforms. Modifications to the BO algorithm have been proposed in the literature to overcome these limitations [133, 134, 135]. Relevant variants include Hyperspace partitioning [136], batch Bayesian optimization [137], NxMCMC [42], and AF optimization over a set of exploratory designs [138]. These parallel BO approaches have been shown to perform better than sequential BO in terms of search time [139]. However, these approaches are limited in the degree of parallelization that can be achieved and can lead to redundant experiments, thus wasting resources and potentially getting trapped in local solutions.

In this chapter, we present a set of new parallel BO paradigms that exploit the structure of the system in order to guide the partitioning of the design space (see Figure 5.1). Our first approach, which we call *level-set partitioning*, decomposes the design space by following the level sets of the performance function. Because the performance function cannot be evaluated (it is unknown), a key feature of this approach is that it leverages a reference function (which can be a low-fidelity model or a physics model) to approximate the level sets and guide the partitioning. Our second approach, called *variable partitioning*, partitions the design space by exploiting partially separable structures that typically result when a system is composed of multiple subsystems (e.g., a chemical process is composed of multiple units). We benchmark the performance of our approaches over sequential BO and state-of-the-art parallel BO variants from the literature using a reactor system. Our results show that the proposed approaches can achieve significant reductions in search time. In addition, we observe improvements in performance values found and in search robustness (sensitivity to initial guess).

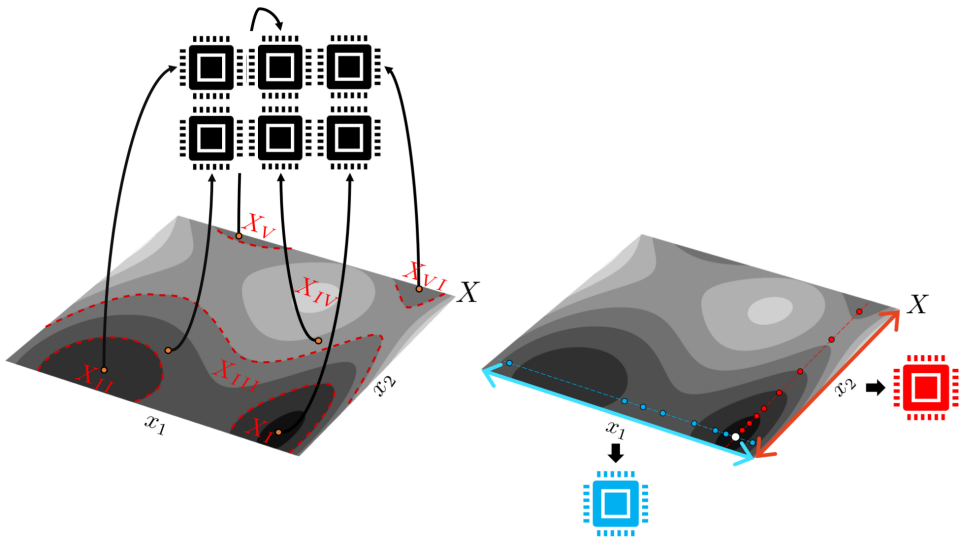

Figure 5.1: Schematic of proposed BO parallelization paradigms using level set partitioning (left) and variable partitioning (right).

5.2 Parallel Bayesian Optimization

The DoE approach most commonly used in HTE platforms is screening. This is a grid-search method that discretizes the design space X into a set of experiments $x_k \in X$, $k \in \mathcal{K} = \{1, ..., K\}$ (we denote this set compactly as $x_\mathcal{K}$). The performance of the system is then evaluated (potentially in parallel) at these points to obtain $f_\mathcal{K}$ and the experiments that achieve the best performance are selected. This screening approach provides good exploratory capabilities, but it is not scalable in the sense that the number of trial experiments needed to cover the design space grows exponentially with the number of design variables, d_x, and with the width of the space X. Moreover, this approach cannot be guaranteed to find an optimal solution. The S-BO framework provides a more sample-efficient (typically require few experiments to identify optimal performance) and closed-loop alternative for DoE, but it is inherently sequential. Several approaches for proposing multiple experiments per cycle have been developed, each with varying degrees of complexity and sample efficiency. These parallel BO variants can grouped into four main parallelization

paradigms: AF optimization over a set of hyperparameters, design space partitioning, fantasy sampling, and AF optimization over a batch of points. The most used approach is the NxMCMC method, which falls under the fantasy sampling paradigm, and is used in popular BO packages such as Spearmint [140]. We now proceed to discuss the specifics of different existing algorithms that are based on these parallelization paradigms.

5.2.1 *Hyperparameter Sampling Algorithm (HP-BO)*

The hyperparameter sampling algorithm (which we refer to as HP-BO) identifies a new *batch* of experiments $x_{\mathcal{K}}^{\ell+1}$ by optimizing the acquisition function $\mathcal{AF}_f^{\ell}(x;\kappa_k)$ using K different values of the exploratory hyperparameter κ_k, $k \in \mathcal{K}$. In other words, we obtain the new experiments by solving:

$$x_k^{\ell+1} \leftarrow \underset{x}{\operatorname{argmin}} \quad \mathcal{AF}_f^{\ell}(x;\kappa_k) \tag{5.1a}$$

$$\text{s.t.} \quad x \in X \tag{5.1b}$$

for $k \in \mathcal{K}$. The hyperparameter values κ_k, can be selected manually or sampled from a distribution. In the approach that we consider here, we generate the values by sampling from an exponential distribution, $\kappa \sim \mathcal{E}(\lambda)$, with rate parameter $\lambda = 1$ as shown in [138]. Once the batch of experiments has been determined, we can evaluate their performance (in parallel) to obtain $f_{\mathcal{K}}^{\ell+1} = f(x_{\mathcal{K}}^{\ell+1})$ and update the dataset $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^{\ell} \cup \left\{x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1}\right\}$. The updated dataset is then used to train a new GP, $\mathcal{GP}_f^{\ell+1}$, which is used to form a new acquisition function, $\mathcal{AF}_f^{\ell+1}$, and to compute a next batch of experiments $x_{\mathcal{K}}^{\ell+2}$. The process is repeated over multiple cycles. The pseudocode for implementing HP-BO is shown in Algorithm 7.

The main advantages of HP-BO are that it is easy to implement, that it is highly parallelizable, and that it allows for the selection of experiments under various exploration and exploitation settings (eliminating the need for tuning κ). The effect of the hyperparame-

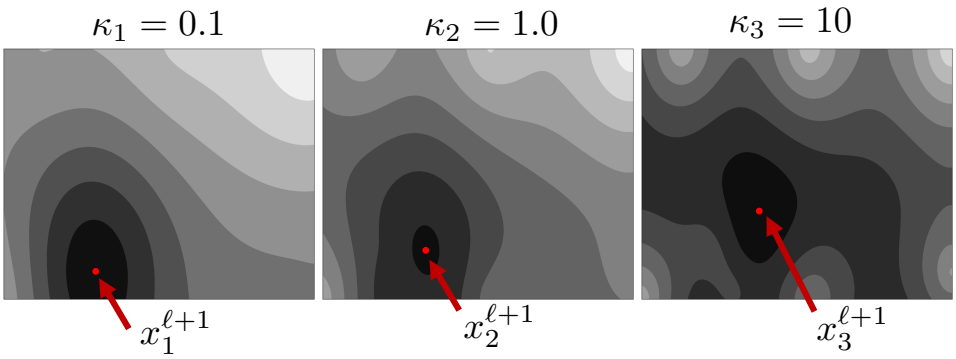

Figure 5.2: HP-BO optimizes the AF for a set of hyperparameters $\kappa_k,\ k \in \mathcal{K}$ to obtain experiments $x_k,\ k \in \mathcal{K}$ that can be evaluated in parallel. Here, we show an example with $K = 3$.

ter κ on the AF is highlighted in Figure 5.2. However, in this approach it is not possible to prevent the proposal of redundant experiments and, as the algorithm converges, the suggested experiments can begin to cluster in a region of low uncertainty (this can cause the algorithm to get trapped at local solutions). The HP-BO algorithm can be easily be extended to incorporate a reference function g. In this case, the GP learns the residual instead of the performance function.

Algorithm 7: Hyperparameter Sampling BO (HP-BO)

Given κ_k, K, L, and $\mathcal{D}^\ell$;
Train GP $\mathcal{GP}_f^\ell$ using $\mathcal{D}^\ell$ and obtain $\mathcal{AF}_f^\ell(x;\kappa_k)$, $k \in \mathcal{K}$;
for $\ell = 1, 2, ..., L$ **do**
  for $k \in \mathcal{K}$ **do**
    Compute experiment $x_k^{\ell+1} \leftarrow \text{argmin}_x\, \mathcal{AF}_f^\ell(x;\kappa_k)$ s.t. $x \in X$;
    Evaluate performance at $x_k^{\ell+1}$ to obtain $f_k^{\ell+1}$;
  end
  Update data $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^\ell \cup \left\{x_\mathcal{K}^{\ell+1}, f_\mathcal{K}^{\ell+1}\right\}$;
  Train GP using $\mathcal{D}^{\ell+1}$ to obtain $\mathcal{GP}_f^{\ell+1}$ and $\mathcal{AF}_f^{\ell+1}(x;\kappa_k)$, $k \in \mathcal{K}$;
end

5.2.2 *HyperSpace Partitioning Algorithm (HS-BO)*

The HyperSpace partitioning algorithm (which we refer to as HS-BO) was presented in [136]. This parallelizes BO by partitioning the design space X into K equally-sized blocks $X_k \subseteq X$, $k \in \mathcal{K}$. Importantly, this approach does not use a surrogate GP model over the entire design space. Instead, a separate GP model is constructed at each partition X_k and is updated using only information collected within this partition. Specifically, each partition $k \in \mathcal{K}$ builds a GP, $\mathcal{GP}^{\ell}_{f,k} \sim \mathcal{N}(m^{\ell}_{f,k}(x), (\sigma^{\ell}_{f,k}(x))^2)$, that is used to construct an acquisition function, $\mathcal{AF}^{\ell}_{f,k}(x;\kappa)$. With this, we can obtain a set of new experiments by solving the following subproblems:

$$x_k^{\ell+1} \leftarrow \underset{x}{\operatorname{argmin}} \quad \mathcal{AF}^{\ell}_{f,k}(x;\kappa) \tag{5.2a}$$

$$\text{s.t.} \quad x \in X_k \tag{5.2b}$$

for $k \in \mathcal{K}$. The domain partitions can also be constructed to have a certain degree of overlap. Specifically, an overlap hyperparameter $\phi \in [0,1]$ is introduced to allow the partitions to share a fraction of the design space. A value of $\phi = 0$ indicates that the partitions are completely separate, while a value of $\phi = 1$ indicates that $X_k = X$ for $k \in \mathcal{K}$ (the partitions are copies of the full design space); this is shown in Figure 5.3. The overlap hyperparameter provides a communication window, allowing the GP model of a given partition to observe system behavior beyond its prescribed partition (share information with other partitions). This, however, introduces a fundamental trade-off. From a parallelization perspective it is desirable that ϕ is small, but from a convergence perspective (e.g., reducing number of iterations) it might be desirable that ϕ is large. A similar trade-off is observed as one decreases or increases the number of partitions, K. As such, there is a complex interplay between the hyperparameters K and ϕ, and these need to be tuned. Similar types of trade-offs have been observed in the context of overlapping decomposition approaches for optimization problems defined over graph domains [141].

In the implementation reported in [136], the number of partitions is set to $K = 2^{d_x}$.

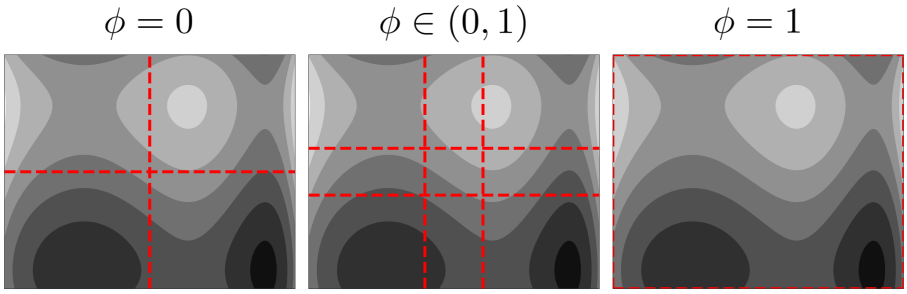

Figure 5.3: HS-BO partitions the domain X into $K = 2^{d_x}$ subdomains and runs a separate instance of S-BO within each partition. A hyperparameter ϕ is introduced to define the degree of overlap in the partitions (the overlapping region aims to share information across subdomains). When $\phi = 0$ there is no overlap between the partitions and when $\phi = 1$ we have that all partitions are the entire domain X.

The HS-BO approach, summarized in Algorithm 8, is easy to implement, is scalable to high-dimensional spaces, and enables the development of GP models for systems that may exhibit different behaviors at various regions of the design space (compared to using a single GP model that captures the entire design space). HS-BO also eliminates redundant sampling by forcing the algorithm to sample from distinct regions of the design space. This results in a more thorough search, which improves the probability that the global solution will be located. Domain partitioning is, in fact, a paradigm widely used in global optimization. A limitation of HS-BO is that the partitions are boxes of equal size (this can limit capturing complex shapes of the performance function); moreover, one needs to tune K and ϕ. In principle, it might be possible to extend this approach to account for automatic tuning and adaptive partitions, but this would require much more difficult implementations that carefully trade-off parallelization and convergence (this is left as a topic of future work). The HS-BO approach can also be easily executed using a reference model by learning the residual instead of the performance function.

Algorithm 8: HyperSpace Partitioning (HS-BO)

Given κ, K, L, ϕ, and $\mathcal{D}^\ell$;
Partition X into $X_k \subseteq X$, $k \in \mathcal{K}$ with overlap ϕ;
Split initial data into domains $\mathcal{D}_k^\ell$, $k \in \mathcal{K}$;
for $k \in \mathcal{K}$ **do**
  Train GP $\mathcal{GP}_{f,k}^\ell$ in X_k using $\mathcal{D}_k^\ell$ and obtain $\mathcal{AF}_{f,k}^\ell$;
end
for $\ell = 1, 2, ..., L$ **do**
  for $k \in \mathcal{K}$ **do**
    Compute experiment $x_k^{\ell+1} \leftarrow \text{argmin}_x \, \mathcal{AF}_{f,k}^\ell(x; \kappa)$ s.t. $x \in X_k$;
    Evaluate performance at $x_k^{\ell+1}$ to obtain $f_k^{\ell+1}$;
    Update data $\mathcal{D}_k^{\ell+1} \leftarrow \mathcal{D}_k^\ell \cup \left\{ x_k^{\ell+1}, f_k^{\ell+1} \right\}$;
    Train GP using $\mathcal{D}_k^{\ell+1}$ to obtain $\mathcal{GP}_{f,k}^{\ell+1}$ and $\mathcal{AF}_{f,k}^{\ell+1}$;
  end
end

5.2.3 *N×MCMC Algorithm (MC-BO)*

The N×MCMC (N times Markov Chain Monte Carlo) algorithm is a popular approach used for proposing multiple experiments [53]. We refer to this approach simply as MC-BO. Assume that we currently have a set of experimental data $\mathcal{D}^\ell$; we use this to generate the GP $\mathcal{GP}_f^\ell$, acquisition function $\mathcal{AF}_f^\ell$, and to compute the next experiment $x_k^{\ell+1}$ for $k = 1$. Our goal is now to obtain the remaining set of experiments $x_k^{\ell+1}$, $k = 2, ..., K$ that we can use to evaluate performance. To do so, we consider a set of *fantasy* predictions obtained by generating S samples, $\hat{f}_s^\ell(x_k^{\ell+1})$, $s \in \mathcal{S}$, from the GP. Here, the term *fantasy* alludes to the fact that the evaluation of performance is based on the GP (and not on the actual system). The fantasy data has the goal of creating an approximate AF. Specifically, for each sample s, we generate a dataset $\hat{\mathcal{D}}_s = \{x_\mathcal{F}^{\ell+1}, \hat{f}_s^\ell(x_\mathcal{F}^{\ell+1})\}$, $\mathcal{F} = \{1, ..., k\}$, and this is appended to the existing dataset $\mathcal{D}^\ell \cup \hat{\mathcal{D}}_s$. This data is then used to obtain a GP $\mathcal{GP}_{f,s}$ and associated acquisition function $\hat{\mathcal{AF}}_{f,s}$. The AFs for all samples $s \in \mathcal{S}$ are collected

and used to compute the mean AF:

$$\overline{\mathcal{AF}}_f(x;\kappa) = \frac{1}{S}\sum_{s\in\mathcal{S}} \hat{\mathcal{AF}}_{f,s}(x;\kappa),\ x \in X. \tag{5.3}$$

The new experiment is then obtained by solving:

$$x_{k+1}^{\ell+1} \leftarrow \underset{x}{\text{argmin}}\ \ \overline{\mathcal{AF}}_f(x;\kappa) \tag{5.4a}$$

$$\text{s.t.}\ \ x \in X \tag{5.4b}$$

To generate another experiment, we repeat the sampling process (using a new set of S samples) and create a different mean acquisition function $\overline{\mathcal{AF}}_f$, and we minimize this to obtain $x_{k+2}^{\ell+1}$. The sampling process is repeated until we obtain the full batch of new experiments, $x_{\mathcal{K}}^{\ell+1}$. Once we have these, we evaluate the performance function at these points (in parallel) to obtain the dataset $\{x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1}\}$, which we append to the data collection $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^{\ell} \cup \{x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1}\}$. We use this new data to re-train the GP of the performance function and repeat the process. The framework for the MC-BO algorithm is presented in Algorithm 9.

The MC-BO algorithm has proven to be an effective parallel extension of the BO algorithm. However, computing the mean AF requires significant computational time (as the GP model needs to be retrained continuously). This algorithm also has the tendency to propose experiments that are close in the design space, especially when it begins to converge. This does not necessarily pose an issue if the algorithm is converging to the global solution. However, if the solution approached is local, this behavior can limit the ability of the algorithm to escape this region. The MC-BO approach can be executed using a reference model by simply learning the residual instead of the performance function.

Algorithm 9: N×MCMC BO Algorithm (MC-BO)

Given κ, K, S, L, and $\mathcal{D}^\ell$;
Train GP $\mathcal{GP}_f^\ell$ using initial dataset $\mathcal{D}^\ell$ and obtain $\mathcal{AF}_f^\ell$;
for $\ell = 1, 2, ..., L$ **do**
  Compute $x_k^{\ell+1} \leftarrow \text{argmin}_x \, \mathcal{AF}_f^\ell(x;\kappa)$ s.t. $x \in X$ for $k = 1$;
  for $k = 1, ..., K-1$ **do**
    for $s \in \mathcal{S}$ **do**
      Generate fantasy dataset $\hat{\mathcal{D}}_s = \left\{ x_{\mathcal{F}}^{\ell+1}, \hat{f}_s^\ell(x_{\mathcal{F}}^{\ell+1}) \right\}$, $\mathcal{F} = \{1, ..., k\}$;
      Use dataset $\mathcal{D}^\ell \cup \hat{\mathcal{D}}_s$ to train GP $\mathcal{GP}_{f,s}$ and obtain $\hat{\mathcal{AF}}_{f,s}(x;\kappa)$;
    end
    Set $\overline{\mathcal{AF}}_f(x;\kappa) \leftarrow \frac{1}{S} \sum_{s \in \mathcal{S}} \hat{\mathcal{AF}}_{f,s}(x;\kappa)$;
    Compute experiment $x_{k+1}^{\ell+1} \leftarrow \text{argmin}_x \, \overline{\mathcal{AF}}_f(x;\kappa)$ s.t. $x \in X$;
  end
  for $k \in \mathcal{K}$ **do**
    Evaluate performance at $x_k^{\ell+1}$ to obtain $f_k^{\ell+1}$;
  end
  Update data $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^\ell \cup \left\{ x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1} \right\}$;
  Train GP using $\mathcal{D}^{\ell+1}$ to obtain $\mathcal{GP}_f^{\ell+1}$ and $\mathcal{AF}_f^{\ell+1}$;
end

5.2.4 *Batch Bayesian Optimization Algorithm (q-BO)*

The q-BO or batch Bayesian optimization algorithm uses a multipoint acquisition function, $\mathcal{AF}_q^\ell(x_\mathcal{K};\kappa)$, like the q-LCB presented in [137], to select a batch of q experiments that can be run in parallel. Unlike most adaptations of BO where the AF is optimized over a single point, the q-LCB is optimized over a set of q points. By selecting the experiments in a batch rather than independently, as in HP-BO, or sequentially, as in MC-BO, q-LCB is able to measure the correlation between the suggested sample locations, allowing it to more easily avoid the issue of redundant sampling. Given a desired batch size, the value of a particular set of experiments $x_\mathcal{K}$ is measured according to:

$$\mathcal{AF}_q^\ell(x_\mathcal{K};\kappa) = \frac{1}{S} \sum_{s=1}^{S} \max \left(m_q^\ell(x_\mathcal{K}) - \kappa \cdot |A_q^\ell(x_\mathcal{K}) z_s| \right) \tag{5.5}$$

where $m_q^\ell(x_\mathcal{K}) \in \mathbb{R}^q$ and $A_q^\ell(x_\mathcal{K}) \in \mathbb{R}^{q\times q}$ are the GP mean and Cholesky factor of the GP covariance ($AA^T = \Sigma$) at a batch of points $x_\mathcal{K}$ respectively, $z_s \in \mathbb{R}^q$ is a random variable with $z_s \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and $|\cdot|$ is the absolute value (element-wise) operator. The new batch is then selected by solving:

$$x_\mathcal{K}^{\ell+1} \leftarrow \underset{x_\mathcal{K}}{\operatorname{argmin}} \quad \mathcal{AF}_q^\ell(x_\mathcal{K}; \kappa) \tag{5.6a}$$

$$\text{s.t.} \quad x_\mathcal{K} \in X \tag{5.6b}$$

where, again, the optimization is done over the entire batch of q points in $x_\mathcal{K}$. The experiments are then run (in parallel) and the collected performance measurements are used to update the dataset $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^\ell \cup \{x_\mathcal{K}^{\ell+1}, f_\mathcal{K}^{\ell+1}\}$. This data is used to retrain the model which enables the selection of the next batch of experiments. The pseudocode for q-BO is summarized in Algorithm 10.

The q-BO algorithm has proven to be especially popular in the multi-objective optimization setting and is, in principle, not difficult to implement. However, unlike single-point AFs, multi-point AFs do not have a closed-form representation. As a result, constructing and optimizing $\mathcal{AF}_q^\ell(x_\mathcal{K}; \kappa)$ requires the use of numerical methods like Monte Carlo, as seen in (5.5), making this an intensive process, especially as q increases. Additionally, while the use of the Cholesky factor ensures that the algorithm cannot select redundant experiments, safeguards must be placed when constructing the AF optimization problem to ensure that the optimizer cannot select identical points as this will result in the covariance matrix $\Sigma(x_\mathcal{K})$ being singular and cause the optimizer to fail. In this work, that safeguard was implemented as a tolerance value, ϵ, that set the minimum allowable distance between any two points within $x_\mathcal{K}$. We should note that while using this strategy, we observed that q-BO can and does occasionally select points that are within ϵ of each other. This can be practically as undesirable as redundant sampling, depending on the value of ϵ. A reference model can also be easily incorporated into the q-BO approach by having the algorithm learn the residual instead of the performance function.

Algorithm 10: Batch Bayesian Optimization Algorithm (q-BO)

Given κ, K, S, L, and $\mathcal{D}^{\ell}$;
Train GP $\mathcal{GP}_f^{\ell}$ using initial dataset $\mathcal{D}^{\ell}$ and obtain $\mathcal{AF}_q^{\ell}$;
for $\ell = 1, 2, ..., L$ **do**
  Compute $x_{\mathcal{K}}^{\ell+1} \leftarrow \text{argmin}_{x_{\mathcal{K}}} \mathcal{AF}_q^{\ell}(x_{\mathcal{K}}; \kappa)$ s.t. $x_{\mathcal{K}} \in X$;
  for $k \in \mathcal{K}$ **do**
    Evaluate performance at $x_k^{\ell+1}$ to obtain $f_k^{\ell+1}$;
  end
  Update data $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^{\ell} \cup \left\{ x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1} \right\}$;
  Train GP using $\mathcal{D}^{\ell+1}$ to obtain $\mathcal{GP}_f^{\ell+1}$ and $\mathcal{AF}_q^{\ell+1}$;
end

5.3 Parallel Bayesian Optimization using Informed Partitioning

We propose new paradigms for parallel BO that conduct informed partitioning of the design space. Specifically, we propose a domain partitioning approach (analogous to HS-BO) that conducts partitioning by following the level sets of the performance function. Because the performance function cannot be easily evaluated, we use a reference model to guide the partitioning. This approach allows us to leverage expert or physical knowledge, which might highlight certain regions of the design space that are promising or non-promising (and with this prioritize). We also propose a variable partitioning approach that aims to exploit partially separable structures that are commonly found in complex systems. Specifically, in these systems the performance function is composed of a collection of functions for different subsystems (but the functions are coupled together via common variables). The key idea is then to search the design space by following this separable structure, while sharing information between the coupling variables. We refer to these paradigms as level-set partitioning BO (LS-BO) and variable partitioning BO (VP-BO).

5.3.1 *Level-Set Partitioning Algorithm (LS-BO)*

LS-BO uses domain partitions of the design space X that follow the levels sets of the reference function g. We recall that the α-level set (sublevel) of this scalar function is:

$$\tilde{X}(\alpha) = \{x \in X \,|\, g(x) \leq \alpha\} \subseteq X \tag{5.7}$$

for any $\alpha \in \mathbb{R}$. We now note that solving the AF optimization problem:

$$\min_{x} \quad \mathcal{AF}_f^{\ell}(x) \tag{5.8a}$$

$$\text{s.t.} \quad x \in \tilde{X}(\alpha) \tag{5.8b}$$

would force the BO algorithm to restrict the search over a restricted subdomain $\tilde{X}(\alpha)$. However, solving this optimization problem can be difficult if g does not have an explicit algebraic form (e.g., low-fidelity simulator) or has a complex form (e.g., physics model). To overcome this limitation, we construct a GP model $\hat{g}$ of g to define the approximate level set:

$$\hat{X}(\alpha) = \{x \in X | \hat{g}(x) \leq \alpha\}. \tag{5.9}$$

We use the previous basic observations to derive our domain partitioning approach; we construct a set of domain partitions $\hat{X}_k \subseteq X$, $k \in \mathcal{K}$ by following different level-sets of the function. Specifically, we construct the subdomains:

$$\hat{X}_k = \{x \in X | \alpha_k \leq \hat{g}(x) \leq \alpha_{k+1}\},\ k \in \mathcal{K}, \tag{5.10}$$

We note that the subdomains are upper and lower bounded in order to obtain non-overlapping partitions. The level set thresholds, α_k, are set by discretizing the range of $\hat{g}(x)$. The simplest method for generating the subdomains is to uniformly discretize the

interval between the extreme lower and upper values of the reference model as follows:

$$\alpha_k = \alpha_1 + (k-1)\Delta,\ k = 2, ..., K \tag{5.11}$$

where $\Delta = \frac{\alpha_{K+1}-\alpha_1}{K}$, $\alpha_1 = \min_{x\in X} \hat{g}(x)$, and $\alpha_{K+1} = \max_{x\in X} \hat{g}(x)$. In cases where additional specificity is desired, the partitions can be further adapted by setting the intervals according to various factors such as a focus on a particular region of the design space, the desired level of exploration vs exploitation, the level of confidence in the quality of the reference model, the geometry of $\hat{g}(x)$, and so on. The partitioning approach is illustrated in Figure 5.4.

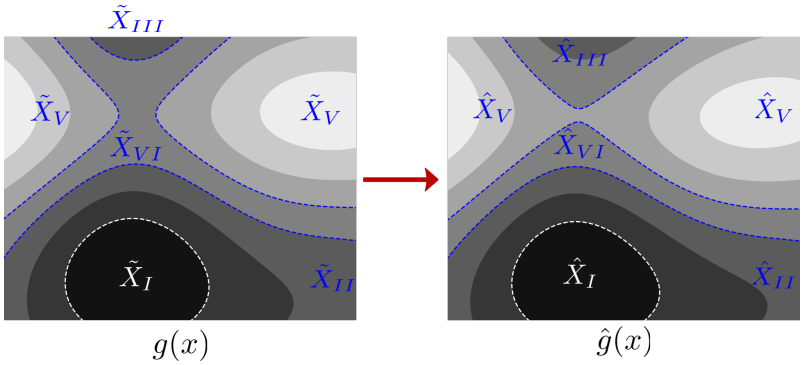

Figure 5.4: Level set partitioning (LS-BO) uses the α-level sets of the reference g to split X into subdomains $\tilde{X}_k$, $k \in \mathcal{K}$. Depending on the complexity of g, enforcing level set constraints in the AF optimization problem can be difficult; therefore, the level sets are approximated using the surrogate model $\hat{g}$.

As in S-BO, we begin with dataset $\mathcal{D}^\ell$, which we use to build the GP $\mathcal{GP}_f^\ell$ and the acquisition function $\mathcal{AF}_f^\ell$. We then obtain a new set of suggested experiments, $x_{\mathcal{K}}^{\ell+1}$, by solving the following collection of optimization problems:

$$x_k^{\ell+1} \leftarrow \underset{x}{\text{argmin}} \quad \mathcal{AF}_f^\ell(x) \tag{5.12a}$$

$$\text{s.t.} \quad x \in \hat{X}_k \tag{5.12b}$$

for $k \in \mathcal{K}$. Using the new experiments $x_{\mathcal{K}}^{\ell+1}$ we evaluate system performance $f_{\mathcal{K}}^{\ell+1}$ (in

parallel) and we append the collected data to the dataset $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^{\ell} \cup \{x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1}\}$. The new dataset is used to update the GP $\mathcal{GP}_f^{\ell+1}$ and the acquisition function $\mathcal{AF}_f^{\ell+1}$. A summary of this procedure is presented in Algorithm 11.

Algorithm 11: Level-Set Partitioning BO (LS-BO)

Given κ, g, L, K and $\mathcal{D}^{\ell}$;
Build surrogate $\hat{g}$ of g;
Construct partitions $\hat{X}_k \subseteq X$, $k \in \mathcal{K}$ using level sets of $\hat{g}$;
Train GP $\mathcal{GP}_f^{\ell}$ with initial dataset $\mathcal{D}^{\ell}$ and obtain $\mathcal{AF}_f^{\ell}$;
for $\ell = 1, 2, ..., L$ **do**
  for $k \in \mathcal{K}$ **do**
    Compute experiment $x_k^{\ell+1} \leftarrow \text{argmin}_x \, \mathcal{AF}_f^{\ell}(x; \kappa)$ s.t. $x \in \hat{X}_k$;
    Evaluate performance at $x_k^{\ell+1}$ to obtain $f_k^{\ell+1}$;
  end
  Update $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^{\ell} \cup \left\{x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1}\right\}$;
  Retrain GP using $\mathcal{D}^{\ell+1}$ to obtain $\mathcal{GP}_f^{\ell+1}$ and $\mathcal{AF}_f^{\ell+1}$;
end

It is important to highlight that the LS-BO approach that we propose uses a GP model of the performance function and an AF that are defined over the entire design space X; this approach thus differs from HS-BO (which uses a different GP and AF in each partition X_k). Moreover, we note that the partitioning of the space follows the level sets of the reference function, and this allows us to concentrate experiments over regions that are most promising. The proposed LS-BO approach can also be implemented in such a way that the reference function is exploited to learn the residual (as opposed to learning the performance function). As such, we can leverage the reference function for constructing the domain partitions and for guiding the search.

5.3.2 *Variable Partitioning Algorithm (VP-BO)*

Many physical systems are typically composed of individual components that are partially interconnected (e.g., they are modular). For instance, a chemical process includes units (e.g., reactors and separations) that are interconnected, and the performance of each

unit contributes to the total system performance. Moreover, the performance of each unit is typically strongly affected by the unit variables and less affected by variables of other units. This partially separable structure can be captured as the following optimization problem:

$$\min_{x} \quad \sum_{k \in \mathcal{K}} f_k(x_k; \mathbf{x}_{-k}) \tag{5.13a}$$

$$\text{s.t.} \quad x \in X \tag{5.13b}$$

where $f_k : X \to \mathbb{R}$ is the performance contribution of component k. The entire set of decision variables is split into K subsets as $x = \{x_1, x_2, ..., x_K\}$, and we define $\mathbf{x}_{-k} = x \setminus \{x_k\}$ (entire set of variables that does not include x_k). We should note that the variable partitions should be non-overlapping subsets (i.e., $x_i \cap x_{j \neq i} = \{0\}, i, j \in \mathcal{K}$). Additionally, we assume that that performance function can be decomposed as $f = \sum_{k \in \mathcal{K}} f_k$.

VP-BO follows a Gauss-Seidel paradigm. Assume we have an initial set of data $\mathcal{D}^\ell = \{x^\ell_{\mathcal{K}}, f^\ell_{\mathcal{K}}\}$ and that we measure $f_1, ..., f_K$ in each experimental module so that $f^\ell_{\mathcal{K}} \in \mathbb{R}^{\ell \times K}$ rather than $\mathbb{R}^\ell$ as in the previous algorithms; note that this means that f^ℓ_k corresponds to the k^{th} column of $f^\ell_{\mathcal{K}}$. We optimize the individual performance of subcomponent k using the variables x_k, while keeping the rest of the variables $\mathbf{x}_{-k}$ constant (to the values of the previous iteration ℓ):

$$\min_{x_k} \quad f_k(x_k; \mathbf{x}^\ell_{-k}) \tag{5.14a}$$

$$\text{s.t.} \quad (x_k; \mathbf{x}^\ell_{-k}) \in X \tag{5.14b}$$

for $k \in \mathcal{K}$. Accordingly, we decompose the AF optimization problem into the subprob-

lems:

$$x_k^{\ell+1} \leftarrow \underset{x_k}{\operatorname{argmin}} \quad \mathcal{AF}_{f_k}^{\ell}(x_k; \mathbf{x}_{-k}^{\ell}, \kappa) \tag{5.15a}$$

$$\text{s.t.} \quad (x_k; \mathbf{x}_{-k}^{\ell}) \in X. \tag{5.15b}$$

for $k \in \mathcal{K}$. Here, $\mathcal{AF}_{f_k}^{\ell}$ is the acquisition function of component k, which is built using the surrogate $\mathcal{GP}_{f_k}^{\ell}$ of the performance f_k. Moreover, $\mathbf{x}_{-k}^{\ell}$ is the value of the variables not in partition k at the current iteration (which are held fixed when optimizing the $\mathcal{AF}_{f_k}^{\ell}$).

We partition the variables by leveraging the reference model g. Specifically, we use this reference to identify which variables have the most impact on individual components of the system; this can be done in various ways. The most straightforward method would be via inspection using a combination of information provided by the reference model and any available expert knowledge over the importance of the various inputs on the subsystems. If such information is not available, $g(x)$ can instead be analyzed with an appropriate feature importance technique, such as sparse principal components analysis (SPCA) [142], automatic relevance determination (ARD) [143], model class reliance (MCR) [144], etc., to determine the appropriate variable-subsystem pairings. Because the partitions must not overlap, the results of this analysis should be checked for instances where an input is paired with multiple subsystems. If this occurs, we recommend that the input in question be paired with the subsystem where it has the highest relative importance. The pseudocode for implementing VP-BO is shown in Algorithm 12.

One of the advantages of the VP-BO approach is that the AF optimization over each partition only uses a subset of variables; this can significantly reduce the computational time of this step. Moreover, this approach is amenable for implementation in a distributed manner (e.g., each subsystem of the network runs its own separate BO algorithm). The VP-BO approach (and the LS-BO approach) also takes system-specific behavior into account when developing partitions (informed by the reference model). As we will show in the next section, the use of prior knowledge can lead to significant reductions in com-

putational time and in the number of experiments performed. Moreover, we will see that such knowledge can help identify solutions and construct surrogate models of higher quality. VP-BO can also be implemented in such a way that reference model is also used to guide the construction of the performance function (by learning the residual instead of the performance function). We also highlight that the VP-BO approach proposed is implemented in a way that each partition has its own GP model and AF. However, it is also possible to implement this approach by building a central GP and AF that are optimized in each partition using a different set of variables.

Algorithm 12: Variable Partitioning BO (VP-BO)

Given κ, g, K, L, and $\mathcal{D}^\ell$;
Decompose $f(x)$ into $f_k(x_k, \mathbf{x}_{-k})$ for $k \in \mathcal{K}$;
Use $\mathcal{D}^\ell$ to train GPs $\mathcal{GP}^\ell_{f_k}$ and obtain $\mathcal{AF}^\ell_{f_k}$, $k \in \mathcal{K}$;
for $\ell = 1, 2, ..., L$ **do**
  for $k \in \mathcal{K}$ **do**
    Compute experiment $x_k^{\ell+1} \leftarrow \text{argmin}_{x_k} \mathcal{AF}^\ell_{f_k}\left(x_k; \mathbf{x}^\ell_{-k}, \kappa\right)$ s.t. $\left(x_k, \mathbf{x}^\ell_{-k}\right) \in X$;
    Evaluate $f_1, ..., f_K$ at $x_k^{\ell+1}$ to obtain $f_{\mathcal{K}}^{\ell+1}[k, :]$;
  end
  for $k \in \mathcal{K}$ **do**
    $\mathbf{x}_{-k}^{\ell+1} \leftarrow \text{argmin}_{\mathbf{x}_{-k}} f_{\mathcal{K}}^{\ell+1}[:, k]$;
  end
  Update $\mathcal{D}^{\ell+1} \leftarrow \mathcal{D}^\ell \cup \left\{x_{\mathcal{K}}^{\ell+1}, f_{\mathcal{K}}^{\ell+1}\right\}$;
  Retrain GPs using $\mathcal{D}^{\ell+1}$ to obtain $\mathcal{GP}^{\ell+1}_{f_k}$ and $\mathcal{AF}^{\ell+1}_{f_k}$ $k \in \mathcal{K}$
end

5.4 Numerical Case Studies

We now present numerical results using the different BO strategies discussed; our goal is to demonstrate that the parallel BO approaches proposed provide significant advantages over S-BO and over other state-of-the-art parallel approaches. Our study simulates the performance of a pair of reactors connected in series; the operating cost of this system is a complex function of the operating temperatures. The detailed physical model used

to simulate the performance of the system is discussed in Appendix A.1.3. To guide our partitioning approaches, we developed a reference model that approximates the physical model. All data and code needed to reproduce the results can be found at https://github.com/zavalab/bayesianopt.

The optimization problem that we aim to solve with BO can be written as:

$$\min_{T_1,T_2} f(T_1,T_2) = f_1(T_1,T_2) + f_2(T_1,T_2) \tag{5.16a}$$

$$\text{s.t.} \quad (T_1,T_2) \in \mathcal{T} \tag{5.16b}$$

Figure 5.5 shows the performance function $f(T_1,T_2)$ over the box domain $\mathcal{T} = [303,423]^2$. The performance function is nonconvex and contains three minima, with local solutions at $(T_1,T_2) = (423,340)$ and $(T_1,T_2) = (423,423)$ and a global solution at $(T_1,T_2) = (333,322)$.

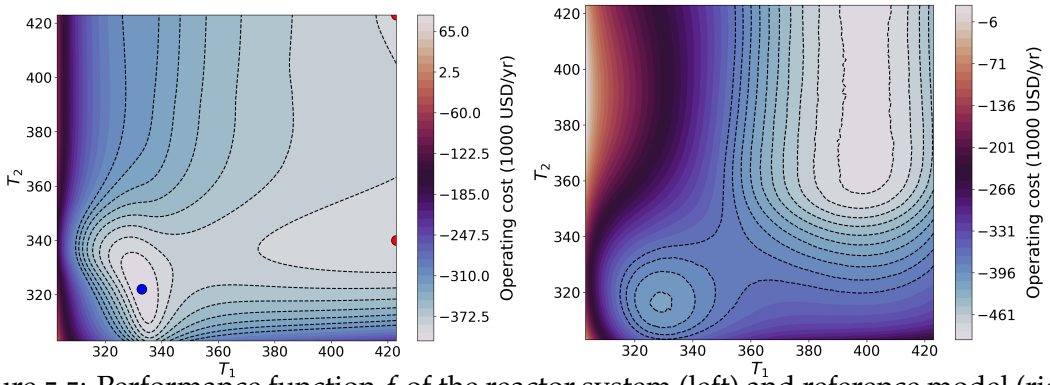

Figure 5.5: Performance function f of the reactor system (left) and reference model (right). Note that the reference model captures the overall (coarse) structure of the performance function but misses some finer details.

The reference model g was derived from a simplified physical model (see the Appendix). However, this model would be difficult to incorporate directly in the AF formulation for the LS-BO and VP-BO approaches because it involves a complex set of algebraic equations. As such, we approximated this model using a GP, $\hat{g}$, and used this as the reference. Figure 5.6 illustrates that the GP $\hat{g}$ is virtually indistinguishable from the sim-

plified physical model g; thus, we can safely use this to guide our search and to guide our partitioning approaches.

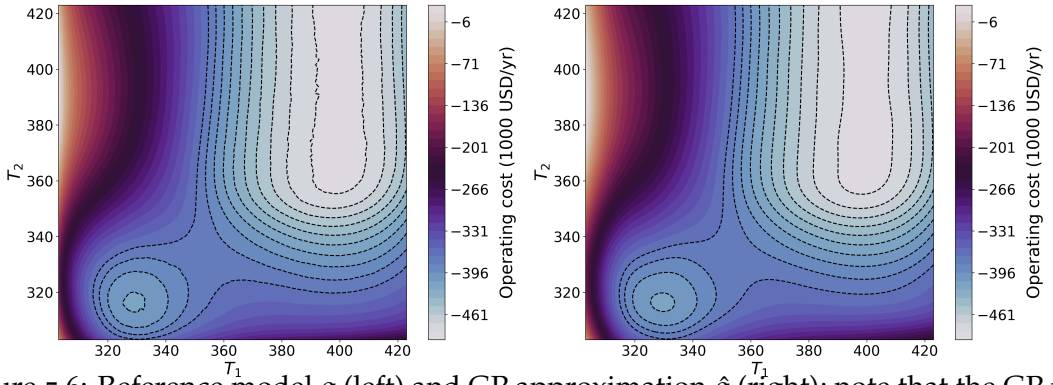

Figure 5.6: Reference model g (left) and GP approximation $\hat{g}$ (right); note that the GP provides an accurate representation and can thus be used to guide partitioning approaches.

The HS-BO algorithm was restricted to $2^2 = 4$ partitions when dealing with a 2D design space. As such, and in order to achieve fair comparisons, we limited the number of parallel experiments performed by MC-BO, HP-BO, q-BO, and LS-BO to 4. The VP-BO algorithm was run using 2 partitions (one for each reactor). All algorithms were implemented in Python 3.7 and the GP modeling was done using the `gaussian_process` package in Scikit-learn. Specifically, we used the built-in `Matern` method as the kernel function. This selection was motivated by the ability of the Mátern kernel to control the smoothness of the resultant function making it highly flexible and capable of accurately modeling systems that exhibit significant nonlinearity and non-smoothness. We set the smoothness parameter $\nu = 2.5$, which tends to be the standard choice. At every iteration, the optimal values of the kernel's hyperparameters, the characteristic length scales l, were updated using the package's built-in optimizer that sets l by solving a log-marginal-likelihood (LML) problem. A more detailed description of the `gaussian_process` package can be found in [100]. The optimization of the AF was done in Scipy [101] using an unconstrained minimization solver (based on L-BFGS-B) for every BO algorithm except LS-BO. The introduction of the reference GP model in the constraints of the AF minimization problem required the selection of a method capable of constrained optimization; for this,

we selected SLSQP. Except for HS-BO, the exploratory parameter of the acquisition function was set to the same fixed value ($\kappa = 2.6$). All algorithms were initialized using the same starting point, and we conducted 25 trials, each with a different starting point selected from a 5×5 grid of $\mathcal{T}$, in order to evaluate robustness. We also ran instances of LS-BO and VP-BO with and without using a reference in the AF (for learning the residual or the performance function). This allowed us to isolate the impacts of the use of the reference model and ensure that observed performance improvements can be attributed to the parallel capabilities. For both LS-BO and VP-BO, the reference function was always used to guide the selection of the partitions.

Figure 5.7 highlights the level sets that we used to partition the design space for the LS-BO approach. These partitions were generated by first locating the minima (local and global) of $\hat{g}(x)$. After determining that there were two, we discretized the range of $\hat{g}(x)$ by building a search interval around each of the minima where the lower bound of the interval was the value of the corresponding minimum. The value of $\hat{g}$ was then evaluated at various points on a line connecting the two minima to determine the spacing of the level sets. This information was used to select the upper bound of these search intervals. We were also able to use this analysis to gauge the size of both of the partitions and observed that the search region around the global minima appeared to cover a significant portion of the design space. As a result, this partition was split along the level set value that resulted in two roughly equal-sized partitions. The fourth and final partition was constructed to search the remaining space outside of the three existing partitions. Figure 5.7 also provides an illustrative summary of this workflow. Note that one of the regions is near the global minimum of f.

Given that the reactors are arranged in series, it is clear that the performance of the first reactor is independent of T_2, while the performance of the second reactor will likely have some dependence on T_1. Figure 5.8 demonstrates this partially-separable structure; note how the first function g_1 is not affected by T_2 (vertical lines), while g_2 does depend on T_1. Using ARD, we confirmed that T_1, which had a characteristic length scale of $l = 0.145$,

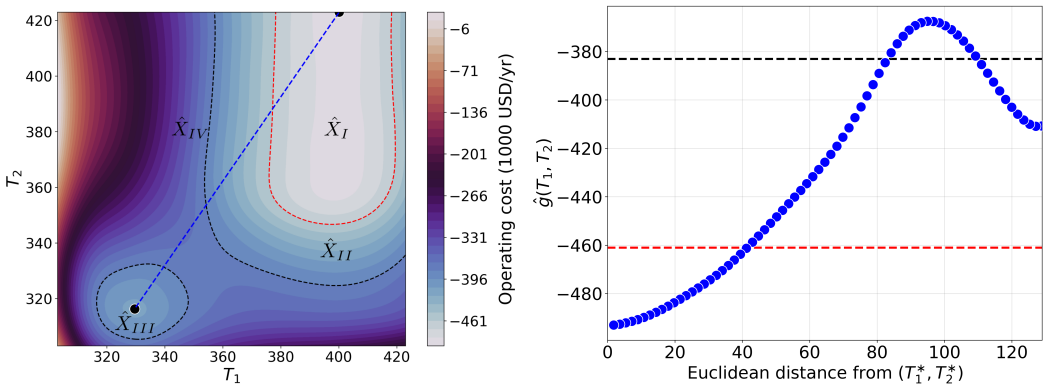

Figure 5.7: Domain partitions for reactor system obtained using reference model $\hat{g}$ (left); the line connecting the two minima of the reference model is shown in blue. Values of $\hat{g}$ along this line (right) indicate that the level set $\hat{g} = -383$ (black line) provides an acceptable split between the two partitions surrounding the minima, while the level set (red line) $\hat{g} = -461$ allows for the partition surrounding the global minimum (denoted as (T_1^*, T_2^*)) to be split into two roughly equal-sized partitions. Note that domain X_{III} is in the region of the global minimum of f.

was a more important input to g_1 than T_2 $(l = 1000)$, while for g_2, T_2 $(l = 0.399)$ was determined to be more important than T_1 $(l = 0.498)$. We thus implemented the VP-BO approach according to the following variable partitions: $x_1 = T_1$, $\mathbf{x}_{-1} = T_2$ and $x_2 = T_2$, $\mathbf{x}_{-2} = T_1$.

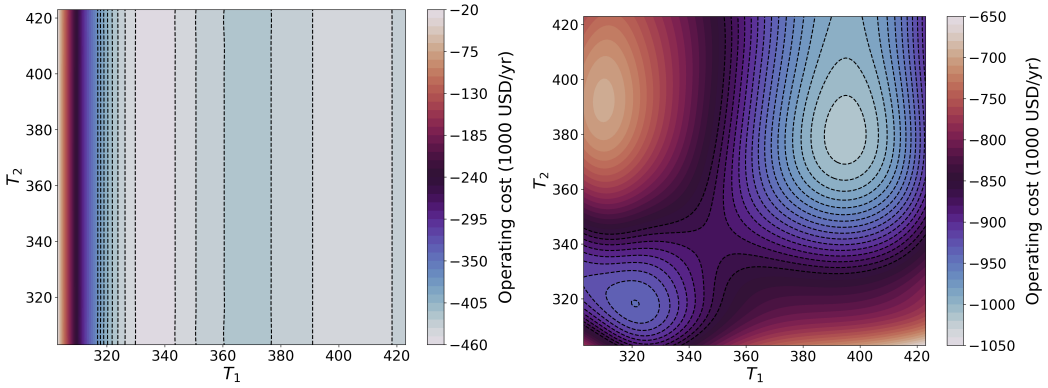

Figure 5.8: Reference model for the first reactor g_1 (left) and for the second reactor g_2 (right). We can see that g_1 is not affected by T_2; the combination of these functions give rise to the reference function $g = g_1 + g_2$.

Figure 5.9 summarizes the average performance (over the 25 runs) of LS-BO and VP-

BO (using reference models) along with the remaining algorithms. Here, we visualize the total experiment time (wall-clock time needed to evaluate performance function) against the best found performance up to the corresponding time. Overall, we observed that all parallel BO variants performed better than standard BO in terms of both speed and best performance found. The performances at the local minima for $(T_1, T_2) = (423, 340)$ and $(T_1, T_2) = (423, 423)$ were approximately -395,000 USD/yr and -387,000 USD/yr respectively, while the performance of the global minimum at $(T_1, T_2) = (333, 322)$ was -410,000 USD/yr. On average, the best performance obtained using BO was -394,500 USD/yr, indicating that this approach converges to a local minimum most of the time. By comparison, all the parallel BO variants found a solution that, on average, was below -400,000 USD/yr. We should also note that this improvement in performance value also comes with a significant reduction in the required wall-clock time: BO took over 500 seconds to converge to its final solution whereas all of the parallel BO variants were able to locate a better solution in approximately 200 seconds.

We note that in this work we used wall-clock time as the comparison metric rather than number of iterations, which is the metric most commonly used in the BO literature. We believe that this allows for a more fair comparison between the parallel and non-parallel versions of the BO algorithm, as every cycle of S-BO and Ref-BO only runs one experiment, while every cycle of the parallel BO algorithms runs 3-4 experiments in tandem. As such, the number of iterations that the parallel BO approaches require to locate a solution can be significantly lower than for the sequential variants (they collect more data per cycle), but the time per iteration can be significantly higher. The use of wall clock time helps standardize the benchmarking of sequential vs. parallel approaches. We recognize, however, that benchmarking algorithms using different metrics can provide valuable insights.

The magnified profiles of Figure 5.9 provide a better comparison between the parallel BO variants. It is clear that LS-BO and VP-BO are significantly faster than all other variants. We also observed that LS-BO, VP-BO, HS-BO and q-BO consistently reached the

global minimum. This illustrates how the redundant sampling seen in MC-BO and HP-BO can degrade performance. Additionally, while the performances of HS-BO and q-BO were similar to LS-BO and VP-BO, they required significantly more experiments to reach this performance level. From these observations we can draw a couple of key conclusions: (i) the use of a reference model for both generating system-specific partitions and simplifying the learning task delivers significant benefit, and (ii) allowing the algorithm to pool the data into a single dataset that is used to build a global surrogate model increases the predictive value of this model, resulting in faster identification of optimal regions.

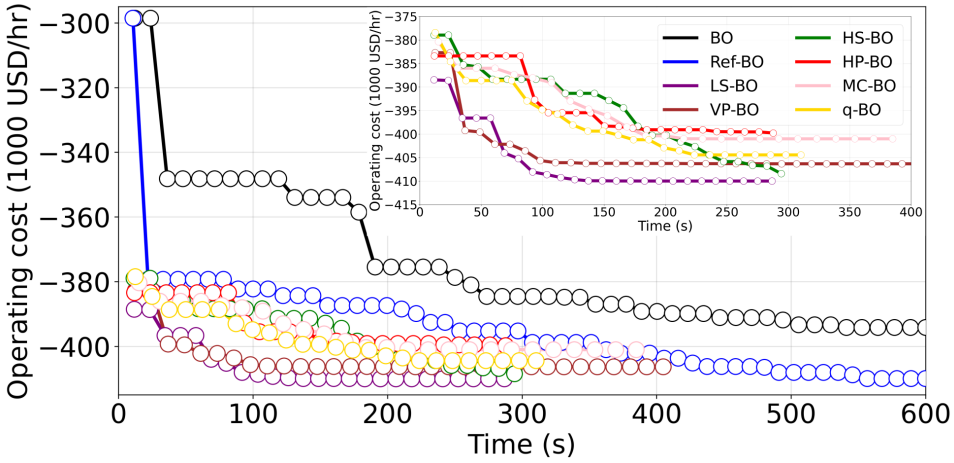

Figure 5.9: Total experiment time against value of best solution for tested algorithms. LS-BO and VP-BO were run using the reference model to partition the domain and guide the search.

Figure 5.10 presents results similar to Figure 5.9, but we run LS-BO and VP-BO without a reference model. By comparing with the results in Figure 5.9 , we observe that using the reference model can help with convergence but not always. LS-BO was 24% slower in terms of the average convergence time, though it maintained its ability to consistently converge to the global minima. VP-BO, meanwhile, converged on average 40% faster compared to when the reference model was used; the solution it returned was also unchanged. These results indicate that g affects LS-BO similarly to S-BO, as outlined in [115]. Namely, that it makes the search more targeted, resulting in more efficient sampling and

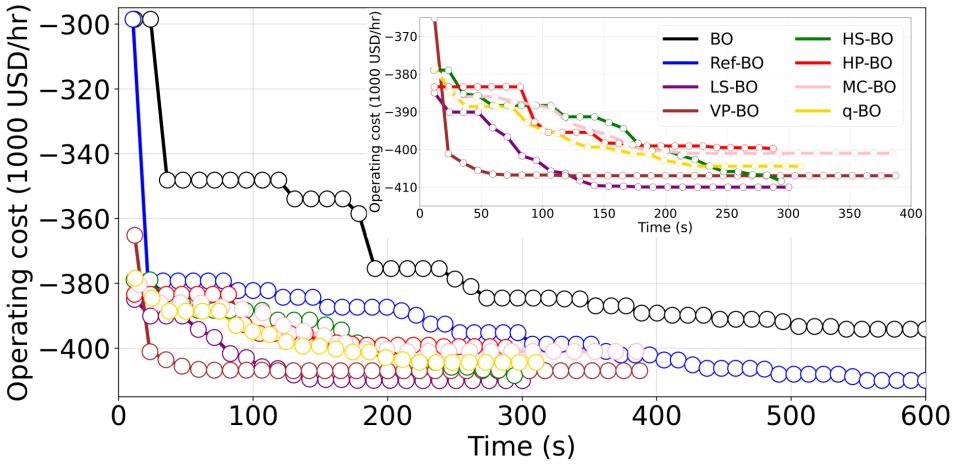

Figure 5.10: Total experiment time against value of best solution for the tested algorithms. LS-BO and VP-BO were run using the reference model to partition the domain but *not* to guide the search.

faster conversion. Meanwhile, with VP-BO, the reference model appeared to encourage more exploration of the domain, which can prevent the algorithm from converging prematurely and potentially returning a suboptimal solution. We base this claim on the fact that, when testing VP-BO without a reference, we observed that while it is not especially sensitive to the initial values of the design variables in a given partition, it is quite sensitive to variable values of other partitions. Overall, however, we observed that both LS-BO and VP-BO still outperformed the remaining parallel algorithms (without or without a reference). These results highlight that using the proposed partitioning approaches has a larger effect on overall convergence. This allows us to confirm that the improvements we observe when using LS-BO and VP-BO can be attributed to the parallelization schemes.

The results presented in Figure 5.9 indicate that LS-BO and VP-BO were consistently more robust and sample efficient than the other approaches. The average values seen in Figures 5.9 and 5.10 provide a measure of robustness: deviations between the final reported average value and one of the three minima are due to the algorithm converging to different solutions during the various runs. For example, the final average reported value for S-BO of 394,500 USD/yr was the result of this algorithm converging to the

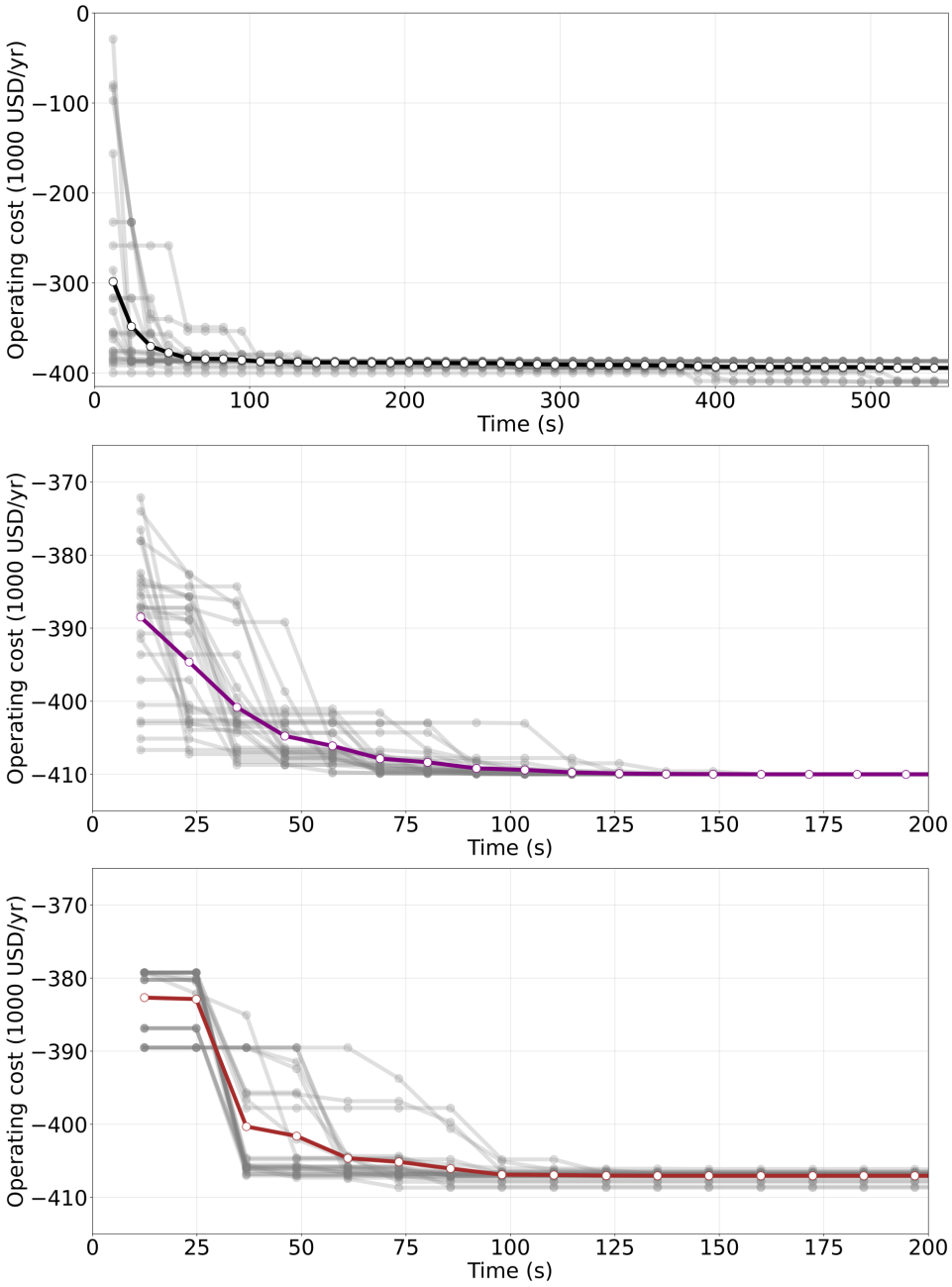

Figure 5.11: Distribution of the performance profiles across the 25 runs for BO (top), LS-BO (middle), and VP-BO (bottom) with the average algorithm performance is shown in color.

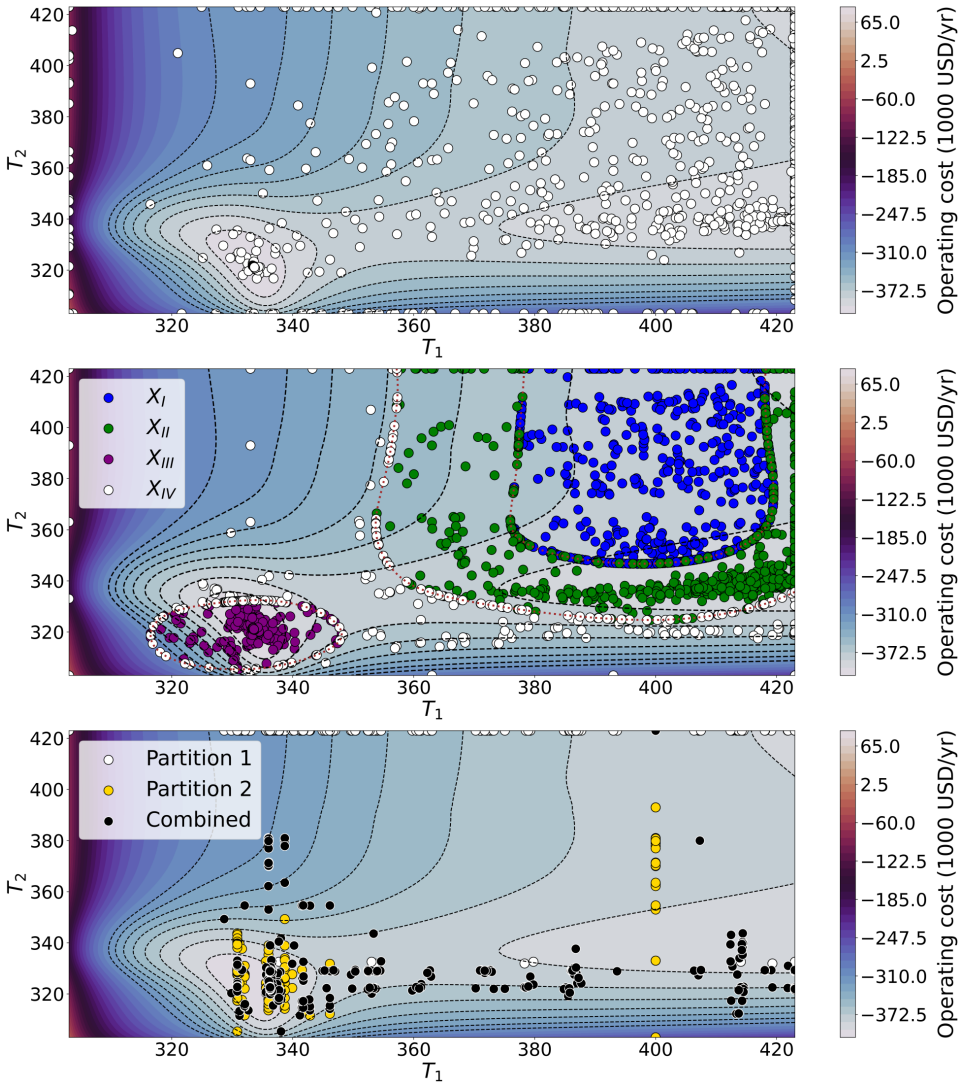

Figure 5.12: Experiment locations across the 25 runs for BO (top), LS-BO (middle), and VP-BO (bottom).

minima at $(T_1, T_2) = (423, 340)$ 13 out of the 25 runs, $(T_1, T_2) = (423, 423)$ for 8 runs, and to $(T_1, T_2) = (333, 322)$ the remaining 4 runs. As a result, the fact that the final reported average values for LS-BO and VP-BO were near the global minimum indicate that these algorithms converged to or near the global solution for most if not all runs (they are robust). The convergence data collected across all runs and shown in Figure 5.11 confirms this. We can see that, regardless of where LS-BO and VP-BO were initialized, they were always able to converge to the same region (unlike S-BO). We also see that convergence of the algorithms was in general fast but, as expected, it was sensitive to the starting point. The sensitivity to the starting point is further indication of why it is important to have expert knowledge (e.g., via use of a reference model) when initializing the search.

Because evaluating the performance function tends to be expensive, reducing the number of experiments (samples) is also essential. Figure 5.12 illustrates how S-BO, LS-BO, and VP-BO compared in terms of sample efficiency. Standard BO sampled in a significantly distributed manner with a considerable number of samples drawn from the boundaries of the domain. For LS-BO, we see that in regions where a solution exists (e.g., regions II and III in our case study), sampling was heavily concentrated at or near the solution. In regions where there is not a solution, the sampling was more distributed. However, the majority of samples tended to cluster around partition boundaries that are located near a solution. Samples drawn from partition X_I appeared to be the most widely distributed, however, this was not surprising as this partition contains a mostly flat region. Another noticeable difference when compared to traditional BO was that there was significantly less sampling at the boundaries of the domain where f has unfavorable (high) values; only 8 out of 2500 samples were taken at the left and bottom bounds. VP-BO exhibited the most clustered sampling; in fact, the vast majority of the samples were drawn from or near the optimal region. Note that, while the majority of samples for X_I (Partition 1) occurred at the top domain boundary, this partition corresponds to reactor 1 which only depends on T_1 as seen in Figure 5.8. Aside from these samples, there was a clear lack of sampling happening at the domain boundaries compared to LS-BO and

traditional BO. This result, coupled with exhibiting the lowest convergence time out of all of the tested algorithms, confirms our belief that the VP-BO algorithm tends to be more exploitative. This is likely due to fact that the partitions for this algorithm are optimized over a lower dimensional space and, for a fixed $\mathbf{x}_{-k}$, VP-BO can find the optimal local variables x_k much faster than the remaining algorithms can find an optimal global variables x. As a result, without the reference model to indicate the potential existence of a solution elsewhere, VP-BO seems more susceptible to settle into the first solution that it finds than algorithms like LS-BO and HS-BO whose partitions force the algorithm to search more widely.

To estimate the computational cost associated with the different algorithms, we measured the total wall-clock time (averaged across the 25 runs). The total wall-clock time includes time for performance evaluation (experiment time) and all time required to conduct other computations (e.g., AF optimization, GP training, and reference model evaluation). The results are shown in Figure 5.13. The closer this time is to the experimental time, the less computationally expensive the algorithm is. For instance, the total wall-clock time of S-BO was 12% higher than the experiment time. We observed that HS-BO and VP-BO were the least computationally intensive methods, with the total wall-clock time being only 14% and 7% higher than the experimental time respectively. We attribute this to the fact that HS-BO runs separate instances of BO across multiple reduced domains and, because the boundaries are rectangular, ensuring that the AF optimizer stays inside of the partition only involves bounding the upper and lower limits of x it is allowed to search over. VP-BO, on the other hand, only optimizes over a subset of variables and this greatly reduces the time required for AF optimization. The wall-clock time of HP-BO was 44% higher than the experiment time, this is because it requires solving multiple AF optimization problems across the entire design space. LS-BO had a total wall-clock time that was 46% higher than the experiment time. This is attributed to the more difficult AF optimization problem that it has to solve (which has constraints defined by a GP model). The total wall-clock time for q-BO was 64% higher than the experiment time, which we

attribute to the fact that AF optimization is done over a set of points, increasing the size of the problem that is solved. Additionally, the calculation involves more complex matrix operations and requires repetitive sampling. MC-BO was the most computationally intensive algorithm, with a total wall-clock time that was 384% higher than the experiment time. We attribute this to the repetitive computations in this algorithm, which require sequential sampling and GP training.

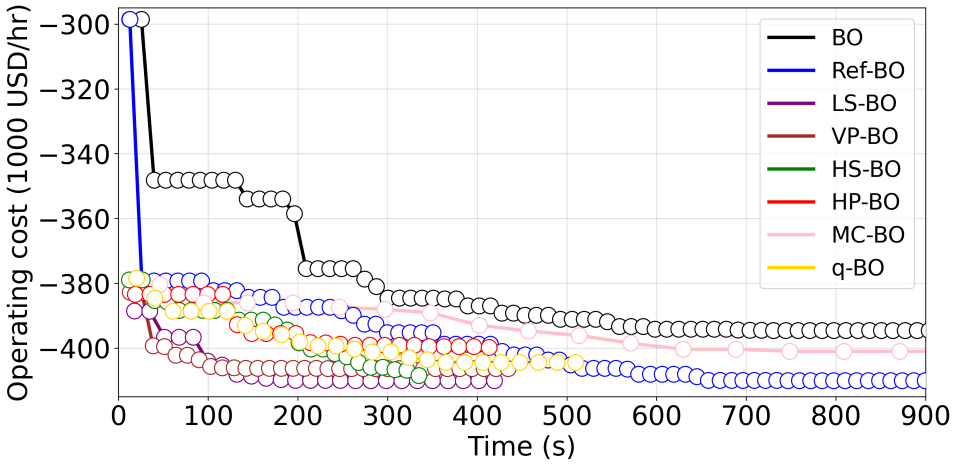

Figure 5.13: Profiles of wall-clock time against performance. Note that this time is comparable to the total experiment time for all algorithms; the only exception is MC-BO, indicating that the AF optimization step (and not the function evaluation) is the bottleneck for this approach.

5.5 Conclusions and Future Work

We have proposed a set of new decomposition paradigms for BO that enable the exploitation of parallel experiments. These approaches decompose the design space by following the level sets of the performance function and by exploiting the partially separable structure of the performance function. A key innovation of these approaches is the use of a reference function to guide the partitions. Using a case study for a reactor system, we found that the proposed approaches outperformed existing parallel approaches in terms of time and quality of solution found. When using LS-BO, we observed that build-

ing partitions that are specialized beyond those that would be generated by the uniform discretization of the range of $\hat{g}(x)$, like those we used in our case study, can require significant user input. Moving forward we would like to explore methods for developing more efficient and automated protocols for generating the partitions. Additionally, we are also interested in incorporating an element of adaptivity to LS-BO and VP-BO via live modification/tuning of the partitions as samples from the system are collected. The proposed parallel paradigms can also open the door to a number of applications and potentially other decomposition paradigms that we will aim to explore in the future. Specifically, we are interested in exploring more complex systems that involve higher-dimensional design spaces and large numbers of parallel experiments. This will allow us to investigate the asymptotic properties of the proposed approaches. Moreover, we are interested in designing alternative paradigms that selectively exchange information between partitions to accelerate the search and that use different types of reference models to guide the search. We are also interested in exploring the application of these approaches to the tuning of complex controllers.

Part III

FINAL THOUGHTS

Chapter 6

CONCLUSIONS AND FUTURE DIRECTIONS

We now conclude this dissertation by summarizing the key findings/contributions of the work presented in Chapters 3-5. We also identify how these contributions can be enriched further with future research.

6.1 Contributions

Incorporating Physics via Reference Models

Chapter 3 presents the reference model, $g(x)$, a low-fidelity representation of the true system that can be sampled more easily (i.e., it is faster and cheaper to evaluate). The aim of the reference model is not necessarily to provide a perfect approximation of the system, but rather to capture coarse trends and identify generally promising regions. Consequently, $g(x)$ can be constructed from a variety of sources, such as simplified physics, empirical correlations, or low-fidelity simulations. The physics knowledge contained within the reference model can be directly incorporated into the Bayesian optimization (BO) framework by embedding $g(x)$ within the acquisition function and shifting the surrogate modeling task to learning the model residual or error, $\varepsilon(x)$, rather than the underlying performance function. The algorithm is then initialized with an approximate representation of the system that highlights potentially promising regions that it can immediately

begin to explore. This results in better sampling efficacy when compared to standard Bayesian optimization (S-BO), since fewer iterations are spent searching in highly suboptimal regions, as demonstrated by the results of our HVAC MPC case study. Additionally, our findings indicate that use of the reference model reduces sensitivity to the initialization point, as the algorithm always starts off with the same ground truth of the system. This results in a more robust performance when compared to S-BO. We also note that shifting to modeling the residual can also yield benefits, as it can be significantly easier to build an accurate estimate of $\varepsilon(x)$ than of the performance function. This facilitates the correction of $g(x)$ and enables the construction of an accurate hybrid system model, further boosting performance.

Composite Function BO via Adaptive Linearizations

In Chapter 4, we consider the scenario where information is available in the form of system connectivity or structure. Shifting to a composite representation of the performance function, where we instead treat a set of intermediates as black-boxes, offers an intuitive method for exploiting this type of information. However, propagating the uncertainty estimates from the surrogate models of the intermediates to the performance metric is often an intractable problem. We introduce a new framework, which we refer to as BOIS, that utilizes an adaptive linearization scheme to overcome this challenge and derive a set of closed-form expressions for the statistical moments of the performance function. Using a pair of case studies, we compare the performance of BOIS against S-BO as well as existing composite function BO paradigms based on sampling (MC-BO) or the use of an auxiliary problem (OP-BO). Each of the case studies highlights the benefits of using a composite function representation of the performance function. We can combine knowledge of system connectivity and fundamental principles with any available subcomponent white-box models to more efficiently specify the intermediates and the inputs of their respective surrogate models. Additionally, the intermediates can be selected so that they are easier to learn than the performance function, thereby facilitating accurate estimation of the system

behavior. Our results demonstrate that BOIS consistently outperforms S-BO and delivers equal or better performance than MC-BO and OP-BO, while also being significantly less computationally intense. Further, we determine that our adaptive linearization scheme accurately estimates the statistical moments of the performance function in a fraction of the time it takes Monte Carlo to generate estimates of comparable accuracy. These results highlight the efficiency of BOIS and suggest that it can facilitate the use of composite functions in a BO setting. This is a promising outcome, given that composite functions can provide a solution to the challenge of scaling BO to higher dimensions by distributing the modeling of the system among the various intermediates.

Informed Partitioning of the Design Space to Enable Parallelization

We present a pair of novel parallelization paradigms, level set BO (LS-BO) and variable partitioning BO (VP-BO), in Chapter 5. The aim of these methods is to address the issues observed in existing parallel BO algorithms, namely the limited degree of parallelization that can be achieved and the occurrence of redundant sampling. This is accomplished by decomposing the design space into a set of distinct partitions along the level sets of the performance function or according to the partially separable structure of the system. However, as the system is often not readily observable, the partitioning is instead guided by a reference model. This allows for an efficient division of high-throughput experiment (HTE) resources, as the search can be concentrated around regions where the system is estimated to perform well. We benchmark the performances of LS-BO and VP-BO against S-BO and a series of parallel approaches found in the literature using a reactor network case study. Our findings indicate that the proposed paradigms are able to outperform the existing algorithms both in terms of speed and the value of the solution. We also observe that our methods are more robust, consistently converging to the region containing the global solution regardless of where the algorithm is initialized. These results, coupled with the relatively low computational intensity we observed, demonstrate that LS-BO and VP-BO provide a set of efficient and targeted parallel experimentation solutions.

6.2 Future Research Directions

Establishing Asymptotic Properties for the Reference Model

The performance improvements observed when using a reference model are empirical, and there is not a clear understanding on how coarse the reference model can be before it no longer provides useful information. While the results of the case studies indicate that there appears to be significant leeway, this is likely system-dependent. Establishing a theoretical minimum information/accuracy criterion for ensuring that the reference model is beneficial would provide guidelines for determining under which conditions Ref-BO remains an effective solution. Solving this problem is obviously complicated by the fact that, in a realistic setting, determining the accuracy of the reference model is not possible as the real system model is not available for comparison. However, work by Kandasamy and colleagues has shown that it is possible to establish conditions under which multi-fidelity BO that can be proven to perform better than S-BO [145, 146, 147]. These are largely based on the effective reduction in the search region provided by the low-fidelity approximations. A similar analysis that determines the degree to which the reference model narrows down the search space coupled with a regret measure—obtained by determining how a reference model performs given an arbitrary residual—can be used to establish the theoretical efficacy of Ref-BO.

Alternative Surrogate Models

While Gaussian Processes (GPs) offer significant flexibility and are very data efficient, they have limitations in placing bounds on prediction values and can struggle to model highly non-smooth and discontinuous functions. This can lead to infeasible predictions, especially when the system operates near a feasibility bound of the modeled quantity or near a discontinuity, which negatively impact the performance of BO by causing it to sample from suboptimal regions. Parametric models like neural networks and polynomials or bounded output models like skewed sparse GPs can provide a solution to this

issue [148, 124, 149, 150]. Additionally, these models scale to higher input dimensions significantly better than GPs, providing a potential solution for deploying our paradigms on large-scale applications.

BO in High-dimensional Design Spaces and Integration with Network Modeling Tools The distribution of the modeling task to various intermediates in composite function BO offers a solution to the input scaling issues encountered by GPs. As shown in Chapter 4, the entire set of inputs does not need to be used to model each of the elements in $y(x)$. By leveraging system connectivity, we can nest the appropriate intermediates within each other to capture the effects of upstream variables. This allows for the creation of a set of interconnected models, each with a custom set of inputs that is more relevant for predicting component behavior. Not only does this reduce the number of surrogate inputs, thereby allowing GPs to continue to be used even at high dimensions, but it also results in more accurate models. Such an approach is commonly used in modeling network systems (e.g., chemical processes), where each intermediate component is often described by its own model with a unique set of inputs and outputs. While some of the inputs are specified at the design variable level (e.g., the size of a unit), others depend on the output of another intermediate (e.g., a feed stream that is the outflow of an upstream unit). BOIS can therefore serve as a natural optimization framework for existing network modeling libraries, facilitating the inclusion of new or difficult-to-model intermediates by treating these as black-boxes while continuing to utilize readily available component models. Thus, exploring the integration of BOIS with available modeling tools is a promising and exciting direction for future research. The results of this work can determine if the distributed modeling approach presented can be used to overcome the curse of dimensionality exhibited by GPs, while also providing these libraries with a powerful optimization strategy.

Automated and Dynamic Design Space Partitioning

Currently, the domain partitions used in LS-BO and VP-BO are set manually. While this provides an additional opportunity to leverage expert knowledge, requiring the user to specify all of the partitions can be unreasonable, especially for a large number of parallel experiments. As discussed in Chapter 5, uniform partitioning between the maximum and minimum of the reference model is a potential solution for automating the decomposition on X in LS-BO. However, this approach does not allow for the concentration of resources around more promising regions. In the case of VP-BO, even with methods like ARD, user input is still required to interpret the results of the analysis and determine the allocation of the variables. Thus, developing a solution that utilizes information such as the number and location of minima in the reference model or structural relations between subcomponents and inputs (i.e., downstream inputs will not affect upstream inputs) to automatically build a set of partitions, while allowing for manual adjustments, can greatly streamline the implementation of these methods. Additionally, incorporating a dynamic element, where the partitions are adaptively modified as more information becomes available, can reduce the number of suboptimal experiments performed by gradually focusing in on the identified optimal regions.

Merging Developments into a Single Paradigm

The capabilities we have developed are currently distributed among four different algorithms. Combining these into a unified paradigm could unlock capabilities that would otherwise be unavailable to each individual method, such as the consideration of variable hierarchies [151], the inclusion of system constraints [152], dynamic adjustments to the system configuration/structure [153], and the construction of surrogates that are accurate across a wide range of the design space [154]. The incorporation of a reference model into either BOIS, LS-BO, or VP-BO can be done rather seamlessly by simply shifting the surrogate modeling task to a residual. Note that in the case of BOIS, the reference model(s) would have to be of the intermediate(s). However, merging these three paradigms into

one is not as straightforward. While BOIS and VP-BO make use of the same type of information, VP-BO optimizes each subcomponent individually, whereas BOIS optimizes the whole system. LS-BO utilizes the level sets of the system-wide performance function. However, it could instead consider component performance or even the values of the intermediate variables themselves (to explore different operating regimes). Thus, future research should focus on determining the most advantageous manner for integrating these three methods, testing the resulting framework across various domains to assess its effectiveness, and optimizing its performance to ensure it can handle large-scale, complex systems efficiently.

Appendix A

SUPPLEMENTARY INFORMATION

This appendix presents supplementary information from select sections of this dissertation.

A.1 System Models

A.1.1 *Reactor-Separator Network Model*

The reactor-separator network model considers the generation of product C from two reagents A and B. As seen in Figure A.1, these reagents are fed into the process where they are individually compressed and heated. The compressors are modeled with the `IsentropicCompressor` module from the BioSTEAM library [155] using the properties of N_2, H_2, and NH_3 for A, B, and C respectively and the `IdealAcitivityCoefficients` and `IdealFugacityCoefficients` methods. The models calculate the temperatures of the compressor outlet as well as the power load of the unit. The heaters are assumed to

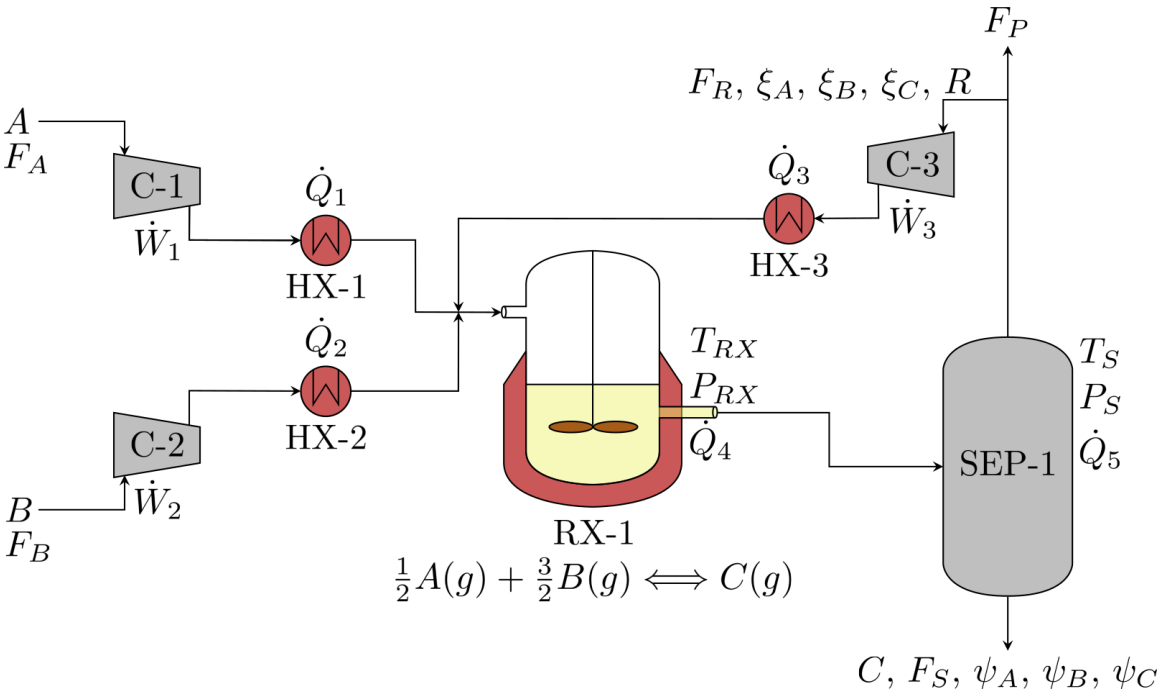

Figure A.1: Schematic diagram of the reactor-separator network modeled in the first case study.

operate at constant pressure and we can calculate their duties as:

$$\dot{Q} = F(H_{\text{out}} - H_{\text{in}}) \tag{A.1a}$$

$$= F(H^\circ + R \cdot \text{ICPH}(T_{\text{out}}) - H^\circ - R \cdot \text{ICPH}(T_{\text{in}})) \tag{A.1b}$$

$$= FR\,(\text{ICPH}(T_{\text{out}}) - \text{ICPH}(T_{\text{in}})) \tag{A.1c}$$

where F is the material flowrate through the heater, T_{in} and T_{out} are the temperatures of the heater inlet and outlet respectively, H° is the standard enthalpy of the streams, and R is the universal gas constant. The ICPH function measures the effect of temperature on enthalpy as is defined as:

$$\text{ICPH}(T) = \Delta\alpha\,(T - T^\circ) + \frac{\Delta\beta}{2}\left(T^2 - (T^\circ)^2\right) + \frac{\Delta\gamma}{3}\left(T^3 - (T^\circ)^3\right) - \Delta\zeta\left(T^{-1} - (T^\circ)^{-1}\right) \tag{A.2a}$$

where T° is the standard temperature (298 K) and

$$\Delta\alpha = \sum_{i\in\{A,B,C\}} x_i\alpha_i \tag{A.3a}$$

$$\Delta\beta = \sum_{i\in\{A,B,C\}} x_i\beta_i \tag{A.3b}$$

$$\Delta\gamma = \sum_{i\in\{A,B,C\}} x_i\gamma_i \tag{A.3c}$$

$$\Delta\zeta = \sum_{i\in\{A,B,C\}} x_i\zeta_i \tag{A.3d}$$

Here, x_i is the molar fraction of species i in the stream and α_i, β_i, γ_i, and ζ_i are a set of molecule-specific coefficients.

The hot streams are fed to a reactor, RX-1, where A and B undergo the following gas-phase reaction:

$$\frac{1}{2}A(g) + \frac{3}{2}B(g) \Longleftrightarrow C(g) \tag{A.4a}$$

The reaction mixture is assumed to be an ideal gas and, at a given temperature and pressure, the composition of the reactor effluent is calculated from the following:

$$\prod_{i\in\{A,B,C\}} (y_i)^{\nu_i} = \prod_{i\in\{A,B,C\}} \left(\frac{n_{i0}+\nu_i\varepsilon}{n_0+\nu\varepsilon}\right)^{\nu_i} = \left(\frac{P_{RX}}{P^\circ}\right)^{-\nu} K \tag{A.5}$$

where y_i is the mol fraction of species i in the reactor outlet and ν_i is its stoichiometric coefficient; n_{i0} are the moles of i in the feed stream, $\nu = \sum_i \nu_i$ and $n_0 = \sum_i n_{i0}$; the extent of the reaction is denoted by ε; P_{RX} is the pressure of the reactor and P° is standard pressure (1 bar). The equilibrium constant, K, is calculated as:

$$\log K = -\frac{\Delta G_{\text{rxn}}}{RT_{RX}} \tag{A.6}$$

where T_{RX} is the reactor temperature. The change in the Gibbs free energy of the reaction,

ΔG_{rxn}, is a function of temperature and has the form:

$$\Delta G_{\text{rxn}} = \Delta H^{\circ}_{\text{rxn}} - \frac{T_{RX}}{T^{\circ}} \left(\Delta H^{\circ}_{\text{rxn}} - \Delta G^{\circ}_{\text{rxn}}\right) + R \left(\text{ICPH}(T_{RX}) - T_{RX} \cdot \text{ICPS}(T_{RX})\right) \tag{A.7}$$

where $\Delta G^{\circ}_{\text{rxn}}$ and $\Delta H^{\circ}_{\text{rxn}}$ are the changes in Gibbs free energy and enthalpy of the reaction measured at standard temperature. Similar to the ICPH function, the ICPS function measures the change in entropy due to temperature effects and is defined as:

$$\begin{aligned} \text{ICPS}(T) = {} & \Delta\alpha \left(\log(T) - \log(T^{\circ})\right) + \Delta\beta \left(T - T^{\circ}\right) \\ & + \frac{\Delta\gamma}{2}\left(T^2 - (T^{\circ})^2\right) - \Delta\zeta\left(T^{-2} - (T^{\circ})^{-2}\right) \end{aligned} \tag{A.8a}$$

Due to the reaction, the ICPH and ICPS coefficients are now defined as:

$$\Delta\alpha = \sum_{i \in \{A,B,C\}} \nu_i \alpha_i \tag{A.9a}$$

$$\Delta\beta = \sum_{i \in \{A,B,C\}} \nu_i \beta_i \tag{A.9b}$$

$$\Delta\gamma = \sum_{i \in \{A,B,C\}} \nu_i \gamma_i \tag{A.9c}$$

$$\Delta\zeta = \sum_{i \in \{A,B,C\}} \nu_i \zeta_i \tag{A.9d}$$

Because the reaction occurs at constant temperature and pressure, the duty of the reactor, $\dot{Q}_4$ is calculated as:

$$\dot{Q}_4 = \dot{r}_C \left(\Delta H^{\circ}_{\text{rxn}} + R \cdot \text{ICPH}(T_{RX})\right) \tag{A.10}$$

where $\dot{r}_C$ is the generation rate of C in the reactor.

The product stream that exits the reactor is fed to a separator, SEP-1, where it is cooled to T_S at pressure P_S to allow for the condensation and recovery of C. This unit is modeled with the `Flash` module from the BioSTEAM library using the same properties and methods as the compressor models. From this model we obtain the flowrates and compositions of the resultant product and vapor streams as well as the energy requirements of

the separator. A fraction, $1 - R$, of the vapor stream that exits SEP-1 is purged and leaves the process. The remainder is re-compressed and heated before being fed back to RX-1.

The performance of the system is expressed as a cost function (negative profit) that measures the consumption of the reagents along with production rate of C and the corresponding utility requirements of the process at various recycle fractions and reactor and separator settings; it is formulated as:

$$f_1(x, y(x)) = \sum_{j \in \{A,B\}} w_{j0} F_j + F_S \sum_{i \in \{A,B,C\}} w_i \psi_i + w_3 \left(\frac{\psi_C F_S - \bar{F}}{\bar{F}} \right)^2 \tag{A.11a}$$

$$f_2(x, y(x)) = \sum_{h=1}^{5} w_h \dot{Q}_h, + w_e \sum_{k=1}^{3} \dot{W}_k \tag{A.11b}$$

$$f(x, y(x)) = f_1(x, y(x)) + f_2(x, y(x)) \tag{A.11c}$$

where the total cost $f(x, y(x))$ is distributed among two functions f_1 and f_2. Here, f_1 considers the the materials flowing in an out of the system: the first term represents the costs of A and B with w_{j0} and F_j denoting the cost and feed rate of each reagent; the second term measures the value of the product stream with F_S denoting its flow and ψ_i and w_i representing the fraction of species i present in this stream and its corresponding unit value. The final term in f_1 is a quadratic penalty term meant to enforce a demand cap for the production of C, which is denoted by $\bar{F}$. The flow of energy through the system, and thereby the cost of utilities, is measured by f_2. The heating and cooling requirements of the feed and recycle heaters, RX-1, and SEP-1 are denoted by $\dot{Q}_h$ and these have a unit cost of w_h. Similarly, compressor k has a power load of $\dot{W}_k$, and this is supplied at a unit cost of w_e.

Table A.1: Thermodynamic Constants for the Reactor-Separator Network Model

Parameter	Value	Units	Notes
R	8.314	J/mol·K	Universal gas constant
T°	298	K	Standard temperature
P°	1.0	bar	Standard pressure
$\Delta H^\circ_{\text{rxn}}$	39200	J/mol	Standard heat of reaction
$\Delta G^\circ_{\text{rxn}}$	32900	J/mol	Change in free energy of reaction at T°
ν_A	$-1/2$	—	Stoichiometric coefficient of A
ν_B	$-3/2$	—	Stoichiometric coefficient of B
ν_C	1	—	Stoichiometric coefficient of C
α_A	3.280	—	α coefficient of A
α_B	3.249	—	α coefficient of B
α_C	5.578	—	α coefficient of C
β_A	0.593×10^{-3}	—	β coefficient of A
β_B	0.422×10^{-3}	—	β coefficient of B
β_C	3.020×10^{-3}	—	β coefficient of C
γ_A	0.000	—	γ coefficient of A
γ_B	0.000	—	γ coefficient of B
γ_C	0.000	—	γ coefficient of C
ζ_A	0.040×10^5	—	ζ coefficient of A
ζ_B	0.083×10^5	—	ζ coefficient of B
ζ_C	-0.186×10^5	—	ζ coefficient of C

Table A.2: Relevant Parameters for the Reactor-Separator Network Model

Parameter	Value	Units	Notes
F_A	1000	kmol/hr	Feed flow of A
F_B	3000	kmol/hr	Feed flow of B
$\bar{F}$	1900	kmol/hr	Target production rate of C
w_{A0}	6.00	USD/kmol	Cost of A
w_{B0}	1.40	USD/kmol	Cost of B
w_A	0.00	USD/kmol	Value of A in product stream
w_B	0.00	USD/kmol	Value of B in product stream
w_C	8.50	USD/kmol	Value of C in product stream
w_h	1.92×10^{-2}	USD/MJ	Cost of heating utility
w_h	5.00×10^{-3}	USD/MJ	Cost of cooling utility
w_e	1.42×10^{-1}	USD/kWh	Cost of power utility

A.1.2 *Biofertilizer Production Process Model*

The biofertilizer production facility we consider, shown in Figure A.2, considers the recovery of consists of 10.59 tonnes/yr of phosphorus (P) from the 20830 tonnes/yr of manure generated at a hypothetical 1000 animal unit dairy farm. The manure is intially fed into an anaerobic digester to generate biogas. The biogas, a mixture largely composed of CH_4, CO_2, and H_2S, is sent to a pair of scrubbers that remove the carbon dioxide and hydrogen sulfide. The resulting product is a high purity methane stream that can either be exported or burned on-site to generate electricity.

The digested manure (also referred to as digestate) is passed through a solids-liquids separator (SLS) that removes the suspended solids present in the digestate. The produced extrudate is then fed to a series of bag photobioreactors (b-PBRs), along with any urea required to provide additional nitrogen (N), where it is used as a growth medium for cyanobacteria (CB) cultivation. The CB growth is assumed to be light-limited and is simulated as described by [5]:

$$X(t) = X_0 \exp(-\kappa t) + X_s \left(1 - \exp(-\kappa t)\right) \tag{A.12a}$$

$$X_S = \frac{\eta \frac{I_0 S}{V}}{m_\nu} \tag{A.12b}$$

$$\kappa = Y_{X\nu} m_\nu \tag{A.12c}$$

Here X is the CB concentration at any given time, X_0 is the initial CB concentration, and X_S is the steady-state $(t = \infty)$ CB concentration; the biomass yield per photon is denoted by $Y_{X\nu}$ and m_ν is the light energy required for cell maintenance; η is the photosynthetic efficiency of the cell (i.e., how much of the energy in each photon is captured and converted into biomass); S and V are the irradiated reactor surface area and total reactor volume respectively, and I_0 is the incident light intensity at the surface of the reactor.

Once the CB culture is ready to be harvested, it is sent to a flocculation tank where self-flocculation is induced. The formed flocs are then fed into a lamella clarifier where

they are allowed to settle out of solution. The CB sludge that forms at the bottom of the clarifier is collected and passed through a pressure filter that produces a concentrated cyanobacteria solution. The solution is transferred to a thermal dryer where the moisture removal process is completed, yielding a dry CB biomass product. The CB-free water that exits the lamella clarifier and pressure filter is mixed, and a fraction of it is recycled back to the reactors while the remainder is purged to prevent the buildup of impurities within the system. Depending on the flowrate of the purge, fresh make-up water may need to be added to the reactors if the water in the incoming manure is not enough to account for this loss.

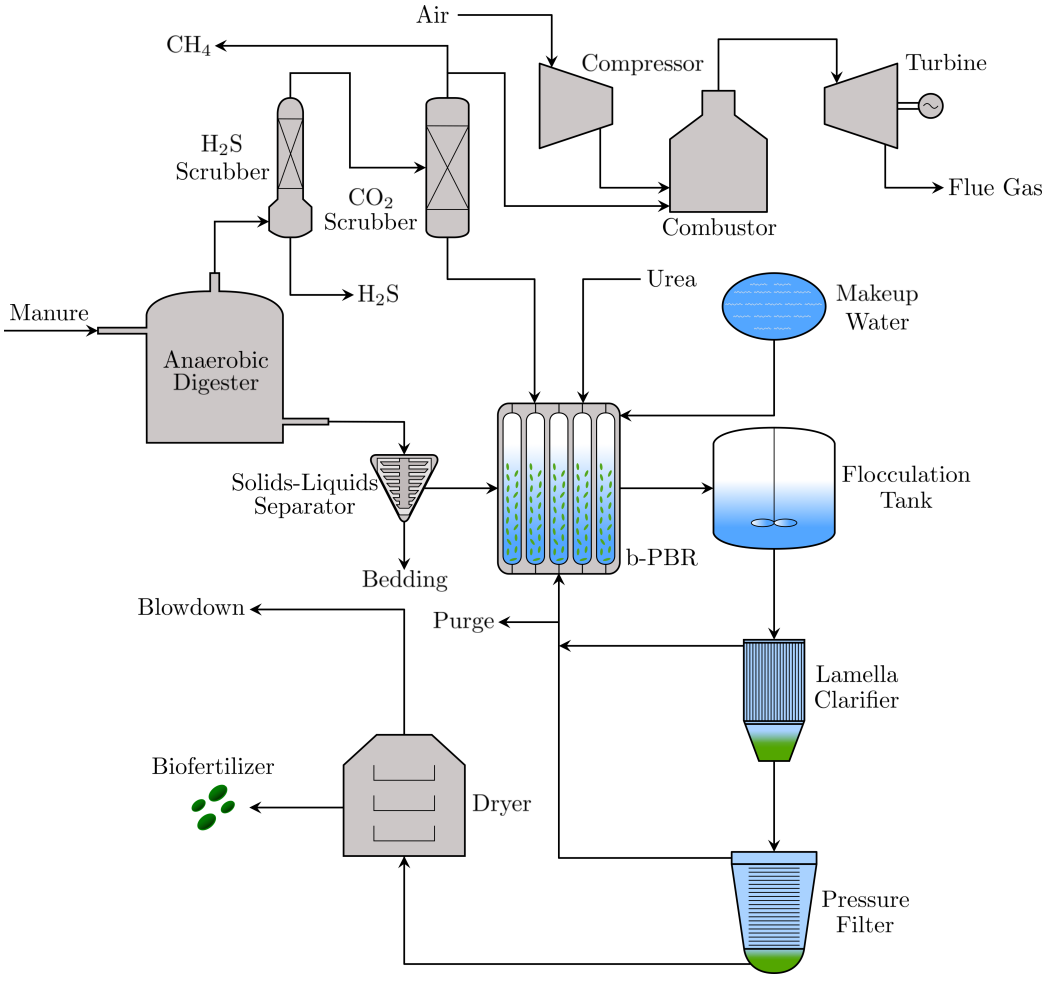

Figure A.2: Flow diagram of the biofertilizer production process.

The CB production rate is set based on the phosphorus density of the bacteria cells, ρ_P (g P/g CB). Given an incoming P flowrate, m_P, the CB needed to fully incorporate the nutrient load is calculated as:

$$m_{CB} = \frac{m_P}{\rho_P} \tag{A.13}$$

From this, we can determine the amount of urea, m_U, required to meet the N demands of the cells based on a specified nitrogen density, ρ_N (g N/g CB):

$$m_U = \max\left\{0, x_{UN}\left(m_{CB}\rho_N - m_N\right)\right\} \tag{A.14}$$

where m_N is the flowrate of N in the reactor feed and x_{UN} is the mass fraction of nitrogen in urea. The amount of fresh makeup water required is also calculated based on the CB production rate and the final titer of the CB culture:

$$m_{FW} = \max\left\{0, m_{CB}\left(\frac{\rho_W}{X(t_b)} - 1\right) - \left(m_{RW} + m_W\right)\right\} \tag{A.15}$$

where t_b is the reactor batch time, ρ_W is the density of water, m_{RW} is the flowrate of the recycled water stream, and m_W is the flowrate of water supplied to the b-PBRs from the manure.

The minimum selling price (MSP) of the biofertilizer is determined by calculating the price, p_{CB}, that it must be sold at to for the process to achieve a specified discounted return on investment (DROI) target (15% in this study). The DROI is a profitability measure that is defined as the discount rate, i, that results in a net present value (NPV) of zero at the end of a process' lifetime. We define the NPV as:

$$\text{NPV} = C + \sum_{j=1}^{T} P(1+i)^{-j} \tag{A.16}$$

where C is the total capital investment (TCI) of the of the process, P is the annual after-tax profit (AATP) and T is the process lifetime (in years).

The TCI is based on the installed costs of the required units and is calculated according to their size. If sizing and costing correlations are available, these are used to calculate these values. Otherwise, they can be obtained from price data found in the literature for similar units using capacity correlations and price indices:

$$c_k = c'_k \left(\frac{S_k}{S'_k}\right)^{\phi_k} \left(\frac{PI}{PI'_k}\right) \tag{A.17}$$

where c_k, S_k, and ϕ_k are cost, size, and scaling factor respectively of the k^{th} unit; c'_k and S'_k are the cost and size of a reference unit; PI is the value of the price index for the year 2020 and PI'_k is the price index for the year in which the quote for the reference unit was made. We can then define the inside battery limit (ISBL) cost of the process as:

$$c_{IS} = \sum_{k}^{K} c_k \tag{A.18}$$

which serves as the basis quantity from which the TCI can be calculated:

$$c_{OS} = 0.4 c_{IS} \tag{A.19a}$$

$$c_{ENG} = 0.3(c_{IS} + c_{OS}) \tag{A.19b}$$

$$c_{CON} = 0.2(c_{IS} + c_{OS}) \tag{A.19c}$$

$$\tag{A.19d}$$

where c_{OS}, c_{ENG}, c_{CON} are the outside battery limits (OSBL), engineering, and contingency costs associated with constructing the process and bringing it online. The total capital investment required is then:

$$C = c_{IS} + c_{OS} + c_{ENG} + c_{CON} \tag{A.20}$$

The AATP is the net income generated by the process and is expressed as:

$$P = (1 - r)\left(p_{CB}m_{CB} + p_{NG}m_{NG} + p_{EL}\dot{w} - O - d\right) + d \tag{A.21}$$

where m_{CB}, m_{NG}, and $\dot{w}$ are the production rates of cyanobacteria, methane, and electricity respectively and these are sold at unit price of p_{CB}, p_{NG}, and p_{EL}. The tax rate levied on the process revenue is denoted by r (21% in the US); d is annual depreciation of the purchased capital equipment and is calculated using a straight-line depreciation scheme assuming a salvage value of 0:

$$d = \frac{c_{IS}}{T} \tag{A.22}$$

The total operating cost (TOC) of the process, O, is composed of two parts, fixed operating costs (FOCs) and variable operating costs (VOCs). FOCs include annual costs that are not tied to production levels and must be paid in full every year the plant operates. This includes expenses such as maintenance, operations fees such as licensing costs and property taxes or leasing costs, overhead costs like insurance, and labor. Maintenance, operations, and overhead costs are calculated as a fraction of the ISBL:

$$c_{MT} = 0.05c_{IS} \tag{A.23a}$$

$$c_{OP} = 0.025c_{IS} \tag{A.23b}$$

$$c_{OV} = 0.05c_{IS} \tag{A.23c}$$

$$\tag{A.23d}$$

and labor costs are calculated as a function of the size of the reaction section, SA (in acres):

$$c_{LB} = c'_{LB}\left(\frac{SA}{SA'}\right) \tag{A.24}$$

where c'_{LB} is the labor cost at a reference facility of size SA'

Variable operating costs consists of items like utilities, raw materials, and storage

and transportation costs that fluctuate with production levels. As we do not consider distribution costs, and we assume that manure and any make-up water required are provided free of charge, the VOCs in this study consist of the utility, urea, and bag replacement costs. The utility requirements are obtained either from correlations or by scaling reported values for similar units from the literature to the desired size, similar to (A.24). We include a scale factor on the bag replacement and reactor mixing costs of the form:

$$\lambda_{SV}\left(\frac{S}{V}\right) = \max\left\{1, 3\left(\frac{\frac{S}{V}}{\left(\frac{S}{V}\right)_0}\right) - 2\right\} \tag{A.25}$$

where $\left(\frac{S}{V}\right)_0$ is the surface area to volume ratio of the base design. This penalty reflects the fact that higher $\frac{S}{V}$ values require more complex geometries which increase the manufacturing costs of the bags and also increase the flow turbulence within the reactors, resulting in a higher mixing energy requirement. A penalty is also added to the labor cost:

$$\lambda_{LB}(t_b) = \max\left\{0, 0.05(t_{b0} - t_b)\right\} \tag{A.26}$$

where t_{b0} is the batch time of the base design and $\lambda_{LB}(t_b)$ has units of MMUSD/yr. This reflects the fact that smaller values of t_b increase frequency at which the b-PBRs must be emptied, cleaned, and refilled and by extension the labor intensity of the process. Finally, we scale the value of X_S by a factor of 0.32 as the model growth model used likely over-predicts the true titer that is achievable. This places the value of X_S within the range of 1-2 g/L for the range of t_b we consider, which is the range of titers commonly seen in practice at commercial CB cultivation facilities.

Table A.3: General process information[2, 3, 4, 5]

Parameter	Value	Units	Notes
m_M	20830	tonnes/yr	Manure feed flowrate
m_P	9.64	tonnes/yr	P in SLS extrudate
m_N	10.60	tonnes/yr	N in SLS extrudate
m_W	18030	tonnes/yr	Water in SLS extrudate
Y_{Xv}	2.02×10^{-9}	kg/µmol	Biomass yield per photon
m_v	255	µmol/kg·s	CB cell maintenance light needs
η	0.24	—	Photosynthetic effiiciency of CB
X_0	0.03	g/L	Initial CB concentration
I_0	350	µmol/m^2·s	Light intensity
$\left(\frac{S}{V}\right)_0$	15.4	m^{-1}	$S:V$ ratio of base design
σ	70	kg/m^2	Areal density of b-PBRs
t_{b0}	30	days	Batch time of base design
ρ_{P0}	0.023	g P/ g CB	P density of base design
ρ_N	0.05	g N/g CB	N denisty of CB
x_{UN}	0.467	—	Mass fraction of nitrogen in urea
PI	596.2	—	Cost index for 2020
T	10	years	Lifetime of ReNuAl process
p_{EL}	0.11	USD/kW-hr	Electricity price
p_{NG}	5.84	USD/1000 SCF	Natural gas price
ρ_W	1000	kg/m^3	Water density
ρ_{BG}	1.2	kg/m^3	Biogas density
ρ_{NG}	0.72	kg/m^3	Natural gas density

Table A.4: Product yield factors

Product	Unit Operation	Yield	Notes
CH_4[156]	Anaerobic Digester	3.09×10^{-2} kg/kg manure	CH_4 from biogas production
CO_2[156]	Anaerobic Digester	1.66×10^{-2} kg/kg manure	CO_2 from biogas production
H_2S[156]	Anaerobic Digester	1.14×10^{-4} kg/kg manure	H_2S from biogas production
Electricity[157]	Electricity Generator	4.33 kW-hr/kg CH_4	assumes gas turbine efficiency of 0.3 and CH_4 energy content of 1000 BTU/scf
Bedding[158]	Solid-Liquid Separator	0.09 kg/kg manure	P and N are assumed to be uniformly distributed in digestate liquid and solid fractions
Primary Dewatering Product[159]	Lamella Clarifier	1.60×10^{-2} kg/L	–
Secondary Dewatering Product[160]	Pressure Filter	0.27 kg/L	–

Table A.5: Capital costs of process units

Item	c'_k	Units	PI'_k	Size Ratio	ϕ_k	Notes
Anaerobic Digester (AD)[161]	$937.1\,(m_{in})^{0.6} + 75355$	USD	539.1	–	–	m_{in} denotes unit capacity (tonne/yr)
Solid-Liquid Separator (SLS)[161]	$14.9 m_{in} + 1786.9 \log(m_{in}) - 9506.6$	USD	556.7	–	–	m_{in} denotes unit capacity (lb/hr)
Electricity Generator[161]	$0.67 c_{AD} x_{CH4}$	USD	539.1	–	–	c_{AD} denotes cost of AD (USD) x_{CH4} denotes fraction of biogas produced used for electricity
H_2S Scrubber[162]	348	USD	521.9	2.59×10^1	0.6	–
CO_2 Scrubber[163]	13.1	MMUSD	444.2	4.37×10^{-4}	0.8	–
Photobioreactors[164, 165]	279	USD	556.8	1.08×10^5	0.6	cost of structural system, bag costs are included in TOC
Flocculation Tank[159]	0.115	MMUSD	585.7	1.57×10^{-3}	0.6	Scaling is in terms of CB mass
Lamella Clarifier[159]	2.50	MMUSD	585.7	1.57×10^{-3}	0.6	Scaling is in terms of CB mass
Pressure Filter[160]	0.137	MMUSD	381.8	2.39×10^{-1}	0.6	–
Dryer[160]	0.706	MMUSD	539.1	2.20×10^3	0.6	–

Table A.6: Variable operating costs of process units

Item	Operating Cost	Units	Utilities	Notes
Anaerobic Digester[161]	$0.096 c_{AD}$	USD/yr	electricity	c_{AD} denotes cost of AD (USD)
Solid-Liquid Separator[161]	$0.488 m_{in} + 0.1 c_{SLS}$	USD/yr	electricity	m_{in} and c_{SLS} denote capacity and cost of SLS (USD)
H_2S Scrubber[162]	66.7	USD/tonne biogas	activated carbon	gas removal via carbon bed adsorption
CO_2 Scrubber[163]	40.0	USD/tonne CO_2	amine solution, steam	gas removal via amine scrubbing
Photobioreactors[164]	12100	USD/acre/yr	electricity, bags, urea, water	–
Flocculation Tank[159]	100	USD/tonne CB	electricity	source includes cost of chemical flocculant
Lamella Clarifier[159]	0.43	USD/tonne CB	electricity	–
Pressure Filter[160]	2.06	USD/tonne CB	electricity	–
Dryer[160]	19.3	USD/tonne water	natural gas	basis is in terms of water removed

A.1.3 *Reactor Network System Model*

Exact Model

The reactor system consists of a pair of CSTRs operating at steady-state and connected in series. In the first reactor, reactant A is converted into a desired product P, which can react further to form an undesired product U. An additional reactant D reacts with U to form A, and is fed to the first reactor to reduce the amount of U formed. In order to further reduce the amount of U present and increase the value of the product stream, the outlet of the first reactor is then fed to a second reactor along with an additional reactant B, which can react with U to form a secondary product E. The reaction mechanism is complex and given by:

$$2A \longleftrightarrow P \tag{A.27a}$$

$$P \longleftrightarrow 2U \tag{A.27b}$$

$$U + B \longleftrightarrow E \tag{A.27c}$$

$$U + D \rightarrow 2A \tag{A.27d}$$

The rates of each reaction are assumed to be elementary and thus:

$$r_1 = k_1 C_A^2 - k_{1r} C_P \tag{A.28a}$$

$$r_2 = k_2 C_P - k_{2r} C_U^2 \tag{A.28b}$$

$$r_3 = k_3 C_U C_B - k_{3r} C_E \tag{A.28c}$$

$$r_4 = k_4 C_U C_D \tag{A.28d}$$

where C_i is the concentration of species i and k_j and k_{jr} are the forward and reverse rate constants of the j^{th} reaction. In our analysis we assumed that $k_{jr} = 0.01 k_j$, indicating that

the forward reaction is favored. The material balances are:

$$0 = F_{in}C_{A_{in}} - F_{out}C_A - 2(r_1 - r_4)V \tag{A.29a}$$

$$0 = F_{in}C_{P_{in}} - F_{out}C_P + (r_1 - r_2)V \tag{A.29b}$$

$$0 = F_{in}C_{U_{in}} - F_{out}C_U + (2r_2 - r_3 - r_4)V \tag{A.29c}$$

$$0 = F_{in}C_{B_{in}} - F_{out}C_B - r_3V \tag{A.29d}$$

$$0 = F_{in}C_{E_{in}} - F_{out}C_E + r_3V \tag{A.29e}$$

$$0 = F_{in}C_{D_{in}} - F_{out}C_D - r_4V \tag{A.29f}$$

where F_{in} and F_{out} are the volumetric flowrates of the reactor feed and outlet respectively, $C_{i_{in}}$ is the concentration of species i in the feed stream, and V is the volume of the CSTR. The reactions in (A.27) are assumed to be exothermic and a cooling jacket is used to remove excess heat and control the temperature inside of the reactors. The jacket uses a fluid entering at a temperature T_{ic} and flowing at a mass flowrate of $\dot{m}_c$ as the coolant. The coolant flowrate required to maintain the desired temperature can be determined from the reactor energy balance:

$$H_{in} = \rho C_p F T_{in} \tag{A.30a}$$

$$H_{out} = \rho C_p F T \tag{A.30b}$$

$$\dot{Q} = -r_1 V \Delta H_1 - r_2 V \Delta H_2 - r_3 V \Delta H_3 - r_4 V \Delta H_4 \tag{A.30c}$$

$$\dot{m}_c = \frac{H_{in} - H_{out} + \dot{Q}}{C_{pc}(T_{oc} - T_{ic})} = \frac{\rho C_p F(T_{in} - T) + \dot{Q}}{C_{pc}(T_{oc} - T_{ic})} \tag{A.30d}$$

where T_{in}, is the temperature of the inlet stream, ΔH_j is the heat of reaction for the j^{th} reaction, and C_{pc} is the specific heat capacity of the coolant. Additionally, we assume that reactions do not change the heat capacity C_{pin} or density ρ_{in} of the reactor inlet. This allows us to set $C_p = C_{pin} = C_{pout}$, $\rho = \rho_{in} = \rho_{out}$, and $F = F_{in} = F_{out}$. The relation

between the rate constants and temperature is described by the Arrhenius equation:

$$k = k_0 \exp\left(\frac{-E_A}{RT}\right) \tag{A.31}$$

where k_0 is the pre-exponential factor, E_A is the activation energy of the reaction, and R is the universal gas constant.

The outlet of the second reactor is fed to series of flash separation units to recover the products from the effluent stream. Product E is recovered in the vapor fraction of the first vessel as stream $\dot{v}_1$, and product P is recovered in the liquid fraction of the second vessel as stream $\dot{l}_2$. The relative volatility of the chemicals is set with respect to the vapor-liquid equilibrium ratio K_P; compositions and flows for the exiting streams can be determined from the following vapor-liquid equilibrium calculation:

$$x_i = \frac{z_i}{f\left(K_P \alpha_i - 1\right) + 1} \tag{A.32a}$$

$$y_i = K_p \alpha_i x_i \tag{A.32b}$$

where z_i, x_i, and y_i are the molar fractions of species i in the feed, liquid, and vapor streams respectively. The relative volatility of each chemical is denoted by α_i and f is the fraction (on a molar basis) of the feed that exits the vessel in the vapor stream. We set f according to the molar fraction of the recovered product in the feed, $f = z_E$ for the first vessel, and $f = 1 - z_P$ for the second vessel. The energy required to vaporize the desired fraction of the flash's feed was supplied by a heater that uses steam as the heating agent. The required flowrate of steam, $\dot{m}_{stm}$, was determined from the flash vessel energy balance

$$\dot{Q}_{vap} = \sum_{i \in \{A,P,U,B,E,D\}} L_i y_i \dot{v} \tag{A.33a}$$

$$\dot{m}_{stm} = \frac{\dot{Q}_{vap}}{L_{H_2O}} \tag{A.33b}$$

where L_i and L_{H_2O} are the latent molar heat of species i and water, respectively.

The performance of the system is expressed as a cost function (negative profit) that measures the quality of the product streams along with the corresponding utility requirements at various temperatures and is formulated as:

$$f_1(T_1, T_2) = \sum_{i \in \{A,P,U,B,E,D\}} w_i y_{i1} \dot{v}_1 + \sum_{i \in \{A,P,U,B,E,D\}} w_i x_{i2} \dot{l}_2 + \sum_{i \in \{A,B,D\}} w_i F_i C_{i0} \tag{A.34a}$$

$$f_2(T_1, T_2) = w_c \left(\dot{m}_c + \dot{m}'_c\right) + w_{stm} \left(\dot{m}_{stm} + \dot{m}'_{stm}\right) \tag{A.34b}$$

$$f(T_1, T_2) = f_1(T_1, T_2) + f_2(T_1, T_2) \tag{A.34c}$$

where T_1 and T_2 are the operating temperatures of the first and second reactor. The molar fraction of species i in first product stream, $\dot{v}_1$, is denoted by y_{i1}, and x_{i2} is the molar fraction of species i in the second product stream $\dot{l}_2$. The price of species i is represented by w_i, and w_c and w_{stm} are the costs of the cooling and heating utilities respectively. The cost of the reagents supplied to the network is captured by the final term in (A.34a) where F_i and C_{i0} are the volumetric flow rate and inlet concentration respectively of species i into the process.

Reference Model

By substituting the Arrhenius expression (A.31) into the rate expressions (A.28), we can determine that reaction rates are functions of temperature and concentration; this is a major source of nonlinearity in the system. We draw inspiration from the use of inferential sensors that are used in industry to correlate the rates directly to temperature (bypassing concentrations) in order to develop a reference model. Specifically, in our reference model we develop a polynomial function that approximates the dependence of the rate on the temperature. We develop our polynomial model based on the following transformation of the rate expression:

$$\log r = \frac{-E_A}{R} \frac{1}{T} + n \log C \tag{A.35}$$

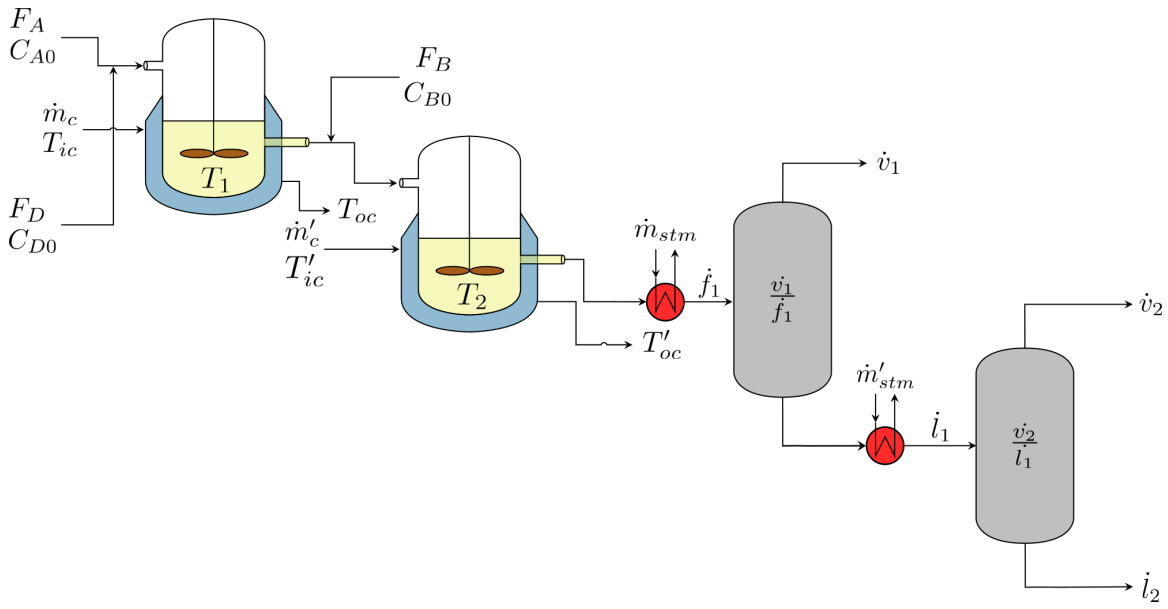

Figure A.3: Schematic diagram of the serial CSTR reactor system and product recovery system

where we ignore the reverse reaction due to the comparatively small k_r values used. From the mass balances, we can also determine that the concentration is an implicit function of the temperature. We choose to capture this relation using higher-order polynomials. During our analysis, we determined that a third-order polynomial provided satisfactory performance, resulting in the following approximation for the rate expression:

$$\log r = \theta_1 \left(\frac{1}{T}\right) + \theta_2 \left(\frac{1}{T}\right)^2 + \theta_3 \left(\frac{1}{T}\right)^3 + \theta_0 \tag{A.36}$$

where θ_0, θ_1, θ_2, and θ_3 are the model coefficients. Using (A.36), we can rewrite the material balances purely as functions of temperature and obtain the following expressions

for the various species concentrations:

$$C_A = C_{A_{in}} - 2(r_1(T) - r_4(T))\frac{V}{F} \tag{A.37a}$$

$$C_P = C_{P_{in}} + (r_1(T) - r_2(T))\frac{V}{F} \tag{A.37b}$$

$$C_U = C_{U_{in}} + (2r_2(T) - r_3(T) - r_4(T))\frac{V}{F} \tag{A.37c}$$

$$C_B = C_{B_{in}} - r_3(T)\frac{V}{F} \tag{A.37d}$$

$$C_E = C_{E_{in}} + r_3(T)\frac{V}{F} \tag{A.37e}$$

$$C_D = C_{D_{in}} - r_4(T)\frac{V}{F} \tag{A.37f}$$

we then substitute these values into a performance function similar to (A.34) to obtain the reference model for the system g. This reference model can be seen as an approximate physical model of the real system (captured by the exact model) and is much easier to evaluate. However, because this model is comprised of a complex set of algebraic equations, we further approximate the dependence of the performance function on the temperatures using a GP model.

Appendix B

SUPPLEMENTARY INFORMATION

This appendix presents supplementary information from select sections of this dissertation.

B.1 Setup of Intermediate Function Gaussian Process Models

B.1.1 *Chemical Process Optimization Study*

In the composite function BO problem, we treat the reactor and separator as black-boxes. The behavior of these units is approximated using a set of GP surrogate models, $\mathcal{GP}_y^\ell$, that estimate the following intermediate functions, $y(x)$: the duties of RX-1 and SEP-1, $\dot{Q}_4$ and $\dot{Q}_5$ respectively, the product to purge ratio of A, the feed to purge ratio of B, and purge to product ratio of C. We define these last three variables as:

$$\eta_A = \frac{\psi_A F_S}{\xi_A F_P} \tag{B.1a}$$

$$\eta_B = \frac{\xi_B F_P}{F_B} \tag{B.1b}$$

$$\eta_C = \frac{\xi_C F_P}{\psi_C F_S} \tag{B.1c}$$

where F_P is the flowrate of the purge and ξ_i is the fraction of species i in this stream. These intermediates have the following lower and upper feasibility bounds:

$$\hat{l}_y = [10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}] \tag{B.2a}$$

$$\hat{u}_y = [1, \infty, \infty, \infty, \infty] \tag{B.2b}$$

Because B is essentially non-condensable and only present in trace amounts in the product stream, $1 - \eta_B$ is equal to the conversion of B in the reactor. As a result, the generation rates of A, B, and C can be calculated from this variable:

$$\dot{r}_B = -(1 - \eta_B) F_B \tag{B.3a}$$

$$\dot{r}_A = \frac{\nu_A}{\nu_B} \dot{r}_B \tag{B.3b}$$

$$\dot{r}_C = \frac{\nu_C}{\nu_B} \dot{r}_B \tag{B.3c}$$

Note that the negative sign in (B.3a) is due to the fact that B consumed by the reaction. Combining these flowrates with η_A and η_C we can calculate the flowrates and compositions of the remaining streams. The purge and recycle streams are specified by:

$$\xi_A F_P = \frac{F_A + \dot{r}_A}{1 + \eta_A} \tag{B.4a}$$

$$\xi_B F_P = \eta_B F_B \tag{B.4b}$$

$$\xi_C F_P = \frac{\eta_C \dot{r}_C}{1 + \eta_C} \tag{B.4c}$$

$$F_P = \xi_A F_P + \xi_B F_P + \xi_C F_P \tag{B.4d}$$

$$F_R = \frac{F_P R}{1 - R} \tag{B.4e}$$

where F_R is flowrate of the recycle stream; note that the purge and recycle streams have the same composition. The product stream can similarly be defined:

$$\psi_A F_S = \frac{\eta_A(F_A + \dot{r}_A)}{1+\eta_A} \tag{B.5a}$$

$$\psi_B F_S = 0 \tag{B.5b}$$

$$\psi_C F_S = \frac{\dot{r}_C}{1+\eta_C} \tag{B.5c}$$

$$F_S = \psi_A F_S + \psi_B F_S + \psi_C F_S \tag{B.5d}$$

$$\tag{B.5e}$$

Combining these expressions with the compressor and heater models (which are assumed to be white-boxes) and the GP estimates for $\dot{Q}_4$ and $\dot{Q}_5$, we can calculate all of the values in (B.1).

Due to the presence of the recycle stream, we initially surmised that η_B should be a function of the five inputs as it is essentially a measure of the extent of the reaction. However, automatic relevance determination (ARD) [143] showed that P_S does not appear to have much of an effect on η_B. This is likely due to fact that T_S is able to capture most of the variability from the separation process. As a result, the GP model of η_B takes only T_{RX}, P_{RX}, R, and T_S as inputs. We then nest η_B within the models of η_A and η_C. These measure the performance of the separator, which is dependent on the composition of the separator feed in addition to T_S and P_S. As the reactor outlet is the separator feed and is directly dependent on the value of η_B, η_A and η_C should exhibit a similar dependence. This allows us to avoid having to explicitly consider the recycle fraction and reactor temperature and pressure (all of which affect the reactor outflow), thereby reducing the input dimension of the GP models for these intermediates from five to three. Using a similar approach, we determine that the reactor heat duty can effectively be modeled using T_{RX}, P_{RX}, η_A and η_B, as the latter two variables are closely correlated to flowrate and composition of the recycle stream, while the former two establish the equilibrium point of the reactor.

Similarly the GP model of $\dot{Q}_5$ uses T_{RX}, R, and η_B along with T_S, and P_S as these inputs are closely correlated to the flowrate and composition of the separator feed as well as the simple and latent heat released by condensation of the product stream. Note that this approach highlights the high degree of customization composite function BO affords in the selection of GP model inputs. In addition to improving model quality, this can allow BO to be extended to high-dimensional systems where standard BO runs into scalability issues as the individual intermediates do not have to be modeled with the full set of inputs.

B.1.2 *Photobioreactor Design Study*

Because of the novelty of the presented CB cultivation system, mature models for this unit operation are not yet available. The remaining systems (AD, biogas purfication, and CB harvesting) are established technologies and models of these units can be found in the literature. As a result, we choose to treat these systems as white-boxes and only model the b-PBR as a black-box. The design variables of interest are the reactor surface area to volume ratio, $\frac{S}{V}$, batch time, t_b, and the phosphorus cell density of the CB, ρ_P.

The intermediate functions we choose to estimate are the required reactor volume V and the CB titer at harvest time X; we define their feasibility bounds to be:

$$\hat{l}_y = [10^{-6}, 10^{-6}] \tag{B.6a}$$

$$\hat{u}_y = [10^{6}, 10] \tag{B.6b}$$

Using these variables, we can fully specify the outlet flow and size of the b-PBRs:

$$SA = \frac{\rho_W V}{\sigma} \tag{B.7a}$$

$$m_{CB} = X\left(\frac{V}{t_b/365}\right) \tag{B.7b}$$

$$m_{PBR} = \rho_W\left(\frac{V}{t_b/365}\right) \tag{B.7c}$$

where m_{PBR} is the mass flow rate out of the reactor. Note that we approximate the density of the culture as ρ_W due to the relatively low concentration of the CB; the division of t_b by 365 is done to convert this qunatity from days to years. The GP models developed for X and V both take in the full set of inputs ($\frac{S}{V}$, t_b, and ρ_P). This is due to the fact that these variables have a significant impact on the througput and size of the reactor. As a result, this case study demonstrates how composite function BO can be used to select sample points that can be used to develop a surrogate models for a unit of interest while driving the system the unit is a part of to an optimal configuration. This essentially allows us to complete two tasks at once and significantly reduces the probability of sampling at points in highly sub-optimal areas where the unit would likely never operate. As a result, the developed model is highly refined in the regions around the located optima and, therefore, is arguably more useful.

BIBLIOGRAPHY

[1] Ranjeet Kumar, Michael J Wenzel, Mohammad N ElBsat, Michael J Risbeck, Kirk H Drees, and Victor M Zavala. Stochastic model predictive control for central HVAC plants. *Journal of Process Control*, 90:1–17, 2020.

[2] Rebecca A. Larson, Mahmoud Sharara, Laura W. Good, Pam Porter, Troy Runge, Victor Zavala, Apoorva Sampat, and Amanda Smith. Evaluation of manure storage capital projects in the Yahara River watershed. Technical report, University of Wisconsin-Extension, University of Wisconsin-Madison College of Agricultural and Life Sciences, Biological Systems Engineering, 2016.

[3] Hui Wang, Horacio A. Aguirre-Villegas, Rebecca A. Larson, and Asli Alkan-Ozkaynak. Physical properties of dairy manure pre- and post-anaerobic digestion. *Applied Sciences*, 9(13):2703, 2019.

[4] Economic Indicators. *Chemical Engineering*, 128(1):56, 2021.

[5] Ryan L Clark, Laura L McGinley, Hugh M Purdy, Travis C Korosh, Jennifer L Reed, Thatcher W Root, and Brian F Pfleger. Light-optimized growth of cyanobacterial cultures: Growth phases and productivity of biomass and secreted molecules in light-limited batch growth. *Metabolic engineering*, 47:230–242, 2018.

[6] Kenneth Lange. *Optimization*. Springer Texts in Statistics. Springer, New York, NY, 2nd edition, 2013.

[7] Jasbir S. Arora. Chapter 10 - Numerical methods for unconstrained optimum design. In Jasbir S. Arora, editor, *Introduction to Optimum Design (Third Edition)*, pages 411–441. Academic Press, Boston, MA, third edition edition, 2012.

[8] Paul T. Boggs and Jon W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1995.

[9] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[10] Andreas Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2002.

[11] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-free Optimization*. SIAM, 2009.

[12] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 01 1965.

[13] Robert Hooke and T. A. Jeeves. "direct search" solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8(2):212–229, 1961.

[14] Pradnya A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265, 2016.

[15] L.N. Vicente and Custódio A.L. Analysis of direct searches for discontinuous functions. *Mathematical Programming*, 133(1):299–325, 2012.

[16] Robert Michael Lewis, Virginia Torczon, and Michael W. Trosset. Direct search methods: Then and now. *Journal of Computational and Applied Mathematics*, 124(1):191–207, 2000.

[17] Virginia Joanne Torczon. *Multidirectional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Rice University, Houston, TX, 1989.

[18] W.H. Swann. Direct search methods. In W. Murray, editor, *Numerical Methods for Unconstrained Optimization*, pages 13–28. Academic Press, New York, NY, 1972.

[19] Juliane Müller. Surrogate model algorithms for computationally expensive black-box global optimization problems. 2012.

[20] Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[21] Jeffrey Larson, Matt Menickelly, and Stefan M. Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.

[22] Ivo D. Dinov. *Black Box Machine-Learning Methods: Neural Networks and Support Vector Machines*, pages 383–422. Springer International Publishing, Cham, Switzerland, 2018.

[23] G. E. P. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society*, 13(1):1–38, 1951.

[24] H.-M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227, 2001.

[25] Harold J. Kushner. A new method for locating the maximum point of an arbitrary multi-peak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

[26] V Saltenis. On a method of multi-extremal optimization. *Automatic Control and Computer Sciences (Avtomatika i Vychislitelnayya Tekchnika)*, 5(3):33–38, 1971.

[27] Jonas Mockus. Bayesian methods of search for an extremum. *Automatic Control and Computer Sciences (Avtomatika i Vychislitelnayya Tekchnika)*, 6(3):53–62, 1972.

[28] R. Garnett. *Baysian Optimization*. Cambridge University Press, 2023.

[29] Allen Jonathan Román, Shiyi Qin, Julio C. Rodríguez, Leonardo D. González, Victor M. Zavala, and Tim A. Osswald. Natural rubber blend optimization via data-driven modeling: The implementation for reverse engineering. *Polymers*, 14(11):2262, 2022.

[30] Runzhe Liang, Haoyang Hu, Yueheng Han, Bingzhen Chen, and Zhihong Yuan. Capbo: A cost-aware parallelized Bayesian optimization method for chemical reaction optimization. *AIChE Journal*, 70(3):e18316, 2024.

[31] Roman Marchant and Fabio Ramos. Bayesian optimisation for informative continuous path planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6136–6143, 2014.

[32] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in Bayesian optimization. *ACM Computing Surveys*, 55(13s):287, 2023.

[33] Joel A. Paulson and Calvin Tsay. Bayesian optimization as a flexible and efficient design framework for sustainable process systems. *arXiv preprint arXiv:2401.16373*, 2024.

[34] Tom Savage, Nausheen Basha, Jonathan McDonough, Omar K. Matar, and Ehecatl Antonio del Rio Chanona. Multi-fidelity data-driven design and analysis of reactor and tube simulations. *Computers & Chemical Engineering*, 179:108410, 2023.

[35] Zahra Zanjani Foumani, Mehdi Shishehbor, Amin Yousefpour, and Ramin Bostanabad. Multi-fidelity cost-aware Bayesian optimization. *Computer Methods in Applied Mechanics and Engineering*, 407:115937, 2023.

[36] Peter Z. G. Qian and C. F. Jeff Wu. Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. *Technometrics*, 50(2):192–204, 2008.

[37] R. Astudillo and P. Frazier. Bayesian optimization of composite functions. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 354–363. PMLR, 09–15 Jun 2019.

[38] J.A. Paulson and C. Lu. COBALT: COnstrained Bayesian optimizAtion of computaionaLly expensive grey-box models exploiting derivaTive information. *Computers & Chemical Engineering*, 160:107700, 2022.

[39] M. Balandat, B. Karrer, D.R. Jiang, S. Daulton, B. Letham, A.G. Wislon, and E. Bashky. BOTORCH: A framework for efficient Monte-Carlo Bayesian optimization. In *Proceedings of the 34th Conference International Conference on Neural Information Processing Systems*, NIPS '20, pages 21524–21538. Curran Associates Inc., Dec 2020.

[40] Wenjie Xu, Yuning Jiang, Bratislav Svetozarevic, and Colin N. Jones. Bayesian optimization of expensive nested grey-box functions. *arXiv preprint arXiv:2306.05150*, 2023.

[41] Matthias Schonlau. *Computer experiments and global optimization*. PhD thesis, University of Waterloo, Waterloo, ON, 1997.

[42] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959. Curran Associates, Inc., 2012.

[43] José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1470–1479. JMLR.org, 2017.

[44] A multi-points criterion for deterministic parallel global optimization based on Gaussian processes. Technical report.

[45] Jian Wu and Peter I. Frazier. The parallel knowledge gradient method for batch Bayesian optimization. *arXiv preprint arXiv:1606.04414*, 2018.

[46] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Parallelised Bayesian optimisation via Thompson sampling. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 133–142. PMLR, 09–11 Apr 2018.

[47] Quoc Phong Nguyen, Zhaoxuan Wu, Bryan Kian Hsiang Low, and Patrick Jaillet. Trusted-maximizers entropy search for efficient Bayesian optimization. *arXiv preprint arXiv:2107.14465*, 2021.

[48] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3330–3338. MIT Press, 2015.

[49] Thomas Bayes. An essay towards solving a problem the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical Transactions*, 53:370–418, 1763.

[50] Yifan Wu, Aron Walsh, and Alex M. Ganose. Race to the bottom: Bayesian optimisation for chemical problems. *Digital Discovery*, 3:1086–1100, 2024.

[51] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[52] Bertil Matérn. Spatial variation : Stochastic models and their application to some problems in forest surveys and other sampling investigations. In *Messages from the State Forestry Research Institute*, volume 49, 1960.

[53] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[54] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feritas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55(1):361–387, 2016.

[55] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[56] Dustin Tran, Rajesh Ranganath, and David M. Blei. The variational Gaussian process. *arXiv preprint arXiv:1511.06499*, 2016.

[57] E. Snelson, Z. Ghahramani, and C. Rasmussen. Warped Gaussian processes. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 337–334. MIT Press, 2004.

[58] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, Toronto, ON, 1995.

[59] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *ICML*, volume 2014, pages 937–945, 2014.

[60] Jenna C Fromer, David E Graff, and Connor W Coley. Pareto optimization to accelerate multi-objective virtual screening. *Digital Discovery*, 3(3):467–481, 2024.

[61] Raul Astudillo and Peter Frazier. Bayesian optimization of function networks. *Advances in neural information processing systems*, 34:14463–14475, 2021.

[62] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

[63] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13:455–492, 1998.

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

[65] Nando De Freitas, Alex Smola, and Masrour Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. *arXiv preprint arXiv:1206.6457*, 2012.

[66] Adam D Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(10), 2011.

[67] AS Yamashita, AC Zanin, and D Odloak. Tuning of model predictive control with multi-objective optimization. *Brazilian Journal of Chemical Engineering*, 33(2):333–346, 2016.

[68] Robert W Koller, Luis A Ricardez-Sandoval, and Lorenz T Biegler. Stochastic back-off algorithm for simultaneous design, control, and scheduling of multiproduct systems under uncertainty. *AIChE Journal*, 64(7):2379–2389, 2018.

[69] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[70] Tamara G Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45(3):385–482, 2003.

[71] Jorge L Garriga and Masoud Soroush. Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research*, 49(8):3505–3515, 2010.

[72] Marco Forgione, Dario Piga, and Alberto Bemporad. Efficient calibration of embedded MPC. *arXiv preprint arXiv:1911.13021*, 2019.

[73] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pages 2171–2180. PMLR, 2015.

[74] Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, and Svetha Venkatesh. Bayesian optimization for adaptive experimental design: A review. *IEEE Access*, 8:13937–13948, 2020.

[75] Aaron Wilson, Alan Fern, and Prasad Tadepalli. Using trajectory data to improve bayesian optimization for reinforcement learning. *The Journal of Machine Learning Research*, 15(1):253–282, 2014.

[76] J. Mockus. *Bayesian Approach to Global Optimization: Theory and Applications*, volume 37. Springer Science & Business Media, 2012.

[77] Farshud Sorourifar, Naitik Choksi, and Joel A. Paulson. Computationally efficient integrated design and predictive control of flexible energy systems using multi-fidelity simulation-based Bayesian optimization. *Journal of Optimal Control Applications and Methods*, 44(4):549–576, 2023.

[78] K. Kandasamy, G. Dasarathy, J. Schnieder, and B. Pózcos. Multi-fidelity Bayesian optimisation with continuous approximations. In D. Precup and Y.W. Teh, editors, *Uncertainty in Artificial Intelligence*, volume 70 of *Proceedings of Machine Learning Research*, pages 1799–1808. PMLR, 06–11 Aug 2017.

[79] Jian Wu, Saul Toscano-Palmerin, Peter I. Frazier, and Andrew G. Wilson. Practical multi-fidelity Bayesian optimization for hyperaparameter tuning. In *Uncertainty in Artifical Intelligence*, pages 788–798. PMLR, 2020.

[80] Rommel G Regis and Christine A Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global optimization*, 31(1):153–171, 2005.

[81] Michael JD Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994.

[82] Andrew R Conn and Sébastien Le Digabel. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, 28(1):139–158, 2013.

[83] José A Caballero and Ignacio E Grossmann. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE journal*, 54(10):2633–2650, 2008.

[84] Burcu Beykal, Fani Boukouvala, Christodoulos A Floudas, Nadav Sorek, Hardikkumar Zalavadia, and Eduardo Gildin. Global optimization of grey-box computational systems using surrogate functions and application to highly constrained oil-field operations. *Computers & Chemical Engineering*, 114:99–110, 2018.

[85] Nestor V Queipo, Javier V Goicochea, and Salvador Pintos. Surrogate modeling-based optimization of SAGD processes. *Journal of Petroleum Science and Engineering*, 35(1-2):83–93, 2002.

[86] Yolanda Mack, Tushar Goel, Wei Shyy, and Raphael Haftka. Surrogate model-based optimization framework: A case study in aerospace design. In *Evolutionary computation in dynamic and uncertain environments*, pages 323–342. Springer, 2007.

[87] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[88] Florian Häse, Matteo Aldeghi, Riley J. Hickman, Loïc M. Roch, and Alán Aspuru-Guzik. GRYFFIN: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 8(3):031406, 2021.

[89] J. Zhang, S.D. Petersen, T. Radivojevic, A. Ramirez, A. Pérez-Manríquez, E. Abeliuk, B.J. Sánchez, Z. Costello, Y. Chen, M.J. Fero, H. Garcia Martin, J. Nielsen, J. D. Keasling, and M.K. Jensen. Combining mechanistic and machine learning models for predictive engineering and optimization of tryptophan metabolism. *Nature Communications*, 11(1):1–13, 2020.

[90] Elvis A Eugene, Xian Gao, and Alexander W Dowling. Learning and optimization with Bayesian hybrid models. In *2020 American Control Conference (ACC)*, pages 3997–4002. IEEE, 2020.

[91] Mina Rafiei and Luis A Ricardez-Sandoval. Stochastic back-off approach for integration of design and control under uncertainty. *Industrial & Engineering Chemistry Research*, 57(12):4351–4365, 2018.

[92] Alberto Lucchini, Simone Formentin, Matteo Corno, Dario Piga, and Sergio M Savaresi. Torque vectoring for high-performance electric vehicles: An efficient MPC calibration. *IEEE Control Systems Letters*, 4(3):725–730, 2020.

[93] Farshud Sorourifar, Georgios Makrygirgos, Ali Mesbah, and Joel A. Paulson. A data-driven automatic tuning method for MPC under uncertainty using constrained Bayesian optimization. *IFAC-PapersOnLine*, 54(3):243–250, 2021. 16th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2021.

[94] Joel A Paulson and Ali Mesbah. Data-driven scenario optimization for automated controller tuning with probabilistic performance guarantees. *IEEE Control Systems Letters*, 5(4):1477–1482, 2020.

[95] Danny Drieß, Peter Englert, and Marc Toussaint. Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations. In

2017 IEEE International Conference on Robotics and Automation (ICRA), pages 902–907. IEEE, 2017.

[96] Marcello Fiducioso, Sebastian Curi, Benedikt Schumacher, Markus Gwerder, and Andreas Krause. Safe contextual Bayesian optimization for sustainable room temperature PID control tuning. *arXiv preprint arXiv:1906.12086*, 2019.

[97] Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomlin. Goal-driven dynamics learning via Bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5168–5173. IEEE, 2017.

[98] Dario Piga, Marco Forgione, Simone Formentin, and Alberto Bemporad. Performance-oriented model learning for data-driven MPC design. *IEEE control systems letters*, 3(3):577–582, 2019.

[99] James B Rawlings, Nishith R Patel, Michael J Risbeck, Christos T Maravelias, Michael J Wenzel, and Robert D Turney. Economic MPC and real-time decision making with application to large-scale HVAC energy systems. *Computers & Chemical Engineering*, 114:89–98, 2018.

[100] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[101] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and

SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[102] Dipti Jasrasaria and Edward O Pyzer-Knapp. Dynamic control of explore/exploit trade-off in Bayesian optimization. In *Science and Information Conference*, pages 1–15. Springer, 2018.

[103] Gavin Towler and Ray Sinnott. *Chemical Engineering Design: Principles, Practice and Economics of Plant and Process Design*. Elsevier, 2012.

[104] George E. P. Box, J. Stuart Hunter, and William G. Hunter. *Statistics for Experimenters: Design, Innovation, and Discovery*. Wiley, Second edition, 2005.

[105] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[106] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[107] Remi R. Lam, Matthias Poloczek, Peter I. Frazier, and Karen E. Willcox. Advances in Bayesian optimization with applications in aerospace engineering. In *Proceedings of the AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Boston, MA, 2018. AIAA.

[108] Tijana Radivojević, Zak Costello, Kenneth Workman, and Hector Garcia Martin. A machine learning automated recommendation tool for synthetic biology. *Nature Communications*, 11(1):4879, 2020.

[109] Rémy Priem, Nathalie Bartoli, and Youssef Diouane. On the use of upper trust bounds in constrained Bayesian optimization infill criterion. In *AIAA Aviation 2019 Forum*, pages 1–10, Dallas, United States, June 2019.

[110] Ankush Chakrabarty, Scott A. Bortoff, and Christopher R. Laughman. Simulation failure-robust Bayesian optimization for data-driven parameter estimation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(5):2629–2640, 2023.

[111] B. Sohlberg and E.W. Jacobsen. Grey-box modeling – Branches and experiences. *IFAC Proceedings Volumes*, 41(2):11415–11420, 2008. 17th IFAC World Congress.

[112] F. Boukouvala and C.A. Floudas. ARGONAUT: AlgoRithms for Global Optimization of coNstrAined grey-box compUTational problems. *Optimization Letters*, 11(5):895–913, 2017.

[113] Ishan Bajaj, Shachit S. Iyer, and M.M. Faruque Hasan. A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point. *Computers & Chemical Engineering*, 116:306–321, 2018.

[114] Burcu Beykal, Fani Boukouvala, Christodoulos A. Floudas, and Efstratios N. Pistikopoulos. Optimal design of energy systems using constrained grey-box multi-objective optimization. *Computers & Chemical Engineering*, 116:488–502, 2018.

[115] Q. Lu, L.D. González, R. Kumar, and V.M. Zavala. Bayesian optimization with reference models: A case study in MPC for HVAC central plants. *Computers & Chemical Engineering*, 154:107491, 2021.

[116] A.K. Uhrenholt and B.S. Jensen. Efficient Bayesian optimization for target vector estimation. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2661–2670. PMLR, 16–18 Apr 2019.

[117] Congwen Lu and Joel A. Paulson. No-regret constrained Bayesian optimization of noisy and expensive hybrid models using differentiable quantile function approximations. *Journal of Process Control*, 131:103085, 2023.

[118] R. Astudillo and P.I. Frazier. Thinking inside the box: A tutorial on grey-box Bayesian optimization. In *Proceedings of the 2021 Winter Simulation Conference*, December 2021.

[119] Leonardo D. González and Victor M. Zavala. BOIS: Bayesian Optimization of Interconnected Systems. *arXiv preprint arXiv:2311.11254*, 2023.

[120] Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *arXiv preprint arXiv:1106.6251*, 2012.

[121] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. Remarks on multi-output Gaussian process regression. *Knowedge-Based Systemsl*, 144:102–121, 2018.

[122] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2008.

[123] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983.

[124] J.C. Thompson, V.M. Zavala, and O.S. Venturelli. Integrating a tailored recurrent neural network with Bayesian experimental design to optimize microbial community functions. *PLOS Computational Biology*, 19(9):1–25, 2023.

[125] Thanh Nhat Nguyen, Thuy Tran Phuong Nhat, Ken Takimoto, Ashutosh Thakur, Shun Nishimura, Junya Ohyama, Itsuki Miyazato, Lauren Takahashi, Jun Fujima, Keisuke Takahashi, and Toshiak Taniike. High-throughput experimentation and catalyst informatics for oxidative coupling of methane. *ACS Catalysis*, 10(2):921–932, 2012.

[126] Steven M. Mennen, Carolina Alhambra, C. Liana Allen, Mario Barberis, Simon Berritt, Thomas A. Brandt, Andrew D. Campbell, Jesús Castañón, Alan H. Cherney, Melodie Christensen, David B. Damon, J. Eugenio de Diego, Susana García-Cerrada, Pablo García-Losada, Rubén Haro, Jacob Janey, David C. Leitch, Ling

Li, Fangfang Liu, Paul C. Lobben, David W. C. MacMillan, Javier Magano, Emma McInturff, Sebastien Monfette, Ronald J. Post, Danielle Schultz, Barbara J. Sitter, Jason M. Stevens, Iulia I. Strambeanu, Jack Twilton, Ke Wang, and Matthew A. Zajac. The evolution of high-throughput experimentation in pharmaceutical development and perspectives on the future. *Organic Process Research & Development*, 23(6):1213–1242, 2019.

[127] Michael J. Smanski, Hui Zhou, Ben Shen Claesen, Michael A. Fischbach, and Christopher A. Voigt. Synthetic biology to access and expand nature's chemical diversity. *Nature Reviews Microbiology*, 14:135–149, 2016.

[128] Joshua A. Selekman, Jun Qiu, Kristy Tran, Jason Stevens, Victor Rosso, Eric Simmons, Yi Xiao, and Jacob Janey. High-throughput automation in chemical process development. *Annual Review of Chemical and Biomolecular Engineering*, 8:525–547, 2017.

[129] Michae Shevlin. Practical high-throughput experimentation for chemists. *ACS Medicinal Chemistry Letters*, 8(6):601–607, 2017.

[130] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[131] Andrew L. Ferguson and Keith A. Brown. Data-driven design and autonomous experimentation in soft and biological materials engineering. *Annual Review of Chemical and Biomolecular Engineering*, 13, 2022.

[132] Arpan Biswas, Anna N. Morozovska, Maxim Ziatdinov, Eugene A. Eliseev, and Sergei V. Kalinin. Multi-objective Bayesian optimization of ferroelectric materials with interfacial control for memory and energy storage applications. *Journal of Applied Physics*, 130(20):204102–1–204102–1, 2021.

[133] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-

suited to parallelize optimization. In *Computation Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

[134] Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(119):4053–4103, 2014.

[135] Sébastian Marmin, Clément Chevalier, and David Ginsbourger. Differentiating the multipoint expected improvement for optimal batch design. In *Machine Learning, Optimization, and Big Data*. 2015.

[136] M. Todd Young, Jacob Hinkle, Arvind Ramanathan, and Ramakrishnan Kannan. Hyperspace: Distributed Bayesian hyperparameter optimization. In *2018 30th International Symposium on Computer Architecture and High Performance Computing*, pages 339–347. IEEE, 2018.

[137] James T. Wilson, Riccardo Moriconi, Frank Hutter, and Marc P. Deisenroth. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

[138] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Parallel algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 55–70. Springer, 2012.

[139] M. Todd Young, Jacob D. Hinkle, Ramakrishnan Kannan, and Arvind Ramanathan. Distributed Bayesian optimization of reinforcement learning algorithms. *Journal of the Parallel and Distributed Computing*, 139(1):43–52, 2020.

[140] Jaspar Snoek, Hugo Larochelle, and Ryan P. Adams. Spearmint. https://github.com/HIPS/Spearmint, 2012.

[141] Sungho Shin, Victor M Zavala, and Mihai Anitescu. Decentralized schemes with

overlap for solving graph-structured optimization problems. *IEEE Transactions on Control of Network Systems*, 7(3):1225–1236, 2020.

[142] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.

[143] Christopher K. I. Williams and Carl E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, volume 8, pages 514–520. MIT Press, 1996.

[144] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

[145] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. *Advances in neural information processing systems*, 29, 2016.

[146] Kirthevasan Kandasamy, Gautam Dasarathy, Junier Oliva, Jeff Schneider, and Barnabas Poczos. Multi-fidelity Gaussian process bandit optimisation. *Journal of Artificial Intelligence Research*, 66:151–196, 2019.

[147] Kirthevasan Kandasamy, Gautam Dasarathy, Barnabas Poczos, and Jeff Schneider. The multi-fidelity multi-armed bandit. *Advances in neural information processing systems*, 29, 2016.

[148] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on Bayesian neural networks—A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[149] Bowen Lei, Tanner Quinn Kirk, Anirban Bhattacharya, Debdeep Pati, Xiaoning Qian, Raymundo Arroyave, and Bani K Mallick. Bayesian optimization with adap-

tive surrogate models for automated experimental design. *Npj Computational Materials*, 7(1):194, 2021.

[150] Kai Chen, Twan van Laarhoven, and Elena Marchiori. Gaussian processes with skewed Laplace spectral mixture kernels for long-term forecasting. *Machine Learning*, 110:2213–2238, 2021.

[151] Markus Grimm, Sébastien Paul, and Pierre Chainais. Process-constrained batch Bayesian approaches for yield optimization in multi-reactor systems. *Computers & Chemical Engineering*, page 108779, 2024.

[152] David Eriksson and Matthias Poloczek. Scalable constrained Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 730–738. PMLR, 2021.

[153] Luca Mencarelli, Qi Chen, Alexandre Pagot, and Ignacio E Grossmann. A review on superstructure optimization approaches in process system engineering. *Computers & Chemical Engineering*, 136:106808, 2020.

[154] Ruth Misener and Lorenz Biegler. Formulating data-driven surrogate models for process optimization. *Computers & Chemical Engineering*, 179:108411, 2023.

[155] Yoel Cortes-Peña, Deepak Kumar, Vijay Singh, and Jeremy S. Guest. BioSTEAM: A fast and flexible platform for the design, simulation, and techno-economic analysis of biorefineries under uncertainty. *ACS Sustainable Chemistry & Engineering*, 8(8):3302–3310, 2020.

[156] Elmar Dimpl. Small-scale electricity generation from biomass: Experience with small-scale technologies for basic energy supply. Technical report, German Federal Ministry for Economic Cooperation and Development (BMZ), 08 2010.

[157] United States Department of Energy (DoE). How Gas Turbine Power Plants Work,

2022. Data retrieved from the Office of Fossil Energy and Carbon Management, https://www.energy.gov/fecm/how-gas-turbine-power-plants-work.

[158] Horacio A Aguirre-Villegas, Rebecca A Larson, and Mahmoud A Sharara. Anaerobic digestion, solid-liquid separation, and drying of dairy manure: Measuring constituents and modeling emission. *Science of the total environment*, 696:134059, 2019.

[159] Jonathan N. Rogers, Julian N. Rosenberg, Bernardo J. Guzman, Victor H. Oh, Luz Elena Mimbela, Abbas Ghassemi, Michael J. Betenbaugh, George A. Oyler, and Marc D. Donohue. A critical analysis of paddlewheel-driven raceway ponds for algal biofuel production at commercial scales. *Algal Research*, 4:76–88, 2014.

[160] Jiaze Ma, Philip Tominac, Brian F. Pfleger, and Victor M. Zavala. Infrastructures for phosphorus recovery from livestock waste using cyanobacteria: Transportation, techno-economic, and policy implications. *ACS Sustainable Chemistry & Engineering*, 9(34):11416–11426, 2021.

[161] Yicheng Hu, Horacio Aguirre-Villegas, Rebecca A. Larson, and Victor M. Zavala. Managing conflicting economic and environmental metrics in livestock manure management. *ACS ES&T Engineering*, 2(5):819–830, 2022.

[162] Suneerat Pipatmanomai, Sommas Kaewluan, and Tharapong Vitidsant. Economic assessment of biogas-to-electricity generation system with H_2S removal by activated carbon in small pig farm. *Applied Energy*, 86(5):669–674, 2009.

[163] Berhane H. Gebreslassie, Randall Waymire, and Fengqi You. Global optimization for sustainable design and synthesis of algae processing network for CO_2 mitigation and biofuel production using life cycle optimization. *AIChE Journal*, 59(5):1599–1621, 2013.

[164] Jennifer N. Clippinger and Ryan E. Davis. Techno-economic analysis for the production of algal biomass via closed photobioreactors: Future cost potential evaluated

across a range of cultivation system designs. Technical report, National Renewable Energy Laboratory, 2019.

[165] Signature Solar. EG4 BrightMount Solar Panel Ground Mount Rack Kit, 4 Panel Ground Mount, Adjustable Angle, 2022. Data retrieved from Solar Signature Mounting Hardware, https://signaturesolar.com/eg4-brightmount-solar-panel-ground-mount-rack-kit-4-panel-ground-mount-adjustable-angle.