# The Interplay of Optimization and Machine Learning to Solve Large-scale Black-box Noisy Functions

Pariyakorn Maneekul

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

Doctor of Philosophy

University of Washington

2025

Reading Committee:

Zelda B. Zabinsky, Chair

Chiwoo Park

Giulia Pedrielli

Program Authorized to Offer Degree:
Industrial and Systems Engineering

University of Washington

## Abstract

The Interplay of Optimization and Machine Learning to Solve Large-scale Black-box Noisy Functions

Pariyakorn Maneekul

Chair of the Supervisory Committee:
Zelda B. Zabinsky
Industrial and Systems Engineering

High-dimensional black-box optimization presents an increasingly prevalent challenge in modern science and engineering. This dissertation addresses this challenge through a novel interplay between optimization and machine learning methods, developing adaptive search algorithms that strike a balance between exploration, exploitation, and estimation. The proposed algorithms leverage machine learning techniques to construct surrogate models, thereby enhancing the efficiency of the optimization process.

The dissertation proposes a multi-level Partitioning and Branch-and-Bound (PBnB) algorithm designed for level-set approximation, enhancing the original PBnB algorithm significantly. This multi-level PBnB algorithm employs importance sampling to strategically identify promising subregions of the partitioned search space. Its performance is further enhanced by integrating Gaussian processes as a surrogate model to guide local sampling exploration. During the process, the target level set is approximated by classifying subregions as either pruned (no intersection with target level set), maintained (contained within target level set), or undecided. This enhanced version of the PBnB algorithm introduces an adaptive sampling probability that strategically directs samples to the most promising regions. Since this importance sampling results in dependency amongst samples, we have applied a statistical method to construct a confidence interval on the probability of correctly classifying a subregion as pruned or maintained. The contribution to the interplay of

optimization and machine learning is the local sampling within each subregion. We incorporate Gaussian processes and regularized quadratic regression, common and successful methods for prediction in machine learning for level-set approximation. The analysis of this multi-level PBnB algorithm quantifies the quality of the level set approximation by deriving probability bounds on the volume of incorrectly pruned or maintained regions, which accounts for the effects of importance sampling.

To address the challenges of high dimensionality, this dissertation introduces the Branching Adaptive Surrogate Search Optimization (BASSO) framework that conceptualizes the use of branching and surrogate modeling for black-box optimization. BASSO generalizes multi-level PBnB and adapts it to optimization as opposed to level-set approximation. A finite-time analysis of BASSO proves that the expected number of BASSO function evaluations needed to first sample a point in the global optimum vicinity is linear in dimension given that two strong assumptions are satisfied. The desired linearity result suggests an algorithm that is scalable to high dimensions in theory. This research explores several variations to implement BASSO and partially satisfy the two assumptions. In this part of the research, methods used in machine learning are introduced to improve the chance of sampling in the improving region. One BASSO implementation incorporates Gaussian processes as a surrogate model and a second uses regularized quadratic regression as a surrogate model to predict where to sample next within a subregion. The synergy between the surrogate model and the optimization algorithm work together to balance exploration and exploitation. The local surrogate model guides sampling within a subregion, while the adaptive subregion probabilities identify promising subregions. This interplay allows the system to effectively use both local subregion information (from the surrogate model) and global information (from the adaptive probabilities) to improve its search.

Numerical experiments of BASSO provide insights into the gap between theoretical ideal performance and the performance of proposed implementation with machine learning techniques to tackling high dimensional black-box problem. This dissertation also explores partitioning, cluster-

ing and decomposition as techniques for high-dimensional optimization.

While the proposed multi-level PBnB algorithm and BASSO framework focus on balancing exploration and exploitation for deterministic, black-box optimization, this dissertation also considers estimation when dealing with a noisy black-box function. The dissertation extends the Single Observation Search Algorithm (SOSA) by incorporating insight from machine learning techniques. The original neighborhood averaging technique for noisy function value estimation of SOSA is replaced with a new quadratic regression, extending the concept of basis expansion. Complementing this, the search strategy is improved by incorporating optimistic sampling, a concept drawn from reinforcement learning, to more effectively guide exploration. This research contributes to the interplay of optimization and machine learning by providing quadratic regression as an estimation method within a single-observation scheme and achieving convergence results while accounting for dependency between samples. Theoretical convergence results for this SOSA extension are presented, and numerical experiments on benchmark problems demonstrate performance gains over the baseline algorithm.

Finally, this dissertation identifies possible applications and future research opportunities arising from the interplay of optimization and machine learning in solving large-scale black-box noisy functions. This includes a discussion of quantum computing approaches for global optimization, considering both their theoretical promises and practical challenges.

# TABLE OF CONTENTS

# LIST OF FIGURES

# ACKNOWLEDGMENTS

Completing this dissertation would not have been possible without the support, guidance, and encouragement of many incredible people. I would like to take this opportunity to express my sincere gratitude.

First and foremost, I would like to express my deepest appreciation to my advisor, Professor Zelda B. Zabinsky for your mentorship, patience, and understanding throughout my PhD. I am incredibly grateful for your guidance, and it has been an honor to be your student. I would also like to thank my thesis committee members: Professor Giulia Pedrielli, Professor Chiwoo Park, and Professor Eardi Lila for your invaluable insights and expertise in making this dissertation complete. My gratitude also goes to Professor Seksan Kiatsupaibul for your advice, support and for generously sharing your wisdom.

To the faculties and staffs in the Department of Industrial and Systems Engineering at the University of Washington, thank you for creating a supportive academic home throughout these years. To my ISE friends and cohort, Anna, Chelsea, Lun, Jiaxin, Daniel, Tianchen, Feng, Danielle, and Serin, it has been a great pleasure to share this journey with you.

To my family, and most of all, to grandma yaya, you are the reason I found the strength to never give up. Thank you for your energy, your love, and for everything you and grandpa sacrificed to raise me up to be the person I am today. To dad and mom, thank you for your hard work to ensure my siblings and I had the best life imaginable. Everything I have accomplished is built on the foundation of your love and dedication.

My gratitude also goes to Amy and Bobby family for the warm hospitality since I arrived in US, and for taking care of me during the final stages of my dissertation writing. Amy has been a constant role model, and for that I am truly thankful. I wish your family lasting happiness and

good health.

To my friends who made Seattle feel like a home away from home, Chaonai, A Supasai, Makham, Shane and all the thai students at UW, thank you for taking care of me and filling this journey with so much joy. Finally, to my friends in Thailand, Ting, Nhae, Tum, Naree, Pinggy, Jojo, Girl, Mot, Prae, Pang, Sarunya, Pitchaya, Suphattra, and Sitanan, disregarding the distance and the challenging time zones, you always made time to nurture and support me across every possible online platform, reminding me that true friendship knows no borders.

# DEDICATION

to Preeyaporn and Sanoh Maneekul

# Chapter 1

# INTRODUCTION

Black–box optimization is broadly applicable across many domains and has been studied in multiple scholarly fields under names including Derivative–free Optimization, Global Optimization, Bayesian Optimization, Sequential Experimental Design, and assorted variants of the multi-armed bandit problem. Black-box optimization considers the design and analysis of algorithms for optimization problems where the structure of the objective function and the constraints defining a feasible solution are unknown or there is no explicit mathematical formulation. In addition to the lack of knowledge regarding the function structure, noisy black-box optimization depends on acquiring or estimating the function value given a design vector, often through a discrete-event simulation, to gain information and create a search path to the optimum. Black-box optimization often has a limited budget for evaluations due to time-consuming function evaluations and limited computational expense, especially when the objective function can only be evaluated with noise. Designing black-box optimization algorithms with a limited budget is challenging due to the trade-offs between exploration, exploitation, and estimation.

Machine learning methods can be integrated with black-box optimization algorithms to address the challenge in prediction of an unknown function, arising from expensive function evaluations. A stochastic adaptive search for black-box optimization adaptively samples points in the feasible region with a bias towards a global optimal solution and uses the information from their corresponding sampled objective functions to predict the global optimum. On the other hand, a statistical method or machine learning algorithm takes sampled pairs of feature vectors (design points) and their corresponding responses (sampled objective functions) and uses them to produce an inferred function to predict the responses of other unseen feature vectors. The two paradigms of prediction share many similarities. Both take sampled pairs of design points and their responses, and both use

the information to predict a target response. A difference between stochastic adaptive search and machine learning is the availability of design points and responses. For example, in engineering design, a response to a design point may involve running a computationally expensive simulation and evaluations may be sequential. As another example, a recommender system with millions of design points and responses available is well suited for a machine learning algorithm. Due to the wide range of applications, statistical methods, global optimization, and machine learning have undergone staggering development over the past decades. A large number of useful tools have been developed. However, machine learning tools have not been well integrated into global optimization algorithms to achieve good performance in high-dimensional black-box problems. From this perspective, this dissertation proposes to enhance global optimization algorithms by leveraging statistical methods and machine learning tools to improve performance in global optimum prediction.

The dissertation proposes methods that integrate tools used in machine learning with global optimization algorithms to solve noisy black-box functions with discrete-time performance analysis. This dissertation introduces a multi-level Partitioning and Branch-and-Bound (PBnB) algorithm [113] for level-set approximation, which enhances the original PBnB method, classifying subregions as pruned (no intersection with target level-set), maintained (contained within target level-set), or undecided to form its approximation. The key features of the proposed algorithm include its refined branching scheme, adaptive subregion probability, and the integration of a Gaussian Process for guiding local sampling. The multi-level approach is motivated by the computational challenge of the original PBnB, which branches all current subregions, leading to a proliferation of smaller subregions and excessive function evaluations. It selectively branches only the most promising "best" and "worst" subregions, thereby increasing the chance of efficiently maintaining or pruning with fewer total subregions and fewer function evaluations. Furthermore, the original PBnB samples uniformly on the domain, by selecting subregions with a probability proportionate to their volume. In contrast, the enhanced version introduces an adaptive sampling probability using importance sampling, which strategically directs more samples to promising regions. To account for this non-uniform sampling induced by adaptive subregion probability, a confidence

interval of the target quantile for subregion classification is adjusted using an empirical distribution. The multi-level PBnB algorithm is further enhanced by integrating Gaussian processes as a surrogate model to guide local sampling within subregions, specifically by sampling points that maximize expected improvement. Numerical results demonstrate significant improvement in the efficiency of this multi-level PBnB compared to the original method. Additionally, the analysis of this multi-level PBnB algorithm quantifies the quality of the level set approximation by deriving probability bounds on the volume of incorrectly pruned or maintained regions, which accounts for the effects of importance sampling.

To address the challenges of high dimensionality, this dissertation introduces the Branching Adaptive Surrogate Search Optimization (BASSO) framework, which conceptualizes the use of partitioning and surrogate modeling for black-box optimization. The proposed BASSO framework is inspired by insights into the impact of high dimensions on the sampling distribution from adaptive search methods, including Pure Adaptive Search (PAS) [110, 116, 118] and Hesitant Adaptive Search (HAS) [19, 108, 115]. In particular, under certain conditions, the expected number of PAS (and HAS) function evaluations required to sample below a specified objective function value increases only linearly with the dimension of the input when optimizing a function without noise. While this "linearity result" is ideal, PAS and HAS are not directly implementable. The conceptual Annealing Adaptive Search (AAS) with a derived cooling schedule for its temperature parameter [95] is another attempt at achieving the linearity results, however, it is still impractical to sample exactly from a Boltzmann distribution. Hence, we propose the BASSO framework, which follows an analogy to AAS by mimicking sampling from a Boltzmann distribution, using branching and a surrogate model to increase the chance of sampling in the improving region. Since prior analysis of PAS, HAS, and AAS are not immediately applicable to surrogate modeling and partition-based algorithms, a key contribution of the BASSO framework is the new finite-time analysis. We prove that BASSO achieves the ideal linearity result under two conditions: specifically, conditions on adaptive subregion probabilities and the probability of improvement with a surrogate model. This linearity result implies that BASSO is scalable to high dimensions in theory. The second contribution involves exploring several intuitive ways to implement BASSO and evaluating

which variations most closely satisfy these conditions. Each variation of BASSO proposed in this research differs in its local surrogate model, which guides sampling within a subregion, and in its adaptive subregion probabilities, which identify promising subregions. This interplay allows the system to effectively use both local subregion information (from the surrogate model) and global information (from the adaptive probabilities) to improve its search and contrast exploitation with exploration for selecting subregions. In this part of the research, machine learning methods are introduced as surrogate models to increase the chance of sampling in the improving region. The variations also allow for a comparison between using a Gaussian process surrogate, regularized quadratic regression, and no surrogate (i.e., uniform sampling). While conditions for linearity are impractical to satisfy completely in practice, we explore them through numerical implementations.

Numerical experiments of BASSO provide insights into the gap between theoretical ideal performance and the performance of proposed implementations with machine learning techniques to tackling high dimensional black-box problem. A key insight is the need to consistently sample within improving level sets. This dissertation also explores decomposition and clustering as techniques for generating improving points for high-dimensional optimization.

While the finite-time analysis of the performance of BASSO focuses on deterministic functions, this dissertation also proposes an enhancement to the stochastic adaptive search algorithm with the presence of noise, extending the Single Observation Search Algorithm (SOSA) [57, 58] by incorporating insights from machine learning. The neighborhood averaging technique for function value estimation of SOSA is replaced with a quadratic regression, extending the concept of basis expansion. Contributing to the interplay of optimization and machine learning, this section provides a convergence analysis for a function value estimate from quadratic regression, which accounts for dependent samples acquired from an adaptive search within a single-observation scheme. The search strategy is also enhanced by incorporating optimistic sampling, a concept borrowed from reinforcement learning, which yields encouraging numerical results and improved performance.

The main goal of this proposed research is to develop an integrated optimization and machine learning methodology to optimize large-scale noisy black-box functions. This research has three specific objectives to reach the main goal:

- Integrate a method used in machine learning with adaptive search methods to optimize large-scale black-box functions. This objective is addressed in:
  *Chapter 3.2* Incorporate a machine learning model (Gaussian process) into variations of Multi-level PBnB algorithm.
  *Chapter 4.2* Incorporate a machine learning model (Gaussian process and regularized quadratic regression) into variations of BASSO implementations.
  *Chapter 6.2* Incorporate a machine learning model (quadratic regression) into SOSA to estimate an objective function using noisy function evaluations.

- Prove convergence of the resulting algorithms and analyze finite-time performance properties of the algorithms. This objective is addressed in:
  *Chapter 3.3* Derive probability bounds on the volume of incorrect pruning and incorrect maintaining for variations of Multi-level PBnB algorithm.
  *Chapter 4.3* Provide finite-time analysis of BASSO and conditions for BASSO to achieve scalability in high dimension.
  *Chapter 6.3* Provide convergence analysis of the function estimates with quadratic regression accounting for the dependency of error term that is a martingale difference.

- Combine theoretical analysis and empirical numerical testing to create practical implementation guidelines. This objective is addressed in:
  *Chapter 3.4* Numerical experiments of Multilevel PBnB variations.
  *Chapter 4.4* Numerical experiments of BASSO variations.
  *Chapter 5.2* Numerical experiments of high-dimensional black-box optimization with partitioning, clustering and decomposition.
  *Chapter 6.4* Numerical experiments of enhanced SOSA algorithm variations with quadratic regression and optimistics sampling following the principle from the reinforcement learning.

The dissertation begins by presenting background and relevant literature in Chapter 2. The core

research then delves into specific methodologies, starting with Multi-level Probabilistic Branch and Bound in Chapter 3, where its efficiency for level set approximation is enhanced through multi-level frameworks, importance sampling, and Gaussian processes. This is followed by the introduction of the Branching Adaptive Surrogate Search Optimization (BASSO) framework in Chapter 4, where a finite-time analysis provides conditions to achieve scalability in high dimensions. Chapter 5 is dedicated to numerical experiments of BASSO implementations, decomposition, and clustering applied to high-dimensional black-box optimization, focusing on scalability. The research then presents enhancements to the Single Observation Search Algorithm (SOSA) in Chapter 6, integrating quadratic regression and optimistic sampling for improved performance. The final chapter summarizes the contributions and outlines future work by discussing the emerging field of Quantum Optimization Algorithms and the potential of Grover's Adaptive Search and Quantum Annealing. In conclusion, this dissertation research contributes to integrating optimization and machine learning to improve algorithms for large-scale noisy black-box optimization.

# Chapter 2

# LITERATURE

## *2.1 Black-box Optimization*

Black-box optimization is broadly applicable across many domains and has been studied in multiple scholarly fields under names including Derivative-free Optimization, Global Optimization, Bayesian Optimization, Sequential Experimental Design, and assorted variants of the multi-armed bandit problem. Methods include variations of simulated annealing, genetic and evolutionary algorithms, partitioning methods, meta-modeling, particle swarm, and covariance matrix adaptation evolution strategy (CMA-ES) [45], to name a few. Review of derivative-free optimization methods can be found in [85] and survey global optimization methods can be found in [65]. [10] discusses black-box optimization methodologies and applications. Methods aimed at simulation-optimization are discussed in [1, 36, 40, 92].

An optimization model can be formulated as

$$\min_{x} \; f(x)$$
$$\text{subject to } x \in S$$

where the decision variables are denoted by a $d$-dimensional vector $x$. To be comprehensive, we allow the decision variables to include integer- and real-values. The domain $S$ has $d$ dimensions, with $d_1$ real-valued variables and $d_2$ integer-valued variables, where $d = d_1 + d_2$. Assume $S$ is closed and bounded, and $S \subset \mathbb{R}^{d_1} \times \mathbb{Z}^{d_2}$.The objective function is typically denoted $f(x)$, $f : S \to \mathbb{R}$. The feasible region $S$ can be determined by the intersection of constraints (e.g., $h_j(x) \geq 0$ for $j = 1, \ldots, J$) or by an oracle, that is, given a solution $x$, the model can return whether $x$ is in $S$ or not. We also consider the objective function to be a black-box determined by an oracle, that is,

given a solution $x$, the oracle can return a value $f(x)$.

Several black-box optimization algorithms have been developed to accommodate a mixed integer/real feasible region. A review of algorithms on optimization problems with mixed integer/real variables is discussed in [78].

### *2.2 Challenge of Expensive Function Evaluation*

The primary challenge for many high dimensional black-box problems is that evaluating the black-box function is extremely expensive. This cost of black-box function evaluation is measured in time, money, or physical resources. A single function evaluation may involve running a complex engineering model (like CFD or FEA) running a high-fidelity simulation model, such as a computational fluid dynamics or finite element analysis model, which can take hours or days, even on a high-performance computing cluster. In materials science or chemistry, an evaluation could correspond to the physical synthesis and testing of a new compound, a process that is both time-consuming and resource-intensive [100].

This high evaluation cost fundamentally changes the priorities of the optimization process. When we optimize the cheap-to-evaluate functions, the computational burden lies within the optimization algorithm itself, to be specific, the overhead times it takes to generate the next candidate point. On the other hands, when function evaluations are expensive, this dynamic is reversed. The computational cost is dominated by the time and resources it takes for the black-box oracle to return a value. As a result, the priorities of expensive black-box optimization algorithm become sample efficiency.

The goal of black-box optimization not only to find the optimum, but also to find a near-optimal solution using a small number of function evaluations. This limited computational budget given rise to surrogate-based or model-based optimization, that attempt to learn an approximation of the expensive function. The idea is to build a surrogate model that is cheaper to evaluate from expensive evaluations then use this surrogate model guide the search for the next point to evaluate. This will balance the need to explore uncertain regions of the space with the desire to exploit regions that already appear promising. Bayesian optimization is one of the well-known methods from the class

of machine-learning-based/surrogate-based optimization methods for expensive black-box functions. Bayesian optimization models the objective function as a gaussian process then iteratively updating its conditional mean and covariance based on the observed samples. Hence, bayesian optimization provides uncertainty for the unexplored landscape, which acquisition function can be used to balance the exploration and exploitation to determine the next sample point. A variety of techniques are used to construct surrogate models, including Polynomial Regression, Artificial Neural Networks (ANN), and Radial Basis Functions (RBF). Among these, Gaussian processes are one of the most prominent and commonly used methods [120].

### *2.3 Curse of Dimensionality and Stochastic Adaptive Search*

The curse of dimensionality is a term first introduced by Richard Bellman to describe the exponential growth in the volume of a space as its number of dimensions increases. The consequence for optimization is that, without structural information about the objective function (beyond smoothness properties like Lipschitz continuity [38]), an optimizer requires exponential time in the worst case to search all points and find the optimum. While expensive black-box evaluations already limit the number of function calls an optimizer can make, the curse of dimensionality exacerbates this problem. Moreover, in the case that the objective function high-dimensional, the larger surrogate model requires more samples to construct a model that is a good representation of objective function.

Then, a crucial step in all of the algorithms for high-dimensional black-box optimization in high-dimensional is to strategically select the next point to evaluate and how to allocate the computational effort to balance exploration of the domain with exploitation of promising regions and estimation of noisy function evaluations. To gain insight into the balancing act of exploration and exploitation, we can analyze the impact of sampling distributions on computational effort.

The sampling distribution of Pure Adaptive Search (PAS), described in [110], is uniform on the set of improving points from the previous point. The idea of having a set of improving points may be interpreted as exploiting the objective function value of the best point sampled so far. Uniform sampling on the set of improving points could be considered exploring, in contrast to

seeking the optimum in the level set. The finite-time analysis of PAS shows that, if one *could* sample uniformly from improving regions, then the expected number of such iterations to achieve a solution arbitrarily close to the global optimum with high probability increases at most linearly in dimension. We refer to this as the "linearity result."

For any global optimization problem with continuous domain in $n$ dimensions, with Lipschitz constant at most $L$, and convex feasible region with diameter at most $D$, the expected number of PAS points to get within $\varepsilon$ of the global optimum is:

$$\mathbb{E}\left(N(y_* + \varepsilon)\right) \leq 1 + n\, ln\left(\frac{LD}{\varepsilon}\right).$$

While PAS is not directly implementable, the introduction of Hesitant Adaptive Search (HAS) [19, 108] and Annealing Adaptive Search (AAS) [88, 96, 107] attempted to narrow the gap between theory and implementation. Annealing adaptive search is an idealized version of simulated annealing, where sampling from a sequence of Boltzmann distributions with decreasing temperature focuses the sampling distribution on improving level sets. With an appropriate cooling schedule, AAS can achieve the desired linearity result of PAS [96], although it is still impractical to sample from a Boltzmann distribution. Many of the black-box optimization algorithms mentioned also use partitioning and surrogate modeling as a means to search for a global minimum.

In Chapter 4 of this dissertation, we presents an adaptive stochastic search algorithm called Branching Adaptive Surrogate Search Optimization (BASSO) that conceptualizes the use of branching and surrogate modeling, demonstrate how strategic partitioning can be used to steer the sampling distribution towards the improving level set of points. A finite time analysis of BASSO shows that the desired linearity result of PAS is achievable with specific assumptions and conditions. The analyses provide insight into the exploration/exploitation trade-off. The sampling distribution must exploit the threshold of the best function value observed but must fully explore the associated level set of improving points.

### *2.3.1 Machine Learning and Optimization*

The interplay between optimization and machine learning has two directions, first, optimization algorithms form the computational engine that builds machine learning models by learning their parameters and tuning their parameters; second, machine learning techniques, such as surrogate modeling, provide powerful tools to solve complex optimization problems by learning the behavior of their objective functions.

In this dissertation, we focus on utilizing methods used in machine learning to assist large-scale black-box optimization, specifically stochastic adaptive search. A stochastic adaptive search for noisy black-box optimization adaptively samples points in the feasible region with a bias towards a global optimal solution and uses the information from their corresponding sampled objective functions to predict the global optimum.

On the other hand, a machine learning algorithm takes sampled pairs of feature vectors (design points) and their corresponding responses (sampled objective functions), and uses them to produce an inferred function to predict the responses of other unseen feature vectors. The two paradigms of prediction share many similarities. Both take sampled pairs of design points and their responses, and both use the information to predict a target response. Due to its wide range of applications, machine learning has undergone a staggering development over the past decades. A large number of useful tools have been developed. From this perspective, in chapter 6, this study proposes the enhancement a stochastic search (SOSA) that leverages machine learning tools to enhance its performance in global optimum prediction of a smooth continuous objective function in noisy black-box optimization.

Given a noisy black-box optimization problem, a stochastic adaptive search sequentially produces sampled design points in the feasible region and observes their corresponding function values. The observed points and observed function values are used to steer the subsequent sampling towards a global optimal solution. Algorithms that fall into this category include stochastic approximation [61, 86], Adaptive Search with Resampling [2], Gaussian process regression [99], and the Single Observation Search Algorithm (SOSA) [57, 58]. Stochastic approximation is a

gradient based algorithm that uses stochastic gradients to guide the search. Adaptive Search with Resampling repeatedly observes function values at each sampled design point and averages the replications to remove the noise. Gaussian process regression and SOSA generalize the information of the preceding search sequence to infer the function values of the unseen design points. The latter two algorithms employ a concept similar to that of the supervised learning methodology.

Supervised learning is a type of machine learning that aims to predict the output (function value) of an input (feature vector or design point) based on an inferred function learned from a set of known input-output pairs. Some of the most well-known supervised learning algorithms include $k$-nearest neighbor [27], linear regression, polynomial regression, the support vector machine (SVM) [14], the regression tree [16] and neural networks [39, 89]. These supervised learning methods are very successful in real applications. Recent advances range from language processing [26, 33] to healthcare and drug discovery [46, 53]. The list of applications keeps expanding. The abstraction of these algorithms employs the concept of basis expansion to achieve more flexible representations of the target functions. Common basis functions include piecewise linear basis functions, polynomial basis functions, radial basis functions, etc.

Consider methods in machine learning to tackling high dimensional problem, regularization is techniques used in machine learning and statistics to addresses this by adding a penalty term to the model's loss function. High-dimensional data, where the number of variables is very large relative to the number of samples, a model can easily find spurious correlations in the training data and overfit, especially when the number of training samples is limited. Regularization is a powerful tool for managing these challenges. L1 Regularization (Lasso Regression), L2 Regularization (Ridge Regression) and Elastic Net Regularization [47] are techniques that adds a penalty equal to the coefficients size. A key characteristic of L1 regularization is its ability to shrink some coefficients to exactly zero. This acts as a form of automatic features/variables selection, removing irrelevant or redundant features from the model. For high-dimensional problems where many features may be noise, Lasso is an efficient method because it creates a sparse model that relies only on the most important predictors. In Chapter 3, chapter 4 and chapter 5, we apply the L1 Regularization or Lasso regression as the surrogate model for high-dimensional black-box optimization.

### *2.3.2 Method for High Dimensional Black-box Optimization*

Achieving scalability in large-scale black-box optimization is hindered by two interconnected challenges. The first challenge arises from the heterogeneity of the objective function across the domain. In black-box optimization, we lack knowledge of the objective function's structure. Consequently, behavior observed in one region of the feasible space provides little to no information about other, unobserved regions, making it difficult to build a single, accurate model to guide the optimization process. A potential strategy to alleviate this is to subdivide the problem space using methods like partitioning and clustering to tackle smaller sub-problems. This issue is compounded by the second challenge: the well-known curse of dimensionality. As the number of input dimensions increases the number of samples needed to search for the global optimum grows exponentially. This leads to major consequences, such as sampling inefficiency, where a greater number of sample points is needed to achieve a "space-filling" design that adequately represents the domain. Moreover, while surrogate models can approximate the objective function to guide the search efficiently, as input dimensions increases, the number of sample points required to train a surrogate model also grows with the dimension, which can make the cost of building a surrogate model more expensive and undermines its purpose of saving costly function evaluations. To address the challenges that come with dimensionality and expensive function evaluations, researchers have developed numerous strategies to make high-dimensional black-box optimization manageable. This section reviews the four common methods used to tackle high-dimensional black-box optimization: partitioning, decomposition, clustering, and embedding.

#### *Partitioning*

To build a good surrogate model or to learn the surface of the objective function on the entire domain require large number of samples. Partitioning is tackling high-dimensional black-box optimization problem is to avoid building a single, complex model of the entire search space. It dynamically divided the search domain into smaller, more regions or partitions. The optimization effort is then strategically allocated to the most promising of these regions, creating a natural mech-

anism to balance global exploration (deciding which region to investigate) with local exploitation (intensively searching within a chosen region). These methods do not require additional assumption about the function's structure but use adaptive strategy to focus the search on the promising partition instead.

The Nested Partitions Method [97] and the Adaptive Hyperbox Algorithm [109] are example methods that partition the feasible region into subregions and mainly search within promising subregions. Partitioning techniques are also used in the Stable Noisy Optimization by Branch and Fit (SNOBFIT) algorithm [51] where the algorithm subdivides the domain and builds local models of the function as in trust region. Another work use partitioning is the Non-linear Optimization with Mesh Adaptive Direct (NOMAD) Search algorithm [63]. NOMAD uses partitioning and refines the search by generating a series of grids with varying discretizations of the space of variables. The adaptive mesh frame acts as a window that constrains the search to a specific region of the space, partitioning within this space focuses the search to the promising area and allows an efficient allocation of computational resources.

Another powerful approach, Latent Action Monte Carlo Tree Search (LA-MCTS) [103] involves hierarchical tree-based partitioning. LA-MCTS methods builds a tree structure that recursively subdivides the search space. At each node in its search tree, LA-MCTS uses the K-means algorithm to cluster the already-evaluated points based on their objective values. It then trains a classifier (a Support Vector Machine) to learn a non-linear decision boundary that separates the "good" points from the "bad" ones, thereby partitioning the space. The algorithm uses Monte Carlo Tree Search with an upper confidence bound selection strategy to decide which branch of the tree or which partition/subregion to explore next. A very recent and innovative development in this area is Hierarchical Optimization with Large Language Models (HOLLM) [91]. HOLLM also uses a tree-based partition and uses a Large Language Model (LLM) as its local sampler. Promising subregions are selected using a scoring mechanism inspired by multi-armed bandit problem, and the LLM is then prompted with information about that subregion to generate new candidate points.

The strength of partitioning methods is less assumption about the function's global structure,

make it more flexible and perform better empirically. The local modeling approach also is more scalable than attempting to fit a single global model. The main disadvantages are that their performance can be sensitive to the hyperparameters that govern the partitioning logic, such as the initial trust region size or the branching strategy in tree-based methods. While they are more sample-efficient than global search, they may still require a significant number of function evaluations to perform the enough initial exploration before they can effectively zoom in on the most promising regions.

The key innovation of the recent partition methods is the partitioning strategy that acts as global navigator to perform the high-level task of deciding where in the entire domain to focus on. Once a promising region (or regions) is identified, the local surrogate model will search within the smaller region. This modular design suggests the future development will be, not only in inventing new partitioning schemes but also exploring possible pairing of partitioning frameworks and different surrogate model choices. In this dissertation, we proposed multi-level Partitioning and Branch-and-Bound (PBnB) algorithm for level-set approximation in Chapter 3 and Branching Adaptive Surrogate Search Optimization (BASSO) framework in Chapter 4.

*Decomposition*

The key idea behind decomposition technique is to divide the original problem into a set of smaller, low dimensional subproblems, which are easier to manage. Although this idea is intuitive and simple, there are three difficult main features of the decomposition-based algorithm that are needed to be addressed namely, how to decompose the problem, how to optimize each subproblem, and how to combine these subproblems. One of the earlier methods in this category is cooperative co-evolutionary algorithm[79]. Cooperative co-evolutionary algorithm decomposes the original problem into a set of lower-dimensional and tractable subproblems, each of which can be solved separately. The CCEAs methods is investigate in a recent survey paper [67]. For Bayesian optimization, many works assume additive black-boxes model, decompose the problem based on initial data, then maximize an acquisition function that is additive under the decomposition [44, 55, 87]. The primary advantage of decomposition methods is their ability to scale to extremely high dimen-

sions by breaking the problem, these methods tackle a series of low-dimensional problems, which can be parallelized to further speed up the process.

The main drawback, however, is that the performance of these methods depends on the correctness of the assumed decomposition. If a function with complex, non-additive interactions is incorrectly modeled, the algorithm's performance can be degraded. The process of identifying the variable interactions or finding the right decomposition is also a challenging and expensive subproblem. Recent works for Bayesian optimization attempt to resolve this analyzing distribution shifts [59] or utilize random tree-based decomposition and an additive acquisition function (RDUCB) [121]. In Chapter 5, we extends the partitioning method to solve high-dimensional black-box optimization with decomposition technique in the numerical experiments.

*Clustering*

Clustering-based methods represent another data-driven approach to manage the search space. The application of clustering typically apply clustering algorithms to the set of points sampled to identify promising region of the search space, then guide subsequent search. The classic use of clustering is in multi-start methods. The GLOBAL algorithm [75] is an example of this approach. It begins by drawing a number of points uniformly at random from the search space. Instead of initiating an expensive local search from every one of these points, it first applies a clustering algorithm to group them. The assumption is that points in same cluster likely lie within the same neighborhood of a single local minimum. Therefore, a local search procedure is initiated from the most promising points within cluster. More recent work has focused on integrating clustering more tightly with modern surrogate-assisted optimization frameworks. The Clustering-based Surrogate-assisted Evolutionary Algorithm (CSRSA) algorithm [9] provides a compelling example. CSRSA first runs an evolutionary algorithm on a surrogate model of the expensive function. This generates a large population of candidate solution. The algorithm then applies a clustering method to this population. The centers of the resulting clusters are identified as high-potential candidate points, which are then selected for evaluation with the true, expensive black-box function. This process uses clustering as a filter to select the most promising candidates generated by a cheaper search

process for expensive evaluation. The advantage of clustering-based methods is their effectiveness on highly multi-modal objective functions, by identifying and exploring distinct promising region. The main disadvantages are that their performance depends on the quality of the clustering itself. The choice of clustering algorithm, its distance metric, and its hyperparameters (e.g., the number of clusters) can have a profound impact on the outcome. Furthermore, the clustering step introduces computational overhead, which must be weighed against the expensive function evaluations.

*Embeddings*

The principle behind embedding methods is the assumption that even though a problem is defined in a high-dimensional space, the objective function is primarily governed by a lower-dimensional space. If this assumption holds, the curse of dimensionality in optimization can be avoided by performing the optimization search within this low-dimensional space, thereby concentrating the limited evaluation budget where it has the most impact. In this category is REMBO (Random EMbedding Bayesian Optimization) [105] uses of random linear embeddings. It randomly generates a low-dimensional subspace and projects it into the high-dimensional ambient space using a fixed random matrix. The Bayesian optimization search is then conducted entirely within this low-dimensional subspace. A framework for bayesian optimization in embedded subspaces is proposed in [73],with the implementation called,the Hashing-enhanced Subspace BO (HeSBO) method for high-dimensional problems. Subsequent research has focused on creating more intelligent embeddings. The goal is to use the data gathered during the optimization process to learn an embedding that is more relevant to the objective function. The primary advantage of embedding methods is their potential for gains in sample efficiency, if the effective dimension assumption holds and the embedding is well-chosen. The fundamental weakness of all embedding methods is when the effective subspace is misidentified, the optimizer will be constrained to a suboptimal region of the search space. Random embeddings offer no guarantee of capturing the correct subspace. Learned embeddings, while more intelligent, can be computationally expensive to train and incline to overfitting when the number of expensive function evaluations is small.

### *2.3.3 Noisy Black-box Optimization*

Dealing with the noisy observation in black-box optimization highlight the importance of how to strike a balance between exploration of new points and estimation of observed good points efficient algorithms. A simple question arises on how to estimate the true objective function with noisy evaluation and what is the most efficient way to distribute the computation budget given noisy observation. The classic stochastic approximation algorithm have the early methods that proven to be successful at optimizing noisy functions on continuous domains [20, 62, 86]. Stochastic approximation is widely used for continuous problems with many applications. It is a first-order method with single or multiple observations per point that typically converges to a first-order stationary point. Sample average approximation [60] has been used in both continuous and discrete stochastic optimization. Sample average approximation (SAA) takes a collection of the random vector from Monte Carlo samples or Latin hypercube designs and then solves an associated problem. Under certain conditions, SAA asymtotically converges to the global optimum. In this dissertation, we enhanced the estimation steps of single observation search algorithm (SOSA) in chapter 6.

To consider the impact of estimation on the trade-offs between exploration and exploitation, Hesitant Adaptive Search with Estimation (HAS-E) [115] was introduced with a finite-time analysis of algorithm performance that combines estimation with a sampling distribution through the use of confidence intervals on the estimated function evaluations.

An interpretation of HAS-E is that there is a tradeoff between sampling from a larger than needed level set (with loose upper confidence bound and fewer replications) and sampling from a more accurate estimate of the current level set (with tight upper confidence bound and more replications). This suggests that algorithms should use few replications as long as the estimation approaches the true function value as the algorithm approaches the global minimum.

While PAS, HAS, AAS, BASSO, and HAS-E are not directly implementable, the analyses provide insights into efficient features of a black-box optimization algorithm. The final insight is to not expend computation on regions with poor performance, and quickly focus the sampling distribution on improving regions.

# Chapter 3

# MULTI-LEVEL PROBABILISTIC BRANCH AND BOUND FOR LEVEL SET APPROXIMATION

This chapter proposed a multi-level Partitioning and Branch-and-Bound (PBnB) algorithm for level-set approximation, which enhances the original PBnB [113] method. The contributions of this chapter to the three main goals of this dissertation are the proposed multi-level PBnB algorithm, new finite-time performance analysis of this multi-level PBnB algorithm and numerical experiments of multi-level PBnB variations. The proposed multi-level PBnB algorithm improves upon the original by incorporating three key features, including a refined branching scheme, adaptive subregion probability, and the integration of a Gaussian process for guiding local sampling. In contrast to the original PBnB, the multi-level PBnB selectively branches only the most promising "best" and "worst" subregions, thereby increasing the chance of efficiently maintaining or pruning with fewer total subregions and fewer function evaluations. Additionally, this enhanced multi-level PBnB introduces an adaptive sampling probability to select subregions to strategically direct more samples to promising regions as opposed to the original PBnB that samples uniformly on the domain, by selecting subregions with a probability proportionate to their volume. The multi-level PBnB algorithm is further enhanced by incorporating a machine learning model (Gaussian process) as a surrogate model to guide local sampling within subregions into variations of multi-level PBnB algorithm. The new analysis of this multi-level PBnB algorithm quantifies the quality of the level set approximation by deriving probability bounds on the volume of incorrectly pruned or maintained regions, which accounts for the effects of importance sampling. This chapter also provides numerical results that demonstrate a significant improvement in the efficiency of this multi-level PBnB compared to the original method.

### *3.1 Introduction and Background*

Probabilistic Branch and Bound (PBnB) is a random search algorithm that uses sampling and partitioning of the solution space [49, 114]. The algorithm is similar to the nested partition framework. The algorithm iteratively updates its confidence interval on the value of the $\varepsilon$-optimal level set or $\delta$-target quantile, and seeks the target level set, that is, the set of solutions within the target quantile.

We follow the notation of [114]. PBnB aims to approximate a level set with respect to a performance function $f(x)$ of a noisy black-box optimization model. The optimization problem is

$$\min_{x \in S} f(x),$$

where $f(x) : S \to \mathbb{R}$, and $S \subset \mathbb{R}^{dim}$. The decision variable $x$ is a vector in $dim$ dimensions, and the values may be integer or real-valued. We are interested in the $\delta$ quantile associated with $f(x)$, denoted $y(\delta)$,

$$y(\delta) = \operatorname*{argmin}_{y} \{P(f(X) \leq y) \geq \delta\}, \text{ for } 0 < \delta < 1,$$

where $X$ is a random variable uniformly distributed on the domain $S$.

We let $L(\delta)$ be our desired set of best $\delta$-quantile solutions, where

$$L(\delta) = \{x \in S : f(x) \leq y(\delta)\}, \text{ for } 0 < \delta < 1.$$

Level set identification is an active field of research. The problem of estimating level sets of black-box objective functions arises in a wide range of applications, including monitoring environmental parameters [82] and aircraft configuration [80]. Sequential learning is one method to solve this problem. Most sequential learning algorithms use a stochastic process model such as a Gaussian process (GP) as a surrogate model, as in [93]. Other methods include the use of Bayesian neural networks [43], linear bandits [68], and problem reduction [8]. Level set approximation via Probabilistic Branch and Bound has been useful in designing a simulated water distribution net-

work for a large city [102], and in optimizing screening and treatment decisions for hepatitis C over a 40 year projected time horizon [50]. Similar approaches also include contour estimation, which uses surrogate models like Gaussian Processes to efficiently find the boundary, or contour, of a region where a function's output is above or below a specific threshold [83].

Probabilistic Branch and Bound (PBnB) uses sampling, branching and classification of subregions to provide a collection of subregions that form the level set approximation. PBnB classifies subregions as maintained (contained in the target level set) and pruned (no intersection with the target level set) with statistical confidence, and updates its collection of current undecided subregions that do not have statistical confidence to be maintained or pruned. A finite-time analysis of PBnB provides probability bounds on incorrectly pruning and maintaining subregions on each iteration. This result quantifies the quality of the solution and helps inform a decision maker when to stop the algorithm with an acceptable quality of the solution.

As illustrated in Figure 3.1, the original PBnB (Algorithm A) has two primary components. The first component is to sample uniformly over the current subregions, and to provide an interval estimate of the objective function value for the target quantile, denoted $y(\delta, S)$ where $\delta$ is the user-input for the target quantile . The second component focuses on classifying subregions into maintained or pruned, and branching undecided subregions for more sampling to glean more information.

The main computational challenge of Original PBnB stems from making too many function evaluations and branching too many subregions. The branching scheme in Original PBnB branches **all** of the current subregions. This scheme leads to a proliferation of smaller subregions that requires a large number of samples to confirm maintaining and pruning. This motivates the concept of Multilevel PBnB to branch **only** the promising best and worst subregions. The idea is that identifying the best subregions increases the chance of maintaining, and identifying the worst subregions increases the chance of pruning, with fewer subregions and fewer function evaluations.

In this study, we hypothesize that focusing more on the promising subregions will improve the efficiency of the PBnB algorithm.

We also note that Original PBnB samples **uniformly** on the current subregions, which is a

conservative sampling distribution that is useful in estimating $y(\delta, S)$ with confidence intervals. We consider a form of importance sampling where more samples are taken in promising subregions. The interval estimation of $y(\delta, S)$ must be adapted to account for non-uniform sampling. We hypothesize that sampling on a posterior distribution based (in contrast to uniform sampling) will improve efficiency.

We define four variations to Original PBnB (Algorithm A), namely multi-level PBnB (Algorithm B) that modifies the branching strategy, Incumbent-based Sampling (Algorithm C) that additionally uses a posterior distribution derived from lowest sampled point in each subregion, Uncertainty-based GP Sampling (Algorithm D) that uses a posterior distribution posterior distribution derived Gaussian process estimation of uncertainty and Uncertainty- and EI-based GP Sampling (Algorithm E) that uses a posterior distribution derived Gaussian process estimation of uncertainty and then samples according to maximum expected improvement.

We present computational results to answer three research questions. First, we determine if branching on only promising subregions will improve efficiency. Second, we examine the sensitivity of the algorithms to two parameter values that impact branching. Third, we determine if sampling on a posterior distribution will improve efficiency. To answer the questions above, we conduct a computational experiment on Original PBnB (Algorithm A), Multilevel PBnB (Algorithm B), Incumbent-based Sampling (Algorithm C), Uncertainty-based GP Sampling (Algorithm D) and Uncertainty- and EI-based GP Sampling (Algorithm E) while varying algorithm parameters over test functions in varying dimensions.

### *3.2 Multi-level PBnB for Level Set Approximation*

The input parameters to PBnB defined by the user include: $\delta, \alpha, \varepsilon, B$, and $c$. The parameter $\delta$, $0 < \delta < 1$, is used to define a $\delta$-quantile for the target level set. For example, the user may be interested in the set of solutions in the best 10%, in which case $\delta = 0.1$.

The following two parameters, $\alpha$ and $\varepsilon$, are used to determine the quality of the level set approximation. The approximation can be wrong in two ways: it could prune a portion of the level set or it could maintain some area that is not in the level set. The parameter $\alpha$, $0 < \alpha < 1$, is used

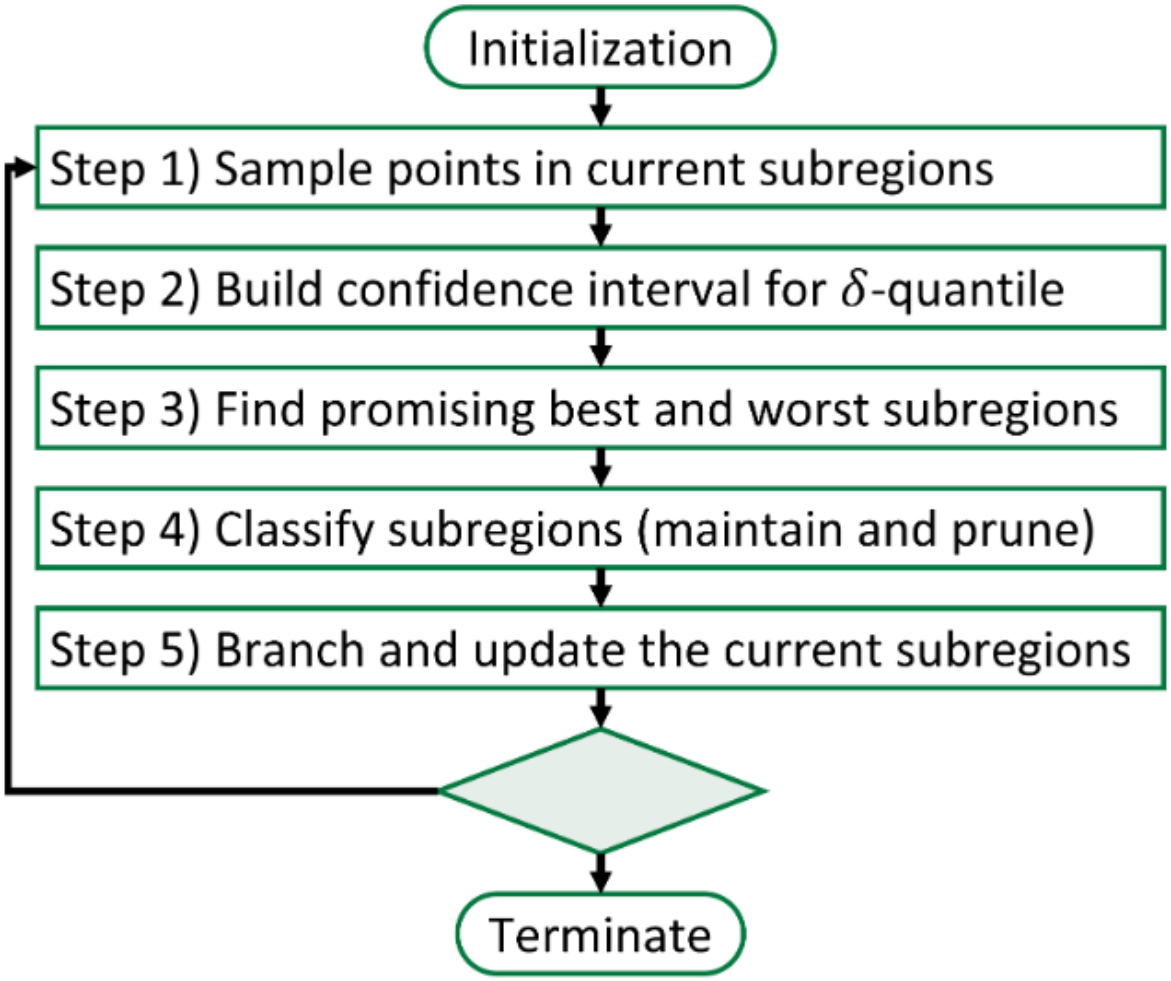



**Figure 3.1:** Procedure of multi-level PBnB variations for level set approximation. The target quantile $\delta$ is a user-input, and the estimated objective function value is denoted $y(\delta, S)$.

in the confidence level of the estimation of $y(\delta, S)$ and in the probabilities of incorrectly pruning or maintaining. The choice of $\alpha$ will influence the sample size. As the confidence level $(1-\alpha)$ increases, a larger sample size is needed. The parameter $\varepsilon > 0$ is the volume of solutions that can be tolerated to be categorized incorrectly. We also expect that a high confidence level (low $\alpha$) will have fewer mistakes, since the probability of the incorrect volume exceeding $\varepsilon$ decreases.

The analytical results of the algorithm [49, 114] provide confidence intervals on the quantile

estimation of $y(\delta, S)$, and probability bounds on incorrectly pruning a volume of size $\varepsilon$, and on incorrectly maintaining a volume of size $\varepsilon$, respectively. To paraphrase, on each iteration, the probability that the volume of the incorrectly pruned region is no greater than $\varepsilon$ is bounded by $(1-\alpha)^4$. Similarly, the probability that the volume of the incorrectly maintained region is no greater than $\varepsilon$ is bounded by $(1-\alpha)^4$. Notice the number of function evaluations per iteration depends on the observed sample points.

The branching scheme is defined by parameter $B$, $B \geq 2$, which is the number of evenly sized subregions to create by subdividing the longest dimension of the subregion. The parameter $c$ is the incremental sample size for sampling on the current subregions in Step 1.

Let $\Sigma^M_{iter}, \Sigma^P_{iter}$ and $\Sigma^C_{iter}$ be collections of subregions that are maintained, pruned, and currently undecided (the set of subregions that are not pruned or maintained), respectively, on iteration $iter$.

In Step 1, Algorithms A and B sample points according to a uniform distribution on current subregions, whereas Algorithm C uses a posterior distribution based on the lowest function value observed in a subregion, then samples uniformly on selected subregion. Algorithm D and Algorithm E uses uncertainty observed in a subregion and selects subregions according to posterior distribution derived from Gaussian process estimation of uncertainty and then samples uniformly on selected subregion. In Step 2, the expressions to estimate the target quantile $y(\delta, S)$ with confidence intervals differ when points are sampled uniformly (Algorithm A and B) or using a posterior distribution (Algorithm C, D and E) . Step 3 is the same for all variations, and uses the confidence intervals from Step 2.

In Step 4, Algorithms A and B implement additional sampling in the best and worst subregions to statistically confirm classification of subregions as maintained or pruned. Algorithm C, D and E do not add more sample points in Step 4, but waits until enough points are accumulated through the posterior distribution to classify subregions with the same level of confidence.

In Step 5, Algorithm A branches *all* current subregions resulting in subregions that are always the same size, whereas Algorithms B and C with multi-level branching branch only the promising best and worst subregions that results in subregions with different sizes. Algorithms D and E branch the promising best and worst subregions and also the subregions with more uncertainty.

PBnB for level set approximation proceeds until all subregions are either maintained, pruned, or reach a user-defined minimum size that we term "unbranchable." Upon termination, PBnB outputs a confidence interval on the target quantile $y(\delta, S)$ and the subregions that have been maintained as an approximation of the target level set.

**Multi-level Probabilistic Branch and Bound (PBnB) for Level Set Approximation**

***Step 0. Initialization***: Input user-defined parameters, $\delta, \alpha, \varepsilon, B,$ and $c$. Also, initialize the maintain, prune, and current subregion collections as $\Sigma_1^M = \emptyset$, $\Sigma_1^P = \emptyset$, $\Sigma_1^C = \{S\}$. Set $\delta_1 = \delta$, $c_0 = 0$,and the iteration counter $iter = 1$.

***Step 1. Sample points in current subregions***: For the current subregions in $\Sigma_{iter}^C$, sample $c$ additional points over all current subregions in $\Sigma_{iter}^C$ and update the total number of points in $\Sigma_{iter}^C$, such that $c_{iter} \leftarrow c_{iter-1} + c$.

***1A and 1B***: To obtain an additional sample point that is uniformly distributed, randomly choose subregion $\sigma_i$ from the subregions in $\Sigma_{iter}^C$ with probability

$$p_i = v(\sigma_i)/v(\Sigma_{iter}^C), \tag{3.1}$$

where $v(\cdot)$ denotes the $dim$-dimensional volume of a set. Then, uniformly generate a sample point within the chosen subregion $\sigma_i$. Update the number of points that have been sampled in $\sigma_i$ as $N^i$.

***1C***: To obtain an additional sample point using importance sampling, randomly choose subregion $\sigma_i$ from the subregions in $\Sigma_{iter}^C$ with probability $\tilde{p}_i$, where

$$\tilde{p}_i = \begin{cases} v(\sigma_i)/v(\Sigma_{iter}^C) & \text{if } iter = 1 \\ \left(\left(\tilde{f}_i^* - \hat{f}^* + 1\right) \sum_{j=1,\ldots,||\Sigma_{iter}^C||} \left(1/\tilde{f}_j^* - \hat{f}^* + 1\right)\right)^{-1} & \text{if } iter > 1, \end{cases}$$

where $\tilde{f}_i^*$ is the lowest sampled value in subregion $\sigma_i$ and $\hat{f}^* = \min_{i=1,\ldots,||\Sigma_{iter}^C||}\{\tilde{f}_i^*\}$. Then, uniformly generate a sample point within the chosen subregion $\sigma_i$. Update the number of points that have been sampled in $\sigma_i$ as $N^i$.

***1D***: To obtain an additional sample point using importance sampling, randomly choose subregion

$\sigma_i$ from the subregions in $\Sigma^C_{iter}$ with probability $\tilde{p}_i$, where

$$\tilde{p}_i = \begin{cases} \nu(\sigma_i)/\nu(\Sigma^C_{iter}) & \text{if } iter = 1 \\ \left(s^{max}_{iter}(\sigma_i)\right)^2 / \sum_{j=1,\ldots,||\Sigma^C_{iter}||} \left(s^{max}_{iter}(\sigma_i)\right)^2 & \text{if } iter > 1, \end{cases}$$

where $(s^{max}_{iter}(\sigma_i))^2$ is the largest predicted model variance $(\hat{s}^i_{iter}(x))^2$ from the Gaussian process constructed on $\sigma_i$ (resulting from a grid search on the subregion). Then, uniformly generate a sample point within the chosen subregion $\sigma_i$. Update the Gaussian process predictor $\hat{g}^i_{iter}(x)$ and model variance $(\hat{s}^i_{iter}(x))^2$ for subregion $\sigma_i$. Update the number of points that have been sampled in $\sigma_i$ as $N^i$.

***1E***: To obtain an additional sample point using importance sampling, randomly choose subregion $\sigma_i$ from the subregions in $\Sigma^C_{iter}$ with probability $\tilde{p}_i$, where

$$\tilde{p}_i = \begin{cases} \nu(\sigma_i)/\nu(\Sigma^C_{iter}) & \text{if } iter = 1 \\ \left(s^{max}_{iter}(\sigma_i)\right)^2 / \sum_{j=1,\ldots,||\Sigma^C_{iter}||} \left(s^{max}_{iter}(\sigma_i)\right)^2 & \text{if } iter > 1, \end{cases}$$

where $(s^{max}_{iter}(\sigma_i))^2$ is the largest predicted model variance $(\hat{s}^i_{iter}(x))^2$ from the Gaussian process constructed on $\sigma_i$ (resulting from a grid search on the subregion). Then, generate a sample point in the chosen subregion, $x \in \sigma_i$ that maximizes the acquisition function over $\sigma_i$,

$$ACK^i(x) = [\tilde{f}^*_i - \hat{g}^i_{iter}(x)]\phi\left(\frac{\tilde{f}^*_i - \hat{g}^i_{iter}(x)}{\hat{s}^i_{iter}(x)}\right) + \hat{s}_{iter}(x)\Phi\left(\left(\frac{\tilde{f}^*_i - \hat{g}^i_{iter}(x)}{\hat{s}^i_{iter}(x)}\right)\right),$$

where $\phi$ is the pdf and $\Phi$ is cdf of the standard normal distribution. Update the Gaussian process predictor $\hat{g}^i_{iter}(x)$ and model variance $(\hat{s}^i_{iter}(x))^2$ for subregion $\sigma_i$. Update the number of points that have been sampled in $\sigma_i$ as $N^i$.

***Step 2. Build confidence interval for*** $y(\delta, S)$: Order all sampled points, $z_{(1)}, \ldots, z_{(c_{iter})}$, in all

current subregions in $\Sigma_{iter}^{C}$ by their function values, so that

$$f(z_{(1)}) \leq \cdots \leq f(z_{(c_{iter})}).$$

Calculate the lower and upper bounds of $\delta_{iter}$ as

$$\delta_{iter}^{L} = \delta_{iter} - \frac{v(\Sigma_{iter}^{P})\varepsilon}{v(S)v(\Sigma_{iter}^{C})} \quad \text{and} \quad \delta_{iter}^{U} = \delta_{iter} + \frac{v(\Sigma_{iter}^{M})\varepsilon}{v(S)v(\Sigma_{iter}^{C})}. \tag{3.2}$$

***2A and 2B***: Calculate the confidence interval with lower and upper confidence limits as

$$CI_l = f(z_{(\tilde{r})}) \quad \text{and} \quad CI_u = f(z_{(\tilde{s})}),$$

where $\tilde{r}$ and $\tilde{s}$ are determined by

$$\tilde{r} = \max r : \sum_{i=0}^{r-1} \begin{pmatrix} c_{iter} \\ i \end{pmatrix} (\delta_{iter}^{L})^{i} (1-\delta_{iter}^{L})^{c_{iter}-i} \leq \frac{\alpha_{iter}}{2}, \text{ and} \tag{3.3}$$

$$\tilde{s} = \min s : \sum_{i=0}^{s-1} \begin{pmatrix} c_{iter} \\ i \end{pmatrix} (\delta_{iter}^{U})^{i} (1-\delta_{iter}^{U})^{c_{iter}-i} \geq 1 - \frac{\alpha_{iter}}{2}, \tag{3.4}$$

where $\alpha_{iter} = \alpha / B^{iter}$. The estimate of $y(\delta, S)$ is

$$\hat{y}(\delta, S) = (CI_l + CI_u)/2.$$

***2C, 2D, 2E***: Calculate the confidence interval of $y(\delta, S)$ with lower and upper confidence limits as

$$CI_l = f(z_{(\tilde{r}')}) \quad \text{and} \quad CI_u = f(z_{(\tilde{s}')}),$$

where $\tilde{r}'$ and $\tilde{s}'$ are determined by,

$$\tilde{r}' = \max r : \sum_{i=0}^{r} \frac{1}{c_{iter}} \frac{p_{m_{(z_{(i)})}}}{\tilde{p}_{m_{(z_{(i)})}}} \leq \frac{\tilde{r}}{c_{iter}}, \text{ and} \tag{3.5}$$

$$\tilde{s}' = \min s : \sum_{i=0}^{s} \frac{1}{c_{iter}} \frac{p_{m_{(z_{(i)})}}}{\tilde{p}_{m_{(z_{(i)})}}} \geq 1 - \frac{\tilde{s}}{c_{iter}}, \tag{3.6}$$

where $\tilde{r}$ and $\tilde{s}$ are from (3.3) and (3.4), respectively. The importance sampling probability of the subregion $m_{(z_{(i)})}$ is $\tilde{p}_{m_{(z_{(i)})}}$, according to Step 1C, 1D, 1E, and 1F, where the subregion $m_{(z_{(i)})}$ is the subregion associated with the ordered sample point $z_{(i)}$. The estimate of $y(\delta, S)$ is

$$\hat{y}(\delta, S) = (CI_l + CI_u)/2.$$

***Step 3 Find best and worst subregions***:

***3A, 3B, 3C, 3D, 3E***: In each subregion $\sigma_i$ in $\Sigma^C_{iter}$, order all sampled points, $x_{i,(1)}, \ldots, x_{i,(N^i)}$, by their function values, so that $f(x_{i,(1)}) \leq \cdots \leq f(x_{i,(N^i)})$. Construct the collections of best and worst subregions $\mathscr{P}_b$ and $\mathscr{P}_w$ using the quantile confidence interval as

$$\begin{aligned} \mathscr{P}_b &= \{\sigma_i | f(x_{i,(N^i)}) < CI_l, \sigma_i \in \Sigma^C_{iter}, i \in 1, \ldots, ||\Sigma^C_{iter}||\} \\ \mathscr{P}_w &= \{\sigma_i | f(x_{i,(1)}) > CI_u, \sigma_i \in \Sigma^C_{iter}, i \in 1, \ldots, ||\Sigma^C_{iter}||\}. \end{aligned}$$

***Step 4. Classify subregions (maintain and prune)***:

***4A and 4B***: For all subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$, uniformly sample additional points such that the total number of points is $N^i_k$ where $k$ is the level of subregion $\sigma_i$,

$$N^i_k = \left\lceil \frac{\ln\left(\alpha/B^k\right)}{\ln\left(1 - \varepsilon/v(S)\right)} \right\rceil.$$

To provide a cap on $N^i_k$, let $N^i_k \leftarrow \min\{N^i_k, 100^{dim} v(\sigma_i)/v(S)\}$. In each subregion, reorder all of its $N^i_k$ sampled points, $x_{i,(1)}, \ldots, x_{i,(N^i_k)}$, by their function values so that $f(x_{i,(1)}) \leq \ldots \leq f(x_{i,(N^i_k)})$. Then update the maintaining indicator functions $M_i$, for $\sigma_i \in \mathscr{P}_b$, and the pruning indicator functions

$P_i$, for $\sigma_i \in \mathscr{P}_w$, as

$$M_i = \left\{ \begin{array}{ll} 1, & \text{if } f(x_{i,(N_k^i)}) < CI_l \\ 0, & \text{otherwise} \end{array} \right\} \text{ and } P_i = \left\{ \begin{array}{ll} 1, & \text{if } f(x_{i,(1)}) > CI_u \\ 0, & \text{otherwise} \end{array} \right\}.$$

Update the maintained set $\Sigma_{iter+1}^M \leftarrow \Sigma_k^M \bigcup_{i \in \mathscr{P}_b : M_i = 1} \sigma_i$, the pruned set $\Sigma_{iter+1}^P \leftarrow \Sigma_k^P \bigcup_{i \in \mathscr{P}_w : P_i = 1} \sigma_i$, and the current set $\Sigma_{iter+1}^C$ to no longer include maintained and pruned subregions.

***4C, 4D, 4E***: For all subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$, if the current number of observed points in subregion $\sigma_i$, $N^i$, exceeds $\left\lceil \ln\left(\alpha / B^k\right) / \ln\left(1 - \varepsilon / v(S)\right) \right\rceil$ where $k$ is the level of subregion $\sigma_i$, then maintain the subregions in $\mathscr{P}_b$ and prune the subregions in $\mathscr{P}_w$. Update the maintained set $\Sigma_{iter+1}^M$, pruned set $\Sigma_{iter+1}^P$, and current set $\Sigma_{iter+1}^C$, accordingly.

***Step 5. Branch and update current subregions***:

***5A***: Branch all remaining subregions in $\Sigma_{iter}^C$.

***5B***: Branch all subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$. If $\mathscr{P}_b$ and $\mathscr{P}_w$ are empty, then branch all remaining subregions in $\Sigma_{iter}^C$.

***5C***: Branch all subregions in $\mathscr{P}_b$ and $\mathscr{P}_w$. If $\mathscr{P}_b$ and $\mathscr{P}_w$ are empty, rank the subregions in $\Sigma_{iter}^C$ by their incumbent function value and branch the best 10% and worst 10%.

***5D, 5E***: Branch all subregions in $\mathscr{P}_b$, $\mathscr{P}_w$, and $\mathscr{P}_u$, where $\mathscr{P}_u$ is a collection of subregions with high uncertainty. If $iter > 1$,

$$\mathscr{P}_u = \{\sigma_i | s_{iter}^{max}(\sigma_i) > \hat{s}_{median}, \text{ for } \sigma_i \in \Sigma_{iter}^C, i \in 1, \ldots, \|\Sigma_{iter}^C\|\},$$

where $\hat{s}_{median}$ is the median of $s_{iter}^{max}(\sigma_i)$ over all $\sigma_i \in \Sigma_{iter}^C$. Update the Gaussian process in each subregion for Algorithms E.

***Step 6. Decision***:

***6A, 6B, 6C, 6D, 6E***: If all subregions $\sigma_i \in \Sigma_{iter}^C$ are not branchable, go to Step 7 and terminate the

algorithm. If there are still branchable subregions in $\Sigma^C_{iter}$, set

$$\delta_{iter+1} = \frac{\delta v(S) - v(\Sigma^M_{iter+1})}{v(\Sigma^C_{iter+1})},$$

and increment the counter $iter \leftarrow iter + 1$, and go to Step 1.

***Step 7. Output results***:
***7A, 7B, 7C, 7D, 7E***: Output $\hat{y}(\delta, S)$, $[CI_l, CI_u]$, and maintained subregions in $\Sigma^M_{iter}$ on the last iteration.

### *3.3 Finite-time Performance Analysis*

We analyze the performance of the multi-level PBnB variations A, B, C and D for level set approximation by deriving confidence intervals on $y(\delta, S)$ and bounds on the probabilities of incorrectly pruning and maintaining subregions at every iteration $iter$. We assume the objective function $f(x)$ can be evaluated exactly. The performance analysis follows the sequence in [114], and we highlight the differences due to the variations introduced in this paper.

First we prove that the lower and upper confidence limits in Step 2 provide an interval estimation of the target quantile. The introduction of multi-level sampling does not alter the main result in Theorem 1, however the analysis must be modified to account for importance sampling in Algorithms C and D. Whereas Algorithms C and D use importance sampling to select a subregion, they sample uniformly on the selected subregion. In contrast, Algorithm E uses EI-sampling on a selected subregion, which does not fit into this finite time analysis.

Then we analyze the quality of the level set approximation by deriving probability bounds on the volume of incorrect pruning and incorrect maintaining in Theorems 2 and 3, for an iteration $iter$. The number of function evaluations on iteration $iter$ is accounted for in $c_{iter}$. The analysis must be modified from the original analysis in [114] to account for multi-level branching as well as importance sampling.

Once we establish the probability bounds for each iteration in Theorems 2 and 3, the accumu-

lation of error over all iterations follows the analysis in [114], and we summarize the final results in Theorems 4 and 5.

We analyze the performance of the PBnB variations A, B, C and D for level set approximation by deriving confidence intervals on $y(\delta, S)$ and bounds on the probabilities of incorrectly pruning and maintaining subregions at every iteration $iter$. We assume the objective function $f(x)$ can be evaluated exactly. The performance analysis follows the sequence in [114], and we highlight the differences due to the variations introduced in this paper.

Theorems 4 and 5 state that for any iteration, and associated $c_{iter}$ function evaluations, the volume of the incorrectly pruned region, and similarly, the volume of the incorrectly maintained region, is at most $\varepsilon$ with a probability of at least $(1-\alpha)^4$. In practice, the parameters $\alpha$ (the significance level in the confidence level of the estimation of $y(\delta, S)$ and in the probabilities of incorrectly pruning and maintaining) and $\varepsilon$ (volume of solutions that can tolerated to be categorized incorrectly) directly influence the number of samples required to statistically confirm whether a subregion should be maintained or pruned. Therefore, the decision maker can use these parameters to empirically manage the trade-off between the accuracy of the level-set approximation and the computational cost. Choosing a smaller $\alpha$ (for higher confidence) or a smaller $\varepsilon$ (for a tighter error bound) will increase the required sample size, thus demanding more function evaluations to classify each subregion.

### *3.3.1 Confidence Interval on $y(\delta, S)$*

In Theorem 1, we let $\varepsilon^M_{iter}$ denote the volume of $\Sigma^M_{iter}$ that is incorrectly maintained, and let $\varepsilon^P_{iter}$ denote the volume of $\Sigma^P_{iter}$ that is incorrectly pruned at iteration $iter$. Since volume is always non-negative, zero is a natural lower bound on $\varepsilon^M_{iter}$ and $\varepsilon^P_{iter}$. Upper bounds on $\varepsilon^M_{iter}$ and $\varepsilon^P_{iter}$ are used in Theorem 1 to provide an interval estimation of the target quantile.

**Theorem 1** For any iteration $iter \geq 1$, suppose $0 \leq \varepsilon^P_{iter} \leq \varepsilon v(\Sigma^P_{iter})/v(S)$ and $0 \leq \varepsilon^M_{iter} \leq \varepsilon v(\Sigma^M_{iter})/v(S)$. Then the bounds of the target quantile are

$$y(\delta^L_{iter}, \Sigma^C_{iter}) \leq y(\delta, S) \leq y(\delta^U_{iter}, \Sigma^C_{iter}) \tag{3.7}$$

where $\delta^L_{iter}$ and $\delta^U_{iter}$ are from (3.2) for the current $\Sigma^C_{iter}$. Therefore an interval estimate of the quantile is

$$P\left(f(z_{(\tilde{r})}) \leq y(\delta, S) \leq f(z_{(\tilde{s})})\right) \geq 1 - \alpha_{iter} \text{ for Algorithms A and B}$$

and is

$$P\left(f(z_{(\tilde{r}')}) \leq y(\delta, S) \leq f(z_{(\tilde{s}')})\right) \geq 1 - \alpha_{iter} \text{ for Algorithms C and D}$$

where $z_{(1)}, \ldots, z_{(c_{iter})}$ are the $c_{iter}$ ordered samples by function values as in Step 2 from the current region $\Sigma^C_{iter}$ at iteration $iter$, $\alpha_{iter} = \alpha/(B)^{iter}$, and $\tilde{r}$ and $\tilde{s}$ are given in Step 2, Equations (3.3) and (3.4), while $\tilde{r}'$ and $\tilde{s}'$ are in Equations (3.5) and (3.6).

**Proof:** To capture the impact of incorrect pruning and maintaining on shifting the $\delta_{iter}$ quantities to correspond to the current set, $\Sigma^C_{iter}$, we provide lower and upper bounds on $\delta_{iter}$ in (3.2) to incorporate the maximum volume error of incorrect maintaining and pruning at iteration $iter$. As in [114][Theorem 1], an upper bound on $y(\delta, S)$ is achieved when $\varepsilon^P_{iter} = 0$ and $\varepsilon^M_{iter} = \varepsilon v(\Sigma^M_{iter})/v(S)$, yielding

$$y(\delta, S) \leq y\left(\delta_{iter} + \frac{\varepsilon v(\Sigma^M_{iter})}{v(S)v(\Sigma^C_{iter})}, \Sigma^C_{iter}\right) = y\left(\delta^U_{iter}, \Sigma^C_{iter}\right) \tag{3.8}$$

and a lower bound is achieved when $\varepsilon^M_{iter} = 0$ and $\varepsilon^P_{iter} = \varepsilon v(\Sigma^P_{iter})/v(S)$, yielding

$$y(\delta, S) \geq y\left(\delta_{iter} + \frac{\varepsilon v(\Sigma^P_{iter})}{v(S)v(\Sigma^C_{iter})}, \Sigma^C_{iter}\right) = y\left(\delta^L_{iter}, \Sigma^C_{iter}\right). \tag{3.9}$$

Therefore,

$$y(\delta^L_{iter}, \Sigma^C_{iter}) \leq y(\delta, S) \leq y(\delta^U_{iter}, \Sigma^C_{iter}). \tag{3.10}$$

Note that when there is no error in pruning and maintaining (i.e., $\varepsilon^P_{iter} = 0$ and $\varepsilon^M_{iter} = 0$), then $y(\delta, S) = y\left(\delta_{iter}, \Sigma^C_{iter}\right)$. Also notice that (3.10) does not depend on the sampling distribution, so is valid for all the PBnB variations.

When we sample uniformly on $\Sigma^C_{iter}$, as in Step 1 of Algorithms A and B, the sampled function values are independent and identically distributed, and each sample function value then acts like a Bernoulli trial and falls in a $\delta^L_{iter}$ or $\delta^U_{iter}$ level set with $\delta^L_{iter}$ or $\delta^U_{iter}$ probability, respectively. Using properties of a binomial distribution, as in [23], we can build a $1-\alpha_{iter}$ quantile confidence interval for $y(\delta,S)$ making use of $\tilde{r}$ and $\tilde{s}$ as in (3.3) and (3.4). The values of $f(z_{(\tilde{r})})$ and $f(z_{(\tilde{s})})$ could be interpreted as the $\tilde{r}/c_{iter}$ quantile and $\tilde{s}/c_{iter}$ quantile, written as $y\left(\tilde{r}/c_{iter},\Sigma^C_{iter}\right)$ and $y\left(\tilde{s}/c_{iter},\Sigma^C_{iter}\right)$, respectively. They provide bounds on $y\left(\delta^L_{iter},\Sigma^C_{iter}\right)$ and $y\left(\delta^U_{iter},\Sigma^C_{iter}\right)$, and coupled with the upper bound in (3.8) and lower bound in (3.9) of $y(\delta,S)$ at iteration $iter$, we can determine the $1-\alpha_{iter}$ confidence interval such that,

$$P\left(f(z_{(\tilde{r})}) \le y\left(\delta^L_{iter},\Sigma^C_{iter}\right) \le y(\delta,S) \le y\left(\delta^U_{iter},\Sigma^C_{iter}\right) \le f(z_{(\tilde{s})})\right) \ge 1-\alpha_{iter} \tag{3.11}$$

when the sampling distribution is uniform on $\Sigma^C_{iter}$.

However, in Algorithms C and D, each subregion is selected with a probability $\tilde{p}_i$ in the current set $\Sigma^C_{iter}$ in Step 1. Therefore, the sampled function values are independent, but not identically distributed. But, once a subregion is selected, the sampled function values in that subregion are independent and identically distributed. To accommodate this importance sampling probability $\tilde{p}$, we adjust $\tilde{r}$ and $\tilde{s}$ and project them to the importance sampling empirical cumulative distribution function (cdf) [37] as $\tilde{r}'$ and $\tilde{s}'$. For the empirical cdf with uniformly distributed sampling (Algorithms A and B), we give equal weight to each sample point to obtain the bounds in (3.11), however now we need to weight the points according to the subregion they are in.

Given the ordered sample,

$$f(z_{(1)}) \le \ldots \le f(z_{(c_{iter})})$$

of the $c_{iter}$ points in Step 2, the importance sampling probability of subregion $m_{(z_{(i)})}$ is denoted $\tilde{p}_{m_{(z_{(i)})}}$ where the subregion $m_{(z_{(i)})}$ is the subregion associated with the ordered sample point $z_{(i)}$.The sampling probability of subregion $m_{(z_{(i)})}$ using uniform sampling (as in Algorithms A and B) is denoted $p_{m_{(z_{(i)})}}$, as in (3.1).

The empirical cdf with importance sampling is adjusted to weight the subregions according to

$\tilde{p}$ in order to achieve a $1-\alpha_{iter}$ confidence interval, as given in (3.11). According to the empirical cdf $\tilde{F}(y)$ with importance sampling proposed in [37],

$$\tilde{F}(y) = \sum_{i=1}^{c_{iter}} \frac{1}{c_{iter}} \frac{p_{m_{(z_{(i)})}}}{\tilde{p}_{m_{(z_{(i)})}}} I(Y \leq y)$$

where $I(Y \leq y)$ is the indicator function that a sampled function value is no greater than $y$. We estimate the $\tilde{r}/c_{iter}$ quantile and $\tilde{s}/c_{iter}$ quantile from uniform sampling with $\tilde{F}^{-1}(\tilde{r}/c_{iter})$ and $\tilde{F}^{-1}(\tilde{s}/c_{iter})$ from importance sampling. This yields $\tilde{r}'$ and $\tilde{s}'$ where

$$\tilde{r}' = \max r : \sum_{i=1}^{r} \frac{1}{c_{iter}} \frac{p_{m_{(z_{(i)})}}}{\tilde{p}_{m_{(z_{(i)})}}} \leq \frac{\tilde{r}}{c_{iter}}, \text{ and } \tilde{s}' = \min s : \sum_{i=1}^{s} \frac{1}{c_{iter}} \frac{p_{m_{(z_{(i)})}}}{\tilde{p}_{m_{(z_{(i)})}}} \geq 1 - \frac{\tilde{s}}{c_{iter}}$$

and $\tilde{r}$ and $\tilde{s}$ are from (3.3) and (3.4), respectively. Hence, we can determine the $1-\alpha_{iter}$ confidence interval such that,

$$P\left( f(z_{(\tilde{r}')}) \leq y\left(\delta^L_{iter}, \Sigma^C_{iter}\right) \leq y(\delta, S) \leq y\left(\delta^U_{iter}, \Sigma^C_{iter}\right) \leq f(z_{(\tilde{s}')}) \right) \geq 1 - \alpha_{iter}$$

for Algorithms C and D. □

### *3.3.2 Probability Bounds for an Iteration*

We next analyze the quality of the level set approximation by deriving probability bounds on the volume of incorrect pruning and incorrect maintaining in Theorems 2 and 3, for an iteration $iter$.

We define several events to ensure the conditions of Theorem 1 are met, and the interval estimate on $y(\delta, S)$ is correct. Theorem 1 assumes that $\varepsilon^M_{iter}$ is bounded by $\varepsilon v(\Sigma^M_{iter})/v(S)$ which we denote as event,

$$A^M_{iter} = \left\{ v(\Sigma^M_{iter}) - v\left( L(\delta,S) \cap v(\Sigma^M_{iter}) \right) \leq \frac{\varepsilon v(\Sigma^M_{iter})}{v(S)} \right\}$$
$$= \left\{ v\left( L(\delta,S) \cap v(\Sigma^M_{iter}) \right) \geq v(\Sigma^M_{iter}) - \frac{\varepsilon v(\Sigma^M_{iter})}{v(S)} \right\}$$

and that $\varepsilon^P_{iter}$ is bounded by $\varepsilon\Sigma^P_{iter}/v(S)$ which we represent by the event

$$A^P_{iter} = \left\{ v\left( L(\delta,S) \cap v(\Sigma^P_{iter}) \right) \leq \frac{\varepsilon v(\Sigma^P_{iter})}{v(S)} \right\}.$$

The event that $y(\delta,S)$ is bounded correctly is denoted by

$$A^{CI}_{iter} = \left\{ f(z_{(\tilde{r})}) \leq y\left(\delta^L_{iter}, \Sigma^C_{iter}\right) \leq y(\delta,S) \leq y\left(\delta^U_{iter}, \Sigma^C_{iter}\right) \leq f(z_{(\tilde{s})}) \right\}$$

for Algorithms A and B, and

$$= \left\{ f(z_{(\tilde{r}')}) \leq y\left(\delta^L_{iter}, \Sigma^C_{iter}\right) \leq y(\delta,S) \leq y\left(\delta^U_{iter}, \Sigma^C_{iter}\right) \leq f(z_{(\tilde{s}')}) \right\}$$

for Algorithms C, and D.

All three events, denoted

$$A_{iter} = \left\{ A^M_{iter} \cap A^P_{iter} \cap A^{CI}_{iter} \right\}, \tag{3.12}$$

ensure that $y(\delta,S)$ is bounded correctly on iteration *iter*, i.e., $f(z_{(\tilde{r})}) \leq y\left(\delta^L_{iter}, \Sigma^C_{iter}\right) \leq y(\delta,S) \leq y\left(\delta^U_{iter}, \Sigma^C_{iter}\right) \leq f(z_{(\tilde{s})})$ holds for Algorithms A and B, and $f(z_{(\tilde{r}')}) \leq y\left(\delta^L_{iter}, \Sigma^C_{iter}\right) \leq y(\delta,S) \leq y\left(\delta^U_{iter}, \Sigma^C_{iter}\right) \leq f(z_{(\tilde{s}')})$ holds for Algorithms C and D, and the volume of incorrect pruning and maintaining is bounded.

**Theorem 2** Consider an iteration *iter* of Algorithms A, B, C, D on problem $(\mathscr{P})$ and suppose $\hat{\sigma}^P_{iter}$ has been pruned on the iteration *iter*. Let $\hat{\sigma}^P_{iter,k,m}$ be a subregion pruned at level $k$, and let $\hat{\sigma}^P_{iter,k} = \bigcup_m \hat{\sigma}^P_{iter,k,m}$ be the set of subregions at level $k$ that have been pruned on the iteration *iter*. Also, $\hat{\sigma}^P_{iter} = \bigcup_k \hat{\sigma}^P_{iter,k}$ is the set of all subregions that have been pruned on the iteration *iter*. Each

subregion $\hat{\sigma}^P_{iter,k,m}$ is selected with probability $\tilde{p}_{iter,k,m}$ in Step 1.

Suppose the event $A_{iter}$ in (3.12) is true. Then, the event that the volume of the incorrectly pruned region, i.e., $v\left(L(\delta,S)\cap\hat{\sigma}^P_{iter}\right)$ is less than or equal to $\sum_k D^P_{iter,k}\varepsilon_k$, where $D^P_{iter,k}$ is the number of subregions of level $k$ pruned at iteration $iter$ and $\varepsilon_k = \varepsilon/B^k$, occurs with probability at least $\prod_k(1-\alpha_k)$, that is

$$P\left(v\left(L(\delta,S)\cap\hat{\sigma}^P_{iter}\right)\leq\sum_k D^P_{iter,k}\varepsilon_k|A_{iter}\right)\geq\prod_k(1-\alpha_k), \tag{3.13}$$

where $\alpha_k = \alpha/B^k$.

The proof below uses the quantile definition to bound the probability that the pruned set does not incorrectly contain the target set up to a maximum error, conditioned on the event $A_{iter}$. The probability statement from Theorem 1 coupled with order statistics for the best sample in the subregion is used to further bound the desired probability. The sample size used in Step 4 for level $k$ is determined to achieve the desired $1-\alpha_k$ probability bound. For Algorithms C and D it is important to keep track of each pruned subregion $m$ at level $k$, $\hat{\sigma}^P_{iter,k,m}$, whereas Algorithms A and B sample uniformly so it is not necessary to keep track of the subregions.

**Proof:** At iteration $iter$, for the pruned subregion $m$ at level $k$, $\hat{\sigma}^P_{iter,k,m}$, we note that the event $\left\{v\left(L(\delta,S)\cap\hat{\sigma}^P_{iter,k,m}\right)\leq\varepsilon_k\right\}$ is equivalent to the event $\left\{v\left(\{x\in\hat{\sigma}^P_{iter,k,m}:f(x)\leq y(\delta,S)\}\right)\leq\varepsilon_k\right\}$ by the definition of $L(\delta,S)$, and therefore, the probability of that event, that is, that the volume of the incorrectly pruned subregion $\hat{\sigma}^P_{iter,k,m}$ is less than or equal to $\varepsilon_k$, given the event $A_{iter}$ is true, can be expressed as

$$P\left(v\left(L(\delta,S)\cap\hat{\sigma}^P_{iter,k,m}\right)\leq\varepsilon_k|A_{iter}\right)=P\left(v\left(\{x\in\hat{\sigma}^P_{iter,k,m}:f(x)\leq y(\delta,S)\}\right)\leq\varepsilon_k|A_{iter}\right). \tag{3.14}$$

Now, consider the probability expression of quantile and let $\delta^p_{iter,k,m}=\varepsilon_k/v(\hat{\sigma}^P_{iter,k})$. We first assume that $y(\delta,S)$ is continuous in $\delta$ and $y(\delta^p_{iter,k,m},\hat{\sigma}^P_{iter,k,m})$ is continuous in $\delta^p_{iter,k,m}$, and that

$v\Big(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) = y\}\Big) = 0, \forall y$. When $X$ is a uniform sample in $\hat{\sigma}^P_{iter,k,m}$, we have

$$\begin{aligned} P\Big(f(X) < y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\Big) &= \frac{v\Big(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) < y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big)}{v(\hat{\sigma}^P_{iter,k,m})} \\ &= \delta^p_{iter,k,m} \\ &= \frac{\varepsilon_k}{v(\hat{\sigma}^P_{iter,k,m})} \end{aligned}$$

then multiplying $v(\hat{\sigma}^P_{iter,k,m})$ on both sides, we have

$$v\Big(\{x \in \hat{\sigma}^P_{iter,k,m}) : f(x) < y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big) = \varepsilon_k.$$

Hence, we have

$$\begin{aligned} \varepsilon_k &= v\Big(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) < y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big) \\ &= v\Big(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \leq y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big) \\ &\quad - v\Big(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) = y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big) \end{aligned}$$

and in the special case that $v\Big(\{x \in \hat{\sigma}^P_{iter,k,m}) : f(x) = y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big) = 0, \forall y$, we have,

$$\varepsilon_k = v\Big(\{x \in \hat{\sigma}^P_{iter,k,m}) : f(x) \leq y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big). \tag{3.15}$$

We substitute the expression for $\varepsilon_k$ from (3.15) into the probability expression in (3.14), yielding,

$$\begin{aligned} &P\Big(v\Big(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \leq y(\delta, S)\}\Big) \leq \varepsilon_k | A_{iter}\Big) \\ &= P\Big(v\Big(L(\delta, S) \cap \hat{\sigma}^P_{iter,k,m}\Big) \leq v\Big(\{x \in \hat{\sigma}^P_{iter,k,m}) : f(x) \leq y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\Big) | A_{iter}\Big) \end{aligned}$$

and from the properties of level sets if $y(\delta,S) \le y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})$ then $\left\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \le y(\delta,S)\right\} \subseteq \left\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \le y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\right\}$, therefore,

$$P\left(v\left(L(\delta,S) \cap \hat{\sigma}^P_{iter,k,m}\right) \le \varepsilon_k | A_{iter}\right) = P\left(y(\delta,S) \le y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m}) | A_{iter}\right)$$

and in the special case that $v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) = y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})\}\right) = 0, \forall y$, we have that

$$P\left(v\left(L(\delta,S) \cap \hat{\sigma}^P_{iter,k,m}\right) \le \varepsilon_k | A_{iter}\right) = P\left(y(\delta,S) < y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m}) | A_{iter}\right)$$

and by the condition $A_{iter}$ and the pruned assumption, we have $y(\delta,S) \le y(\delta^U_{iter}, \Sigma^C_{iter}) \le f(z_{(\tilde{s})}) < f(x_{(p),(1)})$ where $x_{(p),(1)}$ is the best sample out of $N^P_{iter,k,m}$ independent sample in $\hat{\sigma}^P_{iter,k,m}$. Therefore,

$$\begin{aligned} P\left(v\left(L(\delta,S) \cap \hat{\sigma}^P_{iter,k,m}\right) \le \varepsilon_k | A_{iter}\right) &\ge P\left(f(x_{(p),(1)}) \le y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m}) | A_{iter}\right) \\ &= 1 - P\left(f(x_{(p),(1)}) > y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m}) | A_{iter}\right), \end{aligned}$$

and since each of $N^P_{iter,k,m}$ samples in $\hat{\sigma}^P_{iter,k,m}$ are independent and uniformly distributed, we have

$$P\left(v\left(L(\delta,S) \cap \hat{\sigma}^P_{iter,k,m}\right) \le \varepsilon_k | A_{iter}\right) \ \ge 1 - (1 - \delta^p_{iter,k,m})^{N^P_{iter,k,m}}.$$

We know $N^P_{iter,k,m} \ge \left\lceil \ln(\alpha/B^k)/\ln(1-\varepsilon/v(S)) \right\rceil$ in Step 4, and $\delta^p_{iter,k,m} = \varepsilon_k / v(\hat{\sigma}^P_{iter,k,m}) = \varepsilon / v(S)$. Also, $N^P_{iter,k,m} \ge \ln(\alpha/B^k)/\ln(1-\delta^p_{iter,k,m}) \Rightarrow ln(1-\delta^p_{iter,k,m})^{N^P_{iter,k}} \le \ln \alpha_k \Rightarrow (1-\delta^p_{iter,k,m})^{N^P_{iter}} \le \alpha_k$.

Multiplying -1 on both sides and adding one to both sides, the inequality becomes $1-(1-\delta^p_{iter,k,m})^{N^P_{iter,k,m}} \ge 1-\alpha_k$, hence,

$$P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \le y(\delta,S)\}\right) \le \varepsilon_k | A_{iter}\right) \ge 1 - \alpha_k, \tag{3.16}$$

or we can write,

$$P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \leq y(\delta,S)\}\right) \geq \varepsilon_k | A_{iter}\right) < \alpha_k. \tag{3.17}$$

Whereas (3.17) provides a probability bound for one subregion at level $k$, we are interested in a probability bound on all subregions at level $k$. Let $D^P_{iter,k}$ be the number of pruned subregions at level $k$ at iteration $iter$. We consider the union of all events at level $k$ to bound the probability of incorrect pruning at level $k$, written as

$$\begin{aligned}
&P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k} : f(x) \leq y(\delta,S)\}\right) \geq D^P_{iter,k}\varepsilon_k | A_{iter}\right) \\
&\leq P\left(\bigcup_{D^P_{iter,k}} \left\{v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \leq y(\delta,S)\}\right) \geq \varepsilon_k | A_{iter}\right\}\right).
\end{aligned}$$

Since the probability of each subregion incorrectly pruned is bounded we have that

$$\begin{aligned}
&P\left(\bigcup_{D^P_{iter,k}} \left\{v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \leq y(\delta,S)\}\right) \geq \varepsilon_k | A_{iter}\right\}\right) \\
&\leq \sum_{m=1}^{D^P_{iter,k}} P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \leq y(\delta,S)\}\right) \geq \varepsilon_k | A_{iter}\right).
\end{aligned}$$

And, since each subregion $\hat{\sigma}^P_{iter,k,m}$ is selected with probability $\tilde{p}_{iter,k,m}$, we have,

$$\begin{aligned}
&P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k} : f(x) \leq y(\delta,S)\}\right) \geq D^P_{iter,k}\varepsilon_k | A_{iter}\right) \\
&\leq \sum_{m=1}^{D^P_{iter,k}} \frac{\tilde{p}_{iter,k,m}}{\sum_{m=1}^{D^P_{iter,k}} \tilde{p}_{iter,k,m}} P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) \leq y(\delta,S)\}\right) \geq \varepsilon_k | A_{iter}\right) \\
&< \sum_{m=1}^{D^P_{iter,k}} \frac{\tilde{p}_{iter,k,m}}{\sum_{m=1}^{D^P_{iter,k}} \tilde{p}_{iter,k,m}} \alpha_k = \alpha_k.
\end{aligned}$$

Hence, at each level $k$,

$$P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k} : f(x) \leq y(\delta,S)\}\right) \geq D^P_{iter,k}\varepsilon_k | A_{iter}\right) < \alpha_k.$$

Then $P\left(v\left(\{x \in \hat{\sigma}^P_{iter,k} : f(x) \leq y(\delta,S)\}\right) \leq D^P_{iter,k}\varepsilon_k | A_{iter}\right) \geq 1 - \alpha_k$. Since samples in $\hat{\sigma}^P_{iter,k}$ at

each level $k$ are sampled independently,

$$P\left(v\left(\{x \in \hat{\sigma}^P_{iter} : f(x) \leq y(\delta,S)\}\right) \leq \sum_k D^P_{iter,k}\varepsilon_k | A_{iter}\right) \geq \prod_k (1-\alpha_k).$$

Now in the more general case, where $y(\delta,S)$ and $y(\delta^p_{iter,k,m}, \hat{\sigma}^P_{iter,k,m})$ may have discontinuities, the $v\left(\{x \in \hat{\sigma}^P_{iter,k,m} : f(x) = y\}\right)$ may be positive for some $y$, however, the flow of the proof is the same, where the possibility of discontinuities changes some equalities to inequalities. The details are given in the proof of Theorem 2 in [114]. □

**Theorem 3** Consider an iteration $iter$ of Algorithms A, B, C, D, or E on problem $(\mathscr{P})$ and suppose $\hat{\sigma}^M_{iter}$ has been maintained on the iteration $iter$. Let $\hat{\sigma}^M_{iter,k,m}$ be a subregion maintained at level $k$, and let $\hat{\sigma}^M_{iter,k} = \bigcup_m \hat{\sigma}^M_{iter,k,m}$ be the set of subregions at level $k$ that have been maintained on the iteration $iter$. Also, $\hat{\sigma}^M_{iter} = \bigcup_k \hat{\sigma}^M_{iter,k}$ is the set of all subregions that have been maintained on the iteration $iter$. Each subregion $\hat{\sigma}^M_{iter,k,m}$ is selected with probability $\tilde{p}_{iter,k,m}$ in Step 1.

Suppose the event $A_{iter}$ in (3.12) is true. Then, the event that the volume of the correctly maintained region is greater than or equal to $v\left(\hat{\sigma}^M_{iter}\right) - \sum_k D^M_{iter,k}\varepsilon_k$ where $D^M_{iter,k}$ is the number of subregions of level $k$ maintained at iteration $iter$ and $\varepsilon_k = \varepsilon/B^k$ occurs with probability at least $\prod_k (1-\alpha_k)$, or, in other words, the volume of the incorrectly maintained region is less than or equal to $\sum_k D^M_{iter,k}\varepsilon_k$ with probability $\prod_k (1-\alpha_k)$,

$$P\left(v\left(\hat{\sigma}^M_{iter}\right) - v\left(L(\delta,S)\right) \leq \sum_k D^P_{iter,k}\varepsilon_k | A_{iter}\right) \geq \prod_k (1-\alpha_k), \tag{3.18}$$

where $\alpha_k = \alpha/B^k$.

**Proof:** The proof is similar to the proof of Theorem 2.

### *3.3.3 Accumulated Error*

Now that the multi-level branching and importance sampling have been accounted for in Theorems 1, 2 and 3, the accumulated error probability bounds follow directly, as in [114].

**Theorem 4** For any iteration $iter \geq 1$, the volume of the incorrectly pruned region is at most $\varepsilon$ with probability at least $(1-\alpha)^4$, that is

$$P\left( v(L(\delta,S) \cap \Sigma_{iter}^{P}) \leq \varepsilon \right) \geq (1-\alpha)^4. \tag{3.19}$$

**Proof:** (cf. [114][Theorem 5])

**Theorem 5** For any iteration $iter \geq 1$, the volume of the incorrectly maintained region is at most $\varepsilon$ with probability at least $(1-\alpha)^4$, that is

$$P\left( v(\Sigma_{iter}^{M} \backslash L(\delta,S)) \leq \varepsilon \right) \geq (1-\alpha)^4. \tag{3.20}$$

**Proof:** Proof is similar to the proof of Theorem 5.

Although this part of the dissertation assumes the objective function can be evaluated without noise, the addition of noise in [114] can be readily applied to these PBnB variations.

### *3.4 Numerical Experiments*

We ran computational experiments on five algorithms: Original PBnB (Algorithm A), multi-level PBnB (Algorithm B) and multi-level PBnB with Incumbent-based Sampling (Algorithm C), Uncertainty-based GP Sampling (Algorithm D) and Uncertainty- and EI-based GP Sampling (Algorithm E). All five algorithms were run on three different test functions, the Rosenbrock problem, the centered sinusoidal problem, and the shifted sinusoidal problem, for dimensions 2, 5, 7, and 10. The test problems are defined in the appendix. For each test condition, 10 replications were performed with common random number seeds and the results averaged. The average number of function evaluations until the first subregion is maintained is the main metric of interest. For all test conditions, we set the common PBnB parameter values, $\delta = 0.2$, $\alpha = 0.1$, $\varepsilon = 0.025(\text{vol}(S))$, $c = dim * 100$, and the unbranchable size $= 0.025(\text{vol}(S))$. For all algorithms, the branching parameter $B$ was tested with $B = \{2,4\}$.

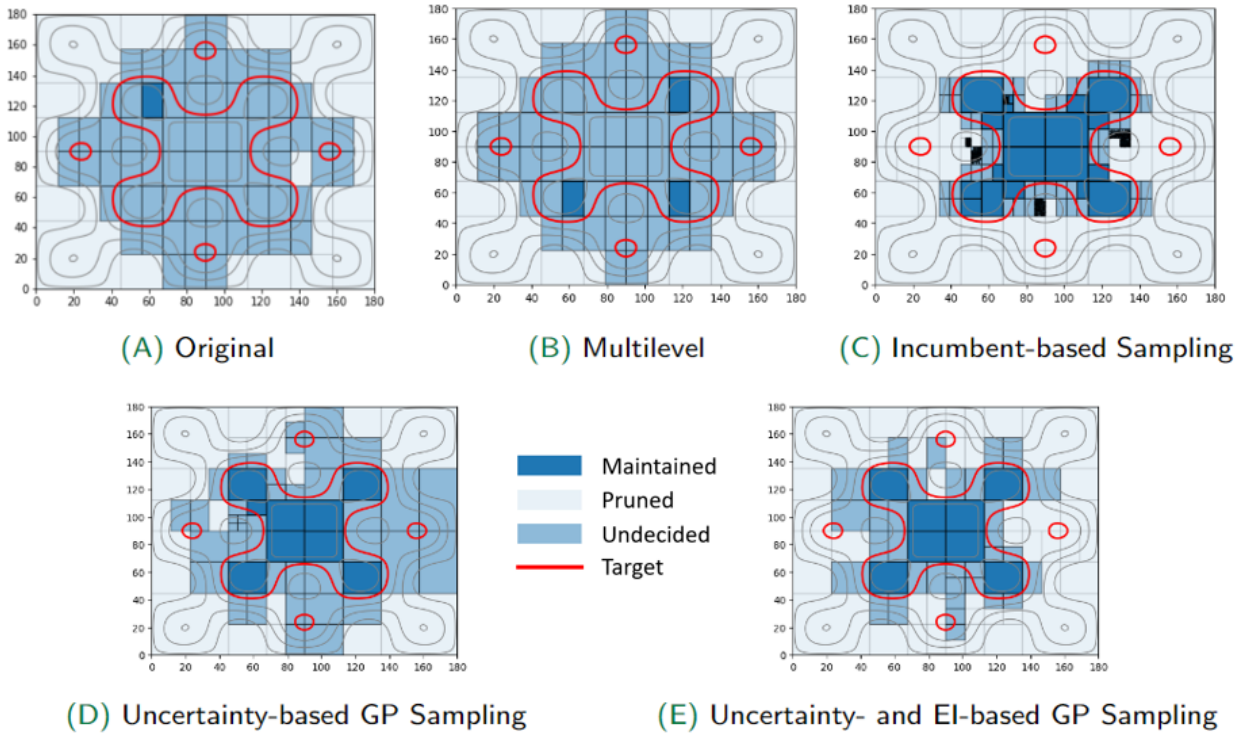



**Figure 3.2:** Approximating the 0.2-quantile level set (contour shown in red) on 5,000 function evaluations, $B = 2$, $c = 100 * dim$, $\delta = 0.2$, $\alpha = 0.1$ , $\varepsilon = 0.025(vol(S))$

We first examine the performance of different variations of PBnB. Figure 3.2 illustrates all algorithms approximating a 20% level set after 5,000 function evaluations on centered sinusoidal functions in two dimensions. Notice that the subregions of Algorithm A are of the same size on each iteration, whereas the subregions in Algorithm B, C, D, and E (multi-level) are of different sizes. In these cases, improvement can be seen as algorithm B, C, D, and E are able to maintain more regions in the desired level set (within the red line).

This suggests that variations of multi-level PBnB is able to prune and maintain faster by narrowing in on promising subregions, whereas Original PBnB focuses on searching the entire space. The ANOVA analysis result of five factors, including test function, algorithm, B, C, and dimension are presented in Figure 3.3. Test of simple main-effects of Algorithms, B and c are significant. The main effect plot also demonstrates the performance improvement upon variations of PBnB.

Figure 3.4 provides the average over 10 replications of the number of function evaluations until

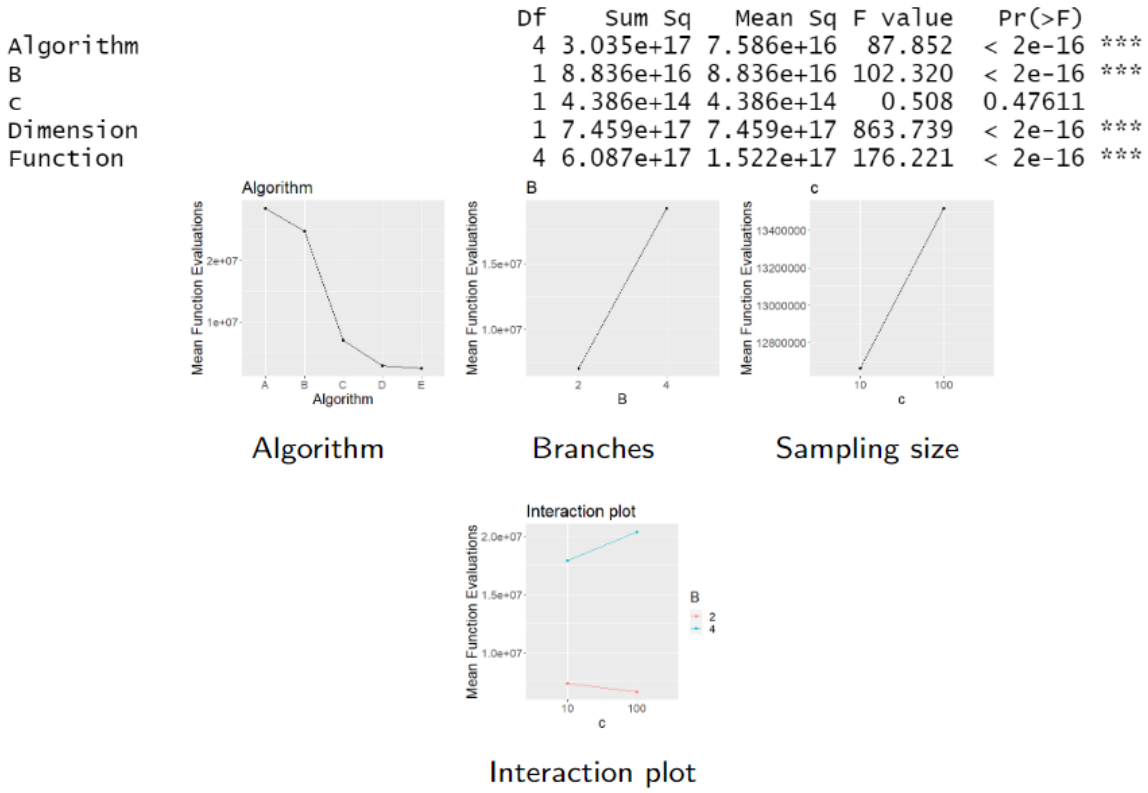



**Figure 3.3:** ANOVA analysis result, main effect plot and interaction effect plot of the five variation of PBnB

first maintaining a subregion with the best experiment design with $B = 2$ and $c = 100 * dim$ . From the t-test analysis, we can conclude that: (1) Multi-level PBnB improves upon Original PBnB (with 75% confidence), (2) Incumbent-based Sampling PBnB improves upon Multi-level PBnB (with 90% confidence), (3) Uncertainty-based GP Sampling PBnB improves upon Incumbent-based Sampling PBnB (with 95% confidence), and (4) Uncertainty and EI-based GP Sampling PBnB improves upon Uncertainty-based GP Sampling PBnB (with 95% confidence).

## *3.5 Conclusion*

This chapter proposes and compares five variations of multi-PBnB for level set approximation. Original PBnB involves branching the problem space until subregions can be pruned or main-

| | dim | Original (Alg. A) Mean | Multilevel (Alg. B) Mean | Incumbent (Alg. C) Mean | Uncert. GP (Alg. D) Mean | EI GP (Alg. E) Mean |
|---|---|---|---|---|---|---|
| Rosenb. | 2 | 4,610 | 3,830 | 1,527 | 1,520 | 695 |
| | 5 | 74,715 | 74,656 | 31,782 | 9,452 | 6,480 |
| | 7 | 1,134,184 | 835,671 | 215,721 | 21,980 | 16,142 |
| | 10 | 17,856,257 | 11,039,015 | 10,765,121 | 4,136,412 | 4,078,984 |
| Cen. Sin. | 2 | 5,327 | 4,476 | 2,498 | 1,640 | 892 |
| | 5 | 323,781 | 251,539 | 187,795 | 6,120 | 4,701 |
| | 7 | 1,836,140 | 1,492,808 | 1,129,481 | 10,570 | 7,610 |
| | 10 | 134,267,137 | 118,484,057 | 56,145,091 | 20,719,432 | 14,411,290 |
| Shif. Sin. | 2 | 3,289 | 2,667 | 1,270 | 1,220 | 1,008 |
| | 5 | 2,988 | 2,187 | 1,490 | 1,275 | 1,290 |
| | 7 | 3,696 | 3,782 | 1,629 | 1,140 | 1,328 |
| | 10 | 96,176 | 71,156 | 29,965 | 24,465 | 18,642 |

**Figure 3.4:** Mean number of function evaluations until first maintained subregion over 10 replications where $B = 2$ and $c = 100 * dim$

tained with statistical confidence. The proliferation of subregions and the extensive number of sample points required to classify subregions as maintained or pruned can be problematic, especially at high dimensions. Multi-level PBnB improves the algorithm by only branching on the most promising subregions, while adding importance sampling improves the algorithm by focusing sampling density on the subregions with better observed function values. Our numerical experiment demonstrates that multi-level PBnB performs better than Original PBnB with statistical confidence in terms of number of function evaluations required to first maintain a subregion, and by adding Uncertainty-based Sampling where the algorithms select subregions according to posterior distributions derived from Gaussian process estimation of uncertainty, the modified algorithms perform even better. This chapter also provides a finite-time performance analysis on the quality of the level set approximation by deriving probability bounds on the volume of incorrect pruning and incorrect maintaining to be a guideline for a decision maker for trading off between accuracy of the level-set approximation and computation cost. Throughout this chapter, the three mains of the dissertation are satisfied.

# Chapter 4

# BRANCHING ADAPTIVE SURROGATE SEARCH OPTIMIZATION (BASSO)

This chapter proposes the Branching Adaptive Surrogate Search Optimization (BASSO) framework to address the challenges of high dimensionality. The contributions of this chapter to the three main goals of this dissertation are the BASSO framework that incorporates partitioning and a machine learning model (Gaussian process and regularized quadratic regression) into variations of BASSO implementations, the finite-time analysis of BASSO, and numerical experiments of BASSO variations. The proposed BASSO framework is inspired by insights into the impact of high dimensions on the sampling distribution from adaptive search methods, including Pure Adaptive Search (PAS) [110, 116, 118] and Hesitant Adaptive Search (HAS) [19, 108, 115]. In this chapter, BASSO conceptualizes framework that use of partitioning and surrogate modeling for black-box optimization to mimic sampling from a Boltzmann distribution to increase the chance of sampling in the improving region. The result of this section is to establish conditions under which the expected number of function evaluations required to reach an $\varepsilon$-optimal solution is bounded by an expression that is linear in dimension implies scalability to high dimension, under two conditions: specifically, conditions on adaptive subregion probabilities and the probability of improvement with a surrogate model. We explore several BASSO implementations and evaluate which variations most closely satisfy these conditions. Each variation of BASSO proposed in this research differs in its local surrogate model, which guides local sampling within a subregion, and in its adaptive subregion probabilities, which identify promising subregions. We investigate machine learning surrogates by comparing a Gaussian process and regularized quadratic regression against a uniform sampling baseline. Numerical results on test problems are presented to illustrate the gap between theory and implementation.

### *4.1 Introduction and Background*

In order to develop scalable algorithms, it is important to gain insight into the impact of high dimensions on the sampling distribution. The impact of high dimensions on computation has been studied through finite-time analyses of several stochastic adaptive search methods, including, Pure Adaptive Search (PAS) [110, 116, 118] and Hesitant Adaptive Search (HAS) [19, 108, 115]. In particular, under certain conditions, the expected number of PAS (and HAS) function evaluations required to sample below a specified objective function value increases only *linearly* in the dimension of the input when optimizing a function without noise. We refer to this as the "linearity result." While PAS and HAS are not directly implementable, the theoretical analyses contribute important insights to global optimization.

The introduction of Annealing Adaptive Search (AAS) [94, 95, 107] was an attempt to narrow the gap between theory and implementation. AAS was created as a conceptual form of simulated annealing where points are sampled over the original domain according to the Boltzmann distribution parameterized by temperature. The desired linearity result can be achieved by the conceptual AAS with a derived cooling schedule for the temperature parameter [95]. However, it is still impractical to sample exactly from a Boltzmann distribution, so an implementation is still illusive.

Another situation where theory was used to aid implementation was in [112]. Since cooling the temperature in simulated annealing too quickly may result in premature convergence, the HAS theory was adapted to model the behavior of simulated annealing and similar stochastic optimization algorithms. The combination of theory and observations are used to determine a stopping and restarting strategy that resulted in an implementable algorithm, Dynamic Multistart Sequential Search (DMSS).

Many optimization algorithms have been developed for black-box optimization that search the whole domain for the global optimum, as reviewed in [4, 5, 85]. Surrogate models of the function to optimize, especially Gaussian processes, have been used widely to determine the next point in the domain to evaluate [35, 41, 84]. Partition-based algorithms, as in [97, 113], create smaller subregions in order to prioritize the exploration of promising areas as a way to dynamically update

the sampling distribution. Partitioning combined with surrogate models, as in [51, 52, 66, 76, 77, 119], have been shown to be successful in many applications. The prior analyses of PAS, HAS, and AAS are not immediately applicable to surrogate modeling and partition-based algorithms. The question remains, *is it possible for surrogate modeling and partition-based algorithms to achieve the ideal linearity result? And if so, what is needed to make these algorithms scalable to high dimensions?*

To address this, we introduce Branching Adaptive Surrogate Search Optimization (BASSO), which conceptualizes the use of branching and surrogate modeling. We present a new finite-time analysis of BASSO and prove that the desired linearity result is achievable with two assumptions. Specifically, we establish conditions under which the expected number of function evaluations required to reach an $\varepsilon$-optimal solution is bounded by an expression that is linear in dimension. While these conditions are impractical to satisfy completely in practice, we explore them with numerical implementations and discuss insights gained. The analysis is valid when the domain is continuous and when the domain is finite, such as a bounded integer lattice.

The main objective of this section is to introduce of BASSO with a finite-time analysis of performance, providing a framework with partitioning schemes and surrogate modeling within subregions that are effective for black-box global optimization. A contribution of the BASSO framework is that it provides conditions that would make an algorithm scalable to high dimensions. The theoretical result allows researchers to make informed trade-offs between exploration and exploitation in designing algorithms.

### *4.2 BASSO*

The BASSO algorithm uses the information obtained on sampled points and their observed objective function values to partition the domain, construct surrogate models, and update a sampling procedure that focuses computational effort on promising subregions.

The optimization problem we consider is

$$\min_{x \in S} f(x) \tag{P}$$

where $f(x) : S \to \mathbb{R}$, and $S \subset \mathbb{R}^d$. Assume $S$ is compact. The feasible region $S$ may be a continuous space, or a finite space. The decision variable $x$ is a vector in $d$ dimensions. We assume a unique minimum, and denote the minimum value and the optimal point in the domain, respectively, as:

$$y_* = \min_{x \in S} f(x) \quad \text{and} \quad x_* = \arg\min_{x \in S} f(x).$$

Similarly, denote the maximum value as $y^* = \max_{x \in S} f(x)$.

The adaptive sampling scheme in BASSO to generate a new sample point on the $k$th iteration has three main components:

(i) Branch a collection of subregions.

(ii) Evaluate the adaptive subregion probabilities $\tilde{p}_i$ for current subregions $\sigma_i$ and select a promising subregion for sampling.

(iii) Generate a point within the selected subregion according to the uniform distribution or a surrogate model.

A motivation for BASSO's adaptive sampling scheme is the effectiveness of a sequence of Boltzmann distributions, as in AAS. Figure 4.1 illustrates the concept of AAS and BASSO on a one-dimensional objective function (shown in the top panel) to be minimized. The adaptive sampling procedure in AAS samples from a Boltzmann distribution where the temperature parameter $T$ is lowered when an improving point is found. The middle panel in Figure 4.1 illustrates how the Boltzmann densities increasingly focus on the optimum as temperature decreases.

In BASSO, the adaptive subregion probability parameter $\tilde{p}$ is a vector that captures the probability of selecting a subregion for further sampling. The bottom panel in Figure 4.1 illustrates how adapting $\tilde{p}$ can focus the sampling distribution on the promising subregions. In the example, the domain is partitioned into three subregions, and initially $\tilde{p}$ is roughly $[1/3, 1/3, 1/3]$, representing a uniform distribution. When the adaptive subregion probability parameter is updated to $\tilde{p} = [0.2, 0.6, 0.2]$, and then to $\tilde{p} = [0.1, 0.8, 0.1]$, there is a higher probability of sampling from the middle subregion with the global optimum. We draw an analogy to sampling from a Boltzmann

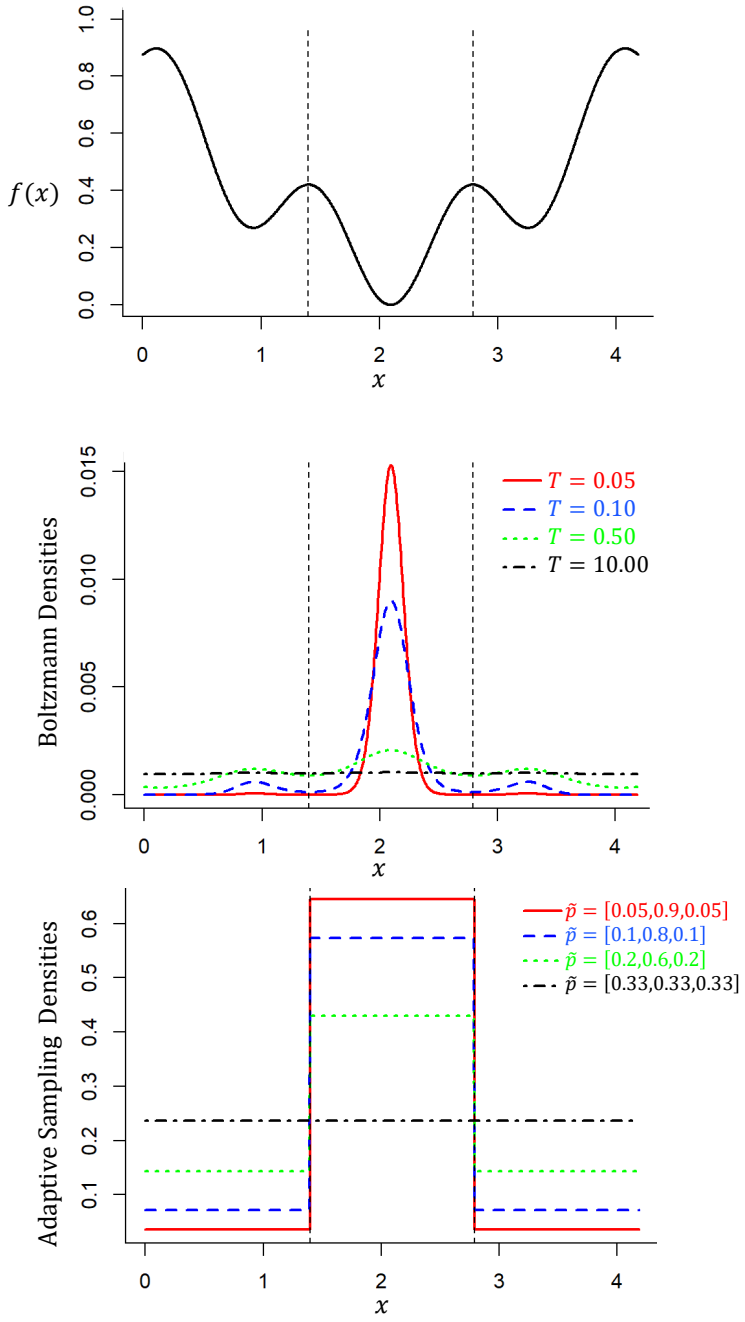



**Figure 4.1:** Sampling from a Boltzmann distribution parameterized by temperature induces the densities to focus more on the optimum as the temperature decreases. In BASSO, the adaptive subregion probability parameter $\tilde{p}$ impacts the sampling distribution.

distribution parameterized by temperature, to sampling from subregions parameterized by $\tilde{p}$. As BASSO progresses, $\tilde{p}$ is adapted to focus on the subregion that contains the optimum. In the analysis of BASSO, Assumption 1 gives a condition on how $\tilde{p}$ is updated that essentially ensures that $\tilde{p}$ is consistent with the contours of the objective function and does not focus on subregions that do not contain the global optimum. In practice, this condition is difficult to always satisfy, as we show in the numerical experiments.

### *4.2.1 BASSO Framework*

The BASSO framework subdivides the feasible region into smaller subregions, using sequential partitioning. Let $\Sigma_k^C$ be the collection of subregions that are produced by branching on the $k$th iteration, $\Sigma_k^C = \{\sigma_1, \ldots, \sigma_{m_k}\}$, where $m_k$ is the number of subregions on the $k$th iteration. We assume that the branching strategy is such that the union of all subregions equals $S$, and that the subregions are mutually exclusive. The subregions do not need to be of equal size. We define the size of subregion $\sigma_i$, denoted $v_{\sigma_i}$, as the $d$-dimensional volume of $\sigma_i$ when $S$ is real-valued, and a count of points in $\sigma_i$ when $S$ is finite. We slightly abuse notation when the domain is finite or has mixed integer- and real-valued variables; technical details can be found in [118] and [108].

We let the adaptive subregion probability $\tilde{p}_i(y)$ be the probability that subregion $\sigma_i$ is selected given an objective function value $y$, $y_* < y \leq y^*$. BASSO keeps track of the best objective function value observed in each subregion $\sigma_i$, denoted $y_i^*$, and the overall best objective function value, denoted $\tilde{y}_k^*$ on iteration $k$, which we call the incumbent value. BASSO uses this incumbent value to parameterize the adaptive subregion probabilities, i.e., $\tilde{p}_i(\tilde{y}_k^*)$. In the analysis, we do not limit the parameter $y$ in $\tilde{p}_i(y)$ to be the incumbent value, because we want to analyze the impact of the parameter on the adaptive subregion probability. It becomes important in establishing a condition in Assumption 1 on the adaptive subregion probabilities to represent the promising subregions.

The steps of the framework follow.

**Branching Adaptive Surrogate Search Optimization (BASSO).**

**Step 0: Initialize.** Set $\sigma_1 = S$, $\Sigma_0^C = \{\sigma_1\}$, $m_0 = 1$, and iteration counter $k = 0$. Sample a point

$X$ from the uniform distribution over $S$. Evaluate $f(X)$ and set $y_1^* = f(X)$. Also, set $\tilde{y}_0^* = f(X)$ to keep track of the incumbent function value observed. Set $\tilde{p}_1(\tilde{y}_0^*) = 1$.

**Step 1: Sample a new point.** Choose a subregion $\sigma_i$ from $\Sigma_k^C$ using the adaptive subregion probabilities $\tilde{p}_i(\tilde{y}_k^*)$ for $i = 1, \ldots, m_k$. Generate a point $X$ on the selected subregion either uniformly or using surrogate modeling, and evaluate $f(X)$. Update the best objective function value $y_i^*$ in the selected subregion $\sigma_i$.

If the new objective function value $f(X)$ is less than the incumbent value $\tilde{y}_k^*$, update the incumbent function value $\tilde{y}_{k+1}^* = f(X)$ and proceed to Step 2. Otherwise, repeat Step 1 for a maximum number of function evaluations before proceeding to Step 2.

**Step 2: Branch subregions.** Branch a collection of subregions in $\Sigma_k^C$ according to a branching strategy. Update the new number of subregions $m_{k+1}$, and update the collection of subregions, $\Sigma_{k+1}^C = \{\sigma_1, \ldots, \sigma_{m_{k+1}}\}$.

**Step 3: Update adaptive subregion probabilities.** Update $\tilde{p}_i(\tilde{y}_{k+1}^*)$ for $i = 1, \ldots, m_{k+1}$.

**Step 4: Evaluate stopping criterion.** If a stopping criterion is met, stop. Otherwise increment the iteration counter $k \leftarrow k+1$ and go to Step 1.

Figure 4.2 illustrates the impact of the adaptive sampling distribution on sampling points with good objective function values. In Figure 4.2, the domain $S$ is partitioned into three subregions, and 100 points are generated independently. For each point, a subregion is selected using the adaptive subregion probabilities, and then a point is generated uniformly within the selected subregion. On left side of Figure 4.2,the adaptive subregion probabilities are $\left[1/3, 1/3, 1/3\right]$, whereas the adaptive subregion probabilities on the right of Figure 4.2 are $[0.1, 0.8, 0.1]$ for the three subregions.

The empirical density range distribution is the number of objective function values that fall in an interval, as illustrated by the vertical histogram in Figure 4.2. This provides a way to visualize the impact of the adaptive sampling distribution. The red dashed curve illustrates the empirical cumulative range distribution, which is the probability the objective function value is below $y$ when a point is sampled according to a sampling distribution. Notice that the range distribution in the right of Figure 4.2 puts more weight on the lower objective function values. The impact of

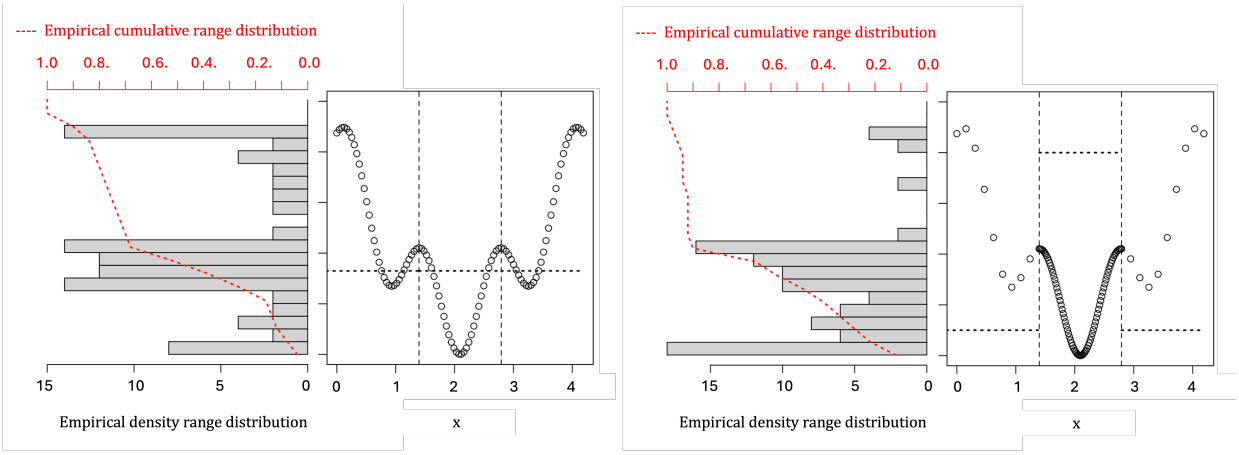



**Figure 4.2:** Empirical density range distribution (gray histogram) and empirical cumulative range distribution (dashed red line) for 100 points sampled independently, where the domain is partitioned into three subregions (i.e., intervals). In (a), the adaptive subregion probability is $\tilde{p} = [0.33, 0.33, 0.33]$, representing a uniform distribution. In (b), the adaptive subregion probability $\tilde{p} = [0.1, 0.8, 0.1]$. Once a subregion is selected, a point is sampled uniformly on that interval.

adaptive sampling on the range distribution is important for the finite-time analysis of BASSO.

### *4.2.2 BASSO Variations*

To implement BASSO, it is important to specify: (i) the branching scheme, (ii) the adaptive subregion probabilities $\tilde{p}_i(\tilde{y}_k^*)$ for subregions $\sigma_i$ in $\Sigma_k^C$, and (iii) the method for generating a point within the selected subregion (surrogate modeling or the uniform distribution). The implementation variations demonstrate the interplay between branching strategy, the adaptive subregion probabilities, and the surrogate model used to generate a point on the selected subregion.

**(i) Branching strategy**

Create a collection of subregions to branch. Add $\lceil 0.1 m_k \rceil$ subregions (where $m_k$ is the number of subregions in $\Sigma_k^C$ on the $k$th iteration) ranked by their best observed function values to the collection, and add additional $\lceil 0.1 m_k \rceil$ of the **remaining** subregions ranked by the largest volume. We branch two types of subregions, those with good observed objective function values and those with large volumes, as an attempt to balance exploitation with exploration.

We subdivide the subregions in the collection coordinate-wise by dividing the largest dimension in half. If any newly created subregion has fewer than two observations in it, sample one or two points uniformly in the subregion so that each subregion has at least two observed points.

**(ii) Adaptive subregion probabilities,** $\tilde{p}$ Four variations of adaptive subregion probabilities are implemented. The first variation (a) is greedy in the sense that it exploits the subregions with the best observed function value. The second variation (b) favors exploration as it uses an estimate of the variance in the probability of selecting subregions. The third variation (c) attempts to balance exploration and exploitation by constructing a one-dimensional Gaussian process to estimate the cumulative range distribution (illustrated as dashed red line in Figure showing range distribution on each subregion.

The fourth variation (d) relaxes the greedy variation by estimating confidence intervals around the incumbent value and each subregion's best observed function value.

**a: Observed best function value.** Choose subregion $\sigma_i$ from the subregions in $\Sigma_k^C$ on the $k$th iteration with probability $\tilde{p}_i(\tilde{y}_k^*)$, where

$$\tilde{p}_i(\tilde{y}_k^*) = \left( \left(y_i^* - \tilde{y}_k^* + 1\right) \sum_{j=1,\ldots,m_k} \left( \frac{1}{y_j^* - \tilde{y}_k^* + 1} \right) \right)^{-1}. \tag{4.1}$$

**b: Sample variance.** Choose subregion $\sigma_i$ from the subregions in $\Sigma_k^C$ on the $k$th iteration with probability $\tilde{p}_i$, where

$$\tilde{p}_i = \begin{cases} v_{\sigma_i} / v_{\Sigma_k^C} & \text{if } k = 1 \\ \left(s_k(\sigma_i)\right)^2 / \sum_{j=1,\ldots,m_k} \left(s_k(\sigma_i)\right)^2 & \text{if } k > 1, \end{cases} \tag{4.2}$$

and $(s_k(\sigma_i))^2$ is the sample variance in subregion $\sigma_i$, that is,

$$(s_k(\sigma_i))^2 = \left(\frac{1}{N_k^i - 1}\right) \sum_{j=1}^{N_k^i} (y_j - \overline{f_i})^2, \tag{4.3}$$

and $y_j$ is the function value of the $j$th previously sampled point in subregion $\sigma_i$, and $\overline{f_i}$ is the sample mean of the $N_k^i$ observed objective function values in subregion $\sigma_i$, i.e., $\overline{f_i} = (1/N_k^i) \sum_{j=1,\ldots,N_k^i} y_j$.

**c: Gaussian process on the range distribution.** For each of the subregions $\sigma_i$ in $\Sigma_k^C$, $i = 1,\ldots,m_k$, we construct a Gaussian process on the empirical cumulative range distribution using the objective function values of the $N_k^i$ points that have been sampled in $\sigma_i$ on the $k$th iteration. Specifically, we use the $N_k^i$ observed objective function values in subregion $\sigma_i$, $y_j$, $j = 1,\ldots,N_k^i$ and the associated empirical cumulative function values $F(y_j)$, using $\mathscr{D} = \{y_j, F(y_j)\}_{j=1}^{N_k^i}$ as the input space. The empirical cumulative function $F(y_j)$ is the count of observed function values less than or equal to $y_j$, divided by $N_k^i$.

The Gaussian process on the range distribution provides a predictor function on subregion $\sigma_i$ as an approximation of the cumulative range distribution on $\sigma_i$, where the mean of the Gaussian process is denoted $\hat{F}_{i,k}(y)$. Note that the Gaussian process predictor is a one-dimensional function (as illustrated in Figure 4.2), and is easily computed.

The adaptive subregion probability of selecting subregion $i$ is,

$$\tilde{p}_i(\tilde{y}_k^* + \delta) = \frac{\hat{F}_{i,k}(\tilde{y}_k^* + \delta)}{\Sigma_{i=1}^{m_k} \hat{F}_{i,k}(\tilde{y}_k^* + \delta)}. \tag{4.4}$$

where $\tilde{y}_k^* + \delta$ is the value of the fifth lowest observed objective function value.

**d: Confidence bounds.** Choose subregion $\sigma_i$ from the subregions in $\Sigma_k^C$ on the $k$th iteration with probability $\tilde{p}_i(\tilde{y}_k^*)$, where

$$\tilde{p}_i(\tilde{y}_k^*) = \begin{cases} 0 & \text{if } LB_i \geq \widetilde{UB}_k \\ \\ \widetilde{UB}_k - LB_i / \sum_{i=1,\ldots,m_k} \widetilde{UB}_k - LB_i & \text{otherwise} \end{cases} \tag{4.5}$$

and $LB_i = y_i^* - s_k(\sigma_i)$, $\widetilde{UB}_k = \tilde{y}_k^* + s_k(\tilde{\sigma})$, $\tilde{\sigma}$ is the subregion containing a point with the incumbent value $\tilde{y}_k^*$, and $s_k(\cdot)$ is the square root of the sample variance as in (4.3).

**(iii) Generate a point on the selected subregion**

Three variations of generating a point within a subregion are implemented. The first variation (A) has no surrogate model and simply samples uniformly on the the selected subregion. The second variation (B) uses a Gaussian process as a surrogate model on the selected subregion, and the third variation (C) uses a regularized quadratic regression as the surrogate model.

**A: Uniform.** Generate a point $X$ uniformly distributed on the selected subregion.

**B: Maximize expected improvement for a Gaussian process on the domain.** For each of the subregions $\sigma_i$ in $\Sigma_k^C$, $i = 1, \ldots, m_k$, we construct a Gaussian process on the domain using the $N_k^i$ points that have been sampled in $\sigma_i$ on the $k$th iteration and their associated function values, using $\mathscr{D} = \{x_j, f(x_j)\}_{j=1}^{N_k^i}$ as the input space. $\hat{s}_{i,k}^2(x)$, for $x \in \sigma_i$. The estimated mean function of the Gaussian process that approximates the objective function value using the sample points in subregion $\sigma_i$ on iteration $k$ is denoted $\hat{g}_{i,k}(x)$ and model variance is denoted $\hat{s}_{i,k}^2(x)$, for $x \in \sigma_i$.

The expected improvement function for subregion $\sigma_i$ uses the best objective function value observed in that subregion $y_i^*$ to capture a better point, and a standard Normal cumulative distribution $\Phi$ to capture uncertainty,

$$EI_i(x) = \left( \frac{y_i^* - \hat{g}_{i,k}(x)}{\hat{s}_{i,k}(x)} \right) + \hat{s}_{i,k}(x)\, \Phi \left( \frac{y_i^* - \hat{g}_{i,k}(x)}{\hat{s}_{i,k}(x)} \right). \tag{4.6}$$

Generate a point $X$ in subregion $\sigma_i$ that maximizes the expected improvement function over

$x \in \sigma_i$. The maximization of $EI_i(x)$ is implemented numerically by performing a grid search on the subregion. This implementation of Variation B is computationally intensive for high dimensions. In the numerical experiments, we executed Variation B on problems in dimensions 20 and 50, and not on the higher dimensions.

**C: Minimize a regularized quadratic regression on the domain.** For each of the subregions $\sigma_i$ in $\Sigma_k^C$, $i = 1, \ldots, m_k$, we construct a regularized quadratic regression on the domain using the $N_k^i$ points that have been sampled in $\sigma_i$ on the $k$th iteration and their associated function values, using $\mathscr{D} = \{x_j, f(x_j)\}_{j=1}^{N_k^i}$ as the input space.

The regularized quadratic regression function for $x \in \sigma_i$ is

$$\hat{q}_k^i(x) = \hat{\beta}_0 + \sum_{\ell=1}^{d} \hat{\beta}_\ell x_\ell + \sum_{\ell=1}^{d} \sum_{m=\ell}^{d} \hat{\beta}_{\ell m} x_\ell x_m \tag{4.7}$$

where the $\ell$th component (decision variable) of a point $x$ is denoted $x_\ell$. The coefficients of the quadratic regression ($\hat{\beta}_0$ is the intercept term, $\hat{\beta}_\ell$ are the coefficients of the linear terms, and $\hat{\beta}_{\ell m}$ are the coefficients of the quadratic terms, $\ell, m = 1, \ldots, d$) are determined by minimizing the loss function given in (4.8).

The loss function uses the $N_k^i$ sampled points $x_j \in \sigma_i$ and their objective function values $y_j$, for $j = 1, \ldots, N_k^i$, as

$$\mathscr{L}(\hat{\beta}) = \frac{1}{2N_k^i} \sum_{j=1}^{N_k^i} (\hat{q}_k^i(x_j) - y_j)^2 + \lambda \left( \sum_{l=1}^{d} |\hat{\beta}_\ell| + \sum_{\ell=1}^{d} \sum_{m=\ell}^{d} |\hat{\beta}_{\ell m}| \right), \tag{4.8}$$

where $\hat{\beta}$ is the vector of coefficients, including $\hat{\beta}_0$, $\hat{\beta}_\ell$, and $\hat{\beta}_{\ell m}$. The first term in the loss function is the squared difference between the quadratic model and the observed value. The second term in the loss function is a penalty term weighted by a hyperparameter $\lambda$, that is a regularization parameter controlling the amount of regularization applied. In the numerical experiments, we update $\lambda$ based on observations. For subregions with more than 50 points, we determine the regularizer $\lambda$ by performing a 5-fold cross-validation on that subregion and use the value of $\lambda$ that minimizes the

cross-validation prediction error. For subregions with 50 or fewer points, we set $\lambda = 1$.

Generate a point $X$ in subregion $\sigma_i$ that minimizes the regularized quadratic regression $\hat{q}_k^i(x)$ over $x \in \sigma_i$. In the numerical experiments, we use a trust region algorithm [22] to approximately minimize $\hat{q}_k^i(x)$ over $\sigma_i$.

### *4.3 Analysis*

The BASSO analysis begins by assuming that sample points are generated according to a uniform distribution on selected subregions. We show in Theorem 6 that the objective function values of BASSO with uniform sampling within a subregion stochastically dominate those of a special case of HAS, HAS$'$, under Assumption 1 on the adaptive subregion probabilities $\tilde{p}_i$ We then relax the assumption of uniform sampling within the subregion and analyze the case of surrogate-driven sampling. Theorem 7 shows that the objective function values of BASSO with surrogate sampling stochastically dominate those of BASSO with uniform sampling on a subregion, under Assumption 2 on the surrogate model.

Using prior analyses of PAS and HAS [19, 108, 116, 118], we then bound the expected number of BASSO function evaluations needed to achieve an $\varepsilon$-optimal solution. Corollary 8 provides the desired linearity result for functions satisfying the Lipschitz condition, and Corollary 9 provides the desired linearity result for functions on a finite domain, specifically a $d$-dimensional integer lattice.

HAS$'$ uses the uniform distribution as its underlying sampling distribution, hence, given a value $z$, the *bettering probability*, i.e., the probability to sample a location with value below $z$, can be specified as:

$$b(z) = \sum_{i=1,\ldots,m_k} \frac{\tilde{p}_i(z)}{v_{\sigma_i}} v_{\sigma_i}(z) \tag{4.9}$$

where $y_* < z \leq y^*$, given subregions $\sigma_i \in \Sigma_k^C$, $m_k$, and probabilities $\tilde{p}_i(z)$ for $i = 1, \ldots, m_k$ with $y_* < z \leq y^*$. As earlier, $v_{\sigma_i}$ is the size of subregion $\sigma_i$ and $v_{\sigma_i}(z)$ is the size of the set $\{x \in \sigma_i : f(x) \leq z\}$ for $y_* < z \leq y^*$ (i.e., size is $d$-dimensional volume when $S$ is real-valued, or a count when $S$ is finite). The bettering probability $b(z)$ depends on the adaptive subregion probability parameter

$\tilde{p}_i(z)$ for $i = 1, \ldots, m_k$ with $y_* < z \leq y^*$, which drives the the sampling distribution.

Let the random variable $Y_k^{BASSOunif}$ be the incumbent function value $\tilde{y}_k^*$ on the $k$th iteration when BASSO generates a point uniformly distributed within a selected subregion. Let $Y_k^{HAS'}$ be the incumbent function value from HAS$'$ on the $k$th iteration.

The parameter $z$ in $\tilde{p}_i(z)$ is the incumbent function value in BASSO. However, in the analysis, we do not limit the parameter $y$ in $\tilde{p}_i(y)$ to be the incumbent value because we need to characterize the conditional probability of improvement as the parameter $y$ in $\tilde{p}_i(y)$ changes. This is important in Assumption 1 where we need to account for the amount of improvement as the incumbent value improves.

We note that the probability of generating a point with objective function value no greater than $y$ for BASSO, given the incumbent value $z$, adaptive subregion probabilities $\tilde{p}_i(z)$, when the uniform distribution is used on the selected subregion, can be expressed as

$$P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z) = \sum_{i=1,\ldots,m_k} \frac{\tilde{p}_i(z)}{v_{\sigma_i}} v_{\sigma_i}(y)$$

for $y_* < y \leq z \leq y^*$. An understanding of this expression is that each subregion has a probability $\tilde{p}_i(z)$ of being selected, and once selected, a point is sampled uniformly on the subregion with a probability of achieving a function value less than or equal to $y$ (i.e., $v_{\sigma_i}(y)/v_{\sigma_i}$). Another interpretation of this expression is viewing the sampling density as $\tilde{p}_i(z)/v_{\sigma_i}$.

The challenge in partitioning and using an adaptive subregion probability is focusing on appropriate subregions. We make the following assumptions regarding the adaptive subregion parameters $\tilde{p}_i(z)$ for $i = 1, \ldots, m_k$ with $y_* < z \leq y^*$.

**Assumption 1: Adaptive subregion probabilities $\tilde{p}_i(z)$**

1. $0 \leq \tilde{p}_i(z) \leq 1$ for $i = 1, \ldots, m_k$ with $y_* < z \leq y^*$

2. $\sum_{i=1,\ldots,m_k} \tilde{p}_i(z) = 1$

3. Given two values $t_z$ and $t_{z'}$ such that $y_* < t_z \leq t_{z'} \leq y^*$, the following holds,

$$\frac{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(t_z)/v_{\sigma_i}\right) v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(t_z)/v_{\sigma_i}\right) v_{\sigma_i}(z)} \geq \frac{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(t_{z'})/v_{\sigma_i}\right) v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(t_{z'})/v_{\sigma_i}\right) v_{\sigma_i}(z)} \tag{4.10}$$

for $y_* < y \leq z \leq y^*$.

The first two parts of Assumption 1 ensure that the $\tilde{p}_i(z)$ form a proper probability. The third part is that the ratio (in the left-hand-side of (4.10)) is non-increasing in $t_z$. This ensures that $\tilde{p}_i(z)$ focuses the sampling distribution on the regions with good objective function values. The ratio with $t_z = z$ can be interpreted as the conditional probability,

$$\begin{aligned} P(Y_{k+1}^{BASSOunif} \leq y | Y_{k+1}^{BASSOunif} \leq z, Y_k^{BASSOunif} = z) \\ = \frac{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(z)/v_{\sigma_i}\right) v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(z)/v_{\sigma_i}\right) v_{\sigma_i}(z)} \end{aligned}$$

for $y_* < y \leq z \leq y^*$, when uniform sampling is used on the selected subregion. The conditional probability captures, not only the probability of improving, but the amount of improvement to a function value $y$ where $y \leq z$. Thus, the sampling distribution parameterized by $\tilde{p}_i(z)$ can improve the conditional probability by increasing the chance of selecting "good" subregions. This is key to a good branching algorithm, although, as illustrated in the numerical experiments, it is challenging to devise a practical implementation of $\tilde{p}_i(z)$ that satisfies the third part of Assumption 1.

We now show that the objective function values of BASSO with uniform sampling within subregions stochastically dominate those of HAS$'$.

**Theorem 6** Given the optimization problem (P) over a continuous/finite domain $S$ with Assumption 1 holding, the function values of BASSO with uniform sampling within subregion stochastically dominate those of HAS$'$, i.e.,

$$P(Y_k^{BASSOunif} \leq y) \geq P(Y_k^{HAS'} \leq y)$$

for $k = 0, 1, \ldots$, and $y_* < y \leq y^*$.

**Proof:** To prove that $Y_k^{BASSOunif}$ stochastically dominates $Y_k^{HAS'}$, for $k = 0, 1, 2, \ldots$, we show that the two sequences satisfy three conditions as stated in Lemma 30 in [94], and included in Appendix **??** for completeness. At iteration $k$, we are given the subregions $\sigma_i$ in $\Sigma_k^C$ for $i = 1, \ldots, m_k$, the incumbent value $Y_k = \tilde{y}_k^*$, and the adaptive subregion probabilities $\tilde{p}_i(\tilde{y}_k^*)$. Let $z = \tilde{y}_k^*$ for ease of notation.

To show Condition 1, we need to prove that

$$P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z) \geq P(Y_{k+1}^{HAS'} \leq y | Y_k^{HAS'} = z)$$

for any fixed $k$, $k = 0, 1, 2, \ldots$, and $y$, $z$ values, for $y_* \leq y, z \leq y^*$. We are interested in the case when $y < z$, because, when $y \geq z$, the conditional probabilities are both equal to 1, and so satisfy the inequality.

For $y < z$, using conditional probabilities, we have,

$$\begin{aligned}
&P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z) \\
&= P(Y_{k+1}^{BASSOunif} \leq y | Y_{k+1}^{BASSOunif} \leq z, Y_k^{BASSOunif} = z) \\
&\quad \cdot P(Y_{k+1}^{BASSOunif} \leq z | Y_k^{BASSOunif} = z) \\
&= \frac{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z)/v_{\sigma_i} v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z)/v_{\sigma_i} v_{\sigma_i}(z)} \cdot \sum_{i=1,\ldots,m_k} \frac{\tilde{p}_i(z)}{v_{\sigma_i}} v_{\sigma_i}(z)
\end{aligned}$$

and by Equation (4.10) in Assumption 1 with $z \leq y^*$,

$$\geq \frac{\sum_{i=1,\ldots,m_k} \tilde{p}_i(y^*)/v_{\sigma_i} v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \tilde{p}_i(y^*)/v_{\sigma_i} v_{\sigma_i}(z)} \cdot \sum_{i=1,\ldots,m_k} \frac{\tilde{p}_i(z)}{v_{\sigma_i}} v_{\sigma_i}(z)$$

and because $\tilde{p}_i(y^*)$ represents a uniform distribution on $S$, i.e., $\tilde{p}_i(y^*) = v_{\sigma_i}/v_S$,

$$= \frac{v_S(y)}{v_S(z)} \cdot \sum_{i=1,\ldots,m_k} \frac{\tilde{p}_i(z)}{v_{\sigma_i}} v_{\sigma_i}(z)$$
$$= \frac{v_S(y)}{v_S(z)} \cdot b(z) = P(Y_{k+1}^{HAS'} \leq y | Y_k^{HAS'} = z).$$

To show Condition 2, we need to prove that $P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z)$ is non-increasing in $z$, that is, we need to prove

$$P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z) \geq P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z') \tag{4.11}$$

for any fixed $k$, $k = 0, 1, 2, \ldots$, and any $y$ and $z$ values, $y_* < y \leq z \leq z' \leq y^*$.

If $y \geq z$, $P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z) = 1$, hence the inequality in (4.11) is satisfied. For $y < z$, we have,

$$P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z) = \sum_{i=1,\ldots,m_k} \frac{\tilde{p}_i(z)}{v_{\sigma_i}} v_{\sigma_i}(y),$$

and noting that $\sum_{i=1,\ldots,m_k} \tilde{p}_i(z)/v_{\sigma_i} v_{\sigma_i}(y^*) = 1$, and by Equation (4.10) in Assumption 1, since $z \leq z'$, we have,

$$P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z) \geq \frac{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z')/v_{\sigma_i} v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z')/v_{\sigma_i} v_{\sigma_i}(y^*)}$$

and again noting that $\sum_{i=1,\ldots,m_k} \tilde{p}_i(z')/v_{\sigma_i} v_{\sigma_i}(y^*) = 1$, we satisfy the condition in (4.11).

For $k = 0$, both *BASSOunif* and $HAS'$ sample according to a uniform distribution over $S$, hence the third condition of the Lemma 30 in [94] is satisfied, and we have $P(Y_k^{BASSOunif} \leq y) \geq P(Y_k^{HAS'} \leq y)$ . □

Let the random variable $Y_k^{BASSOsurr}$ be the incumbent function value $\tilde{y}_k^*$ from BASSO on the $k$th iteration where we generate a point within a selected subregion with surrogate modeling. We make

some assumptions regarding efficacy of the surrogate.

**Assumption 2: Sampling within a subregion**

1. Assume that generating a point $X^{BASSOsurr}$ on subregion $\sigma_i$ with surrogate modeling stochastically dominates uniform sampling, i.e.,

$$\begin{aligned} &P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z, X^{BASSOsurr} \in \sigma_i) \\ &\geq P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z, X^{BASSOunif} \in \sigma_i) \end{aligned} \tag{4.12}$$

2. Assume that a lower incumbent function value improves the performance of the surrogate, that is,

$$\begin{aligned} &P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z, X^{BASSOsurr} \in \sigma_i) \\ &\geq P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z', X^{BASSOsurr} \in \sigma_i) \end{aligned} \tag{4.13}$$

   for $y_* < y \leq z \leq z' \leq y^*$.

Assumption 2.1 is that surrogate modeling does no worse than uniform sampling on a subregion. Notice that the incumbent function value $z$ impacts the adaptive selection probabilities $\tilde{p}_i(z)$ but does not impact uniform sampling on a subregion. For *BASSOunif*, we have

$$P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z, X^{BASSOunif} \in \sigma_i) = \frac{v_{\sigma_i}(y)}{v_{\sigma_i}}$$

for any $z$. It is assumed that surrogate modeling improves upon uniform sampling within a subregion, so (4.12) implies that there exists a constant $c_i(y,z) \geq 0$ such that

$$P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z, X^{BASSOsurr} \in \sigma_i) = \frac{v_{\sigma_i}(y)}{v_{\sigma_i}} + c_i(y,z) \tag{4.14}$$

Assumption 2.2 is that a lower incumbent function value improves (or does no worse) surrogate performance than a higher incumbent function value. Combining (4.12) with (4.13), we have

$$P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z', X^{BASSOsurr} \in \sigma_i) = \frac{v_{\sigma_i}(y)}{v_{\sigma_i}} + d_i(y, z') \tag{4.15}$$

where $c_i(y,z) \geq d_i(y,z')$ for $y_* < y \leq z \leq z' \leq y^*$.

A challenge to implementation is for surrogate modeling to have sufficient coverage in a subregion to improve over uniform sampling on a subregion. This may appear easy to satisfy, however, if the sampling guided by the surrogate is too exploitative, the amount of improvement to a value $y, y_* < y \leq z$ may be worse than uniform sampling. This gives a condition on the balance between exploration and exploitation. The second part of the assumption is that, having improved the incumbent value, the surrogate model will increase its probability for improvement to a value $y$. Thus, the accuracy of the surrogate model and the sampling distribution associated with it must satisfy the assumption that includes both the probability of improvement and the *amount* of improvement.

We now show that the objective function values of *BASSOsurr* stochastically dominate those of *BASSOunif*.

**Theorem 7** Given the optimization problem (P) over a continuous/finite domain $S$ with Assumptions 1 and 2 holding, the function values of BASSO with surrogate sampling model within subregions stochastically dominate those of BASSO with uniform sampling within subregions, i.e.,

$$P(Y_k^{BASSOsurr} \leq y) \geq P(Y_k^{BASSOunif} \leq y)$$

for $k = 0, 1, \ldots$, and $y_* < y \leq y^*$.

**Proof:** To prove that $Y_k^{BASSOsurr}$ stochastically dominates $Y_k^{BASSOunif}$ for $k = 0, 1, 2, \ldots$, we again show that the two sequences satisfy the three conditions as stated in Lemma 30 in [94].

To show Condition 1, we need to prove that

$$P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z) \geq P(Y_{k+1}^{BASSOunif} \leq y | Y_k^{BASSOunif} = z)$$

for any fixed $k$, $k = 0,1,2,\ldots$, and any $y$ and $z$ values, for $y_* \le y, z \le y^*$. We have,

$$P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z)$$
$$= \sum_{i=1,\ldots,m_k} \tilde{p}_i(z)\ P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z, X^{BASSOsurr} \in \sigma_i)$$

and by Equation (4.12) in Assumption 2,

$$\ge \sum_{i=1,\ldots,m_k} \tilde{p}_i(z)\ P(Y_{k+1}^{BASSOunif} \le y | Y_k^{BASSOunif} = z, X^{BASSOunif} \in \sigma_i)$$
$$= P(Y_{k+1}^{BASSOunif} \le y | Y_k^{BASSOunif} = z).$$

To show Condition 2, we need to prove that $P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z)$ is non-increasing in $z$, that is, we need to prove

$$P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z) \ge P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z') \tag{4.16}$$

for any fixed $k$, $k = 0,1,2,\ldots$, and any $y$ and $z$ values, for $y_* < y \le z \le z' \le y^*$. If $y \ge z$, $P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z) = 1$, hence the inequality in (4.16) is satisfied. For $y < z$, we have,

$$P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z)$$
$$= \sum_{i=1,\ldots,m_k} \tilde{p}_i(z)\ P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z, X^{BASSOsurr} \in \sigma_i)$$

and by Assumption 2 and (4.14), $= \sum_{i=1,\ldots,m_k} \frac{\tilde{p}_i(z)}{v_{\sigma_i}} v_{\sigma_i}(y) + \sum_{i=1,\ldots,m_k} \tilde{p}_i(z)\ c_i(y,z).$

We note that $\sum_{i=1,\ldots,m_k} \tilde{p}_i(z)/v_{\sigma_i} v_{\sigma_i}(y^*) = 1$. Therefore, we have,

$$P(Y_{k+1}^{BASSOsurr} \le y | Y_k^{BASSOsurr} = z)$$
$$= \frac{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z)/v_{\sigma_i} v_{\sigma_i}(y) + \sum_{i=1,\ldots,m_k} \tilde{p}_i(z)\ c_i(y,z)}{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z)/v_{\sigma_i} v_{\sigma_i}(y^*)}$$

and by (4.10) in Assumption 1, and by (4.15) in Assumption 2, since $z \leq z'$, we have

$$P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z) \geq \frac{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z')/v_{\sigma_i} v_{\sigma_i}(y) + \sum_{i=1,\ldots,m_k} \tilde{p}_i(z')\, d_i(y,z')}{\sum_{i=1,\ldots,m_k} \tilde{p}_i(z')/v_{\sigma_i} v_{\sigma_i}(y^*)}$$

and again noting that $\sum_{i=1,\ldots,m_k} \tilde{p}_i(z')/v_{\sigma_i} v_{\sigma_i}(y^*) = 1$, we have

$$P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurr} = z) = P(Y_{k+1}^{BASSOsurr} \leq y | Y_k^{BASSOsurg} = z').$$

For $k = 0$, both *BASSOsurr* and *BASSOunif* sample according to a uniform distribution over $S$, hence the third condition of the Lemma 30 in [94] is satisfied, and we have $P(Y_k^{BASSOsurr} \leq y) \geq P(Y_k^{BASSOunif} \leq y)$. □

The following corollaries provide an upper bound on the expected number of BASSO function evaluations to get within $\varepsilon$ of the global optimum that is linear in dimension. The corollaries assume a constant lower bound $B$ on the bettering probability in (4.9). Corollary 8 states conditions when the domain $S$ is continuous, and Corollary 9 is stated for a $d$-dimensional integer lattice.

**Corollary 8** Assume Assumptions 1 and 2 hold, and $B$ is a positive lower bound on $b(z)$ in (4.9) for $y_* + \varepsilon < z \leq y^*$. For a global optimization problem $(P)$, when $f(x)$ satisfies the Lipschitz condition with Lipschitz constant at most $L$ and $S$ is a convex feasible region in $d$ dimensions with a maximum diameter of at most $D$, then the expected number of function evaluations for BASSO, using uniformly distributed points or surrogate modeling, to reach a value $y_* + \varepsilon$, $\varepsilon > 0$ , is bounded by,

$$\mathbb{E}[N^{BASSO}(y_* + \varepsilon)] \leq 1 + d \cdot \left(\frac{1}{B}\right) ln\left(\frac{L \cdot D}{\varepsilon}\right). \tag{4.17}$$

**Proof of Corollary 1** By stochastic dominance in Theorem 6, the expected number of iterations

to achieve a value within $y_* + \varepsilon$ for BASSO is less than or equal to the number for HAS$'$, i.e.,

$$E[N^{BASSO}(y_* + \varepsilon)] \leq E[N^{HAS'}(y_* + \varepsilon)]. \tag{4.18}$$

Using (4.10) in Assumption 1, with $t_z = y$ and $t_{z'} = y^*$, we have

$$\frac{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(y)/v_{\sigma_i}\right) v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(y)/v_{\sigma_i}\right) v_{\sigma_i}(y^*)} \geq \frac{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(y^*)/v_{\sigma_i}\right) v_{\sigma_i}(y)}{\sum_{i=1,\ldots,m_k} \left(\tilde{p}_i(y^*)/v_{\sigma_i}\right) v_{\sigma_i}(y^*)}.$$

Note that the numerator on the left-hand side is $b(y)$ by definition in (4.9), and the denominator is equal to 1, because $v_{\sigma_i}(y^*) = v_{\sigma_i}$.

The right-hand side denominator is also equal to 1. For the right-hand side numerator, $\tilde{p}_i(\tilde{y}_k^*)$ represents a uniform sampling probability, where $\tilde{p}_i(\tilde{y}_k^*) = v_{\sigma_i}/v_S$. We then have

$$b(y) \geq \sum_{i=1,\ldots,m_k} \left(\frac{v_{\sigma_i}}{v_S} \frac{1}{v_{\sigma_i}}\right) v_{\sigma_i}(y) = \frac{v_S(y)}{v_S}$$

for $y_* + \varepsilon \leq y \leq y^*$. Since HAS$'$ uses uniform sampling with a bettering probability bounded below by $B$, from [19] we have

$$E[N^{HAS'}(y_* + \varepsilon)] \leq 1 + \frac{1}{B}\, ln\left(\frac{v_S}{v_S(y+\varepsilon)}\right). \tag{4.19}$$

Combined with the bounds on $v_S/v_S(y_* + \varepsilon)$ in [19], we have the desired bound in (4.17) that is linear in dimension. □

**Corollary 9** Assume Assumptions 1 and 2 hold, and $B$ is a positive lower bound on $b(z)$ in (4.9) for $y_* + \varepsilon < z \leq y^*$. For a global optimization problem $(P)$, when $S$ in $(P)$ is a $d$-dimensional lattice $\{1,\ldots,k\}^d$, the expected number of function evaluations for BASSO, using uniformly distributed points or surrogate modeling, to reach the value $y_*$ is bounded by,

$$\mathbb{E}[N^{BASSO}(y_*)] < 2 + d \cdot ln(k). \tag{4.20}$$

The proof for the finite case of Corollary 2 follows similarly, using [118].

### *4.4 Numerical Experiments*

To assess the impact of dimension on BASSO performance, we implemented 12 variations of BASSO (Aa, Ab, Ac, Ad, Ba, Bb, Bc, Bd, Ca, Cb, Cc, and Cd). These variations combine four adaptive subregion probability methods (a: the best observed function value, b: sample variance, c: Gaussian process on the range distribution, and d: confidence bound on the incumbent value), with three surrogate models (A: uniform, B: Gaussian process, and C: quadratic regression. All twelve BASSO variations use the same branching strategy that is described in Section 4.2.2. The value for the maximum number of function evaluations without improving the incumbent value, in Step 1 of BASSO, is set to 50.

We also implemented two publicly available global optimization algorithms, specifically, the Stable Noisy Optimization by Branch and Fit (SNOBFIT) algorithm [51] and the Stochastic Optimization with Adaptive Restart (SOAR) algorithm [69].

SNOBFIT combines global and local search by branching and fitting a surrogate model on subregions. SNOBFIT builds local models of the function similar to trust region approaches. No guarantees can be given that a global minimum is located.

SOAR utilizes Gaussian processes to adaptively restart a local search and select restart locations using current observations. SOAR is shown to asymptotically converge to a global optimum. Numerical experiments with SOAR use the trust region method as the local search.

To gain insight into whether an implementation of BASSO can satisfy the assumptions needed for the theoretical result, we tested the 12 BASSO implementations on both assumptions. In Section 4.4.1, we evaluate whether the adaptive subregion probabilities $\tilde{p}_i$ (a, b, c, and d) satisfy Assumption 1. The numerical experiments illustrate the impact of exploitation versus exploration in selecting subregions. In Section 4.4.2, we numerically illustrate Assumption 2 by investigating the performance of surrogate modeling (B and C) compared to uniform sampling (A). In Section 4.4.3, we numerically evaluate the impact of dimension on test problems.To relate the numerical experiments to the theory, we plot the mean number of function evaluations to first reach a threshold

within $\varepsilon$ of the global minimum, averaged over 10 replications of the algorithm. We include dimensions 10, 20, 30, 40, and 50 on five test problems. We present experiments with 8 BASSO variations (Ba, Bb, Bc, Bd, Ca, Cb, Cc, Cd), SNOBFIT, and SOAR. We excluded the variations with uniform sampling (A) because they performed poorly relative to the (B) and (C) variations. Although none of these algorithms scale as well as the theoretical ideal, the numerical experiments show the effect of dimension on computational performance. In Section 4.4.4, we compute data profiles for the 8 BASSO variations, SNOBFIT, and SOAR. This provides some insight into how the performance behaves as the number of function evaluations increases. In Section 4.4.6, we apply three BASSO variations (Ba, Bc, and Cc) to a real-world application, the falsification of cyberphysical systems [70]. We use regularized quadratic regression (C) coupled with variation (c) for how to choose subregions because it was the best variation shown in Section 4.4.4. We also use the Gaussian process surrogate (B) to compare to the regularized quadratic regression on a relatively low-dimensional problem. The Gaussian process surrogate (B) is coupled with variations (a) and (c) because these performed the best within the (B) variations. The performance of BASSO variations (Ba, Bc, and Cc) compare favorably to the reported performance of SOAR in [70].

### *4.4.1 Test Assumption 1: Adaptive Subregion Probability Assumption*

The four adaptive subregion probability variations (a, b, c, and d) form a proper probability distribution (satisfy Assumptions 1.1 and 1.2) by construction. To evaluate whether the adaptive subregion probabilities $\tilde{p}_i$ satisfy (4.10) in Assumption 1.3, we perform 250 function evaluations of the twelve variations of BASSO on the shifted sinusoidal problem and the Rosenbrock problem with dimensions $d = 2$ and 5. During the execution of the algorithm at the $k$th function evaluation when $\tilde{p}$ is updated, we compute the conditional probability ratio on the left-hand side of (4.10) with $t_z$ as the incumbent function value, and the right-hand side with $t_{z'}$ as the 20% quantile of the observed objective function value, $t_z < t_{z'}$. The numerical values in Tables 4.1 - 4.4, where the incumbent value is labeled $y$, $y = t_z$, and the 20% quantile is labeled $z$, $t_{z'}$.

Tables 4.1 - 4.4 illustrate the left-hand side (LHS) of the probability ratio in (4.10) and the right-hand side (RHS) for adaptive subregion probabilities (a) - (d), respectively. When the LHS

is greater than the RHS, Assumption 1.3 is satisfied. The iterations where the estimated LHS is less than the estimated RHS are highlighted in Tables 4.1 - 4.4, indicating a likely violation of Assumption 1.3.

We observe in Table 4.3 that the adaptive subregion probability with a Gaussian process on the range distribution (c) performs the best, with fewer violations to Assumption 1.3 than the other three adaptive subregion probabilities. The experiment with adaptive subregion probability (c) has only one event with a greater RHS on the shifted sinusoidal function and four events on the Rosenbrock function. The predictor function of the cumulative range distribution from the Gaussian process on the range appears to be effective in guiding the adaptive subregion probabilities towards promising subregions.

As shown in Table 4.1, the adaptive subregion probability with observed best function value (a) satisfies Assumption 1.3 more frequently on the Rosenbrock test function than on the shifted sinusoidal test function. This may be due to the greedy nature of the adaptive subregion probability (a). The shifted sinusoidal test function has more local minima than the Rosenbrock test function, so intuitively, a greedy approach would work better on a test function with few local minima.

On the other hand, adaptive subregion probabilities (b) and (d) satisfy Assumption 1.3 more frequently on the shifted sinusoidal test function than on the Rosenbrock test function (Tables 4.2 and 4.4). These two variations incorporate uncertainty, resulting in more exploration. Exploration is needed on the shifted sinusoidal test function, where the local minima are widely distributed.

From these numerical experiments, we conclude that the variation with Gaussian process on the range distribution (c) comes closest to roughly satisfying Assumption 1. Intuitively, variation (c) does the best job of balancing exploration with exploitation.

**A Uniform**

| Shifted sinusoidal $d = 2$ | | | | | Shifted sinusoidal $d = 5$ | | | | | Rosenbrock $d = 2$ | | | | | Rosenbrock $d = 5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.415 | 0.348 | 1.63 | 2.769 | 10 | 0.177 | 0.183 | 2.811 | 3.411 | 10 | 0.867 | 0.867 | 185.542 | 232.845 | 10 | 0.721 | 0.721 | 642.486 | 840.008 |
| 18 | 0.173 | 0.174 | 0.998 | 2.229 | 15 | 0.544 | 0.517 | 2.67 | 2.961 | 110 | 0.012 | 0 | 1.256 | 29.326 | 114 | 0.154 | 0.157 | 198.088 | 791.735 |
| 26 | 0.233 | 0.251 | 0.998 | 1.993 | 32 | 0.165 | 0.148 | 2.594 | 3.347 | 212 | 0.019 | 0.129 | 1.256 | 41.415 | 214 | 0.071 | 0.068 | 139.612 | 763.788 |
| 35 | 0.268 | 0.293 | 0.998 | 1.772 | 38 | 0.21 | 0.185 | 2.594 | 3.287 | 312 | 0.059 | 0.042 | 1.256 | 37.179 | 319 | 0.004 | 0.004 | 33.084 | 718.587 |
| 45 | 0.267 | 0.29 | 0.998 | 1.772 | 47 | 0.212 | 0.182 | 2.594 | 3.284 | | | | | | | | | | |
| 55 | 0.32 | 0.344 | 0.998 | 1.63 | 57 | 0.209 | 0.181 | 2.594 | 3.287 | | | | | | | | | | |
| 65 | 0.461 | 0.47 | 0.998 | 1.332 | 67 | 0.08 | 0.068 | 1.92 | 3.284 | | | | | | | | | | |
| 85 | 0.054 | 0.053 | 0.095 | 1.233 | 120 | 0.1 | 0.11 | 1.92 | 3.181 | | | | | | | | | | |
| 129 | 0.004 | 0.004 | 0.008 | 1.165 | 142 | 0.032 | 0.033 | 1.505 | 3.06 | | | | | | | | | | |
| 141 | 0.004 | 0.004 | 0.008 | 1.162 | 159 | 0.005 | 0.004 | 1.168 | 2.978 | | | | | | | | | | |
| 161 | 0.004 | 0.005 | 0.008 | 1.108 | 217 | 0.005 | 0.005 | 1.168 | 2.958 | | | | | | | | | | |
| 180 | 0.008 | 0.008 | 0.008 | 0.909 | 258 | 0.008 | 0.008 | 1.168 | 2.738 | | | | | | | | | | |
| 200 | 0.011 | 0.011 | 0.008 | 0.829 | | | | | | | | | | | | | | | |
| 219 | 0.006 | 0.006 | 0.003 | 0.527 | | | | | | | | | | | | | | | |
| 265 | 0.011 | 0.012 | 0.003 | 0.254 | | | | | | | | | | | | | | | |

**B Gaussian Process**

| Shifted sinusoidal $d = 2$ | | | | | Shifted sinusoidal $d = 5$ | | | | | Rosenbrock $d = 2$ | | | | | Rosenbrock $d = 5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.415 | 0.348 | 1.63 | 2.769 | 10 | 0.466 | 0.474 | 3.286 | 3.5 | 10 | 0.867 | 0.867 | 185.542 | 232.845 | 10 | 0.721 | 0.721 | 642.486 | 840.008 |
| 64 | 0.145 | 0.148 | 1.002 | 2.769 | 20 | 0.066 | 0.075 | 2.531 | 3.489 | 111 | 0.705 | 0.361 | 158.721 | 645.963 | 114 | 0.104 | 0.148 | 358.04 | 2046.24 |
| 116 | 0.161 | 0.181 | 1.002 | 2.179 | 64 | 0.087 | 0.052 | 2.09 | 3.35 | 220 | 0.265 | 0.216 | 7.3 | 185.542 | 217 | 0.073 | 0.134 | 358.04 | 2404.12 |
| 170 | 0.165 | 0.165 | 1.002 | 2.251 | 67 | 0.091 | 0.037 | 2.09 | 3.358 | 327 | 0.061 | 0.063 | 0.542 | 122.652 | 321 | 0.055 | 0.102 | 358.04 | 2587.64 |
| 223 | 0.165 | 0.164 | 1.002 | 2.316 | 73 | 0.019 | 0.02 | 1.443 | 3.325 | | | | | | | | | | |
| 238 | 0.058 | 0.059 | 0.278 | 2.179 | 130 | 0.022 | 0.023 | 1.415 | 3.136 | | | | | | | | | | |
| 260 | 0.069 | 0.074 | 0.278 | 1.932 | 183 | 0.022 | 0.023 | 1.415 | 3.131 | | | | | | | | | | |
| | | | | | 246 | 0.012 | 0.011 | 1.321 | 3.047 | | | | | | | | | | |
| | | | | | 265 | 0.002 | 0.002 | 1.016 | 2.995 | | | | | | | | | | |

**C Quadratic Regression**

| Shifted sinusoidal $d = 2$ | | | | | Shifted sinusoidal $d = 5$ | | | | | Rosenbrock $d = 2$ | | | | | Rosenbrock $d = 5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.415 | 0.348 | 1.63 | 2.769 | 10 | 0.177 | 0.183 | 2.811 | 3.411 | 10 | 0.867 | 0.867 | 185.542 | 232.845 | 10 | 0.721 | 0.721 | 642.486 | 840.008 |
| 18 | 0.173 | 0.174 | 0.998 | 2.229 | 15 | 0.544 | 0.517 | 2.67 | 2.961 | 110 | 0.038 | 0.001 | 1.067 | 5.379 | 114 | 0.66 | 0.66 | 16.327 | 20.32 |
| 26 | 0.233 | 0.251 | 0.998 | 1.993 | 32 | 0.165 | 0.148 | 2.594 | 3.347 | 213 | 1 | 1 | 1.019 | 1.044 | 215 | 0.114 | 0.115 | 16.327 | 43.128 |
| 35 | 0.268 | 0.293 | 0.998 | 1.772 | 38 | 0.21 | 0.185 | 2.594 | 3.287 | 316 | 0 | 0 | 0 | 1.052 | 320 | 0.025 | 0.025 | 3.918 | 20.309 |
| 45 | 0.267 | 0.29 | 0.998 | 1.772 | 47 | 0.212 | 0.182 | 2.594 | 3.284 | | | | | | | | | | |
| 55 | 0.32 | 0.344 | 0.998 | 1.63 | 57 | 0.209 | 0.181 | 2.594 | 3.287 | | | | | | | | | | |
| 65 | 0.461 | 0.47 | 0.998 | 1.332 | 67 | 0.08 | 0.068 | 1.92 | 3.284 | | | | | | | | | | |
| 85 | 0.054 | 0.053 | 0.095 | 1.233 | 120 | 0.1 | 0.11 | 1.92 | 3.181 | | | | | | | | | | |
| 129 | 0.004 | 0.004 | 0.008 | 1.165 | 142 | 0.032 | 0.033 | 1.505 | 3.06 | | | | | | | | | | |
| 141 | 0.004 | 0.004 | 0.008 | 1.162 | 159 | 0.005 | 0.004 | 1.168 | 2.978 | | | | | | | | | | |
| 161 | 0.004 | 0.005 | 0.008 | 1.108 | 217 | 0.005 | 0.005 | 1.168 | 2.958 | | | | | | | | | | |
| 180 | 0.008 | 0.008 | 0.008 | 0.909 | 258 | 0.008 | 0.008 | 1.168 | 2.738 | | | | | | | | | | |
| 200 | 0.011 | 0.011 | 0.008 | 0.829 | | | | | | | | | | | | | | | |
| 219 | 0.006 | 0.006 | 0.003 | 0.527 | | | | | | | | | | | | | | | |
| 265 | 0.011 | 0.012 | 0.003 | 0.254 | | | | | | | | | | | | | | | |

**Table 4.1:** Numerical values for testing Assumption 1.3 with adaptive subregion probabilities (a) observed best function value. Highlighted cells violate Assumption 1.3.

**A Uniform**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.393 | 0.456 | 1.63 | 2.769 | 10 | 0.149 | 0.092 | 2.811 | 3.411 | 10 | 0.491 | 0.489 | 8.298 | 35.649 | 10 | 0.07 | 0.07 | 157.848 | 1012.868 |
| 18 | 0.135 | 0 | 0.998 | 2.229 | 15 | 0.552 | 0.571 | 2.67 | 2.961 | 15 | 0.41 | 0.406 | 8.298 | 35.649 | 15 | 0.171 | 0.172 | 105.59 | 287.408 |
| 25 | 0.23 | 0.129 | 0.998 | 1.912 | 32 | 0.176 | 0.173 | 2.594 | 3.347 | 25 | 0.145 | 0.113 | 3.834 | 35.649 | 25 | 0.157 | 0.156 | 105.59 | 305.584 |
| 35 | 0.014 | 0 | 0.108 | 1.847 | 38 | 0.202 | 0.182 | 2.594 | 3.287 | 45 | 0.087 | 0.069 | 3.017 | 60.589 | 35 | 0.146 | 0.135 | 105.59 | 305.584 |
| 87 | 0.018 | 0 | 0.108 | 1.762 | 47 | 0.199 | 0.176 | 2.594 | 3.284 | 51 | 0.057 | 0.037 | 2.17 | 58.605 | 45 | 0.072 | 0.065 | 105.59 | 497.471 |
| 105 | 0.001 | 0 | 0.008 | 1.722 | 57 | 0.197 | 0.181 | 2.594 | 3.287 | 99 | 0.01 | 0.012 | 0.015 | 57.96 | 55 | 0.072 | 0.064 | 105.59 | 509.156 |
| 113 | 0.001 | 0 | 0.008 | 1.56 | 67 | 0.066 | 0.043 | 1.92 | 3.284 | 115 | 0.009 | 0.01 | 0.015 | 29.507 | 65 | 0.072 | 0.066 | 105.59 | 497.471 |
| 123 | 0.001 | 0 | 0.008 | 1.108 | 120 | 0.042 | 0.001 | 1.92 | 3.272 | 128 | 0.016 | 0.019 | 0.015 | 14.095 | 85 | 0.016 | 0.014 | 43.581 | 384.793 |
| 136 | 0.002 | 0 | 0.008 | 0.838 | 143 | 0.015 | 0 | 1.505 | 3.082 | 188 | 0.018 | 0.022 | 0.015 | 12.251 | 149 | 0.013 | 0.013 | 40.763 | 385.923 |
| 151 | 0 | 0 | 0.008 | 0.521 | 156 | 0.003 | 0 | 1.168 | 2.978 | 252 | 0.016 | 0.019 | 0.015 | 14.095 | 169 | 0.012 | 0.012 | 39.685 | 384.793 |
| 166 | 0 | 0 | 0.002 | 0.361 | 214 | 0.004 | 0 | 1.168 | 2.871 | | | | | | 185 | 0.002 | 0.002 | 18.616 | 340.773 |
| 232 | 0 | 0 | 0.001 | 0.398 | 257 | 0.004 | 0 | 1.168 | 2.742 | | | | | | 246 | 0.002 | 0.002 | 18.616 | 379.67 |
| 264 | 0 | 0 | 0.001 | 0.348 | | | | | | | | | | | 268 | 0.002 | 0.002 | 18.616 | 332.86 |

**B Gaussian Process**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.393 | 0.456 | 1.63 | 2.769 | 20 | 0.077 | 0.055 | 1.883 | 3.209 | 10 | 0.491 | 0.489 | 8.298 | 35.649 | 10 | 0.07 | 0.07 | 157.848 | 1012.868 |
| 64 | 0.094 | 0 | 1.002 | 2.636 | 74 | 0.04 | 0.015 | 1.883 | 3.517 | 15 | 0.41 | 0.406 | 8.298 | 35.649 | 60 | 0.032 | 0.03 | 157.848 | 2414.04 |
| 116 | 0.161 | 0.009 | 1.002 | 2.116 | 140 | 0.002 | 0.001 | 1.207 | 3.533 | 25 | 0.145 | 0.113 | 3.834 | 35.649 | 117 | 0.028 | 0.024 | 157.848 | 2207.16 |
| 174 | 0.158 | 0.101 | 1.002 | 2.116 | 202 | 0.002 | 0 | 1.207 | 3.637 | 53 | 0.073 | 0.055 | 3.62 | 72.723 | 175 | 0.009 | 0.005 | 82.823 | 2085.52 |
| 227 | 0.158 | 0.088 | 1.002 | 2.145 | 261 | 0.001 | 0 | 1.207 | 3.686 | 59 | 0.189 | 0.161 | 3.62 | 35.649 | 230 | 0.009 | 0.006 | 82.823 | 1988.828 |
| 259 | 0.182 | 0.13 | 1.002 | 2.116 | | | | | | 68 | 0.067 | 0.061 | 0.416 | 15.787 | 256 | 0.009 | 0.005 | 82.823 | 2045.24 |
| | | | | | | | | | | 122 | 0.049 | 0.048 | 0.416 | 35.649 | | | | | |
| | | | | | | | | | | 186 | 0.061 | 0.062 | 0.416 | 33.315 | | | | | |
| | | | | | | | | | | 256 | 0.064 | 0.065 | 0.416 | 31.607 | | | | | |

**C Quadratic Regression**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.393 | 0.456 | 1.63 | 2.769 | 10 | 0.149 | 0.092 | 2.811 | 3.411 | 10 | 0.491 | 0.489 | 8.298 | 35.649 | 10 | 0.07 | 0.07 | 157.848 | 1012.868 |
| 18 | 0.135 | 0 | 0.998 | 2.229 | 15 | 0.552 | 0.571 | 2.67 | 2.961 | 15 | 0.41 | 0.406 | 8.298 | 35.649 | 15 | 0.171 | 0.172 | 105.59 | 287.408 |
| 25 | 0.23 | 0.129 | 0.998 | 1.912 | 32 | 0.176 | 0.173 | 2.594 | 3.347 | 25 | 0.145 | 0.113 | 3.834 | 35.649 | 25 | 0.157 | 0.156 | 105.59 | 305.584 |
| 35 | 0.014 | 0 | 0.108 | 1.847 | 38 | 0.202 | 0.182 | 2.594 | 3.287 | 45 | 0.087 | 0.069 | 3.017 | 60.589 | 35 | 0.146 | 0.135 | 105.59 | 305.584 |
| 87 | 0.018 | 0 | 0.108 | 1.762 | 47 | 0.199 | 0.176 | 2.594 | 3.284 | 51 | 0.057 | 0.037 | 2.17 | 58.605 | 45 | 0.072 | 0.065 | 105.59 | 497.471 |
| 105 | 0.001 | 0 | 0.008 | 1.722 | 57 | 0.197 | 0.181 | 2.594 | 3.287 | 99 | 0.01 | 0.012 | 0.015 | 57.96 | 55 | 0.072 | 0.064 | 105.59 | 509.156 |
| 113 | 0.001 | 0 | 0.008 | 1.56 | 67 | 0.066 | 0.043 | 1.92 | 3.284 | 115 | 0.009 | 0.01 | 0.015 | 29.507 | 65 | 0.072 | 0.066 | 105.59 | 497.471 |
| 123 | 0.001 | 0 | 0.008 | 1.108 | 120 | 0.042 | 0.001 | 1.92 | 3.272 | 128 | 0.016 | 0.019 | 0.015 | 14.095 | 85 | 0.016 | 0.014 | 43.581 | 384.793 |
| 136 | 0.002 | 0 | 0.008 | 0.838 | 143 | 0.015 | 0 | 1.505 | 3.082 | 188 | 0.016 | 0.018 | 0.015 | 14.4 | 149 | 0.013 | 0.013 | 40.763 | 385.923 |
| 151 | 0 | 0 | 0.008 | 0.521 | 156 | 0.003 | 0 | 1.168 | 2.978 | 248 | 0.016 | 0.018 | 0.015 | 14.825 | 169 | 0.012 | 0.012 | 39.685 | 384.793 |
| 166 | 0 | 0 | 0.002 | 0.361 | 214 | 0.004 | 0 | 1.168 | 2.871 | 261 | 0.02 | 0.022 | 0.015 | 9.111 | 185 | 0.002 | 0.002 | 18.616 | 340.773 |
| 232 | 0 | 0 | 0.001 | 0.398 | 257 | 0.004 | 0 | 1.168 | 2.742 | | | | | | 246 | 0.002 | 0.002 | 18.616 | 379.67 |
| 264 | 0 | 0 | 0.001 | 0.348 | | | | | | | | | | | 268 | 0.002 | 0.002 | 18.616 | 332.86 |

**Table 4.2:** Numerical values for testing Assumption 1.3 with with adaptive subregion probabilities (b) sample variance. Highlighted cells violate Assumption 1.3.

**A Uniform**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.475 | 0 | 1.63 | 2.769 | 10 | 0.593 | 0.624 | 3.429 | 3.583 | 4 | 0.249 | 0.203 | 0.971 | 4.073 | 4 | 0.918 | 0.918 | 1278.517 | 1398.628 |
| 15 | 0.351 | 0.33 | 1.259 | 2.442 | 22 | 0.126 | 0.042 | 2.068 | 3.443 | 104 | 0.025 | 0.013 | 0.167 | 27.401 | 106 | 0.024 | 0.027 | 68.333 | 648.264 |
| 25 | 0.113 | 0.089 | 0.63 | 2.15 | 32 | 0.035 | 0.03 | 1.513 | 3.331 | 204 | 0.029 | 0.018 | 0.167 | 41.594 | 206 | 0.007 | 0.003 | 30.144 | 636.586 |
| 48 | 0.099 | 0.043 | 0.229 | 1.685 | 42 | 0.049 | 0.006 | 1.513 | 3.246 | 304 | 0.030 | 0.021 | 0.167 | 46.290 | 308 | 0.010 | 0.007 | 30.144 | 623.913 |
| 54 | 0.189 | 0.066 | 0.229 | 1.685 | 52 | 0.052 | 0.006 | 1.513 | 3.246 | | | | | | | | | | |
| 63 | 0.148 | 0.007 | 0.096 | 1.4 | 60 | 0.025 | 0.005 | 1.306 | 2.972 | | | | | | | | | | |
| 78 | 0.092 | 0.078 | 0.053 | 0.788 | 110 | 0.013 | 0.012 | 1.115 | 2.33 | | | | | | | | | | |
| 93 | 0.111 | 0.005 | 0.053 | 0.653 | 120 | 0.013 | 0.004 | 1.115 | 2.395 | | | | | | | | | | |
| 110 | 0.023 | 0.001 | 0.006 | 0.512 | 137 | 0.014 | 0.006 | 1.115 | 2.33 | | | | | | | | | | |
| 172 | 0.02 | 0 | 0.006 | 0.535 | 155 | 0.014 | 0.008 | 1.115 | 2.249 | | | | | | | | | | |
| 227 | 0.013 | 0.006 | 0.006 | 0.512 | 175 | 0.019 | 0.01 | 1.115 | 2.1 | | | | | | | | | | |
| 262 | 0.024 | 0 | 0.006 | 0.377 | 195 | 0.004 | 0.001 | 0.708 | 1.967 | | | | | | | | | | |
| | | | | | 255 | 0.004 | 0.002 | 0.708 | 1.967 | | | | | | | | | | |

**B Gaussian Process**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.515 | 0.515 | 1.209 | 1.778 | 10 | 0.402 | 0.125 | 2.727 | 3.222 | 10 | 0.109 | 0.110 | 1.382 | 23.492 | 10 | 0.325 | 0.315 | 196.646 | 416.047 |
| 65 | 0.215 | 0 | 1.209 | 4.479 | 18 | 0.176 | 0.06 | 2.627 | 3.29 | 20 | 0.114 | 0.072 | 1.382 | 23.492 | 20 | 0.141 | 0.139 | 120.619 | 416.047 |
| 76 | 0.2 | 0.014 | 1.1 | 3.229 | 28 | 0.386 | 0.053 | 2.627 | 3.322 | 30 | 0.022 | 0.026 | 0.102 | 34.613 | 130 | 0.101 | 0.098 | 91.088 | 404.833 |
| 86 | 0.2 | 0.099 | 1.1 | 2.524 | 33 | 0.334 | 0.203 | 2.627 | 3.322 | 140 | 0.043 | 0.016 | 0.102 | 47.682 | 240 | 0.104 | 0.093 | 91.088 | 400.131 |
| 96 | 0.102 | 0.034 | 0.242 | 2.163 | 43 | 0.206 | 0.184 | 2.441 | 3.29 | 250 | 0.019 | 0.001 | 0.102 | 31.743 | 350 | 0.098 | 0.076 | 91.088 | 400.131 |
| 156 | 0.37 | 0 | 0.242 | 2.516 | 100 | 0.097 | 0.01 | 1.549 | 3.369 | 360 | 0.045 | 0.032 | 0.102 | 17.118 | | | | | |
| 167 | 0.05 | 0 | 0.114 | 1.864 | 154 | 0.146 | 0.048 | 1.549 | 3.238 | | | | | | | | | | |
| 187 | 0.014 | 0.001 | 0.009 | 1.745 | 215 | 0.185 | 0.018 | 1.549 | 3.14 | | | | | | | | | | |
| 257 | 0.002 | 0.001 | 0.009 | 1.825 | 259 | 0.245 | 0.046 | 1.549 | 3.039 | | | | | | | | | | |

**C Quadratic Regression**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.49 | 0 | 1.65 | 2.77 | 50 | 0.121 | 0.088 | 2.077 | 3.246 | 10 | 0.375 | 0.262 | 1.006 | 5.158 | 10 | 0.967 | 0.967 | 1143.565 | 1180.265 |
| 15 | 0.02 | 0.02 | 0.15 | 1.65 | 76 | 0.023 | 0.015 | 1.574 | 3.246 | 115 | 0.308 | 0.000 | 0.453 | 3.366 | 110 | 0.305 | 0.193 | 14.028 | 25.120 |
| 30 | 0 | 0 | 0.03 | 1.19 | 126 | 0.028 | 0.023 | 1.574 | 3.26 | 215 | 0.000 | 0.000 | 0.000 | 1.046 | 215 | 0.166 | 0.167 | 12.824 | 27.317 |
| 40 | 0.01 | 0.01 | 0.03 | 1.26 | 176 | 0.037 | 0 | 1.437 | 3.265 | 315 | 0.000 | 0.000 | 0.000 | 1.009 | 315 | 0.022 | 0.022 | 3.992 | 20.138 |
| 50 | 0.01 | 0.01 | 0.03 | 1.23 | 227 | 0.044 | 0.019 | 1.423 | 3.158 | | | | | | | | | | |
| 60 | 0.01 | 0.01 | 0.03 | 1.19 | | | | | | | | | | | | | | | |
| 66 | 0.01 | 0 | 0.01 | 0.9 | | | | | | | | | | | | | | | |
| 127 | 0.01 | 0 | 0.01 | 0.81 | | | | | | | | | | | | | | | |
| 182 | 0.01 | 0 | 0.01 | 0.77 | | | | | | | | | | | | | | | |
| 237 | 0.02 | 0.01 | 0.01 | 0.74 | | | | | | | | | | | | | | | |
| 257 | 0.01 | 0 | 0.01 | 0.62 | | | | | | | | | | | | | | | |

**Table 4.3:** Numerical values for testing Assumption 1.3 with adaptive subregion probabilities (c) Gaussian process on the range. Highlighted cells violate Assumption 1.3.

**A Uniform**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.445 | 0.41 | 1.63 | 2.769 | 10 | 0.155 | 0.169 | 2.811 | 3.411 | 10 | 0.238 | 0.235 | 7.63 | 123.944 | 10 | 0.093 | 0.096 | 319.605 | 2467.76 |
| 15 | 0.351 | 0.32 | 1.259 | 2.442 | 15 | 0.556 | 0.551 | 2.67 | 2.961 | 55 | 0.242 | 0.241 | 6.874 | 104.011 | 18 | 0.067 | 0.067 | 143.092 | 854.155 |
| 25 | 0.08 | 0.075 | 0.63 | 2.15 | 32 | 0.204 | 0.176 | 2.594 | 3.347 | 56 | 0.171 | 0.186 | 6.874 | 104.011 | 25 | 0.061 | 0.061 | 143.092 | 943.599 |
| 48 | 0.086 | 0.066 | 0.229 | 1.319 | 38 | 0.275 | 0.225 | 2.594 | 3.287 | 62 | 0.212 | 0.221 | 6.44 | 100.838 | 35 | 0.069 | 0.069 | 143.092 | 854.155 |
| 56 | 0.084 | 0.064 | 0.229 | 1.319 | 47 | 0.254 | 0.22 | 2.594 | 3.284 | 68 | 0.231 | 0.225 | 4.275 | 57.553 | 45 | 0.063 | 0.063 | 143.092 | 937.713 |
| 66 | 0.073 | 0.061 | 0.172 | 1.259 | 57 | 0.24 | 0.213 | 2.594 | 3.287 | 84 | 0.022 | 0.02 | 0.196 | 62.788 | 55 | 0.027 | 0.027 | 88.036 | 854.155 |
| 77 | 0.045 | 0.034 | 0.053 | 1.028 | 67 | 0.157 | 0.077 | 1.92 | 3.284 | 91 | 0.027 | 0.023 | 0.196 | 57.553 | 106 | 0.025 | 0.027 | 88.036 | 830.225 |
| 96 | 0.009 | 0.007 | 0.008 | 0.829 | 110 | 0.086 | 0.041 | 1.51 | 3.038 | 111 | 0.051 | 0.047 | 0.196 | 18.663 | 158 | 0.029 | 0.028 | 88.036 | 873.822 |
| 156 | 0.021 | 0.02 | 0.008 | 0.353 | 121 | 0.106 | 0.039 | 1.51 | 3.078 | 123 | 0.059 | 0.053 | 0.07 | 16.368 | 201 | 0.008 | 0.008 | 45.281 | 827.083 |
| 188 | 0.017 | 0.026 | 0.008 | 0.267 | 139 | 0.112 | 0.048 | 1.51 | 2.98 | 187 | 0.022 | 0.02 | 0.07 | 15.322 | 209 | 0.009 | 0.009 | 45.281 | 740.015 |
| 202 | 0.009 | 0.014 | 0.004 | 0.248 | 158 | 0.124 | 0.067 | 1.51 | 2.811 | 247 | 0.011 | 0.01 | 0.003 | 15.322 | 221 | 0.01 | 0.01 | 45.281 | 645.793 |
| 233 | 0 | 0.002 | 0.001 | 0.196 | 177 | 0.038 | 0.015 | 1.221 | 2.61 | 257 | 0.019 | 0.016 | 0.003 | 11.401 | 241 | 0.004 | 0.004 | 28.19 | 554.874 |
| 253 | 0 | 0.003 | 0.001 | 0.172 | 218 | 0.017 | 0.013 | 1.105 | 2.171 | | | | | | 265 | 0.006 | 0.006 | 28.19 | 411.649 |
| | | | | | 235 | 0.017 | 0.003 | 0.673 | 2.053 | | | | | | | | | | |
| | | | | | 261 | 0.037 | 0.003 | 0.673 | 1.958 | | | | | | | | | | |

**B Gaussian Process**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.445 | 0.41 | 1.63 | 2.769 | 10 | 0.155 | 0.169 | 2.811 | 3.411 | 10 | 0.502 | 0.503 | 8.298 | 35.649 | 10 | 0.399 | 0.4 | 427.597 | 863.028 |
| 64 | 0.17 | 0.14 | 1.002 | 2.636 | 15 | 0.548 | 0.534 | 2.489 | 2.811 | 15 | 0.416 | 0.426 | 8.298 | 35.649 | 62 | 0.23 | 0.234 | 427.597 | 1407.291 |
| 94 | 0.141 | 0.131 | 0.848 | 2.097 | 25 | 0.37 | 0.349 | 2.44 | 2.961 | 25 | 0.176 | 0.182 | 3.834 | 35.649 | 118 | 0.226 | 0.23 | 427.597 | 1407.291 |
| 102 | 0.18 | 0.157 | 0.848 | 1.933 | 78 | 0.101 | 0.06 | 2.068 | 3.411 | 53 | 0.047 | 0.051 | 1.22 | 72.152 | 173 | 0.216 | 0.216 | 427.597 | 1533.6 |
| 109 | 0.217 | 0.183 | 0.848 | 1.9 | 133 | 0.124 | 0.056 | 2.068 | 3.442 | 59 | 0.098 | 0.098 | 1.22 | 33.315 | 225 | 0.199 | 0.199 | 427.597 | 1695.92 |
| 119 | 0.27 | 0.239 | 0.848 | 1.615 | 188 | 0.094 | 0.049 | 2.068 | 3.463 | 68 | 0.066 | 0.065 | 0.416 | 15.14 | 254 | 0.177 | 0.186 | 427.597 | 1761.241 |
| 130 | 0.355 | 0.286 | 0.572 | 1.331 | 239 | 0.091 | 0.041 | 2.068 | 3.506 | 123 | 0.068 | 0.064 | 0.416 | 29.507 | | | | | |
| 191 | 0.167 | 0.113 | 0.262 | 1.456 | 273 | 0.153 | 0.051 | 2.068 | 3.463 | 184 | 0.079 | 0.075 | 0.416 | 21.804 | | | | | |
| 259 | 0.125 | 0.058 | 0.105 | 1.361 | | | | | | 250 | 0.081 | 0.077 | 0.416 | 20 | | | | | |
| | | | | | | | | | | 266 | 0.053 | 0.052 | 0.243 | 11.429 | | | | | |

**C Quadratic Regression**

| Shifted sinusoidal $d=2$ | | | | | Shifted sinusoidal $d=5$ | | | | | Rosenbrock $d=2$ | | | | | Rosenbrock $d=5$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ | $k$ | LHS | RHS | $y$ | $z$ |
| 10 | 0.039 | 0.038 | 0.493 | 2.228 | 10 | 0.155 | 0.169 | 2.811 | 3.411 | 10 | 0.238 | 0.235 | 7.63 | 123.944 | 10 | 0.093 | 0.096 | 319.605 | 2467.76 |
| 60 | 0.235 | 0.235 | 0.493 | 0.913 | 15 | 0.556 | 0.551 | 2.67 | 2.961 | 39 | 0.104 | 0.113 | 4.086 | 100 | 18 | 0.067 | 0.067 | 143.092 | 854.155 |
| 110 | 0.301 | 0.3 | 0.493 | 0.904 | 32 | 0.204 | 0.176 | 2.594 | 3.347 | 40 | 0.065 | 0.071 | 2.563 | 72.526 | 25 | 0.061 | 0.061 | 143.092 | 943.599 |
| 162 | 0.399 | 0.372 | 0.493 | 0.913 | 38 | 0.275 | 0.225 | 2.594 | 3.287 | 67 | 0.114 | 0.116 | 1.653 | 29.617 | 35 | 0.069 | 0.069 | 143.092 | 854.155 |
| 171 | 0.094 | 0.078 | 0.077 | 0.912 | 47 | 0.254 | 0.22 | 2.594 | 3.284 | 75 | 0.062 | 0.065 | 0.83 | 28.374 | 45 | 0.063 | 0.063 | 143.092 | 937.713 |
| 226 | 0.104 | 0.072 | 0.077 | 0.898 | 57 | 0.24 | 0.213 | 2.594 | 3.287 | 128 | 0.065 | 0.067 | 0.83 | 27.782 | 55 | 0.027 | 0.027 | 88.036 | 854.155 |
| 251 | 0.075 | 0.068 | 0.077 | 0.895 | 67 | 0.157 | 0.077 | 1.92 | 3.284 | 173 | 0.036 | 0.044 | 0.643 | 16.368 | 106 | 0.025 | 0.027 | 88.036 | 830.225 |
| | | | | | 125 | 0.181 | 0.139 | 1.92 | 2.982 | 182 | 0.041 | 0.047 | 0.48 | 10.173 | 158 | 0.03 | 0.029 | 88.036 | 830.225 |
| | | | | | 135 | 0.073 | 0.036 | 1.452 | 2.981 | 244 | 0.004 | 0.005 | 0.028 | 13.268 | 207 | 0.013 | 0.012 | 49.609 | 618.208 |
| | | | | | 152 | 0.079 | 0.036 | 1.452 | 2.961 | 266 | 0.005 | 0.006 | 0.028 | 9.159 | 216 | 0.01 | 0.01 | 41.59 | 567.647 |
| | | | | | 170 | 0.087 | 0.048 | 1.452 | 2.792 | | | | | | 262 | 0.013 | 0.013 | 41.59 | 449.238 |
| | | | | | 184 | 0.083 | 0.052 | 1.423 | 2.67 | | | | | | | | | | |
| | | | | | 221 | 0.032 | 0.015 | 1.192 | 2.496 | | | | | | | | | | |
| | | | | | 261 | 0.034 | 0.022 | 1.192 | 2.273 | | | | | | | | | | |

**Table 4.4:** Numerical values for testing Assumption 1.3 with adaptive subregion probabilities (d) confidence bound. Highlighted cells violate Assumption 1.3.

### *4.4.2 Test Assumption 2: Surrogate Modeling Assumption*

We perform another experiment to numerically test Assumption 2 using Gaussian processes (B) and quadratic regression (C) as surrogate models and compared to uniform (A). In order to test Assumption 2, we need to compute the probability of improvement at different objective function values and with different methods.

To calculate the probability of improvements, we constructed a one-dimensional centered sinusoidal problem with two sets of sample points, where each subregion has four sample points. Three of the sample points coincide, however the best function value in each subregion differs. We call one set the base case sample set and the other the better-incumbent sample set. See Figure 4.3.

The probability of improvement using a uniform distribution is calculated by estimating $v_{\sigma_i}(y)/v_{\sigma_i}$. The probability of improvement using Gaussian processes is

$$PI_i(x) = \Phi\left(\frac{y - \hat{g}_{i,k}(x)}{\hat{s}_{i,k}(x)}\right) \tag{4.21}$$

for $x \in \sigma_i$, where $\hat{g}_{i,k}(x)$ and $\hat{s}_{i,k}(x)$ are constructed from the Gaussian process for both the base case sample set and the better-incumbent sample set in each subregion.The probability of improvement is assessed at the point that maximizes the expected improvement function $EI_i(x)$.

Figure 4.4 illustrates the expected improvement functions $EI_i(x)$, as in (4.6), with maximal points for each subregion and for both sample sets, where Figure 4.4(a) is for the better-incumbent sample set and Figure 4.4(b) is for the base case sample set. The new sample point for each subregion is the point that maximizes the expected improvement function, illustrated by the vertical dotted line. Notice that there is a different point selected between the base case sample set and the better-incumbent sample set, due to the different best function values.

The probability of improvement using regularized quadratic regression is calculated using

$$PI_i(x) = P\left(T(x) < \frac{y - \hat{q}_{i,k}(x)}{\sqrt{MSE_i + (SE_i(x))^2}}\right), \tag{4.22}$$

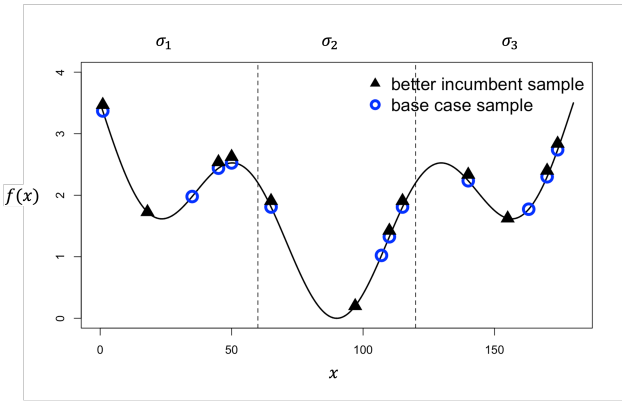



**Figure 4.3:** Two different sets of sample points (blue circle base case sample set and black triangle better-incumbent sample set) are used to build the Gaussian processes with 3 of the 4 sample points in each subregion coinciding. The black triangle sample set has better incumbent points than the blue circle sample set.

for $x \in \sigma_i$, where $y$ is the 5th best function value of both sample sets. The random variable $T(x)$ represents the true objective function value at $x$, $f(x)$, and follows a $t$ distribution with degrees of freedom set to $N_k^i - 3$.

The mean squared error for subregion $\sigma_i$ is

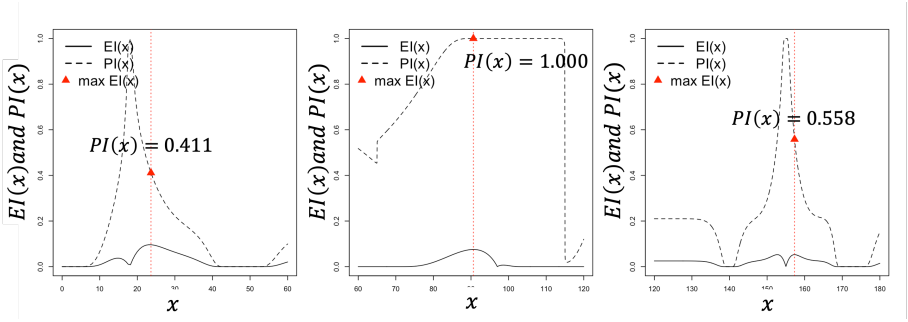
$$MSE_i = \frac{1}{N_k^i} \sum_{j=1}^{N_k^i} (\hat{q}_k^i(x_j) - f(x_j))^2,$$

and the standard error at $x \in \sigma_i$ is

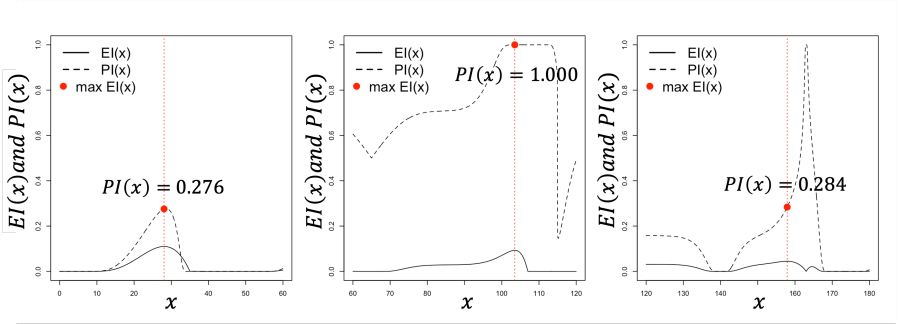
$$SE_i(x) = \sqrt{MSE_i(Q^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}Q)},$$

where $Q$ is a vector of quadratic regression terms at point $x$. The size of $Q$ is 3 for $d = 1$ and the size is $1 + 2d + {}^dC_2$ for $d > 1$. For example,

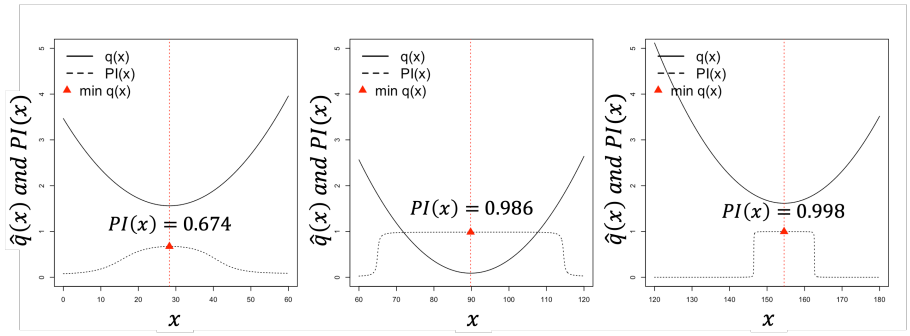
- when $x$ is 1-dimensional, $Q = \{1, x_1, x_1^2\}$,
- when $x$ is 2-dimensional, $Q = \{1, x_1, x_2, x_1^2, x_2^2, x_1x_2\}$,



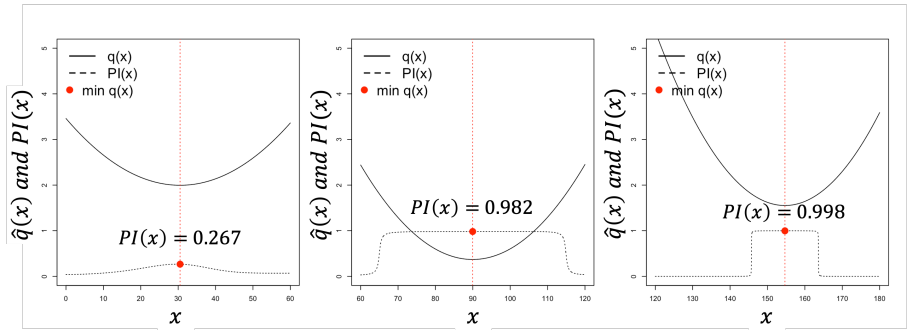

(a) Gaussian process using the black triangle better incumbent sample set.




(b) Gaussian process using the blue circle base case sample set.




(c) Quadratic Regression using the black triangle better incumbent sample set.




(d) Quadratic regression using the blue circle base case sample set.

**Figure 4.4:** Testing Assumption 2.

- and when $x$ is 3-dimensional, $Q = \{1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3, x_1x_3\}$,

where $x_\ell$ is the $\ell$th component (decision variable) of point $x$.

The design matrix $\mathbf{X}$ has a row for every point sampled in subregion $\sigma_i$, so the size of $\mathbf{X}$ is $N_k^i \times (1 + 2d + {}^dC_2)$.

The probability of improvement is assessed at the point that minimizes the quadratic regression function. Figures 4.4(c) and 4.4(d) illustrate the quadratic regression $\hat{q}(x)$, with minimal points, for both sample sets. Notice the different locations between the quadratic regressions in Figures 4.4(c) and 4.4(d)and those using Gaussian processes in Figures 4.4(a) and 4.4(b).

As shown in Table 4.5, the probabilities of improvement using uniform sampling are smaller than those using the base case sample set for the Gaussian process and for quadratic regression. This supports Assumption 2.1 in (4.12) that surrogate modeling with Gaussian processes and quadratic regression can outperform uniform sampling. We also observe that the probabilities of improvement using the better-incumbent sample set (with $z$) are higher than those using the base case sample set (with $z'$). We calculate the probability of improvement within a subregion to be below a value $y$, where $y$ is the 5th best function value of both sample sets. This supports Assumption 2.2 in (4.13) that a better incumbent function value improves surrogate performance.

For this simplified numerical experiment, we see that both surrogate models (Gaussian processes and quadratic regression) outperform uniform sampling, as anticipated. However, we do not expect that Assumption 2 will hold for more challenging problems. We now turn to numerical experiments to assess the expected performance as dimension increases.

### *4.4.3 Expected Performance by Dimension*

The theoretical result in Corollary 8 states that, with the assumptions, the expected number of function evaluations for BASSO to reach a value $y_* + \varepsilon,\ \varepsilon > 0$ , is bounded by a linear function of dimension. To assess the impact of dimension on BASSO performance, we graph the average number of function evaluations for BASSO with the two surrogate models and four ways to choose subregions (Ba, Bb, Bc, Bd, Ca, Cb, Cc, Cd) to visualize whether the performance is linear in

| Estimate | $\sigma_1$ | $\sigma_2$ | $\sigma_3$ |
|---|---|---|---|
| $P(Y_{k+1}^{BASSOunif} \leq y \mid Y_k^{BASSOunif} = z, X^{BASSOunif} \in \sigma_i)$ | 0.252 | 0.833 | 0.252 |
| $P(Y_{k+1}^{BASSOsurrGP} \leq y \mid Y_k^{BASSOsurrGP} = z', X^{BASSOsurrGP} \in \sigma_i)$ | 0.276 | 1.000 | 0.284 |
| $P(Y_{k+1}^{BASSOsurrGP} \leq y \mid Y_k^{BASSOsurrGP} = z, X^{BASSOsurrGP} \in \sigma_i)$ | 0.411 | 1.000 | 0.558 |
| $P(Y_{k+1}^{BASSOsurrQ} \leq y \mid Y_k^{BASSOsurrQ} = z', X^{BASSOsurrQ} \in \sigma_i)$ | 0.267 | 0.982 | 0.998 |
| $P(Y_{k+1}^{BASSOsurrQ} \leq y \mid Y_k^{BASSOsurrQ} = z, X^{BASSOsurrQ} \in \sigma_i)$ | 0.674 | 0.986 | 0.998 |

**Table 4.5:** Result from numerical experiment to test Assumption 2. Observe that the probability of improvement of BASSO with the surrogate modeling (Gaussian processes and quadratic regression) can outperform uniform sampling. The probabilities of improvement using the better-incumbent sample set (with $z$) are higher than those using the base case sample set (with $z'$).

dimension. We include SNOBFIT and SOAR in this experiment.

We perform the numerical experiments on five benchmark black-box optimization functions namely Branin, Hartmann, shifted sinusoidal, centered sinusoidal and Ackley, for dimensions 10, 20, 30, 40, and 50. The number of function evaluations to reach $y_* + \varepsilon$ objective function value are averaged over 10 replications of each algorithm and illustrated in Figure 4.5.

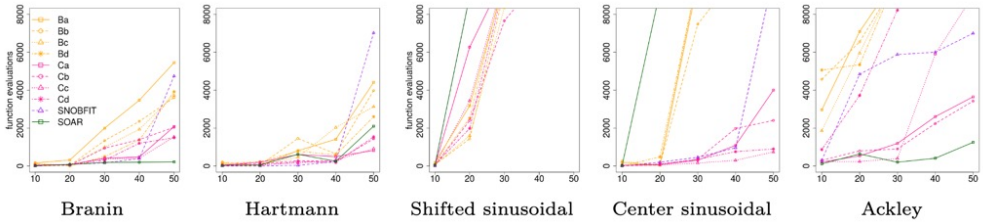



**Figure 4.5:** Number of function evaluations, averaged over 10 replications, until first achieving a near-optimal function value of $y_* + \varepsilon$ for 8 BASSO variations (Ba, Bb, Bc, Bd, Ca, Cb, Cc, Cd), SNOBFIT, and SOAR across dimensions 10, 20, 30, 40 and 50, and five test functions, (a) Branin for $y_* + \varepsilon = 22$ , (b) Hartmann for $y_* + \varepsilon = -1$, (c) shifted sinusoidal for $y_* + \varepsilon = 3.3$, (d) centered sinusoidal for $y_* + \varepsilon = 3.3$, and (e) Ackley function for $y_* + \varepsilon = 16.5$.

We observe that the desired linearity result does not hold across all algorithms and all test functions. However, there are some hints that linearity in dimension is possible in some cases. The shifted sinusoidal function is the hardest of the test functions. It is more challenging to find the improving level set as dimension increases, especially when the level set is not contiguous (as in the sinsoidal test functions). Since the BASSO variation with regularized quadratic regression (C),

while not focusing on high frequency components of the objective, it still appears to capture an overall trend within subregion, resulting in improved scalability.

Our intuition is that approximating the high frequency variations, as with a Gaussian proccess as a surrogate model, does not guide the algorithm to sample within the entire improving level set. Using a Gaussian proccess as a surrogate model may result in too much exploitation and not enough exploration. The regularized quadratic regression is not as accurate in predicting the objective function as a Gaussian process, however, it appears to balance exploration with exploitation.

We next turn to examine data profiles of the same algorithms to give more information on the how performance is impacted by the number of function evaluations.

### *4.4.4 Data Profiles*

To compare the performance of the different algorithms, we use data profiles, as in [71]. We include 10 algorithms: 8 BASSO variations (Ba, Bb, Bc, Bd, Ca, Cb, Cc, Cd), SNOBFIT, and SOAR. The set of problems has 25 problems; 5 test functions each at dimension 10, 20, 30, 40, 50.

We are interested in the fraction of problems $p \in P$ that can be solved within a given tolerance $\tau$ by solver $s \in S$ in $\alpha$ function evaluations. We replicate each algorithm 10 times and use common random seeds for the replications. We average the objective function values over the 10 replications for each algorithm. Let $f(x_0)$ be the average function value (of the 10 replications) of the starting point, $f_s(x_k)$ is the average incumbent function value (over 10 replications) reported by solver $s$ on the $k$th function evaluation, and $f_L$ is the smallest average objective function value obtained by any solver within the computational budget limit $\mu_f$ of 10,000 function evaluations.

For a given tolerance $0 \leq \tau \leq 1$, a problem is considered 'solved' by a solver if its solution improved the starting point by at least a fraction of $(1-\tau)$ of the largest possible reduction by any solver in the set of interest within a given number of function evaluations, i.e., the convergence test in (4.23) is satisfied in the given number of function evaluations. The convergence test is

$$f(x_0) - f_s(x_k) \geq (1-\tau)(f(x_0) - f_L). \tag{4.23}$$

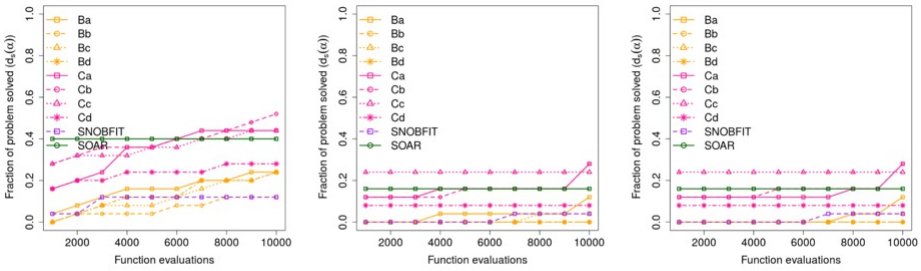
Given a computation budget of $\mu_f = 10,000$ function evaluations, we define $t^{\tau}_{p,s}$ as,

$$t^{\tau}_{p,s} = \begin{cases} \text{number of function evaluations that first satisfy the convergence test by} \\ \text{solver } s \text{ on problem } p \text{ at tolerance } \tau \\ \\ \infty \text{ if the convergence test is not satisfied within } \mu_f \text{ function evaluations.} \end{cases}$$

We define the data profile of solver $s$ at threshold $\alpha$ with tolerance $\tau$ as in [71],

$$d_s(\alpha) = \frac{1}{|P|} size\{p \in P : \frac{t^{\tau}_{p,s}}{n_p + 1} \leq \alpha\},$$

where $n_p$ is the dimension of problem $p$.




(a) $\tau = 0.1$ (b) $\tau = 0.001$ (c) $\tau = 0.00001$

**Figure 4.6:** Data profiles for 10 algorithms on 25 benchmark problems (5 test problems in dimensions 10, 20, 30, 40, and 50), with (a) $\tau = 0.1$, (b) $\tau = 0.001$, and (c) $\tau = 0.00001$.

From the data profiles in Figure 4.6, we see that the BASSO variation (Cc) with the regularized quadratic regression as a surrogate model (C) and the Gaussian process on the range distribution as the adaptive subregion probability (c) outperforms the other BASSO variations. This is consistent with the finding that BASSO subregion variation (c) came closest to satisfying Assumption 1, as discussed in Section 4.4.1. As discussed in Section 4.4.3, the BASSO variations with a regularized quadratic regression as a surrogate model (C) dominated those with a Gaussian process as a surrogate model (B), possibly by achieving a balance between exploration and exploitation.

It is interesting that SOAR performs very well overall. To contrast SOAR with BASSO (Cc), SOAR uses a Gaussian process to obtain a restart location for a local search while BASSO (Cc) uses partitioning and adaptive subregion probabilities to focus a subregion search. The Gaussian process in SOAR is very effective for problems with dimensions 10 and 20, as illustrated in Figure 4.5, but more samples are needed in higher dimensions, limiting the scalability.

### *4.4.5 Experiment with higer dimensions*

We performed additional numerical experiments with 12 variations of BASSO numerically evaluate the impact of dimension on test problems with higer dimensions. We included experiments with three algorithms with available code (CMA-ES [45], SNOBFIT [51], and NOMAD[63]). We chose these three algorithms because they ranked highly with good performance in [66]. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [45] is a population-based evolutionary algorithm that samples new points from a multivariate Gaussian distribution and evaluates their objective function values. The points are sorted by function values and the distribution parameters (i.e., the mean vector and the covariance matrix) are updated based on the ranking of function values. The Stable Noisy Optimization by Branch and Fit algorithm (SNOBFIT) [51] combines global and local search by branching and fitting a surrogate model on subregions. SNOBFIT builds local models of the function similar to trust region approaches. No guarantees can be given that a global minimum is located. The Non-linear Optimization with Mesh Adaptive Direct Search algorithm (NOMAD) [63] has been used for simulation-based optimization. The Mesh Adaptive Direct Search (MADS) algorithm [6] generates a series of grids with varying discretizations of the space of variables. The adaptive mesh frame acts as a window that constrains the search to a specific region of the space, and allows an efficient allocation of computational resources.Although none of these algorithms scale as well as the theoretical ideal, the numerical experiments show the effect of dimension on computational performance.

All twelve BASSO variations use the same branching strategy that is described in Section 4.2.2. The value for the maximum number of function evaluations without improving the incumbent value, in Step 1 of BASSO, is set to 50. The stopping criterion is a fixed number of function

evaluations.

### *4.4.6 Application to Falsification of Cyberphysical Systems*

To demonstrate the efficacy and relevance of the BASSO algorithm to engineering applications, we developed an add-on BASSO module to the open source Search Based Test Generation (SBTG) Python package PSY-TALIRO[101]. PSY-TALIRO implements a similar architecture to S-TALIRO [3] and Matlab tools [29]. Currently, the system requirements can be expressed in a fragment of Timed Propositional Temporal Logic (TPTL) [28] which has polynomial time complexity and is strictly more expressive than Signal or Metric Temporal Logic (STL/MTL) [12]. In addition, the tool can use the Python packages RTAMT [74] and TLTk [24] that provide optimized algorithms for STL robustness monitoring.

The Search Based Test Generation (SBTG) tool requires a formal specification (in TPTL or STL), a blackbox model of the system under test (SUT), and a finite and bounded domain for the search parameters. When a vector $\mathbf{x}$ in the domain is generated by BASSO (or any other optimizer), the vector $\mathbf{x}$ is separated into a vector that represents static model parameters, e.g., initial conditions and/or other system parameters, and a vector that parameterizes signals. In the latter case, the (finite) vector that corresponds to the signals is interpolated by a user selected function (e.g., splines, piecewise linear, etc.) to produce an input signal for the SUT. Then, the SBTG executes the SUT using the provided parameters and signals, receives the SUT output trajectories, and computes the specification robustness. At that point, the robustness value is returned to the BASSO algorithm, and the process repeats until the BASSO algorithm has reached one of its terminating conditions or the maximum allowed testing budget. Further details on requirements driven SBTG can be found in [12, 32, 56].

In order to demonstrate BASSO variations on this application, we selected the F-16 benchmark (version 88ABW-2020-2188) [48] from the ARCH competition [31]. The F16 benchmark provides both a Matlab/Simulink and a Python version of a simplified F16 Ground Control Avoidance System (GCAS). The GCAS system uses piece-wise non-linear differential equations to perform autonomous maneuvers to avoid hitting the ground. The F16 benchmark instance defines three static

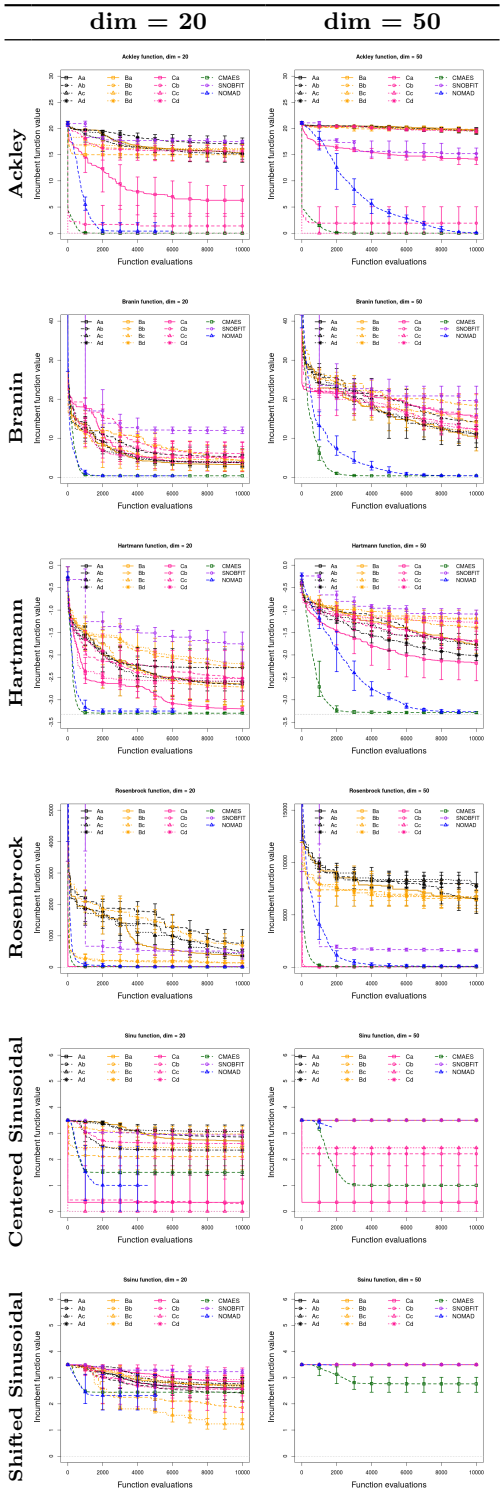



**Figure 4.7:** Incumbent function value for 6 test problems in dimensions 20 and 50

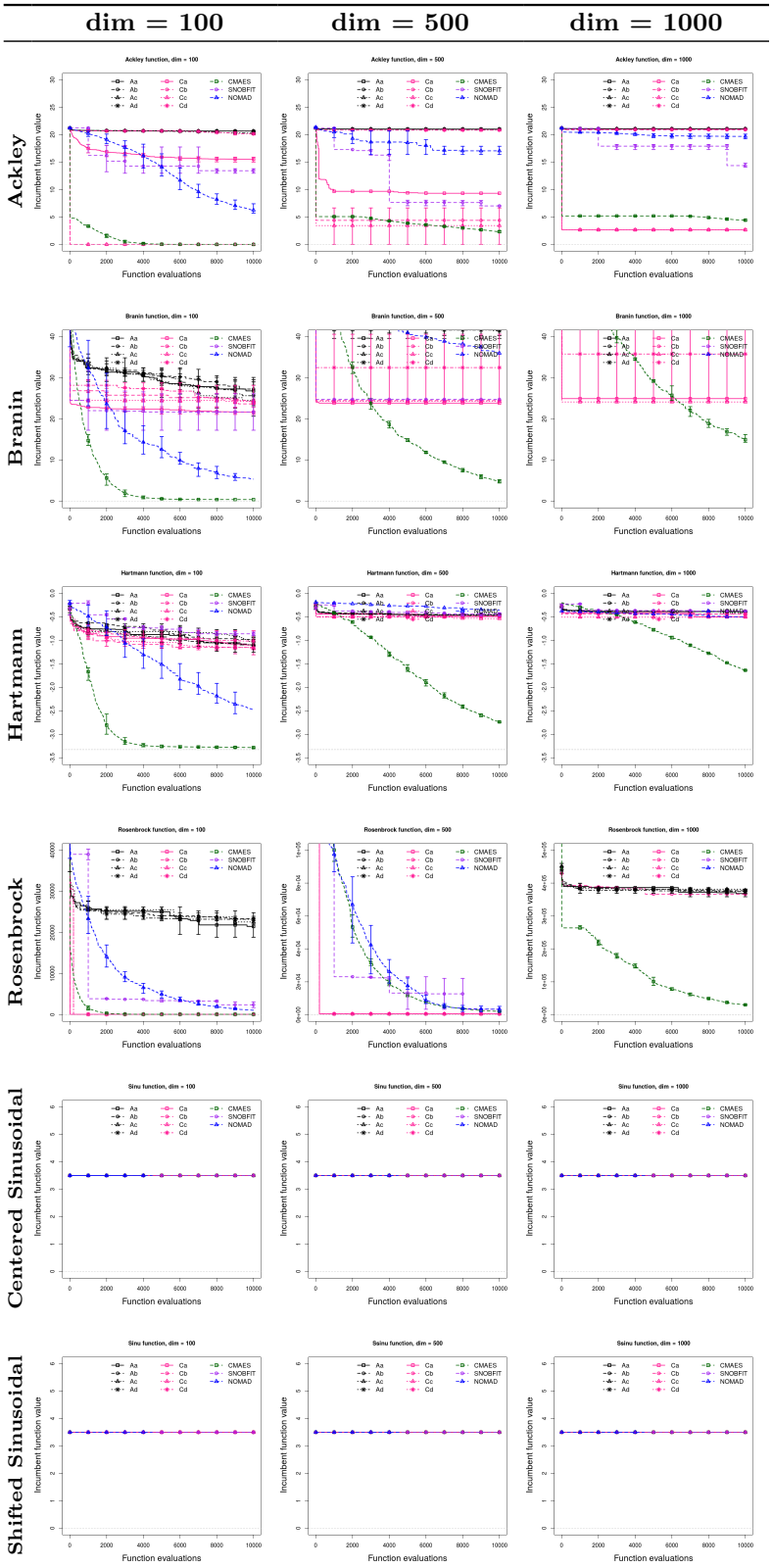



**Figure 4.8:** Incumbent function value for 6 test problems in dimensions 100, 500 and 1000

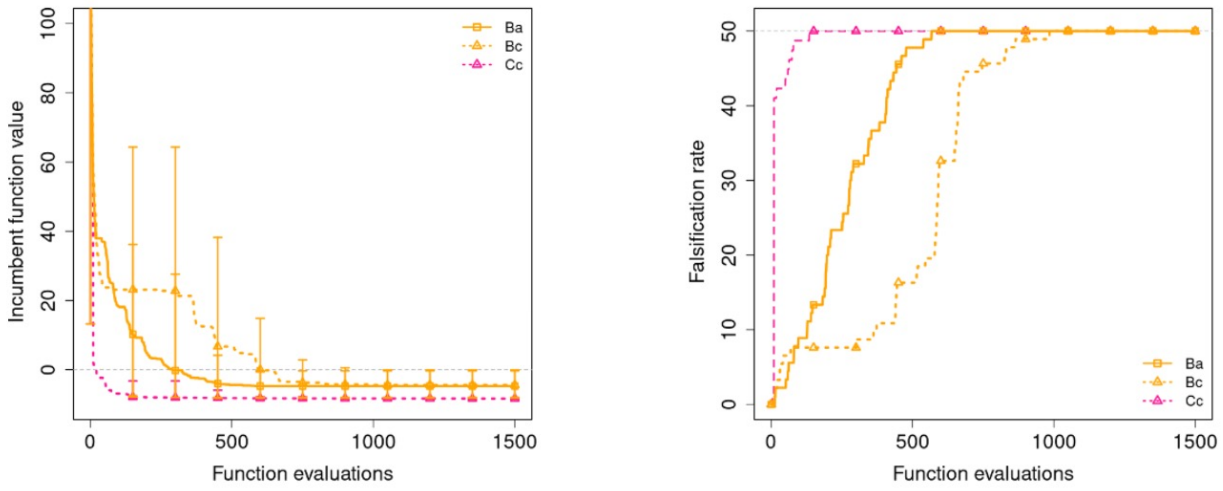
inputs $\phi$, $\theta$ and $\psi$, which are the initial roll, pitch and yaw angles of the aircraft, and there are no time-varying inputs. The three angles can range in the intervals $[0.2\pi, 0.2833\pi]$, $[-0.5\pi, -0.54\pi]$, and $[0.25\pi, 0.375\pi]$, respectively. These angle ranges constitute the domain of decision variables in this experiment. For this demonstration case study, we focus only on the altitude of the aircraft over the time horizon $[0, 15]$ after the GCAS has been activated. That is, the F-16 is initialized at an altitude and in a pose where a collision with the ground is possible. Hence, GCAS is activated to prevent the collision. The verification of a safety requirement simply states that the altitude should always be above 0 during the first 15 seconds after GCAS has been activated. If a value less than 0 is discovered in the optimization algorithm, then the safety requirement is falsified. This indicates that there may be a problem in the GCAS system in satisfying the safety requirement.

The F16 GCAS model is known to be falsifiable (i.e., the requirement is violated by at least one input) at altitudes 2300 ft and it appears to be non-falsifiable at altitude 2400 ft.

BASSO was applied to the F16 GCAS model at altitude 2330 ft, and the experiments were run on a AMD EPYC 9654 96-Core Processor 2.40 GHz. We ran BASSO on the F16 GCAS model for 50 replications, where each replication generated an initial point uniformly on the domain.

We selected BASSO variations Ba, Bc, and Cc to apply to this application because BASSO variation Cc performed the best on the data profiles, as shown in Section 4.4, and BASSO variations Ba and Bc performed well on low-dimensional problems.

Figure 4.9 shows the results of running BASSO Ba, Bc, and Cc on the F-16 benchmark where Figure 4.9a presents the average incumbent function value over 50 replications, with the maximum and minimum value. As illustrated, BASSO Cc outperforms Ba and Bc. We also note that the maximum and minimum for BASSO Ba and Bc indicate more variability over the 50 replications than for BASSO Cc. We speculate that the variability is due to the sensitivity of the Gaussian process variations (Ba and Bc) to the observed points. Figure 4.9b presents the falsification rate, that is, the number of replications out of 50 replications that the algorithm found a negative value. The three BASSO variations discovered a negative value on all 50 replications before the budget of 1500 function evaluations, however, BASSO Cc discovered a negative value much faster than Ba or Bc.




(a) (b)

**Figure 4.9:** BASSO variations Ba, Bc, and Cc applied to the cyberphysical systems falsification problem. (a) The average incumbent function value (over 50 replications), with the maximum and minimum value. (b) Falsification rate (number out of 50 that found a negative value).

Table 4.6 tabulates the results of BASSO variations Ba, Bc, and Cc for comparison with SOAR on the F-16 benchmark (from [31]).Table 4.6 presents the falsification rate (number of replications out of 50 found a negative value), and the mean and median of the number of function evaluations to first achieve a negative value (hitting time) over 50 replications.

The improved performance of BASSO compared to the state of the art SBTG approach (with SOAR) can be attributed to the partitioning aspect of BASSO. Several systems exhibit switching modes which result in high rates of variation of the output function. Approaches that automatically restrict sampling to smaller subregions enjoy increased homogeneity of the function thus achieving improved performance. The results seem to confirm this intuition.

These findings demonstrate BASSO's efficiency with fewer function evaluations. Furthermore, they underscore the benefit of using regularized quadratic regression as a surrogate model, especially given its good performance even when sample sizes are limited.

| Method | Falsification Rate (1500 FEs) | Hitting Time | |
|---|---|---|---|
| | | Mean | Median |
| BASSO Ba | 50 | 83.40 | 74.0 |
| BASSO Bc | 50 | 170.69 | 81.0 |
| BASSO Cc | 50 | 20.36 | 19.0 |
| SOAR | 10 | 123.0 | 125.0 |

**Table 4.6:** Falsification rate is the number of replications out of 50 that found a negative value within 1500 function evaluations. Hitting time is the number of function evaluations to first discover a negative value, with mean and median over 50 replications.

### *4.5 Discussion of BASSO and Quantum Computing*

Our numerical experiments reveal a gap between theory linearity results and its practical implementation, as we have not yet achieved the theoretical linearity. This gap stems from the difficulty of satisfying the two assumptions in the implementation. A first reason for this gap is result of trying to sample in thedisconnected improving level set. While BASSO's sampling distribution, $\tilde{p}$, is designed to help the sample jump between these disconnected promising regions, this mechanism alone appears insufficient to bridge them efficiently in practice. To better address this, our future work will explore clustering techniques to explicitly capture these separate regions. This also, can be viewed as a multi-start strategy. In the next chapter, to overcome the stalling found in our numerical experiment result, we integrate a stopping and restarting strategy known as the Hesitant Adaptive Search with Power-Law Improvement Distribution (HASPLID) [112] in our experiment. experiment to guide on when to restart the search.

An alternative explanation lies in the limitations of surrogate modeling. Models like Gaussian processes require many sample points to achieve reliable prediction accuracy, especially in high-dimensional spaces. For our goal, however, a perfect model is not necessary. We only require a surrogate that provides information about the subregions to locate the improving level set and performs at least as well as uniform sampling. This introduces a trade-off, that is a highly refined model might offer local precision but fail at global exploration, whereas the goal is a model that is

"good enough" to globally guide the search.

Whereas BASSO have been attempting to approximate the ideal performance of PAS that is scalable in dimension, quantum computing provides opportunity that this goal may be achievable in the future. Quantum computers inherently capture randomness, which is an important characteristic of stochastic adaptive search. In contrast to conventional computing, where, given an input $x$ an output $f(x)$ is returned, in a quantum computer, the output is a probability distribution with mean $f(x)$. Quantum computing appears to be able to provide a natural implementation of sampling distributions that focus on improving level sets. [18] introduces Grover's adaptive search that provides a quantum implementation of PAS and HAS. The Quantum Annealing Algorithm (QAA) uses quantum tunneling to implement a quantum version of simulated annealing. A brief discussion of Grover's Search and the Quantum Annealing Algorithm follow.

### *4.5.1 Grover's Search*

Grover's [42] and Shor's [98] algorithms are breakthroughs in quantum computing that have demonstrated its potential. Grover's algorithm, in particular, locates a single item in $N$ items using $O(\sqrt{N})$ iterations of a Grover rotation, composed of a selective phase shift and a Grover operator. For finding $m$ solutions out of $N$, the required number of Grover rotations is bounded by $\lceil(\pi/4)\sqrt{N/m}\rceil$. Grover's search algorithm has been applied to a discrete optimization problem, in particular, finding the minimum among an unsorted set of $N$ different objects by [30]. Their optimization implementation of Grover's search uses exponential searching [15] by randomly selecting a possible solution, using its functional evaluation as the threshold in the selective phase shift operator, and applying a certain number of Grover rotations for each optimum search iteration.

Grover's adaptive search (GAS) is a framework that combines PAS with Grover's search algorithm [18]. GAS uses an adaptive search strategy to dynamically change the number of Grover rotations and demonstrates how GAS can be an implementation of PAS. It can also be viewed as a quantum-computational implementation of HAS. Grover Adaptive Search finds the optimum value of an objective function by using the best-known value from the previous run as a threshold. Setting a threshold for a new iteration from an earlier iteration in GAS is analogous to finding an

improving level set in PAS. The adaptive oracle used in GAS recognizes all values above or below the current threshold (for maximize and minimize respectively), decreasing the size of the search space every iteration the threshold is updated, until an optimum is found. GAS then performs an amplitude amplification (inverting the amplitude of the current qubits state), which increases the probability of landing in the improving level set. If a better solution is found, the threshold is updated and a new iteration is started until the stopping criterion is met.

[11] refines GAS where the number of Grover rotations for each iteration is determined by maximizing the benefit-cost ratio of the expected gain to the number of rotations. [17] then addresses the application of Grover's algorithm with local search techniques where Grover's algorithm is used to locate the promising region that contains the global optimum solution, thereby combining the benefits of a multistart method with GAS. [64] provided a different strategy to determine the benefit-cost ratio with Bayesian update. GAS has been extended to continuous optimization problems by discretizing the space using fixed-point representation [81]. [72] recently proposes Quantum Adaptive Distribution Search (QuADS), an extension of GAS using an adaptive multivariate normal distribution, mimicking CMA-ES for the initial state in quantum search. Numerical experiments conducted on QuADS demonstrate promising outcomes, highlighting the potential of quantum computing in tackling continuous optimization problems.

### *4.5.2 Quantum Annealing Algorithm*

The quantum annealing algorithm (QAA) is an optimization algorithm that makes use of simulated quantum (rather than thermal) fluctuations and tunneling, thus providing a quantum-inspired version of simulated annealing. The property of quantum computing that allows the implementation of QAA is quantum tunneling. In sharp contrast to particles, quantum wave functions can tunnel through high potential barriers with significant probability, and this is formally known as quantum tunneling. The basic idea of QAA is to map the optimization problem to a physical system, such as a network of coupled qubits, and then find the state of the system that corresponds to the minimum energy by gradually evolving the system to its ground state, using a process similar to the annealing process in macroscopic physics. Intuitively, this can be viewed as global evolu-

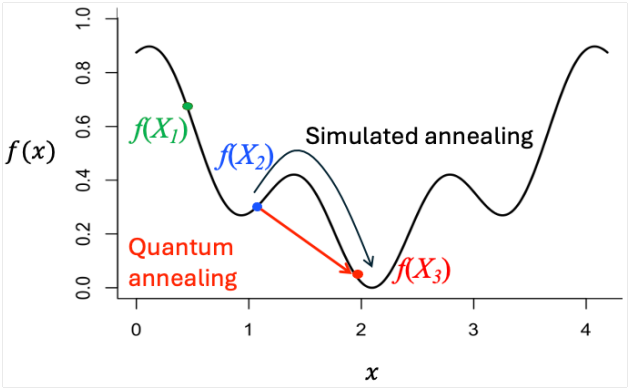



**Figure 4.10:** Comparison of the working principle of simulated annealing and QAA, as in Wang et al. (2024).

tion and superposition of quantum states, which is capable of acquiring global information of the objective function. In contrast, for SA, if the objective function contains several high potential barriers, it may fall into local optima, as the thermal transition depends on the height of the potential barriers. To solve the problem, quantum tunneling is used to go through potential barriers. Figure 4.10 illustrates quantum tunneling versus simulated annealing, as done in [104]. A comparison between QAA and SA [13, 25, 54, 90] suggests that QAA can be exponentially faster than SA in some cases. To apply quantum computing to optimization, an efficient encoding of variable and constraint spaces is necessary. This involves an intricate mapping of the optimization problem to the available number of qubits while preserving problem structure, which is challenging. Moreover, a challenge to quantum computing is the limited coherence durations and the need to maintain fragile quantum states for the duration of the execution. While quantum error correction offers a theoretical pathway to mitigate decoherence effects, its practical implementation at scale remains a substantial engineering and algorithmic hurdle. Consequently, the development of quantum optimization algorithms that are both theoretically and practically sound within the operational limitations of near-term quantum hardware constitutes a crucial area of ongoing development.

## *4.6 Conclusion*

This chapter proposes BASSO, a framework for adaptive random search that conceptualizes partitioning and surrogate modeling to solve black-box optimization problems. The finite time analysis of BASSO is provided as we derive an upper bound on the expected number of function evaluations until a specified $\varepsilon$ above the global minimum value is reached. It is shown that under certain conditions, the expected number of function evaluations is bounded by a linear function of the domain dimension, achieving scalability to problems in high dimensions. This conceptual framework balances exploration through an adaptive subregion probability to focus the search on subregions with improving function values, and with exploitation through a surrogate model that optimizes within a subregion. Several implementations of BASSO is applied on the test problems and we numerically conclude that the best variation uses a regularized quadratic regression as a surrogate, and a one-dimensional Gaussian process on the range distribution for the adaptive subregion probability. The combination of exploration through the adaptive subregion probability and exploitation via the quadratic regression has potential for scalability.We demonstrate the practical relevance of BASSO implementations by exploring their numerical performance on a real-world application of falsification of cyberphysical systems. Throughout this chapter the main goals of the dissertation is satisfied by incorporating a machine learning model (Gaussian process and regularized quadratic regression) into variations of BASSO implementations, providing finite-time analysis of BASSO and conditions for BASSO to achieve scalability in high dimension then we presents numerical experiments of BASSO variations for both synthetic optimization problems and application to the real-world problems.

# Chapter 5

# APPROACHES FOR HIGH-DIMENSIONAL BLACK-BOX OPTIMIZATION

Following the numerical results in Chapter 4 for Branching Adaptive Search Optimization (BASSO) framework, the BASSO numerical result in high-dimensional problem provides insight into the gap between BASSO's theoretical ideal performance and its practical performance when combined with machine learning techniques. This chapter extends the numerical experiment on the high-dimensional problem, combining the methods used in high-dimensional optimization included, partitioning, decomposition, and clustering.

## *5.1 Introduction and Background*

High-dimensional black-box optimization presents a significant challenge, as locating improving level sets becomes exponentially more difficult with increasing dimensions. Our previous work introduced the BASSO framework, a conceptual approach that balances global exploration, by adaptively focusing the search on promising subregions, with local exploitation, by using a surrogate model to optimize within them. While numerical experiments in the previous chapter showed that the desired linearity in performance scaling is not achieved in all BASSO variation, they offer important insights. We observed that a BASSO variant using regularized quadratic regression demonstrated superior scalability. Although this surrogate is less accurate at predicting the objective function than a Gaussian process, it appears to be better at striking a balance between exploration and exploitation, capturing trends within subregions. Building on these findings, this chapter aims to close the gap between current implementation and ideal theoretical performance. The experimental design in this chapter aims to test the methods currently used in high-dimensional black-box optimization to guide the development of a practical algorithm capable of scalability into

solve high-dimensional black-box optimization problems.

### *5.2 Methods for High-dimensional Black-box Optimization*

Our primary objective in this chapter is to develop a algorithm that can effectively scale to solve high-dimensional black-box optimization problems. Our approach is to incorporate methods used in high-dimensional optimization, each chosen to address challenges inherent in high-dimensional spaces. We use partitioning and clustering techniques to subdivide the problem domain into smaller subregions. This strategy, inspired by successful algorithms such as NOMAD, SNOBFIT, and CMA-ES [45, 51, 63], allows for more efficient sample management. The partitioning techniques used in SNOBFIT and NOMAD are described in Chapter 2. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a population-based evolutionary algorithm that samples new points from a multivariate Gaussian distribution and evaluates their objective function values. The points are sorted by function values and the distribution parameters (i.e., the mean vector and the covariance matrix) are updated based on the ranking of function values. In CMA-ES, the mean and covariance matrix of the top-performing points can be viewed as a statistical model of a promising cluster. Instead of performing a traditional local search, CMA-ES generates new candidate points by sampling from the multivariate normal distribution described by these parameters, effectively focusing the search within this promising area. The empirical success of the CMA-ES algorithm also inspires our use of clustering. In this chapter, we perform the BASSO algorithm on specific clusters, which are defined by bounding boxes placed over promising regions identified through a clustering technique.

As observed in our previous numerical studies, achieving a good prediction in the search space is helpful, but it is more important to have a mechanism that reliably focuses the global search on the set of improving levels. Partitioning and clustering addresses the challenge of function heterogeneity by allowing us to specify the promising subregions, then we can utilize scalable surrogate models, such as regularized quadratic regression for exploitation without the need to construct a single, highly complex, and computationally expensive model of the entire domain.

A third approach to address scalability of when optimize high-dimensional function is the de-

composition method that creates low-dimensional subproblems. This reduces the computational burden of constructing surrogate models on high-dimensional spaces.

While the previous chapter identified a promising approach, the BASSO-Cc variant, which combines an adaptive subregion probability from Gaussian process with a regularized quadratic regression as local surrogate model, it also revealed its weakness. We observed that this method, while is able to find "good enough" solutions in the early stages, frequently stalled as the optimization progressed. The search becomes trapped around local minima, suggesting the algorithm had fully exploited the information in a given region but struggled to transition back to a broader, more exploratory search. This failure to escape local optima significantly lowers the probability of discovering the true global optimum.

To overcome this stalling, we integrate a stopping and restarting strategy known as the Hesitant Adaptive Search with Power-Law Improvement Distribution (HASPLID) [112] in our experiment. This strategy provides a data-driven method for deciding when to terminate a current run and restart the search or terminate the entire algorithm. By modeling the optimization algorithm's progress as a stochastic process, HASPLID manages the tradeoff between investing further computational effort in the current search and the probability of achieving a better result by restarting. A key advantage of HASPLID is that its model is tractable to analyze and captures both the probability and magnitude of improvement in two easily estimated parameters. This strategy operates without requiring any specific knowledge or assumptions about the characteristics of the objective function, making HASPLID broadly applicable solution for black-box problems.

This chapter aims to improve high-dimensional optimization performance by combining three methods with BASSO, including clustering, restart (with HASPLID), and decomposition. The experiments evaluate the effectiveness of the eight algorithm variants, which are listed below.

**Algorithms without clustering**

1. Partitioning (BASSO Cc)

2. Partitioning with Restart (BASSO Cc-HASPLID)

3. Partitioning-Decomposition (BASSO Cc-Decomp)

4. Partitioning-Decomposition-Restart (BASSO Cc-Decomp-HASPLID)

**Algorithms with clustering**

5. Partitioning with Clustering (Cluster-BASSO Cc)

6. Partitioning with Clustering and Restart (Cluster-BASSO Cc-HASPLID)

7. Partitioning-Clustering-Decomposition (Cluster-BASSO Cc-Decomp)

8. Partitioning-Clustering-Decomposition-Restart (Cluster-BASSO Cc-Decomp-HASPLID)

The implementation details of each algorithm are as follows.

### *5.2.1 Partitioning*

This section describes two algorithms, BASSO Cc and Cluster-BASSO Cc, that use partitioning to break down a problem into smaller, more manageable subproblems.

The first algorithm, BASSO Cc, is the variation detailed in Chapter 4, Section 4.2.2. The second, Cluster-BASSO Cc, first divides the problem domain into smaller subregions, or ”clusters subregion,” using a clustering algorithm. It then applies the BASSO Cc method to each subregion individually. This approach is designed to focus the search on the most promising regions of the problem space.

**BASSO Cc**

BASSO Cc, is the variation detailed in Chapter 4, Section 4.2.2 that uses adaptive subregion probability from the Gaussian process on the range distribution and generate a point on the selected subregion by minimizing a regularized quadratic regression on the domain. The only difference in this implementation is the initialization use $n_0 = dim$ points. The branching strategy is also defined in section 4.2.2. The stopping criteria is the number of function evaluation that is set it to 10,000 for all test problems.

**Cluster-BASSO Cc**

**Step 0: Initialize** Set algorithm parameters $d$ is the dimension of the space $S$, $K(d)$ is the number of clusters $K(d)$ is set to 5 for numerical experiment. Initialize $n_0 > K(d)$ points in $S$ (uniformly, or Latin hypercube) and evaluate their function values. Let $x^*$ and $y^*$ be incumbent points and its corresponding incumbent function value. Initialize a cluster $C_1$ with $n_0$ points.

**Step 1: Multivariate Normal Sampling and create cluster subregion**

***Step 1.1: Multivariate Normal Sampling*** For each cluster $C_m$, let $x_m$ be the $n_m \times d$ matrix of $n_m$ sample points that belongs to cluster $C_m$. We write $X_m = (x_{ij})$, that $X_m$ is a matrix whose $i, j$ entry is $x_{ij}$, where $j$ is the dimension of $j$ coordinates of $i$ sample point, for $j = 1, \ldots, d$ and $i = 1, \ldots, n_m$. The d-vector, $\mu_m$ is the mean vector of cluster $C_m$. Let $\mu_m = (\bar{x}_1, \ldots \bar{x}_d)$,, wheere $\bar{x}_j$ is the average of column $j$ of matrix $x_m$. Set the covariance matrix for each cluster $C_m$ for $m = 1, \ldots, , K(d)$, to be $S_m$, where $S_m$ is the $d \times d$ matrix with each entry,

$$s_{jk} = s_{kj} = \frac{1}{n_m - 1} \Sigma_{i=1}^{n_m} (x_{ij} - \bar{x_j})(x_{ik} - \bar{x_k}), \quad j,k = 1, \ldots, d$$

For each cluster $C_m$, sample $n_{\text{cluster}=10}$ new points using a multivariate normal distribution with mean vector $\mu_m$ and covariance matrix $K_m$. Add the new points to cluster $C_m$.

***Step 1.2: Create cluster subregion*** For each cluster $m$, create a cluster subregion $\sigma_m$ by finding a lower bound and upper bound on each dimension over the points in that cluster. Then perform optimization (BASSO Cc) with HASPLID within each subregion $\sigma_m$. Also, set $\tilde{y}_m^*$ to keep track of the incumbent function value observed in that cluster subregion $\sigma_m$.

**Step 2: Branch and update adaptive subregion probabilities within each cluster subregion**

***Step 2.1: Branch cluster subregion***: For each cluster subregion, branch the cluster subregion. Branch a collection of subregions in $\Sigma_m^C$ according to a branching strategy. Update the new number of subregions $l$.

***Step 2.2: Update adaptive subregion probabilities*** For each cluster subregion, update adaptive subregion probabilities $\tilde{p}$, $\tilde{p}_i(\tilde{y}_{k+1}^*)$ for $i = 1, \ldots, l$.

**Step 3: Choose a subregion using $\tilde{p}_m$ and sample a point**

For each cluster subregion, choose a subregion $\sigma_m^i$ from $\Sigma_m^C$ using the adaptive subregion probabilities $\tilde{p}_{i,m}(\tilde{y}_m^*)$ for $i = 1,\ldots,l$. Generate a point $X$ on the selected subregion $\sigma_m^i$, either uniformly or using surrogate modeling, and evaluate $f(X)$. Update the best objective function value $y_i^*$ in the selected subregion $\sigma_m^i$. If the new objective function value $f(X)$ is less than the incumbent value within cluster subregion $\tilde{y}_m^*$, update the incumbent function value. If not repeat this step until the number of samples in this step reach $n_{opt} = 50$

**Step 4: Select sample points and recluster**

Select sample points and recluster. To select points for reclustering, we select sample points that have their function value better than a $\delta = 50\%$ quantile of the function value observed over the whole run. Then, for each dimension $j = 1,\ldots,d$, we select the points that are separated by some threshold. Then recluster those selected point into $K(d) = 5$ clusters.

**Step 5: Check stopping criterion** If the stopping criterion is met stop and report the incumbent function value, otherwise go to step 1.

### *5.2.2 Multi-start*

This section introduces two algorithms that integrate HASPLID to model the range behavior of an optimization process. HASPLID provides criteria for stopping a single run and restarting it to address stalling, which occurs when an algorithm gets stuck near a local optimum and struggles to find an improving level set. The goal of this mechanism is to balance exploration of new areas with the exploitation of regions that have already been explored. The two algorithms are BASSO Cc-HASPLID and Cluster-BASSO Cc-HASPLID. In BASSO Cc-HASPLID, the restart process involves discarding all previous subregions while retaining the top 50% of observed points, ranked by their function values. Similarly, the Cluster-BASSO Cc-HASPLID applies this same restart procedure to the instance of BASSO Cc running within each individual cluster.

**BASSO Cc-HASPLID**

BASSO Cc-HASPLID performs BASSO Cc variation as detailed in Section 4.2.2 The difference

in this implementation is the step 0: initialization where we set the three HASPLID parameters: $\varepsilon$ represents the target proportion of the domain;$\delta$ represents the failure tolerance parameter that we want to reach the target with probability at least $1-\delta$; the improvement parameter $\alpha$. Set $p_{fail} = 1, R = 0, \zeta = 1$. Set the HASPLID parameters as suggested in [112].

To initialize BASSO Cc with HASPLID, Increment $R$, set $j_R = 1, k_R = 1, n_{restart} = n_{LARGE}$

**Step 0: Initialize.** Set $\sigma_1 = S$, $\Sigma_0^C = \{\sigma_1\}$, $m_0 = 1$, and iteration counter $k = 0$. Sample a point $X$ from the uniform distribution over $S$. Evaluate $f(X)$ and set $y_1^* = f(X)$. Also, set $\tilde{y}_0^* = f(X)$ to keep track of the incumbent function value observed. Set $\tilde{p}_1(\tilde{y}_0^*) = 1$. Set the three HASPLID parameters: $\varepsilon = (0.01)^d$ represents the target proportion of the domain;$\delta = 0.001$ represents the failure tolerance parameter that we want to reach the target with probability at least $1-\delta$; the improvement parameter $\alpha = 1/d$. Set $p_{fail} = 1, R = 0, \zeta = 1$. To initialize BASSO Cc with HASPLID, Set $j_R = 1, k_R = 1, n_{restart} = n_{LARGE} = 10{,}000$

**Step 1: Sample a new point.** Choose a subregion $\sigma_i$ from $\Sigma_k^C$ using the adaptive subregion probabilities $\tilde{p}_i(\tilde{y}_k^*)$ for $i = 1, \ldots, m_k$. Generate a point $X$ on the selected subregion either uniformly or using surrogate modeling, and evaluate $f(X)$. Update the best objective function value $y_i^*$ in the selected subregion $\sigma_i$.

If the new objective function value $f(X)$ is less than the incumbent value $\tilde{y}_k^*$, update the incumbent function value $\tilde{y}_{k+1}^* = f(X)$. Update $n_{restart}$, increment $k_R$ and $j_R$. Then, decide if we should go to the next step or stop the run If $n_{restart}$ ¿ $k_R$ go to step 2, otherwise go to Step 4.

**Step 2: Branch subregions.** Branch a collection of subregions in $\Sigma_k^C$ according to a branching strategy. Update the new number of subregions $m_{k+1}$, and update the collection of subregions, $\Sigma_{k+1}^C = \{\sigma_1, \ldots, \sigma_{m_{k+1}}\}$.

**Step 3: Update adaptive subregion probabilities.** Update $\tilde{p}_i(\tilde{y}_{k+1}^*)$ for $i = 1, \ldots, m_{k+1}$.

**Step 4: Evaluate stopping criterion.** Solve for the estimate of HASPLID parameter ratio $\zeta$ and update $p_{fail}$. If $p_{fail} < \delta$ increment $R$, reset $j_R = 1$ and $k_R = 1$, set $\sigma_1 = S$, $\Sigma_0^C = \{\sigma_1\}$, $m_0 = 1$. Order all samples by their observed function value, keep the best 50% the sample and restart the BASSO run at step 1. Increment the iteration counter $k \leftarrow k+1$. If $p_{fail} > \delta$ , stop the BASSO with HASPLID and output the $\tilde{y}^*$.

**Cluster-BASSO Cc-HASPLID**

The difference of Cluster-BASSO Cc-HASPLID to the Cluster-BASSO Cc is we define the HASPLID run as Step 2 and 3 of the Cluster-BASSO Cc algorithm. The Cluster-BASSO Cc perform step 2 and 3, for each subregion cluster, for a fixed number of function evaluations $n_{opt} = 50$. While Cluster-BASSO Cc-HASPLID use HASPLID to decide when to stop the run in step 2 and 3 and progress to step 4.

### *5.2.3 Decomposition*

This section introduces two algorithms that integrate decomposition to BASSO with subspace decomposition. One of the inherent limitations of many surrogate modelling methods is that computational cost becomes intractable for problems in high dimensions. We decompose the high dimensional problem into many low-dimensional problems to enable surrogate modelling within practical limits. The combination of decomposition and partitioning addresses heterogeneity of the black-box function, and aids in focusing the search on promising regions. The two algorithms are BASSO Cc-Decomp and Cluster-BASSO Cc-Decomp. BASSO Cc-Decomp implementation. The Cluster-BASSO Cc-Decomp algorithm implementation do BASSO CC-Decomp on each cluster subregion.

**BASSO Cc - Decomp**

**Step 0: Initialize Decomposition** Set decomposition parameters; $d$ is the dimension of the space $S$, $P$ is the number of sub-spaces, and $d^p$ ($d^p = 5, P = d/5$) is the number of sub-space dimensions $p$, $p = 1,\ldots,P$. Create the set of sub-space dimensions for sub-space $p$ by sampling from the index set $\{1,\ldots,d\}$ without replacement. Let $\mathbb{X}^p$ be the sub-space for sub-space $p$, such that $S = \mathbb{X}^1 \times \cdots \times \mathbb{X}^P$.

Initialize $n_0$ points in $S$ (uniformly, or Latin hypercube). Rank order the $n_0$ points by their function evaluations, and create a set of points with their function values.

Let $\mathbb{Q}_0^*$ be the set of $n_0$ points and their function evaluations, $\{x_j, f(x_j)\}$ for $j = 1,\ldots,n_0$. Let

$\tilde{y}_0^* = min_{j=1,\ldots,n_0} f(x_j)$ to keep track of the incumbent function value observed.

Set BASSObudget, for the number of function evaluations to perform for each sub-space in an iteration.

Initialize BASSO for each sub-space $p, p = 1,\ldots,P$. Set $\Sigma_{p,0}^C = \{\mathbb{X}^p\}$, and $\sigma_{p,1} = \mathbb{X}^p$, and $m_{p,0} = 1$ ($m_{p,k}$ is the number of subregions of sub-space $p$ on iteration $k$). Set iteration counter $k = 1$. Set $\tilde{p}_{p,1}(\tilde{y}_0^*) = 1$.

**Step 1: Perform BASSO for each subspace**

**Step 1.1: Sample $N$ new points using belief vectors (adaptive probability $\tilde{p}$)**

For $n = 1,\ldots,N$, do:

**Step 1.1.1: Choose $N$ subregions using belief vectors (adaptive probability $\tilde{p}$** For each sub-space $p = 1,\ldots,P$, choose a subregion from $\Sigma_{p,k}^C$, the set of subregions of sub-space $p$ using $\tilde{p}_{p,i}(\tilde{y}_{k-1}^*)$, for subregions $i$ of sub-space $p$. Call this subregion $\sigma_{p,j}$.

**Step 1.1.2: Sample a point in the chosen subregion uniformly** For each sub-space $p = 1,\ldots,P$, sample a point uniformly distributed on the selected subregion $\sigma_{p,i}$. Construct a full dimensional point by concatenating the points of each of the sub-spaces and evaluate its objective function value. This yields $N$ full dimensional points. Update the set $\mathbb{Q}_k^*$ with the $N$ points and their function evaluations $\{x_j, f(x_j)\}$ for $j = 1,\ldots,N$.

**Step 1.2: Construct surrogate models and generate a new point**

For each sub-space $p = 1,\ldots,P$, choose a subregion from $\Sigma_{p,k}^C$, the set of subregions of sub-space $p$ using $\tilde{p}_{p,k}(\tilde{y}_{k-1}^*)$. Call this subregion $\sigma_{p,i}$.

**Step 1.2.1: Construct a surrogate model in the chosen subregions** Construct a surrogate model in the chosen subregion $\sigma_{p,i}$ using the points that have been sampled and their associated function values (from $Q_k^*$).

**Step 1.2.2: Generate a point using surrogate modeling** Generate a sample point $X_{p,i}^*$ in the chosen subregion $\sigma_{p,i}$ that minimizes the regularized quadratic regression over $\sigma_{p,i}$.

Construct a full dimensional point by concatenating the generated point of each of the sub-spaces and evaluate its objective function value. Add this full dimensional point to $\mathbb{Q}_k^*$. **Step 1.3: Branch and update current subregions for each sub-space p Step 1.4:Update adaptive**

**subregion probability** $\tilde{p}_{p,i}$

**Step 2: Evaluate the stopping criterion**

### *5.2.4 Partitioning Decomposition and Multi-start*

This section introduces two algorithms that integrate HASPLID to model the range behavior of optimization processes of BASSO Cc-Decomp and a modified version of Cluster-BASSO Cc-Decomp. The two algorithms are BASSO Cc-Decomp-HASPLID and Cluster-BASSO Cc-Decomp-HASPLID. For both algorithms, we calculate each HASPLID run will restart and randomly reassign each dimension to a new sub-space.

## *5.3 Numerical Experiments*

In this section, we perform the numerical experiments designed to explore and assess methods that can effectively address the challenges of high-dimensional black-box optimization. The experiments were structured to test the impact of three primary features—Clustering, Decomposition, and HASPLID—when integrated into algorithms for black-box optimization problem. This combination resulted in the following eight algorithms: We numerically tested these eight algorithms on a six test problems, with the dimensions 20, 50, 100, 500, and 1000. We performed 10 independent replications for each problem instance, with each replication starting from a common random seed. We also implemented three publicly available optimization algorithms Covariance matrix adaptation evolution strategy (CMA-ES) [45], Stable Noisy Optimization by Branch and FIT (SNOBFIT) [51], and The Non-linear Optimization with Mesh Adaptive Direct Search algorithm (NOMAD) [63] . We chose the first three algorithms because they ranked highly with good performance in [66].

### *5.3.1 Result*

This section analyzes the performance of the proposed algorithms, first by comparing them against each other and then by benchmarking the top-performing method against established algorithms. Comparison of Proposed Algorithms The incumbent function value found as performance measure of the eight proposed algorithms as shown in Figures 5.1–5.5, the results show that BASSO Cc-HASPLID perform the best in most problems, closely tracked by Cluster-BASSO Cc-HASPLID. Algorithms integrated with the HASPLID restart mechanism consistently outperform their counterparts without it. It is particularly evident when comparing the results of BASSO Cc with BASSO Cc-HASPLID. While the HASPLID integration yields better overall results, it also introduces greater variance in the observed function values during the search. Conversely, methods that relied on decomposition generally exhibited poorer performance in most test cases.

Given its better performance, we compare BASSO Cc-HASPLID the other three optimization algorithms: CMAES, SNOBFIT, and NOMAD. The results are presented in Figures 5.6–5.10. Compared to the benchmark algorithms, BASSO Cc-HASPLID demonstrates a distinct advantage in identifying good solutions early in the optimization process compare to other three algorithms. The HASPLID mechanism provides noticeable improvement in later iterations by helping the algorithm escape local optima. However, it is noted that this mechanism does not completely resolve stalling behavior on all test functions.

## ***5.4 Conclusion***

In this chapter, we extend the numerical experiments from Chapter 4 to high-dimensional problems by integrating high-dimensional techniques including partitioning, clustering, restart and decomposition. Motivated by insights from Chapter 4 on the gap between theoretical and practical performance, we introduced the HASPLID mechanism to guide restart of BASSO, in order to prevent search from over-exploitation of promising regions. The numerical experiments offer encouraging results. Algorithms incorporating HASPLID consistently outperformed their counterparts. Furthermore, the strategy of partitioning the domain using clustering proved effective, with its per-

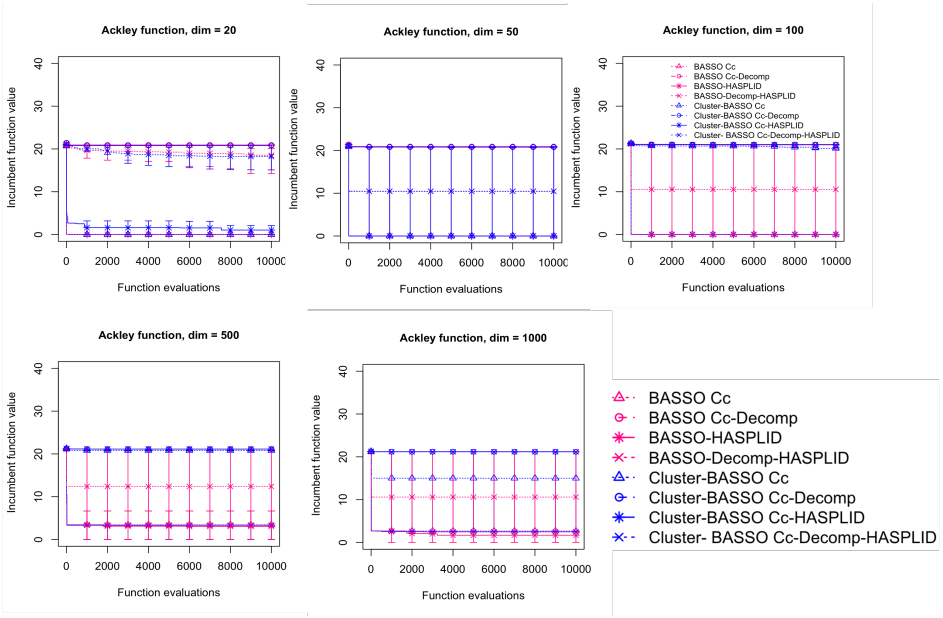



**Figure 5.1:** Incumbent values for Ackley function at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

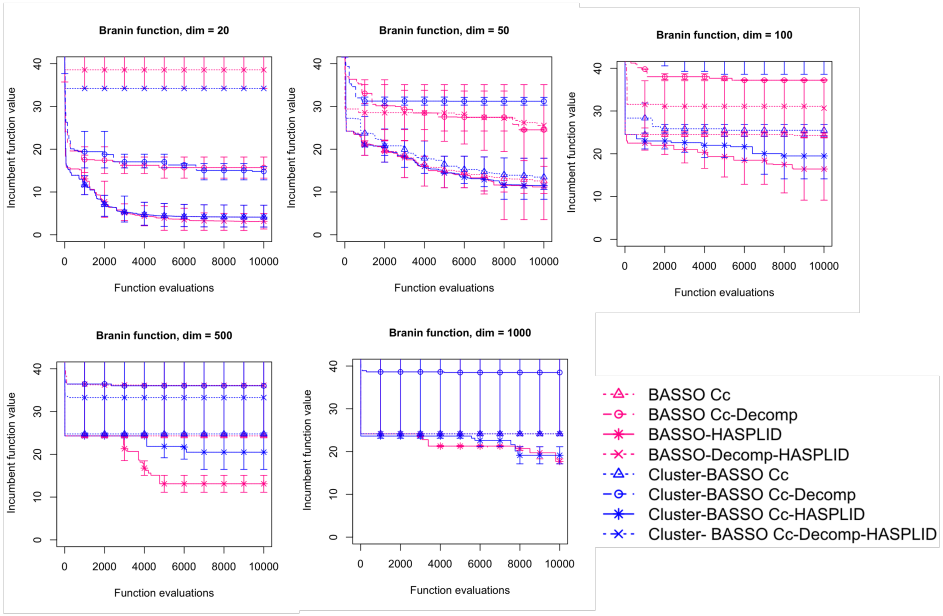



**Figure 5.2:** Incumbent values for Branin function at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

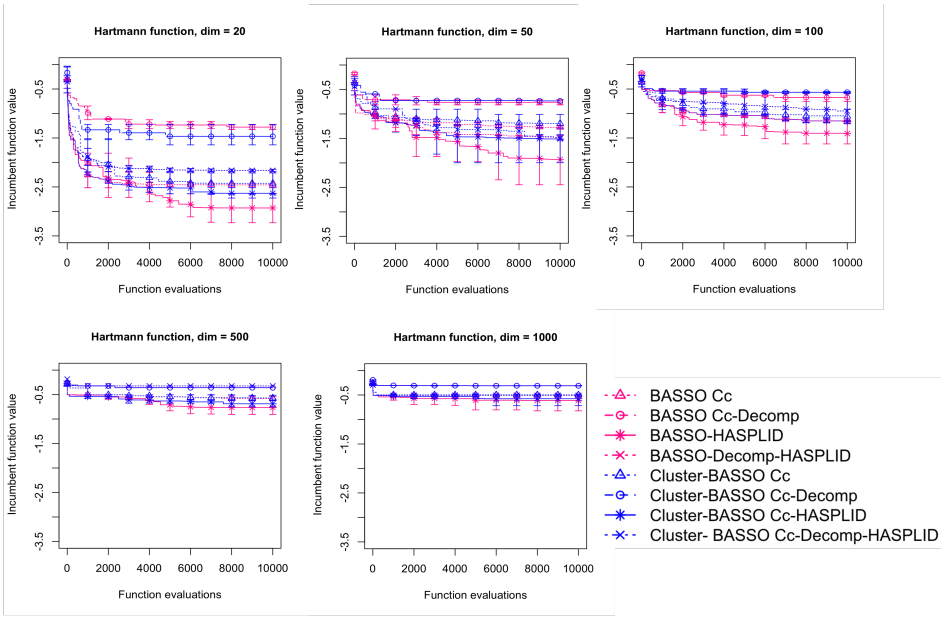



**Figure 5.3:** Incumbent values for Hartmann function at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

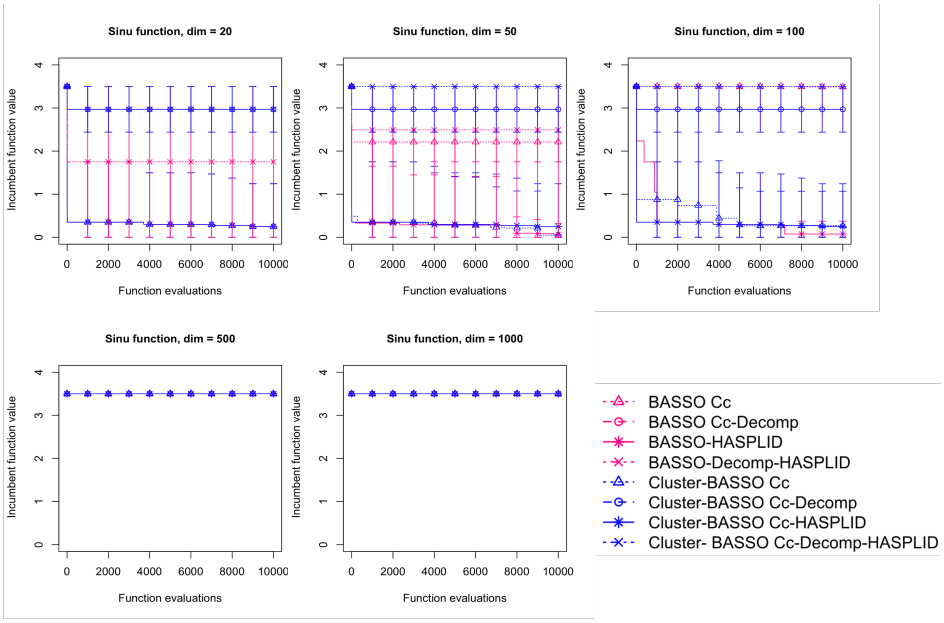



**Figure 5.4:** Incumbent values for Centerted Sinusoidal function at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

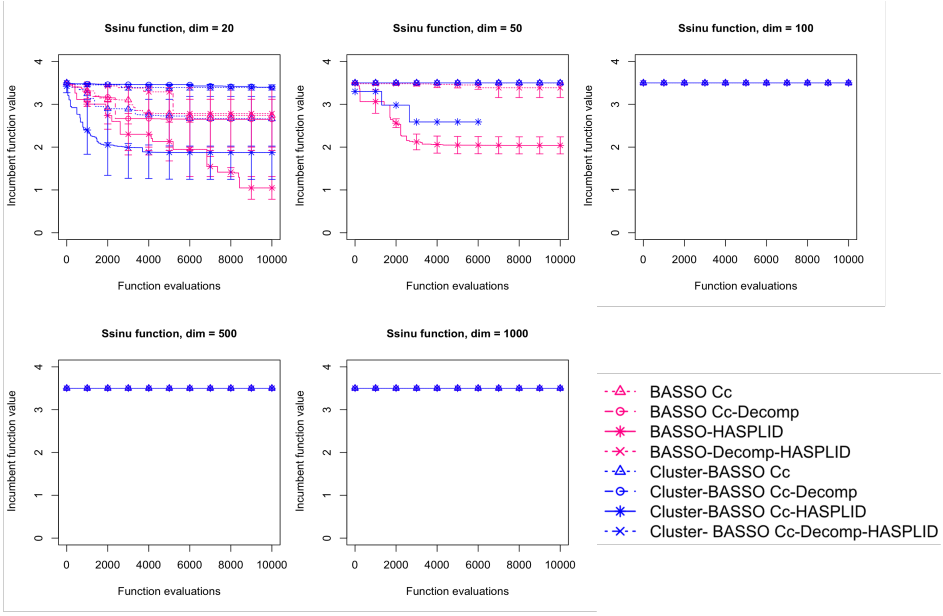



**Figure 5.5:** Incumbent values for Shifted Sinusoidal function at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

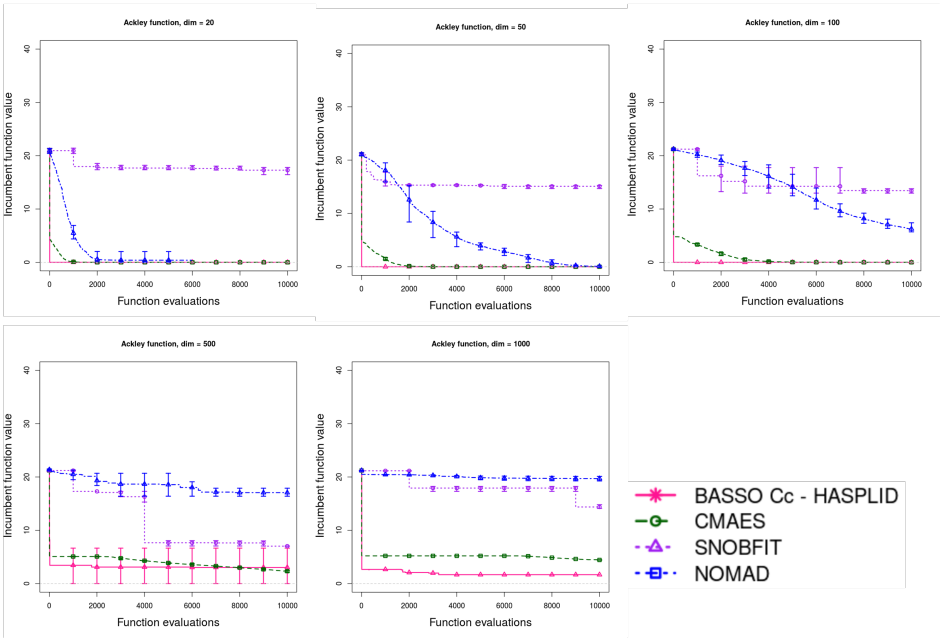



**Figure 5.6:** Incumbent values for Ackley function from BASSO Cc - HASPLID, CMAES SNOBFIT and NOMAD at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

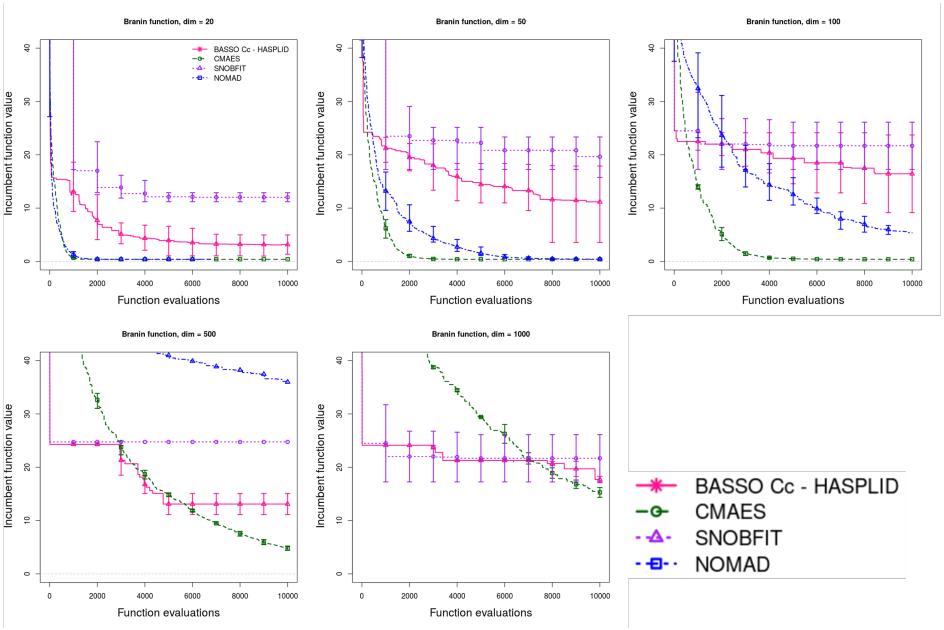



**Figure 5.7:** Incumbent values for Branin function from BASSO Cc - HASPLID, CMAES, SNOBFIT, and NOMAD at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

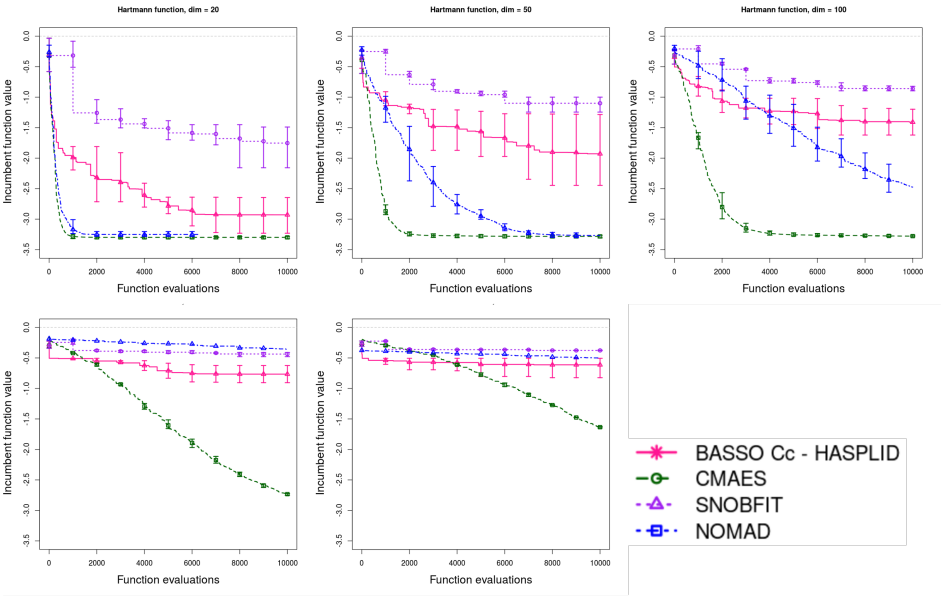



**Figure 5.8:** Incumbent values for Hartmann function from BASSO Cc - HASPLID, CMAES, SNOBFIT, and NOMAD at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

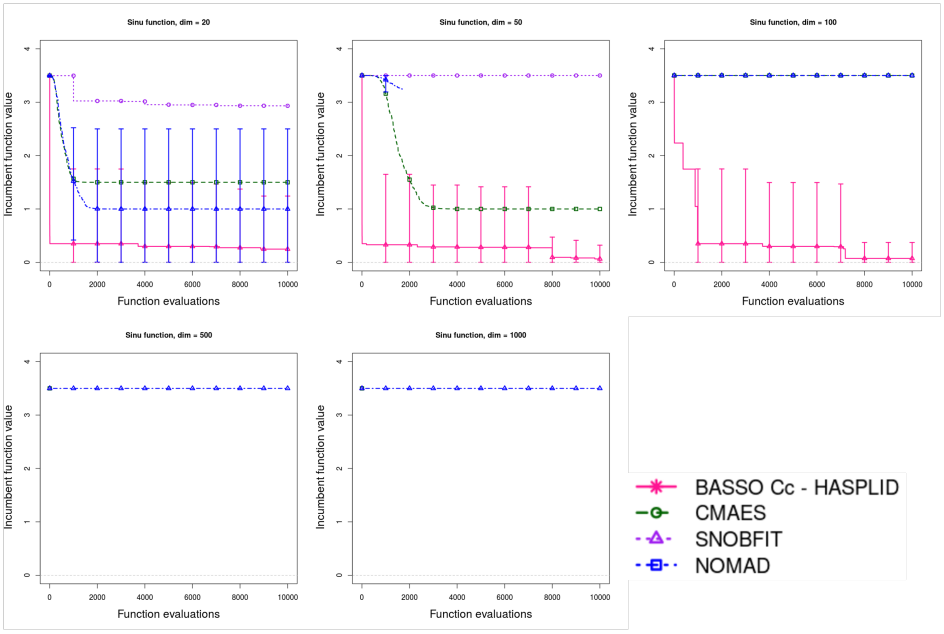



**Figure 5.9:** Incumbent values for Centered Sinusoidal function from BASSO Cc - HASPLID, CMAES, SNOBFIT, and NOMAD at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

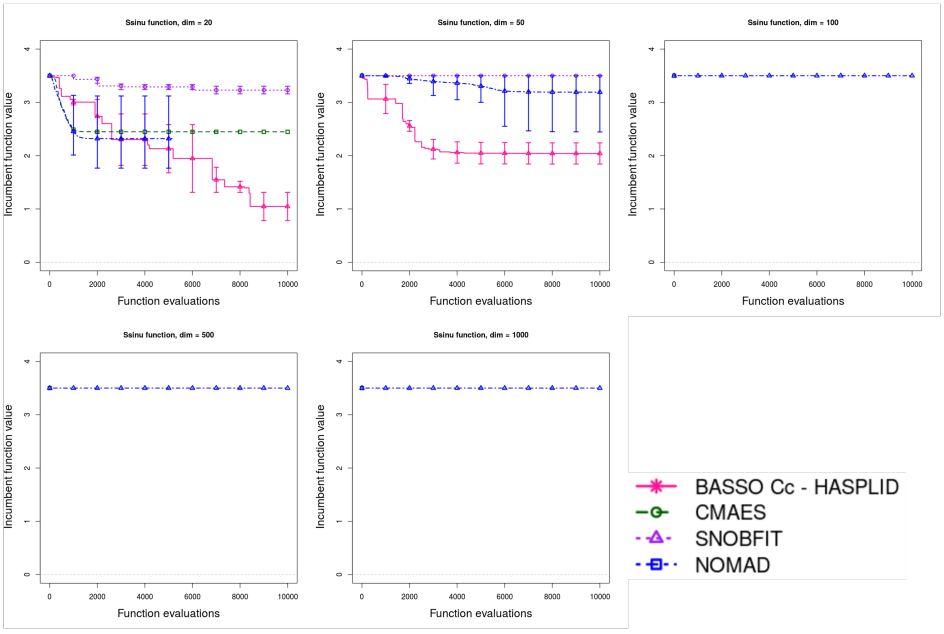



**Figure 5.10:** Incumbent values for Shfited Sinusoidal function from BASSO Cc - HASPLID, CMAES, SNOBFIT, and NOMAD at dimension 20, 50, 100, 500 and 1000, averaged over 10 simulation rounds across 10,000 evaluations, with maximum and minimum value

formance closely tracking that of the best-performing algorithm, BASSO Cc-HASPLID. However, a limitation was identified in the experimental design as the number of clusters was fixed at 5 clusters throughout all experiments. This fixed number of clusters may be insufficient to capture the complex, often disconnected improving level sets in higher-dimensional spaces. Therefore, future work should focus a clustering strategy where the number of clusters is a function of dimensions.

# Chapter 6

# SINGLE OBSERVATION ADAPTIVE SEARCH (SOSA) FOR CONTINUOUS STOCHASTIC OPTIMIZATION WITH MACHINE LEARNING

This part of the research proposes the use of machine learning in a stochastic adaptive search algorithm for efficient noisy black-box optimization. This section propose novel variants of the Single Observation Search Algorithm (SOSA) by integrating machine learning techniques. The baseline SOSA estimates a sample point's objective value via neighborhood averaging and employs an adaptive stochastic search that gravitates toward the best current estimate. The contribution is to enhance both components: the proposed method apply generalization principles to improve value estimation and use reinforcement learning to guide the search algorithm more effectively. Numerical experiments demonstrate that these machine learning enhancements yield significant performance gains for the SOSA framework. In this chapter, the research question is whether the proposed algorithms that adopt a quadratic regression in the objective function value estimation step and reinforcement learning technique improve the performance of the algorithm, and maintain convergence results.

While the previous chapter focuses on deterministic functions, this chapter proposes an enhancement to the stochastic adaptive search algorithm with the presence of noise, extending the Single Observation Search Algorithm (SOSA) [57, 58] by incorporating insights from machine learning. This chapter focuses on how to balance exploration, exploitation with estimation by utilizing the technique and insight from machine learning. The neighborhood averaging technique for estimation of SOSA is replaced with a quadratic regression, extending the concept of basis expansion. This study contributes to the interplay of optimization and machine learning, providing a convergence analysis for a function value estimate from quadratic regression, which accounts

for dependent samples acquired from an adaptive search within a single-observation scheme. The search strategy is also enhanced by incorporating optimistic sampling, a concept borrowed from reinforcement learning, which yields encouraging numerical results and improved performance.

### *6.1 Introduction and Background*

The single observation search algorithm (SOSA) is a class of stochastic adaptive search algorithms for noisy black-box optimization that requires exactly one function evaluation (e.g., simulation) per sampled design point [57, 58]. The algorithm estimates the objective function of a sampled design point by averaging the single observations of the function values at different nearby design points. SOSA was initially proposed for continuous problems in [57], and extended to problems with integer and/or real-valued decision variables in [58]. Under mild regularity conditions, the optimal value estimate from the SOSA framework converges to the true optimal value with probability one for both continuous and mixed integer problems. Numerical experiments in [57, 58] compare the single observation framework of SOSA to a sample average framework with repeated function evaluations using two samplers, Improving Hit-and-Run (IHR) [111] and AP sampler [2]. Both IHR and AP samplers satisfy the mild regularity conditions for SOSA to ensure convergence to the true optimal value. The numerical results provide support to the efficiency of the single observation framework on noisy black-box optimization problems.

Independently created for noisy black-box optimization, the structure of SOSA inadvertently resembles the $k$-nearest neighbor algorithm in machine learning, making SOSA a machine learning algorithm for noisy black-box optimization. However, SOSA employs averaging as its prediction function which is fairly primitive and can be improved.

In this study, we propose to extend SOSA with a basis expansion, using quadratic regression to estimate the objective. The concept of basis expansion in this enhanced SOSA is to expand the basis from sample averaging to the polynomial basis as an attempt to capture a non-linear relationship of the decision variables and the objective function. Additionally, the numerical experiments with BASSO variations also demonstrates that quadratic regression proves to be an efficient model for capturing local behavior, empirically showing the benefit of the quadratic regression as local

surrogate model.

### *6.2 SOSA with Machine Learning*

We consider the stochastic optimization problem

$$\min_{x \in S} \quad f(x) \tag{6.1}$$

where the feasible region is $S \subset \Re^d$. We assume the feasible region $S$ is compact. We address the case where the objective function $f(x)$ cannot be evaluated directly. Instead, only a noisy evaluation is available. The performance at a design point $x \in S$ is given by $g : S \times \Omega \to \Re$, where $U$ is a random element defined on a probability space $(\Omega, \mathscr{A}, \mathbb{P})$. The true objective function is the expected value of these noisy evaluations:

$$f(x) = \mathbb{E}\left[g(x,U)\right]. \tag{6.2}$$

We assume the expected value is finite for all $x \in S$ and $U \in \Omega$. A minimum value $f^*$ is guaranteed to exist on the compact set $S$. Let $\mathscr{X}^* = \text{Arg}\min_{x \in S} f(x)$ denote the set of optimal solutions. For any point $x \in S$, we define B(x,r) as a ball of radius $r$ centered at $x$. The algorithm will use a sequence of radii, $\{r_t\}_{t=1}^{\infty}$, that decreases and converges to zero.

The original SOSA algorithm sequentially samples design points, requiring exactly one function evaluation per point. Let $(\mathscr{X}_t, \mathscr{Y}_t)$ be the sets of paired sample points and their corresponding function evaluations up to iteration $t$ obtained in the course of SOSA. For any point $x \in S$, the objective function estimate, $\hat{f}_t(x)$, is calculated as the sample average of evaluations $y_t$ for all historical points $x_t$ that fall within the ball $B(x, r_t)$. The algorithm's estimate of the optimal value is then the minimum of these function estimates over all sampled points.

The original SOSA framework has two primary shortcomings that this paper aims to address. First, its estimation method—sample averaging—does not leverage specific characteristics of the objective function. We will make an approximation locally by a quadratic regression to improve

the estimation of the noisy function. This allows for a more refined estimation technique using quadratic regression. Second, the original sampling strategy using IHR makes use of the best point found so far, which can cause the algorithm to become trapped in the neighborhood of a local optimum. Therefore, a more robust exploration scheme is needed to mitigate this risk.

To address the first shortcoming, we introduce a quadratic estimation method. At iteration $t$, we define the neighborhood points of $x$ as the set of all previously sampled points that fall into the ball $B(x, r_t)$. Let $L_t(x)$ be the number of these points, which we call the contribution to the estimate at $x$. If this contribution $L_t(x)$ exceeds a predefined threshold $l^*$, say $l^* = d^2$, then we estimate the function value by solving the following least-squares problem:

**Problem** $\mathcal{Q}(x,t)$

$$\min_{A,b,c} \quad \sum_{\{i: x_i \in B(x,r_i),\ i \le t\}} \Big[ y_i - (x_i^\top A x_i + b^\top x_i + c) \Big]^2 \tag{6.3}$$

where $(x_i, y_i)$, $i = 1, \ldots, t$, are the sample points and their function observations in the ball around $x$ up to iteration $t$, $A$ is a $d \times d$ matrix, $b$ is a $d$-vector, and $c$ is a scalar. Solving this problem yields the coefficients $A^*_{x,t}$, $b^*_{x,t}$ and $c^*_{x,t}$ of the local quadratic model. We propose an enhanced algorithm, which we call SOSA with Quadratic Regression (SOSA-Q), that replaces the sample average used in the original SOSA with a quadratic model to estimate the objective function value.

The sampling at each iteration uses Hit-and-Run that is initiated from the so-far best point in the original SOSA and the SOSA-Q. We first identify the point with the best estimated objective function value. A random direction is then generated, and a new point is sampled from the line segment that passes through this best point along that direction. From the multi-armed bandit perspective, this greedy approach can cause the algorithm to get trapped at a local optimal point. To improve the sampling strategy, we propose an optimistic approach that balances exploration and exploitation, drawing an analogy to the acquisition functions used in Bayesian optimization [34, 106]. Instead of initiating Hit-and-Run from the point with the best-estimated objective value, we bias Hit-and-Run towards the point with the best lower confidence bound, following the principle from the reinforcement learning literature [7]. However, assessing this bound for every sampled point at each iteration is computationally impractical. We therefore introduce a heuristic: we first identify

a candidate set of the $K$ best-performing points based on their estimated function values. Then, we calculate the lower confidence bound for only these $K$ points and select the one with the best bound as a point for Hit-and-Run sampler. We called the variations with sampling with Hit-and Run that biased toward the best lower confidence bound, SOSA with Optimistic Bias Sampling (SOSA-O) and SOSA-Q-O SOSA with Quadratic Estimation and Optimistic Bias Sampling (SOSA-Q-O).

In this study, we focus on comparing the estimation method used in SOSA framework. The first is the original SOSA, which serves as our baseline. The other three are variants that incorporate the quadratic estimation and optimistic sampling enhancements.

- Original SOSA (SOSA): Use sample averaging to estimate the function value and uses Hit-and-Run that is initiated from the so-far best point to sample the next point

- SOSA with Quadratic Estimation (SOSA-Q): Use the quadratic regression to estimate the function value and uses Hit-and-Run that is initiated from the so-far best point to sample the next point

- SOSA with Optimistic Sampling (SOSA-O): Use sample averaging to estimate the function value and uses Hit-and-Run that is initiated from the point with the best lower confidence bound to sample the next point

- SOSA with Quadratic Estimation and Optimistic Sampling (SOSA-Q-O): Use the quadratic regression to estimate the function value and uses Hit-and-Run that is initiated from the point with the best lower confidence bound to sample the next point

**Single Observation Search Algorithms with Variations**

We are given:

- A continuous initial sampling density for search on $S$, $q_1(x)$, and a family of continuous adaptive search sampling distributions on $S$ with density

$$q_t(x \mid x_{t-1}),\ t = 2,3,\ldots,$$

.

- Radius schedule: $r_t > 0, t = 1,2,\ldots$

- Slowing integer sequence: $i_t < t, t = 1,2,\ldots$

- Top ranking integer: $K$

**Step 0:** Sample $x_1$ from $q_1$, observe $y_1 = g(x_1,u_1)$, where $u_1$ is a sample value having the same distribution as $U$ and independent of $x_1$. Set $(\mathscr{X}_1,\mathscr{Y}_1) = \{(x_1,y_1)\}$. Also, set $\hat{f}_1(x_1) = \hat{f}_1^*(x_1) = y_1$, $L_1(x_1) = 1$ and $x_1^* = x_1$. Set $t = 2$.

**Step 1:** Given the exploration point $\dot{x}_{t-1}$, sample the next point, $x_t$, from $q_t$. Obtain a sample value $u_t$ having the same distribution as $U$ and independent of $(x_1,y_1),\ldots,(x_{t-1},y_{t-1})$, and evaluate the objective function value $y_t = g(x_t,u_t)$.

**Step 2:** Update $(\mathscr{X}_t,\mathscr{Y}_t) = (\mathscr{X}_{t-1} \cup \{x_t\}, \mathscr{Y}_{t-1} \cup \{y_t\})$. Update the contribution at $x_t$ to each $x \in \mathscr{X}_t$,

$$\begin{aligned} L_t(x) &= |\{k \le t :\ x_k \in B(x,r_k)\}| \\ &= \begin{cases} L_{t-1}(x) & \text{if } x_t \notin B(x,r_t) \\ L_{t-1}(x)+1 & \text{if } x_t \in B(x,r_t) \end{cases} . \end{aligned} \tag{6.4}$$

**Step 2a [SOSA]:** Estimate the objective function using the average of the neighborhood points'

function observations. For each $x \in \mathscr{X}_t$, estimate the objective value $\hat{f}_t(x)$ as follows,

$$\begin{aligned} \hat{f}_t(x) &= \frac{\sum_{\{k \le t:\ x_k \in B(x,r_k)\}} y_k}{|\{k \le t:\ x_k \in B(x,r_k)\}|} = \frac{\sum_{\{k \le t:\ x_k \in B(x,r_k)\}} y_k}{L_t(x)} \\ &= \begin{cases} \hat{f}_{t-1}(x), & \text{if } x_t \notin B(x,r_t) \\ \hat{f}_{t-1}(x) + 1/L_t(x)(y_t - \hat{f}_{t-1}(x)), & \text{if } x_t \in B(x,r_t) \end{cases}. \end{aligned} \tag{6.5}$$

Estimate the optimal value as

$$\hat{f}_t^* = \min_{x \in \mathscr{X}_{i_t}} \hat{f}_t(x) \tag{6.6}$$

and estimate the optimal solution as

$$\hat{x}_t^* \in \operatorname*{argmin}_{x \in \mathscr{X}_{i_t}} \hat{f}_t(x) \tag{6.7}$$

where $\mathscr{X}_{i_t}$ is the subset of $\mathscr{X}_t$ that only includes points 1 through $i_t$.

**Step 2b [SOSA-Q]:** Estimate the objective function value at $x$ using the quadratic regression if there are sufficient points in the ball. For each $x \in \mathscr{X}_t$, estimate the objective value as follows.

$$\tilde{f}_t(x) = \begin{cases} \hat{f}_t(x) & \text{if } L_t(x) < l^* \\ x^\top A_{x,t}^* x + {b_{x,t}^*}^\top x + c_{x,t}^*, & \text{if } L_t(x) \ge l^* \end{cases} \tag{6.8}$$

where $A_{x,t}^*$, $b_{x,t}^*$, $c_{x,t}^*$ are from the solution to Problem $\mathscr{Q}(x,t)$ in (6.3). Estimate the optimal value as

$$\tilde{f}_t^* = \min_{x \in \mathscr{X}_{i_t}} \tilde{f}_t(x) \tag{6.9}$$

and estimate the optimal solution as

$$\tilde{x}_t^* \in \operatorname*{argmin}_{x \in \mathscr{X}_{i_t}} \tilde{f}_t(x). \tag{6.10}$$

**Step 3:** If a stopping criterion is met, stop. Otherwise, update $t \leftarrow t+1$ and go to Step 1.

### *6.3 Convergence Analysis of the Function Estimates*

SOSA-Q generalize the martingale single observation approach, incorporating quadratic regression to estimate the function value. The original SOSA analysis [57] implies the convergence of the original SOSA's function value estimate to the true function value. To establish the convergence of SOSA-Q, we rely on proving the consistency of its quadratic regression estimate. Theorem 10 shows that the objective function value estimate from quadratic regression in the case of martingale difference errors converges to the true objective function value, under some regularity conditions.

Let $X_t$ and $Y_t$ denote the sample point and its corresponding objective function evaluation at iteration $t$, for $t = 1,2,\ldots$. Then $Y_t = g(X_t,U_t)$, where $\{U_t, t = 1,2,\ldots\}$ are random elements, i.i.d. and have the same distribution as $U$. We can construct a filtration, starting with $\mathscr{F}_0 = \sigma(X_1)$, the $\sigma$-field generated by $X_1$, and for $t = 1,2,\ldots,\mathscr{F}_t = \sigma(X_1,Y_1,\ldots,X_t,Y_t,X_{t+1})$. Observe that $X_t$ is $\mathscr{F}_{t-1}$ measurable. According to (6.8) in the algorithm, $\tilde{f}_t(x)$, for a fixed $x \in S$, is an $\mathscr{F}_t$-random variable. The objective of the convergence analysis is to show that for any $x \in S$, $\tilde{f}_t(x)$ converges strongly to $f(x)$.

At iteration $t$, let $\varepsilon_t$ denote the random error,

$$\varepsilon_t = Y_t - f(X_t) = g(X_t,U_t) - f(X_t).$$

Observe that $\varepsilon_t$, for each $t$, is a martingale difference. The analysis requires the following assumptions.

**Assumption 1** The random error $\varepsilon_t$ has bounded variance.

**Assumption 2** The shrinking ball radius, $r_t \to 0$, as $t \to \infty$, and $r_t$ goes to zero at the rate bounded above by $1/t^{\alpha}$, $\alpha > 0$.

**Assumption 3** For all $x \in S$, the contribution to the estimate at $x$, $L_t(x) \to \infty$ with probability 1.

Before stating the final Assumption 4, some notations are required. Let $Q(x)$ be the vector of terms used in a quadratic expansion at point $x$, i.e.,

$$\begin{aligned} Q(x) &= [1, x_1^2, \ldots, x_d^2, 2x_1x_2, 2x_1x_3, \ldots, x_1, \ldots, x_d]^\top \\ &= [Q(x)_1, \ldots, Q(x)_M], \end{aligned}$$

where $Q(x)_1 = 1, Q(x)_2 = x_1^2, Q(x)_3 = x_2^2, \ldots, Q(x)_M = x_d$, and $M = 1 + 2d + d(d-1)$. The integer $M$ is the dimension of the quadratic expansion. Introduce the following matrices,

$$\boldsymbol{Y}_t = [Y_1, \ldots, Y_t]^\top, \quad \boldsymbol{Q}_t = [Q(X_1), \ldots, Q(X_t)]^\top, \quad \boldsymbol{P}_t = \boldsymbol{Q}_t^\top \boldsymbol{Q}_t. \tag{6.11}$$

The matrices $\boldsymbol{Q}_t$ and $\boldsymbol{P}_t$ are of size $t \times M$ and $M \times M$, respectively.

The least square estimator of $\beta$ based on the $t$ observations is then

$$\hat{\beta}_t = \boldsymbol{P}_t^{-1} \boldsymbol{Q}_t^\top \boldsymbol{Y}_t, \tag{6.12}$$

which is the analytical solution to the least square problem. Let $\boldsymbol{D}_t$ be the diagonal matrix containing diagonal of $\boldsymbol{P}_t$, i.e.,

$$\boldsymbol{D}_t = diag(d_{t1}^2, \ldots, d_{tM}^2), \tag{6.13}$$

where $d_{ti}^2 = \sum_{k=1}^t Q(X_k)_i^2 = (\boldsymbol{P}_t)_{ii}, i = 1, \ldots, M$, with $Q(X_t)_i$ being the term $i^{th}$ of the quadratic expansion $Q(X_t)$. We state the final assumption.

**Assumption 4** The matrices $\boldsymbol{P}_t^{-1} \boldsymbol{D}_t$ for $t \geq \tilde{t}$ are bounded uniformly in $t$ w.p. 1.

**Theorem 10** If Assumptions 1-4 are satisfied, then, for all $x \in S$, the objective function value estimate using the quadratic regression (SOSA-Q), $\tilde{f}_t(x)$, converges to the true objective function value $f(x)$ as $t$ goes to $\infty$ with probability one, i.e., for all $x \in S$,

$$\tilde{f}_t(x) \to f(x) \text{ w.p. } 1.$$

*Proof.* Fix $x \in S$. If the objective function $f(x)$ is twice differentiable, i.e., $f(x)$ can be represented by a quadratic expansion in the form,

$$f(z) = f(x) + (z-x)^\top A(z-x) + b^\top (z-x) + \|z-x\|^2 \rho(x, \|z-x\|^2),$$

where, $\lim_{z \to x} \rho(x, \|z-x\|^2) = 0$. Let $Q(x)$ be the quadratic expansion at point $x$, i.e.,

$$Q(x) = [1, x_1^2, \ldots, x_d^2, 2x_1x_2, 2x_1x_3, \ldots, x_1, \ldots, x_d]^\top.$$

The quadratic expansion becomes,

$$\begin{aligned} f(z) &= f(x) + (z-x)^\top A(z-x) + b^\top (z-x) + \|z-x\|^2 \rho(x, \|z-x\|^2) \\ &= Q(z)\beta + \|z-x\|^2 \rho(x, \|z-x\|^2), \end{aligned}$$

where $\beta$ is a vector of the quadratic expansion coefficients. Since $\lim_{z \to x} \rho(x, \|z-x\|^2) = 0$,

$$f(x) = \lim_{z \to x} f(z) = Q(x)\beta. \tag{6.14}$$

For any $x \in S$, let $B(x, r_t)$ be the ball centered at $x$ with radius $r_t$. We use quadratic function to estimate $f(x)$ of $x$ within $B(x, r_t)$. Without loss of generality, we are interested in the probability one event that the contribution to the estimate $L_t(x)$ is greater than critical sample size $l^*$. We estimate the objective function $f(x)$ using the quadratic regression $\tilde{f}_t(x)$. Solving problem $\mathscr{Q}(x,t)$ in (6.3) using the $L_t(x)$ sample points in the ball around $x$ at iteration $t$, results in the least square estimator $\hat{\beta}_t$. The estimate of the function value at a sample point $x$ can be expressed as,

$$\tilde{f}_t(x) = Q(x)\hat{\beta}_t. \tag{6.15}$$

In proving the convergences of the estimate $\tilde{f}(x)$ to the true objective function value $f(x)$, observe

that, by (6.14) and (6.15),

$$\begin{aligned} \tilde{f}_t(x) - f(x) &= Q(x)\hat{\beta}_t - Q(x)\beta \\ &= Q(x)(\hat{\beta}_t - \beta). \end{aligned}$$

To proof of Theorem 10 is equivalent to prove that

$$\hat{\beta}_t \to \beta \text{ w.p. } 1 \text{ as } t \to \infty.$$

Given matrices $\boldsymbol{Y}_t, \boldsymbol{Q}_t, \boldsymbol{P}_t$ in (6.11) and $\boldsymbol{D}_t$ in (6.13), introduce the following vectors,

$$\boldsymbol{\varepsilon}_t = (\varepsilon_1, \ldots, \varepsilon_t)^\top, \quad \boldsymbol{\omega}_t = (\omega_1, \ldots, \omega_t)^\top,$$

where, for each $t$,

$$\omega_t = \|X_t - x\|^2 \rho(x, \|X_t - x\|^2)$$

The response $\boldsymbol{Y}_t$ can be written in the form

$$\boldsymbol{Y}_t = \boldsymbol{Q}_t \beta + \boldsymbol{\varepsilon}_t + \boldsymbol{\omega}_t. \tag{6.16}$$

Recall least square estimator of $\beta$ based on the $t$ observations is defined as

$$\hat{\beta}_t = \boldsymbol{P}_t^{-1} \boldsymbol{Q}_t^\top \boldsymbol{Y}_t.$$

Hence from (6.16), the estimation error becomes,

$$\hat{\beta}_t - \beta = \boldsymbol{P}_t^{-1} \boldsymbol{Q}_t^\top (\boldsymbol{\varepsilon}_t + \boldsymbol{\omega}_t). \tag{6.17}$$

Recall $\boldsymbol{D}_t$ in (6.13). Observe that Equation (6.17) can be written in the form,

$$\hat{\beta}_t - \beta = \boldsymbol{P}_t^{-1}\boldsymbol{D}_t z_t + \boldsymbol{P}^{-1}\boldsymbol{D}_t w_t, \tag{6.18}$$

where the $i^{th}$ component of random vector $z_t$ and $w_t$ is given by,

$$z_{ti} = d_{ti}^{-2}\sum_{k=1}^{t} Q(X_k)_i \varepsilon_k, \tag{6.19}$$

$$w_{ti} = d_{ti}^{-2}\sum_{k=1}^{t} Q(X_k)_i \omega_k. \tag{6.20}$$

Consider (6.19). We first show that, as $t \to \infty$,

$$d_{ti}^2 \to \infty \quad \text{w.p. 1.} \tag{6.21}$$

From Assumption 2, let $\tilde{t}$ be the time that $r_t$ small enough to make $|Q(X_t)_i|$ bounded away from 0, for all $i$, when $t > \tilde{t}$. Therefore, $c_1 \leq |Q(X_t)_i| \leq c_2$, for some $0 < c_1 < c_2$ when $t > \tilde{t}$. Consequently,

$$d_{ti}^2 = \sum_{k=1}^{\tilde{t}} Q(X_k)_i^2 + \sum_{k=\tilde{t}+1}^{t} Q(X_k)_i^2, \tag{6.22}$$

and

$$(L_t - L_{\tilde{t}})c_1^2 \leq \sum_{k=\tilde{t}+1}^{t} Q(X_k)_i^2 \leq (L_t - L_{\tilde{t}})c_2^2. \tag{6.23}$$

By Assumption 3, $L_t \to \infty$ w.p. 1 implies $\sum_{k=\tilde{t}+1}^{t} Q(X_k)_i^2 \to \infty$ w.p.1. Hence, $d_{ti}^2 \to \infty$ w.p. 1.

Since $\varepsilon_t$ is martingale difference, by Lemma 1 in [21], Equation (6.21) implies $z_t \to 0,$ w.p. $1,$ as $t \to \infty$. We assume that $o_t \to 0$ as $t \to \infty$. The goal is to prove that $w_t \to 0$, w.p. 1, as $t \to \infty$, i.e.,

$$\frac{1}{d_{ti}^2}\sum_{k=1}^{t} Q(X_k)_i \omega_k \to 0 \tag{6.24}$$

By Kronecker's lemma, it is equivalent to show that

$$\left|\sum_{k=1}^{t}\frac{1}{d_{ti}^2}Q(X_k)_i\omega_k\right|<\infty. \tag{6.25}$$

Let $\tau_1,\tau_2,\ldots,\tau_{L_t}$ be the time that $X_{\tau_k}\neq\vec{0}$ up to time $t$. Without loss of generality, let $\tau_1>\tilde{t}$. Therefore, there exists $0<c_1<c_2$ such that $c_1\leq|Q(X_t)_i|\leq c_2$, and, hence,

$$d_{\tau_j i}^2=\sum_{k=1}^{j}Q(X_{\tau_k})_i^2\geq c_1^2 j.$$

Therefore,

$$\begin{aligned}
\left|\sum_{k=1}^{t}\frac{1}{d_{ki}^2}Q(X_k)_i\omega_k\right| &= \left|\sum_{j=1}^{L_t}\frac{1}{d_{\tau_j i}^2}Q(X_{\tau_j})_i\omega_{\tau_j}\right| \\
&\leq \sum_{j=1}^{L_t}\frac{1}{d_{\tau_j i}^2}\left|Q(X_{\tau_j})_i\right|\left|\omega_{\tau_j}\right| \\
&\leq \sum_{j=1}^{L_t}\frac{1}{c_1^2 j}c_2\left|\omega_{\tau_j}\right| \\
&\leq \frac{c_2}{c_1^2}\sum_{j=1}^{L_T}\frac{1}{j}\|X_{\tau_j}-x_i\|^2\left|\rho(x,\|X_{\tau_j}-x\|^2)\right| \\
&\leq \frac{c_2}{c_1^2}\sum_{j=1}^{L_T}\frac{1}{j}r_{\tau_j}^2\left|\rho(x,\|X_{\tau_j}-x\|^2)\right| \\
&\leq \frac{c_2}{c_1^2}\bar{\rho}\sum_{j=1}^{L_T}\frac{1}{j}r_{\tau_j}^2 \text{ w.p. 1, because } \rho(x,\|X_{\tau_j}-x\|^2)\to 0 \text{ w.p. 1,} \\
&\leq \frac{c_2}{c_1^2}\bar{\rho}\sum_{j=1}^{L_T}\frac{1}{j}r_j^2 \text{ w.p. 1, because } \tau_j\geq j \implies r_{\tau_j}^2\leq r_j^2, \\
&\leq \frac{c_2}{c_1^2}\bar{\rho}\sum_{j=1}^{L_T}\frac{1}{j}\frac{1}{j^{2\alpha}} \text{ w.p. 1, where } \alpha>0 \text{ by Assumption 2,} \\
&= \frac{c_2}{c_1^2}\bar{\rho}\sum_{j=1}^{L_T}\frac{1}{j^{1+2\alpha}} \text{ w.p. 1.}
\end{aligned}$$

Therefore,

$$\lim_{t\to\infty}\left|\sum_{k=1}^{t}\frac{1}{d_{ki}^2}Q(X_k)_i o_k\right| \leq \lim_{L_t\to\infty}\frac{c_2}{c_1^2}\bar{\rho}\sum_{j=1}^{L_t}\frac{1}{j^{1+2\alpha}} < \infty \quad \text{w,p. 1.} \tag{6.26}$$

□

## *6.4 Numerical Experiments*

In this section, we evaluate the performance of the four SOSA variants using the Hit-and-Run (HR) sampler [117]. The algorithms are applied to three well-known global optimization test problems, each modified to include a noisy objective function and a convex feasible region. The four algorithms under comparison were defined in Section 6.2 and are referred to as SOSA, SOSA-Q, SOSA-O, and SOSA-Q-O.

### *6.4.1 The Sampler*

The Hit-and-Run (HR) sampler is selected for this study because it has a positive probability of sampling any point within the feasible space. As shown in [57], this property ensures that all SOSA variants satisfy Assumption 3 when applied to a convex feasible region. This assumption is critical to ensure that the quadratic estimator (Theorem 10) and the original mean estimator [57] are both consistent estimators. Furthermore, HR provides the flexibility to bias the search toward a chosen point, a property that allows for the straightforward integration of our optimistic sampling strategy. The HR sampler is therefore integrated into the SOSA variants by replacing the generic Step 1a for SOSA and SOSA-Q and Step 1b for SOSA-O and SOSA-Q-O with the following specific procedure.

**Step 1a:** Given the point $\hat{x}_{t-1}^*$ sample the next point $x_t$ by sampling a direction $v$ from the uniform distribution on the surface of a $d$-dimensional hypersphere, and sampling $x_t$ uniformly distributed on the line segment $\Lambda$ where

$$\Lambda = \{\hat{x}_{t-1}^* + \lambda v : \lambda \in \Re\} \cap S.$$

Sample $u_t$ with the same distribution with $U$ and evaluate the objective function value $y_t = g(x_t, u_t)$.

**Step 1b:** Order all sampled points $x \in \mathscr{X}_t$, $x_{(1)},\ldots,x_{(t)}$, by their estimated function values $\hat{f}_t(x)$, so that for SOSA-O,

$$\hat{f}_t(x_{(1)}) \leq \cdots \leq \hat{f}_t(x_{(t)}). \tag{6.27}$$

For SOSA-Q-O, order the estimated function value with the estimated objective function value from quadratic regression $\tilde{f}_t(x)$. Collect a fixed number, $K$, of best performers by their estimated function values, evaluating the lower confidence bounds of these points $x_{(1)},\ldots,x_{(K)}$ according to [7] as follows. For all $\tau \in \{1,\ldots,K\}$,

$$\hat{f}_t^L(x_{(\tau)}) = \hat{f}_t(x_{(\tau)}) - \sqrt{\frac{2\ln(t)}{L_t(x_{(\tau)})}}. \tag{6.28}$$

Assign the point with the lowest confidence bound as $x_t^L$

$$x_t^L \in \underset{\tau\in\{1,\ldots,K\}}{\operatorname{argmin}} \hat{f}_t^L(x_{(\tau)}). \tag{6.29}$$

Given point $x_{t-1}^L$, sample the next point $x_t$ by sampling a direction $v$ from the uniform distribution on the surface of a $d$-dimensional hypersphere, and sampling $x_t$ uniformly distributed on the line segment $\Lambda$ where

$$\Lambda = \{x_{t-1}^L + \lambda v : \lambda \in \Re\} \cap S.$$

Sample $u_t$ with the same distribution with $U$ and evaluate the objective function value $y_t = g(x_t, u_t)$.

### *6.4.2 The Parameters*

We adopt the parameter values for $r_t$ and $i_t$ as suggested in [57]. For the top-rank parameter K, we selected a value that avoids significant computational overhead. The chosen parameters are as follows:

- Radius schedule: $r_t = \begin{cases} 0.1\, t^{-0.009}, & \text{for Problem 1 and 2,} \\ 0.04\, t^{-0.009}, & \text{for Problem 3.} \end{cases}$,

- Slowing integer sequence: $i_t = \left\lfloor t^{0.9} \right\rfloor$,

- Top ranking integer: $K = 15$.

### *6.4.3 Test Functions*

The performance of the algorithms is evaluated on three benchmark problems, each in 5 dimensions: the shifted sinusoidal problem, the Rosenbrock problem, and the Rastrigin problem. To simulate a noisy environment appropriate for stochastic optimization, the true objective function value at each point is corrupted by additive uniform noise scaled to 10% of the function's value. The test problems are described in detail below.

*Problem 1: Shifted Sinusoidal Problem*

$$\begin{aligned} &\min \quad \mathbb{E}\left[f(x) + (1 + |f(x)|)U\right] \qquad \text{s.t.} \quad 0 \le x_i \le \pi, \quad i = 1, \ldots, 5 \\ &\text{where} \;\; f(x) = -\left(2.5 \prod_{i=1}^{d} \sin(x_i - \pi/6) + \prod_{i=1}^{d} \sin(5(x_i - \pi/6))\right) + 3.5 \end{aligned}$$

and $x \in \Re^5$ ($d = 5$) and $U \sim \text{Uniform}[-0.1, 0.1]$. The global minimum is at $x^* = (4\pi/6, \ldots, 4\pi/6)$ and $f(x^*) = 0$.

*Problem 2: Rosenbrock Problem*

$$\begin{aligned} &\min \quad \mathbb{E}\left[f(x) + (1 + |f(x)|)U\right] \qquad \text{s.t.} \quad -2 \le x_i \le 2, \quad i = 1, \ldots, 5 \\ &\text{where} \;\; f(x) = (0.1) \times \sum_{i=1}^{d-1} \left((1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2\right) \end{aligned}$$

and $x \in \Re^5$ ($d = 5$) and $U \sim \text{Uniform}[-0.1, 0.1]$. The global minimum is at $(1, \ldots, 1)$ and $f(x^*) = 0$.

*Problem 3: Rastrigin problem*

$$\min \quad \mathbb{E}\left[f(x)+(1+|f(x)|)U\right] \qquad \text{s.t.} \quad -5.12 \le x_i \le 5.12, \quad i=1,\ldots,5$$
$$\text{where} \;\; f(x) = 10d + \sum_{i=1}^{d}(x_i^2 - 10cos(2\pi x_i))$$

and $x \in \Re^5$ ($d=5$) and $U \sim$ Uniform$[-0.1, 0.1]$. The global minimum is at $(0,\ldots,0)$ and $f(x^*)=0$.

### *6.4.4 Computational Results*

For each test problem, we run each of the four SOSA variants for a total of 12,000 function evaluations. The first 1,000 evaluations of each run constitute a warm-start period. During this initial phase, all algorithms, including the enhanced variants, use the simple averaging method (Step 2a) for function value estimation. This warm-start period ensures that a sufficient number of neighborhood points are collected to allow for stable quadratic estimations later in the run. After 1,000 evaluations, the specialized procedures for the enhanced variants are activated: SOSA-Q switches to Step 2b, SOSA-O to Step 2c, and SOSA-Q-O to Step 2d for the remainder of the search.

To ensure the robustness of our results, we perform 10 independent replications for each algorithm-problem pair. The following four performance metrics are recorded at each function evaluation $t$ (for $t = 1,\ldots,12000$) and then averaged across the 10 replications:

- Estimated Optimal Value ($\hat{f}_t^*$): The best objective value found by the algorithm's estimation method.

- True Objective at Best Point ($f(\hat{x}_t^*)$): The true, noise-free objective function value of the algorithm's current best candidate solution, $\hat{x}_t^*$.

- Estimation Error ($\hat{e}_t$): The average noise of the optimal solution estimate.

- Distance to Optimum ($D_t^*$): The Euclidean distance from the algorithm's best candidate solution to the true global optimum.

The averaged trajectories of these four performance metrics for the Shifted sinusoidal, Rosenbrock, and Rastrigin problems are presented in Figures 6.1, 6.2, and 6.3, respectively.

Figures 6.1 shows the four performance measurements of the four algorithms (SOSA,SOSA-Q, SOSA-O, and SOSA-Q-O) applied to Shifted sinusoidal problem. The SOSA algorithms with quadratic estimation, SOSA-Q and SOSA-Q-O, demonstrate superior performance compared to SOSA and SOSA-O. As observed in panels (a) and (b) of Figures 6.1, both the optimal value estimates and the objective function values of the optimal solution estimates for SOSA-Q and SOSA-Q-O converge more quickly to the optimum (target) than those of SOSA and SOSA-O. As in panel (c) of Figures 6.1, the average noise in the optimal value estimates stabilizes near zero as the algorithm progresses. Consequently, panel (d) demonstrates that the best candidate solutions identified by SOSA-Q and SOSA-Q-O achieve closer proximity to the true optimum than those from SOSA and SOSA-O.

Figures 6.2 show the four performance measurements of the four algorithms applied to the Rosenbrock problem. Again, SOSA-Q-O and SOSA-O demonstrate better performance than SOSA-Q and SOSA. As observed in panels (a) and (b) of Figures 6.2, both the optimal value estimates and the objective function values of the optimal solution estimates SOSA-Q-O and SOSA-O maintain consistently better performance across nearly the entire optimization trajectory. Panel (c) of Figures 6.2 shows that the average noise in the optimal value estimates stabilizes near zero over the course of the algorithm. Panel (d) in Figures 6.2 shows that the best candidate solutions from SOSA-Q-O and SOSA-O approach the true optimum closer than the best candidate solutions from SOSA and SOSA-Q.

Figures 6.3 show the four performance measurements of the four algorithms applied to the Rastrigin problem. SOSA-O demonstrate the best performance. As observed in Panels (a) and (b) of Figures 6.3, SOSA-O exhibited a slight improvement in performance over the other three algorithms. Similar to other problems, Panel (c) of Figures 6.3 shows that the average noise in the

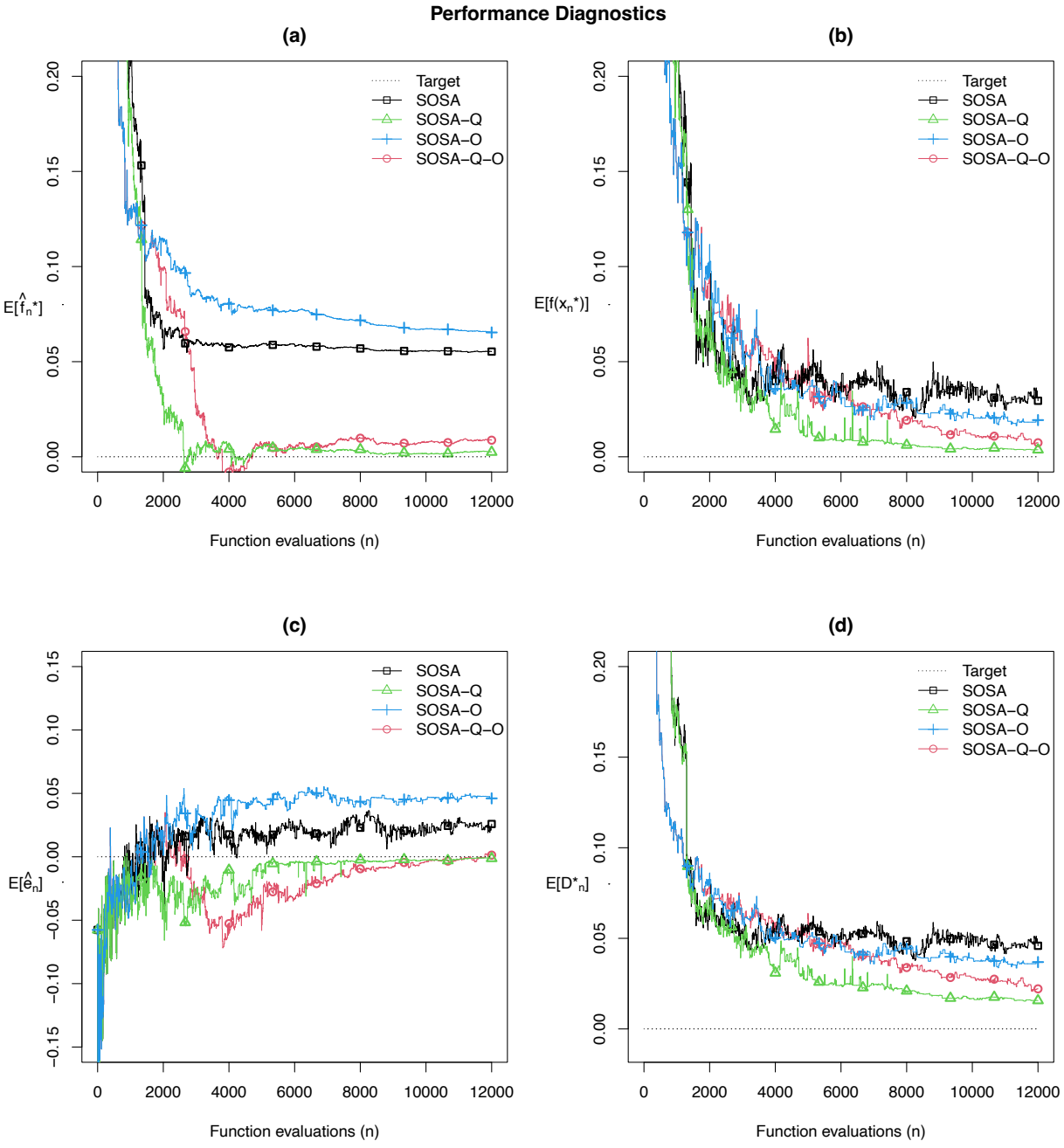



**Figure 6.1:** Performance Diagnostics for SOSA, SOSA-Q, SOSA-O and SOSA-Q-O applied to the shifted sinusoidal problem.

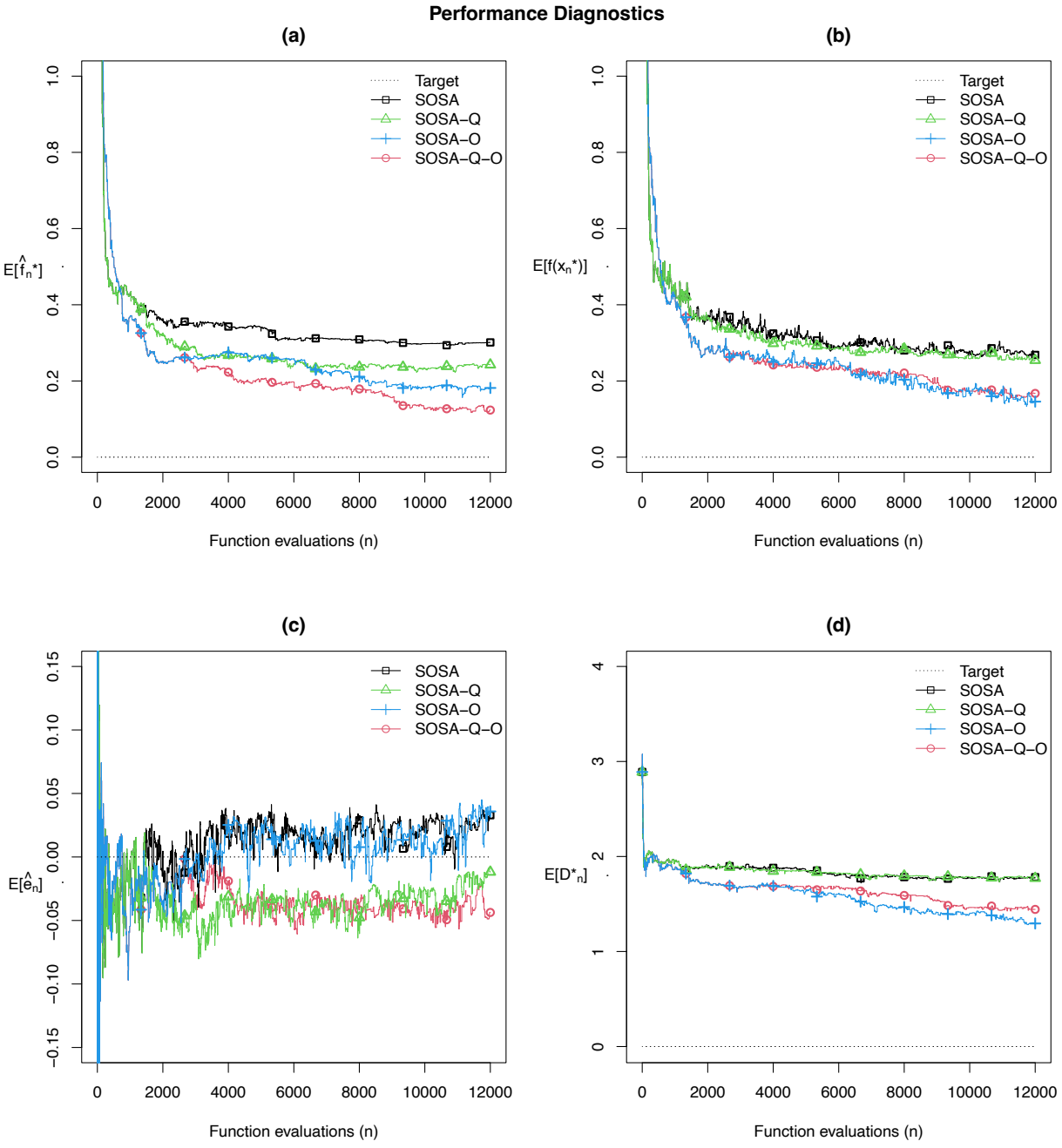



**Figure 6.2:** Performance Diagnostics for SOSA, SOSA-Q, SOSA-O and SOSA-Q-O applied to the Rosenbrock problem.

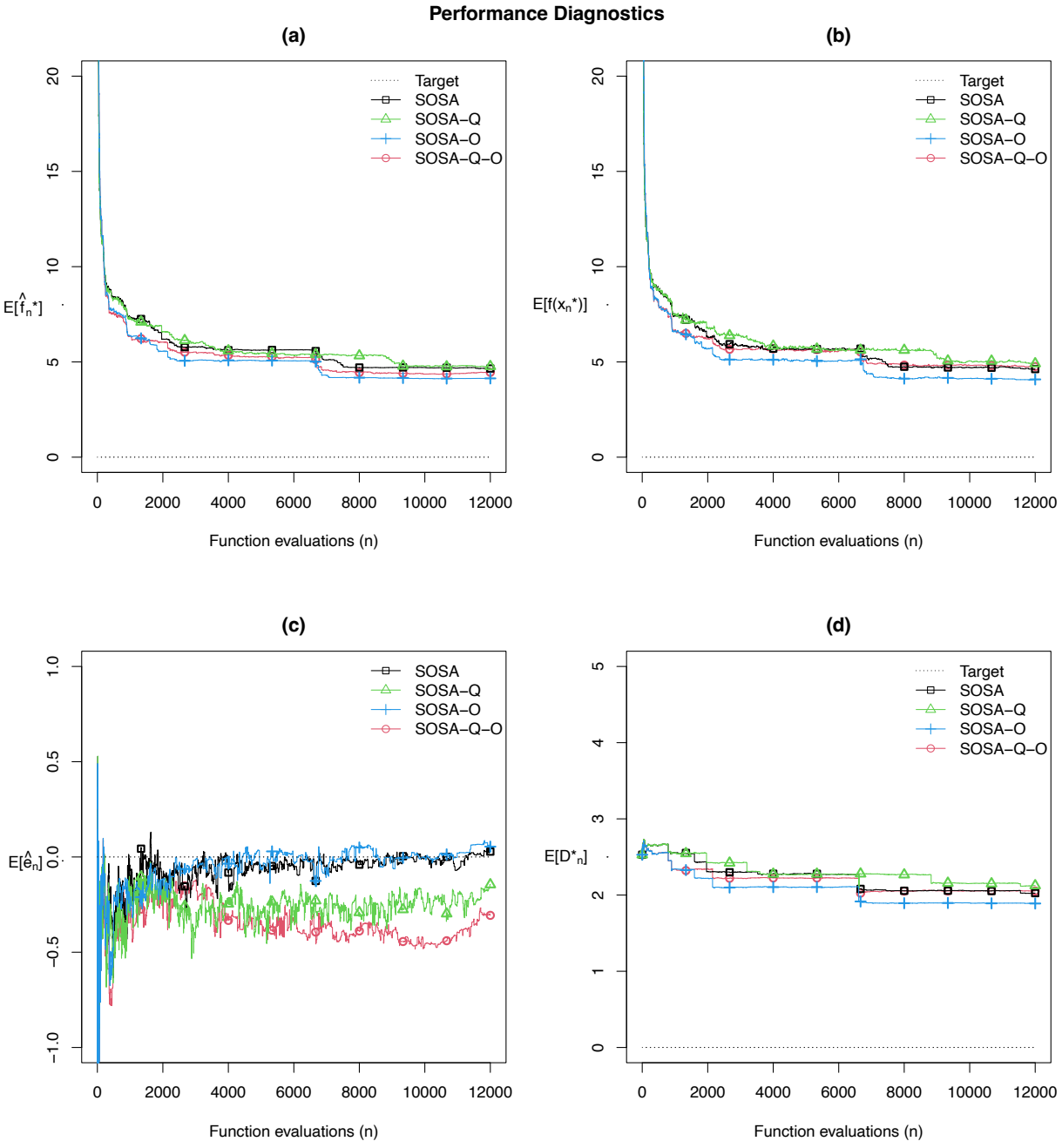



**Figure 6.3:** Performance Diagnostics for SOSA, SOSA-Q, SOSA-O and SOSA-Q-O applied to the Rastrigin problem.

optimal value estimates stabilizes near zero over the course of the algorithm. Consequently, Panel (d) in Figures 6.3 shows that the best candidate solution of SOSA-O approach the true optimum slightly closer than the best candidate solution of the other three algorithms.

Observe that in Figures 6.1, 6.2 and 6.3, SOSA-Q and SOSA-Q-O consistently demonstrate a more pronounced negative bias in the average noise of their optimal value estimates compared to SOSA and SOSA-O, particularly in later stages. This is a common character of adaptive algorithms that prioritize candidate points with minimum estimates, this character amplified in SOSA-Q and SOSA-Q-O due to their incorporation of quadratic regression in the estimation process.

It is worth noting that the true objective function value of the algorithm's current best candidate, $f(\hat{x}_t^*)$ presented in panel (b) of Figures 6.1, 6.2, and 6.3, serves as the primary performance measurement. This is because the true objective function without noise, represents the ground truth of our experiments. Across all optimization problems, we observe that at least one combination of our proposed heuristics outperforms the original SOSA in panel (b). This finding provides evidences that integrating machine learning and reinforcement learning methods into the SOSA framework substantially improves its finite-time performance.

| | SOSA | | SOSA-Q | | SOSA-O | | SOSA-Q-O | | Max Improv. |
|---|---|---|---|---|---|---|---|---|---|
| | $f(\hat{x}_T^*)$ | % | $f(\hat{x}_T^*)$ | % | $f(\hat{x}_T^*)$ | % | $f(\hat{x}_T^*)$ | % | % |
| Problem 1 | 0.0295 | 100% | 0.0037 | 12.5% | 0.0193 | 65.5% | 0.0074 | 25.1% | 87.5% |
| Problem 2 | 0.2683 | 100% | 0.2544 | 94.8% | 0.1458 | 54.3% | 0.1674 | 62.4% | 45.7% |
| Problem 3 | 4.63 | 100% | 4.91 | 106.2% | 4.08 | 88.2% | 4.76 | 102.8% | 11.8% |

**Table 6.1:** Average function values of optimal solution estimates at termination iteration

Table 6.1 shows the average function values of the optimal solution estimated at the termination iteration $T = 12000$ of the four algorithms at termination when applied to Problem 1, 2 and 3. The machine learning enhancements provided a substantial performance advantage, with the proposed variants outperforming the baseline SOSA by 12% to 87% on the final "true objective at best point" metric. Our observations suggest that optimistic sampling enhances performance during the exploration stage by preventing the search from getting trapped in a local optimum, while quadratic estimation enhances performance as the search approaches the solution.

## *6.5 Conclusion*

This section propose variants of the Single Observation Search Algorithm (SOSA) for continuous simulation optimization, through the integration of machine learning and reinforcement learning techniques. Our approach enhances SOSA by applying generalization principles to refine objective function value estimation and leveraging reinforcement learning to guide the search process. We show that under some regularity conditions, the objective function value estimate from quadratic regression in the case of proposed algorithm converges to the true objective function value. Numerical experiments performed on benchmark global optimization problems yield supporting evidence that our proposed heuristic enhances the performance of the SOSA framework. Our observations suggest that optimistic sampling enhances performance during the exploration stage by preventing the search from getting trapped in a local optimum, while quadratic estimation enhances performance as the search approaches the solution. Throughout this chapter, the main goal of this dissertation is satisfied with the proposed SOSA implementation with quadratic regression, convergence analysis of the objective function value estimate and numerical experiments on benchmark global optimization problems.

# Chapter 7

# CONCLUSION

This dissertation has explored the integration of machine learning methods with global optimization algorithms to tackle the challenges of noisy, high-dimensional black-box optimization problems. With the lack of explicit function structure, the black-box optimization methodsdesign focuses on balancing between exploration, exploitation, and estimation. The central idea of this dissertation is that by leveraging the predictive power of statistical and machine learning tools to enhance adaptive search algorithms.

This dissertation presented fulfilled its three main goals. We demonstrates the integration of machine learning models with adaptive search methods through the development of the Multi-level PBnB algorithm, the Branching Adaptive Surrogate Search Optimization (BASSO) framework, and the enhanced Single Observation Search Algorithm (SOSA). Theoretical contributions were made by providing finite-time performance analyses and convergence proofs for these new algorithms, including probability bounds for the Multi-level PBnB, finite-time analysis of BASSO, and a convergence analysis for function estimates in SOSA that accounts for dependent samples. Finally, extensive numerical experiments provided practical implementation guidelines and illustrated the performance of these integrated methodologies.

This research has shown that the two paradigms of global optimization and machine learning can work together to create algorithms superior to their individual components. In the Multi-level PBnB and BASSO frameworks, machine learning models, acting as local surrogates, specifically Gaussian Processes and regularized quadratic regression, are proved to be effective tools to guide the local search. Similarly, the enhancement of SOSA replaced neighborhood averaging with quadratic regression, providing a more sophisticated function value estimate from noisy observations. This fusion allows algorithms to build and refine a global understanding of the search space

with partitioning and adaptive probabilities while simultaneously using localized, data-driven models or the surrogates to make more intelligent sampling decisions, thereby increasing the probability of sampling within an improving region.

**Limitations and Future Research Directions**

Furthermore, while the BASSO framework established theoretical conditions for scalability in high-dimensional problems, a gap remains between this ideal performance and current implementations. Tackling high-dimensional black-box functions remains a challenge. The path forward likely does not only lie in finding a single superior technique, but rather in determining the effective combination of partitioning, surrogate modeling, decomposition, and sampling strategies for different problem classes.

Looking ahead, there is a possibility for implementations to move closer to their theoretical ideals through the field of quantum computing. Conceptual algorithms like Annealing Adaptive Search (AAS) are powerful in theory but impractical due to the difficulty of sampling from a perfect Boltzmann distribution. Quantum annealing and other quantum optimization algorithms, such as Grover's Adaptive Search, offer a potential pathway to physically realize these sampling distributions more efficiently. Exploring the integration of quantum computation could represent the next frontier, potentially bridging the gap between the theory of adaptive search and its practical, scalable implementation.

# BIBLIOGRAPHY

[1] Satyajith Amaran, Nikolaos V Sahinidis, Bikram Sharda, and Scott J Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240:351–380, 2016.

[2] Sigrún Andradóttir and Andrei A Prudius. Adaptive random search for continuous simulation optimization. *Naval Research Logistics*, 57(6):583–604, 2010.

[3] Yashwanth Annpureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *TACAS*, 2011.

[4] C Audet and W Hare. *Derivative-free and blackbox optimization. Springer series in operations research and financial engineering. Springer, Cham*. Springer Cham, Switzerland, 2017.

[5] Charles Audet. *Blackbox Optimization*, pages 1–6. Springer International Publishing, Cham, 2020.

[6] Charles Audet and John E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

[7] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[8] François Bachoc, Tommaso Cesari, and Sébastien Gerchinovitz. The sample complexity of level set approximation. In *Proceedings of the 24$^{th}$ International Conference on Artificial Intelligence and Statistics*, 2021. April 13$^{\text{th}}$-15$^{\text{th}}$, held virtually, 424-432.

[9] Fusheng Bai, Dongchi Zou, and Yutao Wei. A surrogate-assisted evolutionary algorithm with clustering-based sampling for high-dimensional expensive blackbox optimization. *Journal of Global Optimization*, 89(1):93–115, 2024.

[10] Ishan Bajaj, Akhil Arora, and M. M. Faruque Hasan. Black-box optimization: Methods and applications. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, volume 170. Springer, Cham, January 2021.

[11] William P Baritompa, David W Bulger, and Graham R Wood. Grover's quantum algorithm applied to global optimization. *SIAM Journal on Optimization*, 15(4):1170–1184, 2005.

[12] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Nickovic, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS, Springer*, pages 128–168. 2018.

[13] Demian Battaglia, Lorenzo Stella, Osvaldo Zagordi, Giuseppe E Santoro, and Erio Tosatti. Deterministic and stochastic quantum annealing approaches. *Quantum Annealing and Other Optimization Methods*, pages 171–206, 2005.

[14] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. Association for Computing Machinery, 1992.

[15] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.

[16] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[17] David W Bulger. Combining a local search and grover's algorithm in black-box global optimization. *Journal of optimization theory and applications*, 133:289–301, 2007.

[18] David W. Bulger, William P. Baritompa, and Graham R. Wood. Implementing pure adaptive search with Grover's quantum algorithm. *Journal of optimization theory and applications*, 116(3):517–529, 2003.

[19] David W. Bulger and Graham R. Wood. Hesitant adaptive search for global optimisation. *Mathematical Programming*, 81(1):89–102, 1998.

[20] Marie Chau and Michael C Fu. An overview of stochastic approximation. *Handbook of simulation optimization*, pages 149–178, 2014.

[21] Norbert Christopeit and Kurt Helmes. Strong consistency of least squares estimators in linear regression models. *The Annals of Statistics*, 8(4):778–788, 1980.

[22] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, Philadelphia, PA, 2000.

[23] William Jay Conover. *Practical nonparametric statistics*, volume 350. john wiley & sons, 1999.

[24] Joseph Cralley, Ourania Spantidi, Bardh Hoxha, and Georgios Fainekos. TLTk: A toolbox for parallel robustness computation of temporal logic specifications. In *Runtime Verification*, volume 12399 of *LNCS*, pages 404–416, 2020.

[25] Elizabeth Crosson and Aram W Harrow. Simulated quantum annealing can be exponentially faster than classical simulated annealing. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 714–723. IEEE, 2016.

[26] R Dale. What's it good for? *Natural Language Engineering*, 1:113–118, 2021.

[27] L Devroye, L Gyorfi, A Krzyzak, and G Lugosi. On the strong universal consistency of nearest neighbor regression function estimates. *The Annals of Statistics*, 22(3):1371–1385, 1994.

[28] Adel Dokhanchi, Bardh Hoxha, Cumhur Erkan Tuncali, and Georgios Fainekos. An efficient algorithm for monitoring practical TPTL specifications. In *14th ACM-IEEE International Conference on Formal Methods and Models for System Design*, pages 184–193, 2016.

[29] Alexandre Donze. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification (CAV)*, volume 6174 of *LNCS, Springer*, pages 167–170, 2010.

[30] Christoph Dürr and Peter Hoyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

[31] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Aniruddh Chandratre, Alexandre Donz\'e, Georgios Fainekos, Goran Frehse, Khouloud Gaaloul, Jun Inoue, Tanmay Khandait, Logan Mathesen, Claudio Menghi, Giulia Pedrielli, Marc Pouzet, Masaki Waga, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. Arch-comp 2021 category report: Falsification with validation of results. In Goran Frehse and Matthias Althoff, editors, *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, volume 80 of *EPiC Series in Computing*, pages 133–152, 2021.

[32] Georgios Fainekos, Bardh Hoxha, and Sriram Sankaranarayanan. Robustness of specifications and its applications to falsification, parameter mining, and runtime monitoring with s-taliro. In *Runtime Verification (RV)*, volume 11757 of *LNCS*, pages 27–47, 2019.

[33] L. Floridi and M. Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds & Machines*, 30:681–694, 2020.

[34] Peter I Frazier. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pages 255–278. Informs, 2018.

[35] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[36] Michael C. Fu. *Handbook of Simulation Optimization*, volume 216. Springer New York, New York, NY, 2015.

[37] Peter W Glynn et al. Importance sampling for monte carlo estimation of quantiles. In *Mathematical methods in stochastic simulation and experimental design: Proceedings of the 2nd st. petersburg workshop on simulation*, pages 180–185. Citeseer, 1996.

[38] Allen A Goldstein. Optimization of lipschitz continuous functions. *Mathematical Programming*, 13(1):14–22, 1977.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[40] Abhijit Gosavi. *Simulation-Based Optimization, Second Edition*. Springer, New York, 2015.

[41] Robert B Gramacy. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. CRC Press, Taylor & Francis Group, Boca Raton, FL, 2020.

[42] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[43] Huong Ha, Sunil Gupta, Santu Rana, and Svetha Venkatesh. High dimensional level set estimation with bayesian neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. February 2nd-9th, held virtually, 12095-12103.

[44] Eric Han, Ishank Arora, and Jonathan Scarlett. High-dimensional bayesian optimization via tree-structured additive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7630–7638, 2021.

[45] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[46] Stephanie A. Harmon, Thomas H. Sanford, Sheng Xu, et al. Artificial intelligence for the detection of covid-19 pneumonia on chest ct using multinational datasets. *Nature Communications*, 11:4080, 2020.

[47] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *An introduction to statistical learning*. Springer, 2009.

[48] Peter Heidlauf, Alexander Collins, Michael Bolender, and Stanley Bak. Verification challenges in f-16 ground collision avoidance and other automated maneuvers. In *5th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH)*, volume 54, pages 208–217, 2018.

[49] Hao Huang and Zelda B. Zabinsky. Adaptive probabilistic branch and bound with confidence intervals for level set approximation. In R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, editors, *Proceedings of the 2013 Winter Simulation Conference*, pages 4146–4157, Washington, DC, 2013. Institute of Electrical and Electronics Engineers, Inc.

[50] Hao Huang, Zelda B. Zabinsky, Yuankun Li, and Shan Liu. Analyzing hepatitis C screening and treatment strategies using probabilistic branch and bound. In T. M. K. Roeder, P. I. Frazier, R. Szechtman, T. Huschka E. Zhou, and S. E. Chick, editors, *Proceedings of the 2016 Winter Simulation Conference (WSC)*, pages 2076–2086. Institute of Electrical and Electronics Engineers, Inc., 2016.

[51] Waltraud Huyer and Arnold Neumaier. Snobfit–stable noisy optimization by branch and fit. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):1–25, 2008.

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

[53] J. Jumper, R. Evans, A. Pritzel, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021.

[54] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[55] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pages 295–304. PMLR, 2015.

[56] James Kapinski, Jyotirmoy V Deshmukh, Xiaoqing Jin, Hisahiro Ito, and Ken Butts. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems Magazine*, 36(6):45–64, 2016.

[57] Seksan Kiatsupaibul, Robert L Smith, and Zelda B Zabinsky. Single observation adaptive search for continuous simulation optimization. *Operations research*, 66(6):1713–1727, 2018.

[58] Seksan Kiatsupaibul, Robert L. Smith, and Zelda B. Zabinsky. Single observation adaptive search for discrete and continuous stochastic optimization. *Operations Research Letters*, 48(5):666–673, 2020.

[59] Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. Distributionally robust bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2174–2184. PMLR, 2020.

[60] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12:479 – 502, 2002.

[61] Harold Kushner and G. George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, New York, NY, 2nd edition, 2003.

[62] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106, 1964.

[63] Sébastien Le Digabel. Algorithm 909: Nomad: Nonlinear optimization with the mads algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 37(4):1–15, 2011.

[64] Yipeng Liu and Gary J Koehler. Using modifications to grover's search algorithm for quantum global optimization. *European Journal of Operational Research*, 207(2):620–632, 2010.

[65] Marco Locatelli and Fabio Schoen. (Global) optimization: Historical notes and recent developments. *EURO Journal on Computational Optimization*, 9:100012, 2021.

[66] Kaiwen Ma, Luis Miguel Rios, Atharv Bhosekar, Nikolaos V Sahinidis, and Sreekanth Rajagopalan. Branch-and-model: a derivative-free global optimization algorithm. *Computational Optimization and Applications*, 85(2):337–367, 2023.

[67] Xiaoliang Ma, Xiaodong Li, Qingfu Zhang, Ke Tang, Zhengping Liang, Weixin Xie, and Zexuan Zhu. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2018.

[68] Blake Mason, Romain Camilleri, Subhojyoti Mukherjee, Kevin Jamieson, Robert Nowak, and LalitJain. Nearly optimal algorithms for level set estimation. *arXiv*, pages 1–32, 2021. https://doi.org/10.48550/arXiv.2111.01768.

[69] Logan Mathesen, Giulia Pedrielli, Szu Hui Ng, and Zelda B Zabinsky. Stochastic optimization with adaptive restart: A framework for integrated local and global learning. *Journal of Global Optimization*, 79:87–110, 2021.

[70] Logan Mathesen, Shakiba Yaghoubi, Giulia Pedrielli, and Georgios Fainekos. Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 991–997. IEEE, 2019.

[71] Jorge J Moré and Stefan M Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[72] Kohei Morimoto, Yusuke Takase, Kosuke Mitarai, and Keisuke Fujii. Continuous optimization by quantum adaptive distribution search. *Physical Review Research*, 6(2):023191, 2024.

[73] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for bayesian optimization in embedded subspaces. In *International Conference on Machine Learning*, pages 4752–4761. PMLR, 2019.

[74] Dejan Nickovic and Tomoya Yamaguchi. RTAMT: Online robustness monitors from STL. In *Automated Technology for Verification and Analysis (ATVA)*, volume 12302 of *LNCS, Springer*, pages 564–571, 2020.

[75] László Pál, Tibor Csendes, Mihály Csaba Markót, and Arnold Neumaier. Black box optimization benchmarking of the global method. *Evolutionary computation*, 20(4):609–639, 2012.

[76] Remigijus Paulavičius, Lakhdar Chiter, and Julius Žilinskas. Global optimization based on bisection of rectangles, function values at diagonals, and a set of lipschitz constants. *Journal of Global Optimization*, 71:5–20, 2018.

[77] Giulia Pedrielli, Tanmay Khandait, Yumeng Cao, Quinn Thibeault, Hao Huang, Mauricio Castillo-Effen, and Georgios Fainekos. Part-x: A family of stochastic algorithms for search-based test generation with probabilistic guarantees. *IEEE Transactions on Automation Science and Engineering*, 2023.

[78] Nikolaos Ploskas and Nikolaos V Sahinidis. Review and comparison of algorithms and software for mixed-integer derivative-free optimization. *Journal of Global Optimization*, 82(3):433–462, 2022.

[79] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International conference on parallel problem solving from nature*, pages 249–257. Springer, 1994.

[80] Remy Priem, Hugo Gagnon, Ian Chittick, Stephane Dufresne, Youssef Diouane, and Nathalie Bartoli. An efficient application of bayesian optimization to an industrial mdo framework for aircraft design. In *AIAA Aviation 2020 Forum*, 2020. June 15-19, held virtually, 1-16.

[81] V Protopopescu and J Barhen. Quantum algorithm for continuous global optimization. In *Optimization and Control with Applications*, pages 293–303. Springer, 2005.

[82] Mohammad Rahimi, Richard Pon, William Kaiser, Gaurav Sukhatme, Deborah Estrin, and Mani Srivastava. Adaptive sampling for environmental robotics. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2004. April 26$^{th}$-May 1$^{st}$, New Orleans, LA, 3537–3544.

[83] Pritam Ranjan, Derek Bingham, and George Michailidis. Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, 50(4):527–541, 2008.

[84] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. The MIT Press, Cambridge, Mass, 2005.

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

[86] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, 22:400–407, 1951.

[87] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics*, pages 298–307. PMLR, 2018.

[88] H. E. Romeijn and R. L. Smith. Simulated annealing and adaptive search in global optimization. *Probability in the Engineering and Informational Sciences*, 8:571–590, 1994.

[89] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

[90] Giuseppe E Santoro, Roman Martonák, Erio Tosatti, and Roberto Car. Theory of quantum annealing of an ising spin glass. *Science*, 295(5564):2427–2430, 2002.

[91] Andrej Schwanke, Lyubomir Ivanov, David Salinas, Fabio Ferreira, Aaron Klein, Frank Hutter, and Arber Zela. Improving llm-based global optimization with search space partitioning. *arXiv preprint arXiv:2505.21372*, 2025.

[92] Sara Shashaani. Simulation optimization: An introductory tutorial on methodology. In *2024 Winter Simulation Conference (WSC)*, pages 1308–1322. IEEE, 2024.

[93] Shubhanshu Shekhar and Tara Javidi. Multiscale gaussian process level set estimation. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 3283–3291. PMLR, 16–18 Apr 2019.

[94] Yanfang Shen. Annealing Adaptive Search With Hit-and-Run Sampling Methods for Global Optimization. *University of Washington Dissertation*, 2005.

[95] Yanfang Shen, Seksan Kiatsupaibul, Zelda B. Zabinsky, and Robert L. Smith. An analytically derived cooling schedule for simulated annealing. *Journal of Global Optimization*, 38(3):333–365, 2007.

[96] Yanfang Shen, Seksan Kiatsupaibul, Zelda B. Zabinsky, and Robert L. Smith. An analytically derived cooling schedule for simulated annealing. *Journal of Global Optimization*, 38(3):333–365, 2007.

[97] L. Shi and S. Òlafsson. *Nested Partitions Method, Theory and Applications*. Springer, Boston, MA, 2009.

[98] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[99] Wenjie Sun, Zhaolin Hu, and L. Jeff Hong. Gaussian mixture model-based random search for continuous optimization via simulation. In *Proceedings of the 2018 Winter Simulation Conference*, page 2003–2014. IEEE, 2018.

[100] Kei Terayama, Masato Sumita, Ryo Tamura, and Koji Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

[101] Quinn Thibeault, Jacob Anderson, Aniruddh Chandratre, Giulia Pedrielli, and Georgios Fainekos. PSY-TaLiRo: A python toolbox for search-based test generation for cyber-physical systems. In *Formal Methods for Industrial Critical Systems (FMICS)*, 2021.

[102] Yi-An Tsai, Giulia Pedrielli, Logan Mathesen, Zelda B. Zabinsky, Hao Huang, Antonio Candelieri, and Riccardo Perego. Stochastic optimization for feasibility determination: An application to water pump operation in water distribution. In M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, editors, *Proceedings of 2018 Winter Simulation Conference (WSC)*, pages 1945–1956. Institute of Electrical and Electronics Engineers, Inc., 2018.

[103] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *Advances in Neural Information Processing Systems*, 33:19511–19522, 2020.

[104] Xiaojun Wang, Zhenghuan Wang, and Bowen Ni. Mapping structural topology optimization problems to quantum annealing. *Structural and Multidisciplinary Optimization*, 67(5):74, 2024.

[105] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando De Feitas. Bayesian

optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

[106] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for bayesian optimization. *Advances in neural information processing systems*, 31, 2018.

[107] Graham R. Wood and Zelda B. Zabinsky. Stochastic adaptive search. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization*, volume 2, pages 231–249. Springer US, Boston, MA, 2002.

[108] Graham R. Wood, Zelda B. Zabinsky, and Birna P. Kristinsdottir. Hesitant adaptive search: the distribution of the number of iterations to convergence. *Mathematical Programming*, 89(3):479–486, 2001.

[109] Jie Xu, Barry L Nelson, and L Jeff Hong. An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS Journal on Computing*, 25(1):133–146, 2013.

[110] Zelda B Zabinsky. *Stochastic adaptive search for global optimization*. Springer Science & Business Media, New York, 2003.

[111] Zelda B Zabinsky. Stochastic search methods for global optimization. *Wiley Encyclopedia of Operations Research and Management Science*, 2011:1:10, 2011.

[112] Zelda B. Zabinsky, David Bulger, and Charoenchai Khompatraporn. Stopping and restarting strategy for stochastic sequential search in global optimization. *Journal of Global Optimization*, 46:273–286, 2010.

[113] Zelda B. Zabinsky and Hao Huang. A partition-based optimization approach for level set approximation: Probabilistic branch and bound. In Alice Smith, editor, *Women in Industrial and Systems Engineering: Key Advances and Perspectives on Emerging Topics*, pages 113–158. Springer Nature, Cham, Switzerland, 2020.

[114] Zelda B. Zabinsky and Hao Huang. A partition-based optimization approach for level set approximation: Probabilistic branch and bound. In Alice E. Smith, editor, *Women in Industrial and Systems Engineering: Key Advances and Perspectives on Emerging Topics*, pages 113–155. Springer Nature, 2020.

[115] Zelda B. Zabinsky and David D. Linz. Hesitant adaptive search with estimation and quantile adaptive search for global optimization with noise. *Journal of Global Optimization*, 87:31–55, 2023.

[116] Zelda B. Zabinsky and Robert L. Smith. Pure adaptive search in global optimization. *Mathematical Programming*, 53(1-3):323–338, 1992.

[117] Zelda B Zabinsky, Robert L Smith, J Fred McDonald, H Edwin Romeijn, and David E Kaufman. Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3(2):171–192, 1993.

[118] Zelda B. Zabinsky, Graham R. Wood, Mike A. Steel, and William P. Baritompa. Pure adaptive search for finite global optimization. *Mathematical Programming*, 69(1-3):443–448, 1995.

[119] Jianyuan Zhai and Fani Boukouvala. Data-driven spatial branch-and-bound algorithms for box-constrained simulation-based optimization. *Journal of global optimization*, 82(1):21–50, 2022.

[120] Zongzhao Zhou, Yew Soon Ong, My Hanh Nguyen, and Dudy Lim. A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In *2005 IEEE congress on evolutionary computation*, volume 3, pages 2832–2839. IEEE, 2005.

[121] Juliusz Krzysztof Ziomek and Haitham Bou Ammar. Are random decompositions all we need in high dimensional bayesian optimisation? In *International Conference on Machine Learning*, pages 43347–43368. PMLR, 2023.