LUND UNIVERSITY

Bayesian optimization in high dimensions

a journey through subspaces and challenges

Papenmeier, Leonard

2025

Link to publication

Citation for published version (APA):
Papenmeier, L. (2025). *Bayesian optimization in high dimensions: a journey through subspaces and challenges*. Computer Science, Lund University.

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Download date: 12. Jun. 2026

Bayesian Optimization in High Dimensions

Bayesian Optimization in High Dimensions

A Journey Through Subspaces and Challenges

by Leonard Papenmeier

LUND
UNIVERSITY

Thesis for the degree of Doctor of Philosophy
Thesis advisors: Associate Professor Luigi Nardi, Professor Jacek Malec
Faculty opponent: Associate Professor Roman Garnett

To be presented, with the permission of the Faculty of Engineering (LTH) of Lund University, for public criticism in room E:1406 at the Department of Computer Science on Thursday, the 12th of June 2025 at 13:15.

Organization
LUND UNIVERSITY

Department of Computer Science
Box 118
Klas Anshelms väg 10 – SE–223 63 Lund
Sweden

Document name
DOCTORAL DISSERTATION

Date of disputation
2025-06-12

Sponsoring organization The Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Author(s)
Leonard Papenmeier

Title and subtitle
Bayesian Optimization in High Dimensions: A Journey Through Subspaces and Challenges

Abstract
This thesis explores the challenges and advancements in high-dimensional Bayesian optimization (HDBO), focusing on understanding, quantifying, and improving optimization techniques in high-dimensional spaces. Bayesian optimization (BO) is a powerful method for optimizing expensive black-box functions, but its effectiveness diminishes as the dimensionality of the search space increases due to the curse of dimensionality. The thesis introduces novel algorithms and methodologies to make HDBO more practical. Key contributions include the development of the `BAxUS` algorithm, which leverages nested subspaces to optimize high-dimensional problems without estimating the dimensionality of the effective subspace. Additionally, the `Bounce` algorithm extends these techniques to combinatorial and mixed spaces, providing robust solutions for real-world applications. The thesis also explores the quantification of exploration in acquisition functions, proposing new methods of quantifying exploration and strategies to design more effective optimization approaches. Furthermore, this work analyzes why simple BO setups have recently shown promising performance in high-dimensional spaces, challenging the conventional belief that BO is limited to low-dimensional problems. This thesis offers insights and recommendations for designing more efficient HDBO algorithms by identifying and addressing failure modes such as vanishing gradients and biases in model fitting. Through a combination of theoretical analysis, empirical evaluations, and practical implementations, this thesis contributes to the field of BO by advancing our understanding of high-dimensional optimization and providing actionable methods to improve its performance in complex scenarios.

Key words
Optimization, Bayesian Optimization, Black-box optimization, Gaussian process, Machine Learning, Artificial Intelligence

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title
1404 – 1219

ISBN
978-91-8104-547-5 (print)
978-91-8104-548-2 (pdf)

Recipient's notes

Number of pages
318

Price

Security classification

I, the undersigned, being the copyright owner of the abstract of the above-mentioned dissertation, hereby grant to all reference sources the permission to publish and disseminate the abstract of the above-mentioned dissertation.

Signature _____ Date 2025-5-12

DOKUMENTDATABLAD enl SIS 61 41 21

Bayesian Optimization in High Dimensions

A Journey Through Subspaces and Challenges

by Leonard Papenmeier

LUND
UNIVERSITY

.

Cover illustration: Andrea Papenmeier

Funding information: This thesis was funded by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

© Leonard Papenmeier 2025

Faculty of Engineering (LTH), Department of Computer Science

ISBN: 978-91-8104-547-5 (print)
ISBN: 978-91-8104-548-2 (pdf)
ISSN: 1404 – 1219
DISSERTATION 81, 2025
LU-CS-DISS: 2025-03
Printed in Sweden by Tryckeriet i E-Huset, Lund 2025

Contents

List of publications ix
Acknowledgements xi
Popular summary in English xii

Bayesian Optimization in High Dimensions: A Journey Through Subspaces and Challenges **1**
1 Introduction 1
1.1 Motivation 1
1.2 Research Questions 3
1.3 Thesis Outline 4
2 Bayesian Optimization 6
2.1 The Optimization Objective 6
2.2 Gaussian Processes 8
2.3 Gaussian Process Fitting. 12
2.4 Acquisition Functions 17
2.5 Combinatorial Optimization 21
2.6 Acquisition Function Maximization 22
3 High-Dimensional Bayesian Optimization 26
3.1 Additive Models 26
3.2 Subspace Methods 27
3.3 Trust-Region Methods 32
3.4 Monte-Carlo Tree Search 35
3.5 Long Length Scales 37
3.6 Vanishing Gradients 38
4 Contributions 41
4.1 RQ 1: Understanding and Addressing Failure Modes in High-Dimensional Bayesian Optimization 41
4.2 RQ 2: Practical High-Dimensional Bayesian Optimization 43
5 Conclusions and Future Work 45

1 Scientific publications **63**
Author contributions 63

Paper I: Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces 64
Paper II: High-dimensional Bayesian Optimization with Group Testing 64
Paper III: Bounce: Reliable High-Dimensional Bayesian Optimization for Combinatorial and Mixed Spaces 64
Paper IV: Exploring Exploration in Bayesian Optimization 64
Paper V: A Unified Framework for Entropy Search and Expected Improvement in Bayesian Optimization 65
Paper VI: Understanding High-Dimensional Bayesian Optimization 65

Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces **69**
1 Introduction 70
2 Background 71
3 The BAxUS Algorithm 73
3.1 The sparse BAxUS subspace embedding 75
3.2 Trust-region approach 78
3.3 Splitting Strategy 78
3.4 Controlling the Number of Accepted Failures 79
4 Experimental Evaluation 80
4.1 Experimental Results 81
5 Discussion 83
A Theoretical Foundation for the BAxUS Embedding 85
A.1 Proof of Theorem 1 85
A.2 Proof of Corollary 2 87
A.3 Proof of Corollary 1 92
B Consistency of BAxUS 94
C Additional Empirical Evaluations 96
C.1 Ablation Study for the BAxUS Embedding 96
C.2 Evaluation on an Additional Lasso Benchmark 99
C.3 Evaluation on Additional MuJoCo Benchmarks 100
D The Nested Family of Random Embeddings 100
E Additional Details on the Implementation and the Empirical Evaluation 102

High-dimensional Bayesian Optimization with Group Testing **113**
1 Introduction 114
2 Background 115
2.1 High-Dimensional Bayesian Optimization 115
2.2 Low-Dimensional Subspace Bayesian Optimization 116
2.3 Group Testing 117

3 Group Testing for Bayesian Optimization 118
4 Computational Experiments 123
4.1 Experimental Setup 123
4.2 Performance of the Group Testing 124
4.3 Optimization of Real-World and Synthetic Benchmarks . . 125
5 Discussion 127
A The GTBO Algorithm 129
B Comparison with Feature Importance Algorithms 130
C Number of Active Variables 131
D Hyperparameter Analysis 131
E Run Times 132

Bounce:
Reliable High-Dimensional Bayesian Optimization for Combinatorial and Mixed Spaces **143**
1 Introduction 144
2 Background and Related Work 145
3 The Bounce Algorithm 147
3.1 The Subspace Embedding of Mixed Spaces 151
3.2 Maximization of the Acquisition Function 153
4 Experimental Evaluation 153
4.1 50D Low-Autocorrelation Binary Sequences (LABS) 154
4.2 Industrial Maximum Satisfiability: 125D ClusterExpansion benchmark 154
4.3 25D Categorical Pest Control 155
4.4 SVM – a 53D AutoML task 155
4.5 Bounce's efficacy for batch acquisition 156
4.6 The sensitivity of BODi and COMBO to the location of the optima 156
5 Discussion 158
A Consistency of Bounce 161
B Additional Experiments 163
B.1 Bounce and other algorithms on additional benchmarks . 163
C An evaluation of Bounce on continuous problems 166
D Batched evaluations on mixed and combinatorial problems 167
E Low-sequency version of Bounce 168
F Effect of trust-region management 169
G Implementation Details 170
G.1 Optimization of the acquisition function 172
G.2 Kernel choice 175
H Additional Analysis of BODi and COMBO 175

H.1 Analysis of BODi . . . 175
H.2 COMBO on Categorical Problems . . . 180

Exploring Exploration in Bayesian Optimization **193**
1 Introduction . . . 194
2 Background and Related Work . . . 195
2.1 Gaussian Processes . . . 195
2.2 Acquisition Functions . . . 195
2.3 Expressions of Exploration . . . 197
3 Quantifying Exploration . . . 198
3.1 Analysis of Tribal Knowledge . . . 198
3.2 Exploration Methods . . . 199
3.3 Exploration Bounds . . . 202
4 Experiments . . . 203
4.1 Experimental Setup . . . 204
4.2 Empirical Validation of OTSD and OE . . . 205
4.3 Synthetic Benchmarks . . . 206
4.4 Real-World Benchmarks . . . 207
4.5 Exploration Taxonomy . . . 208
5 Conclusion . . . 209
A Proofs . . . 210
A.1 OTSD Upper Bound . . . 210
A.2 OE Upper Bound . . . 211
A.3 KL Estimation Consistency and Bias . . . 212
B Details on the Experimental Setup . . . 213
C Empirical Verification of the OTSD Bound . . . 213
D Additional Experiments . . . 214
D.1 Empirical Validation of OTSD and OE . . . 214
D.2 Error Bars for Main Text Figures . . . 215
D.3 TRs, RAASP, and batching on the Real-World Benchmarks 217
D.4 Impact of the Dimensionality . . . 218
D.5 Optimizer Variations on Different Acquisition Functions . 219
D.6 Optimization Performance . . . 223
E Runtime Analysis . . . 228

A Unified Framework for Entropy Search and Expected Improvement in Bayesian Optimization **239**
1 Introduction . . . 240
2 Background and Related Work . . . 241
2.1 Gaussian Processes . . . 241
2.2 Acquisition Functions . . . 242
2.3 Related Work . . . 243

3 Variational Entropy Search . . . 244
3.1 Entropy Search Lower Bound . . . 244
3.2 EI Through the Lens of the VES Framework . . . 245
3.3 VES-Gamma: A Generalization of EI . . . 247
4 Results . . . 249
4.1 Experimental Setup . . . 249
4.2 Comparing VES-Exp and EI . . . 251
4.3 Performance of VES-Gamma . . . 252
5 Conclusion . . . 254
A Proofs . . . 256
A.1 ESLB Proof . . . 256
A.2 VES-Exp and EI Algorithmic Equivalence . . . 256
B Additional Experimental Results . . . 257
B.1 Synthetic Test Functions . . . 257
C Kolmogorov-Smirnov Test Statistic . . . 258
D VES-Gamma Computational Acceleration . . . 259

Understanding High-Dimensional Bayesian Optimization **267**
1 Introduction . . . 268
2 Problem Statement and Related Work . . . 269
3 Facets of the Curse of Dimensionality . . . 270
3.1 Vanishing Gradients at Model Fitting . . . 271
3.2 Vanishing Gradients of the Acquisition Function . . . 273
3.3 Bias-Variance Trade-off for fitting the GP . . . 277
4 Discussion . . . 279
5 Conclusions and Future Work . . . 283
A A Review of Gaussian Processes and Bayesian Optimization . . . 284
A.1 Gaussian Processes . . . 284
A.2 Bayesian Optimization . . . 285
B Additional Experiments . . . 287
B.1 Ranking of Optimization Algorithms . . . 287
B.2 MLE Gradients for Real-World Experiments . . . 287
B.3 High-dimensional Benchmark Functions . . . 288
B.4 Hard Optimization Problems . . . 291
C Popular Benchmarks Seem Simpler Than Expected . . . 292

Acronyms

AF	acquisition function
ARD	automatic relevance determination
BA	Barber-Agakov
BO	Bayesian optimization
CDF	cumulative density function
COD	curse of dimensionality
DKL	deep kernel learning
DM	deterministic selection
DoE	design of experiment
DSP	dimensionality-scaled prior
ECDF	empirical cumulative distribution function
EETO	exploration-exploitation trade-off
EI	expected improvement
ELBO	evidence lower bound
EM	expectation maximization
ES	entropy search
ESLBO	entropy search lower bound
EETO	exploration-exploitation trade-off
GD	gradient descent
GMM	Gaussian mixture model
GP	Gaussian process
GT	group testing
HDBO	high-dimensional Bayesian optimization
HPO	hyperparameter optimization
HP	hyperparameter
JES	joint entropy search
KG	knowledge gradient
KL	Kozachenko-Leonenko
KS	Kolmogorov-Smirnov

L-BFGS-B limited-memory Broyden-Fletcher-Goldfarb-Shanno with bound constraints

MAB multi-armed bandit

MAP maximum a-posteriori estimation

MCMC Markov-Chain Monte-Carlo

MDP Markov decision process

MES max-value entropy search

MI mutual information

ML machine learning

MLE maximum-likelihood estimation

MLL marginal log-likelihood

MCTS Monte-Carlo tree search

NN neural network

NUTS no u-turn sampler

OE observation entropy

OTSD observation traveling salesman distance

PDF probability density function

PES predictive entropy search

PI probability of improvement

PLS partial least squares

psd positive semi-definite

qEI q-expected improvement

RAASP random axis-aligned subspace perturbations

RBF radial basis function

RL reinforcement learning

RS random search

RV random variable

SMC sequential Monte Carlo

SOTA state-of-the-art

SVM support vector machine

SVR support vector regression

TCD total center distance

TR	trust region
TS	Thompson sampling
TSP	traveling salesman problem
UCB	upper-confidence bound
VAE	variational autoencoder
VarPro	variable projection
VES	variational entropy search
VI	variational inference

List of publications

This thesis is based on the following publications, referred to by their Roman numerals:

I **Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces**

Leonard Papenmeier, Luigi Nardi, Matthias Poloczek

Advances in Neural Information Processing Systems 35, NeurIPS 2022

II **High-dimensional Bayesian Optimization with Group Testing**

Erik Hellsten*, Carl Hvarfner*, Leonard Papenmeier*, Luigi Nardi

Submitted to the Fourth International Conference on Automated Machine Learning

III **Bounce: Reliable High-Dimensional Bayesian Optimization for Combinatorial and Mixed Spaces**

Leonard Papenmeier, Luigi Nardi, Matthias Poloczek

Advances in Neural Information Processing Systems 36, NeurIPS 2023

IV **Exploring Exploration in Bayesian Optimization**

Leonard Papenmeier*, Nuojin Cheng*, Stephen Becker, Luigi Nardi

Submitted to the Forty-First Conference on Uncertainty in Artificial Intelligence

V **A Unified Framework for Entropy Search and Expected Improvement in Bayesian Optimization**

Nuojin Cheng*, Leonard Papenmeier*, Stephen Becker, Luigi Nardi

Accepted at the Forty-Second International Conference on Machine Learning

VI **Understanding High-Dimensional Bayesian Optimization**

Leonard Papenmeier, Matthias Poloczek, Luigi Nardi

Accepted at the Forty-Second International Conference on Machine Learning

All papers are reproduced with permission of their respective publishers.

*Equal contribution.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my wife, Andrea, for her support, patience, and encouragement throughout this journey. Your steady presence and calm perspective helped me through every setback and you celebrated every success by my side. You have always believed in me, even when I doubted myself. And beyond your emotional support, you contributed to this thesis in such a personal and beautiful way by creating the cover illustration. I truly could not have done this without you.

I am immensely grateful to my advisor, Luigi, from whom I have learned so much – far beyond what is written in these pages. Navigating the world of academia is not easy, but you have always been there to guide me. Your balanced personality and calm leadership helped us stay productive even under great pressure. You consistently set the highest standards and encouraged me to meet them. Your constructive criticism and insightful guidance have been invaluable and have helped me to grow as a researcher.

I would like to thank Matthias, who has played an instrumental role in shaping my PhD. From the very beginning, your input challenged me to think more critically and rigorously. You brought clarity and structure to my early work and were a sounding board for many ideas, big and small. Your feedback always came with depth and precision, and your involvement, especially in the beginning of my PhD, came close to that of a co-supervisor.

I would also like to thank Jacek, who has always been patient and supportive, even when things turned out differently than expected. Thank you for being especially supportive during the final stages of my PhD.

I am deeply thankful to my parents for their lifelong support and the values they instilled in me. They taught me the importance of education, and I am forever grateful for their encouragement and belief in me.

To my colleagues Carl, Erik, and Kenan – thank you for the many insightful discussions and your ongoing support. Finally, to all those who have supported me in ways big and small – thank you.

Leonard Papenmeier
Lund 2025

Popular summary in English

Optimization pervades various disciplines: a car should have as little wind resistance as possible, a financial portfolio should have a maximum expected return, and a new material should be as stable as possible. For the example of the car, a simple strategy could modify the car body and measure the wind resistance after each modification. Each such trial is expensive, imposing a low limit on the number of trials.

Bayesian optimization is a method for solving such problems automatically by iteratively interleaving two steps: 1) proposing and evaluating new configurations and 2) refining a model of the function one aims to optimize using all configurations evaluated so far. In the example of the car, the function models the wind resistance of a given configuration. In cases where only a few design variables (or "parameters") define a car body, Bayesian optimization is a popular approach since it often requires fewer trials to find a given solution than other methods. This is referred to as Bayesian optimization being "sample efficient". However, as the number of parameters increases, Bayesian optimization requires more trials so that Bayesian optimization quickly becomes prohibitively expensive, even for a moderate number of parameters. Problems with hundreds or even thousands of parameters are what we refer to as high-dimensional problems

In many cases, parameters cannot take arbitrary values. They may only allow for integers (for instance, the number of windows of a car) or categorical values (for example, the color of the car body). Paradoxically enough, finding the optimal values for such integer-valued or categorical parameters is often harder than if they allowed for arbitrary values. This thesis proposes new methods to carry over the sample efficiency of Bayesian optimization to high dimensions for real-valued problems and problems with categorical or integer-valued parameters.

Additionally, we study the problem of proposing new configurations to test. We propose methods that analyze important properties of common techniques for proposing new configurations and propose new strategies that, in many scenarios, are more sample efficient than other methods.

Finally, we investigate other works tackling high-dimensional Bayesian optimization and identify previously unknown reasons why Bayesian optimization struggles in such settings. Based on these insights, we propose simple strategies to avoid these issues that can be combined with many existing techniques, as they are foundational in nature and can therefore be combined with other techniques.

By examining the problem from multiple angles, we seek a holistic view of the possibilities and limitations of high-dimensional Bayesian optimization.

Bayesian Optimization in High Dimensions: A Journey Through Subspaces and Challenges

1 Introduction

1.1 Motivation

Many impactful applications require the optimization of *black-box* problems where the expression of the function one aims to optimize is unknown. Examples include chemical engineering [44, 103, 39, 100, 9], materials science [115, 31, 87, 106, 42, 40, 47], aerospace engineering [69, 62, 70], hyperparameter optimization [104, 59, 6, 50], neural architecture search [58, 97], vehicle design [55, 12], hardware design [79, 26], drug discovery [82, 109], robotics [67, 11, 90, 10, 73], and life sciences [111, 101, 13]. For instance, in aerospace engineering, one is interested in finding lightweight aircraft structures, e.g., an optimal wing structure at minimum weight. The ideal wing configuration remains unknown, and new wing structures can only be evaluated with an expensive simulation. In such a scenario, one tries to optimize the structure with as few tries as possible [70]. Another example is material sciences, where certain materials (so-called halide perovskites) can produce cheaper and more efficient solar cells. A halide perovskite has a fixed structural formula ABX_3, and the task is to find atoms A, B, and X so that the resulting material has the best possible properties [71, 89]. For instance, halide perovskites are candidate materials for solar cells, i.e., the goal is to find materials with desirable photovoltaic properties. Screening candidates is an expensive process involving synthesizing and testing the material, which limits the number of evaluations that can be run [89]. Finally, in drug design, one needs to find promising molecules with as few trials as possible, as every trial not only requires expensive lab material but also involves a process that can take several days [81].

While cheaper-to-produce solar cells, lighter aircraft, or novel drugs have many societal benefits, the challenges in solving those black-box problems are manifold. First, many problems have *constraints*. One cannot reduce the weight of an aircraft at all costs, but needs to consider structural requirements. Second, many problems are *noisy*. Synthesizing and testing a material is a complex process, and repeating the same process can yield different results. Third, many problems require optimizing *several and potentially competing objectives*. New solar cells should be effective but also cheap to produce. Finally, many problems are *high-dimensional*. Aircraft structures can be modeled with hundreds of design variables [70], and drugs can be parametrized with thousands of features [109].

Bayesian optimization (BO) [74, 75, 102, 30, 34] is a technique for addressing the aforementioned challenges. It uses a probabilistic surrogate to model the unknown objective function. After observing some values, BO chooses new points to evaluate. This is done using an acquisition function (AF) that suggests points likely to have good values or lie in unexplored regions of the search space, trading off exploitation and exploration. This tradeoff makes BO act globally: the search not only focuses on promising regions but also aims at learning more about the function in regions that have not yet been visited. By iteratively observing new points and refining the surrogate model, BO can optimize functions of unknown form. Additionally, BO can naturally handle noisy observations and has been extended to constrained and multi-objective settings.

Whether BO will be successful for a specific application mainly hinges on three factors: 1) if the underlying objective function can be modeled by a chosen class of surrogate models, 2) the ability of the AF to propose promising candidates, and 3) whether a global optimum can be found with a limited number of function evaluations. This thesis focuses on the latter two aspects.

If an AF chooses "good candidates" depends highly on the specific scenario. In cases where one can afford to evaluate the function thousands of times and the problem's dimensionality is low, one may aim at learning the function as accurately as possible to find all local minima. In contrast, if the problem is high-dimensional and only a few dozen evaluations are feasible, trying to learn the function globally is likely hopeless. In such cases, a possible strategy is to focus on finding a single local optimum rather than uncovering all local and global minima. Ideally, the AF's behavior should suit the scenario. In practice, however, it's often unclear how a given AF will behave. This uncertainty leads practitioners to repeatedly use a familiar default AF, due to a lack of insight into the alternatives. Furthermore, the relationship between different AFs is poorly understood.

For many applications, whether the function can be learned globally is connected

to a problem's dimensionality. While problems that are noisy, constrained, or multi-objective often remain challenging, high-dimensional Bayesian optimization (HDBO) has been considered a particularly difficult problem: BO was believed to be limited to approximately 20 dimensions for common evaluation budgets [30, 76]. Extending BO to higher-dimensional spaces was considered as one of the "holy grails" [120] and "one of the most important goals" [80] of the field. The root of the problem of HDBO is the *curse of dimensionality (COD)* [60], which, with linearly growing dimensionality, demands exponentially more function observations to maintain the same level of accuracy. This conflicts with the prevalent scenario of data scarcity in BO [30], constituting a fundamental dilemma. Recent advances have proposed techniques to make BO amenable to problems with hundreds of input parameters [120, 80, 29, 28, 64, 57]. However, most approaches require the objective function to have a specific structure – for instance, an additive structure or a low-dimensional effective subspace outside of which function values remain constant. To make things worse, most subspace-based approaches require the user to provide the often *unknown* dimensionality of the effective subspace as an input to the algorithm [120, 80, 29, 64]. This severely limited the usability of these algorithms in many practical domains, opening a gap for robust HDBO methods without additional assumptions on the objective.

In many cases, the problem is not only high-dimensional but also exhibits combinatorial parameters – such as binary, categorical, or ordinal parameters. For example, in hyperparameter optimization, the learning rate of an optimizer is a continuous parameter, while the number of training epochs is an integer-valued parameter. The ability to cover these complex domains is an additional requirement for practical HDBO algorithms.

Finally, despite the widely accepted rule that in more than 20 dimensions, BO severely suffers from the COD due to the limited evaluation budget, recent works have shown state-of-the-art (SOTA) empirical performance on well-established problems with hundreds or thousands of parameters using simple BO setups [51, 127] – raising the question of whether it was a misbelief that BO only works well for low-dimensional problems and requires additional assumptions on the objective function to scale to higher dimensions.

1.2 Research Questions

The behavior of BO in high dimensions is poorly understood. The widespread belief that BO does not scale to high dimensions led to the development of different classes of algorithms, exploiting different properties of the black-box function such as additivity or a low-dimensional effective subspace. However, this belief has been

recently put into question by two works reporting that simple BO setups mostly work "out of the box" for high-dimensional problems [51, 127]. This poses questions on whether more sophisticated algorithms with additional assumptions are even necessary, and what the reasons are for the widespread belief that BO does not scale well to high-dimensional domains.

Moreover, it is often unclear how different classes of AFs behave, and what the effect of a problem's dimensionality is on their behavior. This lack of insight makes it difficult to choose an appropriate AF for a given HDBO problem.

Finally, existing HDBO methods are often difficult to deploy in practice. Several methods require the user to specify the *effective dimensionality* of a problem, a property that is usually unknown [120, 80, 64]. Other approaches require large amounts of unsupervised training data [113, 76, 68, 72], which is not always available. However, most approaches for combinatorial HDBO rely on unsupervised training data, leaving a gap for *data-agnostic* HDBO methods such as subspace-based approaches for combinatorial variables. In short, there is a need for easy-to-deploy methods that support continuous and combinatorial spaces.

This leads to the following research questions:

RQ 1 : How can we deepen our understanding of high-dimensional Bayesian optimization methods and prevent common failure modes?

RQ 2 : How can we mitigate the need for prior knowledge in high-dimensional Bayesian optimization methods to achieve greater ease of use while maintaining a high level of performance?

- RQ 2.1 : How can we design SOTA high-dimensional Bayesian optimization methods that can operate on a diverse set of applications of varying complexity without making explicit prior assumptions about the structure of the black-box function f?
- RQ 2.2 : How can we learn relevant information about the structure of a high-dimensional application and exploit this information to guide the optimization?

1.3 Thesis Outline

The thesis consists of two parts. The first part serves as the *kappa* (Swedish for *coat*). It introduces the research field and aligns the work conducted in this thesis with the broader field. The second part contains the individual papers that comprise the thesis. The structure is as follows:

- Section 1 motivates the relevance of the research topic and places this thesis in the context of the broader field.
- Section 2 introduces the framework of *Bayesian optimization*, its main components, as well as necessary steps for the practical deployment of the surrogate model and acquisition function.
- Section 3 discusses the subproblem and core topic of this thesis, *high-dimensional Bayesian optimization*, highlights important techniques, and discusses their relation to this thesis.
- Section 4 outlines the contributions of this thesis and how they relate to the research field.
- Section 5 gives final conclusions and discusses avenues for future work.

The following sections present the papers that compose the thesis.

- Paper I addresses RQ 2.1 by introducing `BAxUS`, an algorithm for practical HDBO in continuous spaces. `BAxUS` leverages the assumption of an effective subspace to carry out the optimization in subspaces of increasing dimensionality.
- Paper II addresses axis-aligned problems, where only a subset of parameters influences the function value. `GTBO` identifies these active dimensions and leverages them to steer the optimization, directly tackling RQ 2.2.
- Paper III presents `Bounce`, a HDBO algorithm that extends `BAxUS`' nested subspaces to combinatorial and mixed spaces. `Bounce` also improves over `BAxUS` on purely continuous spaces by revising several design decisions and addressing RQ 2.1.
- Paper IV introduces two new methods, observation traveling salesman distance (OTSD) and observation entropy (OE), to study the degree of exploration in a BO strategy's behavior. The two methods are used to analyze and relate popular BO strategies and can detect failure modes of BO. This paper addresses RQ 1.
- Paper V contributes a novel AF, variational entropy search (VES), which bridges the gap between two different types of AFs (*myopic* and *non-myopic*) and improves over the SOTA on several problems. This paper addresses RQ 1.
- Paper VI identifies common failure modes of HDBO methods by focusing on fundamental aspects of BO like the model fitting and the optimization of the AF. Based on these insights, paper VI presents a novel and simple, yet effective, BO algorithm. This paper addresses RQ 1.

2 Bayesian Optimization

2.1 The Optimization Objective

The goal of Bayesian optimization (BO) [74, 75, 102, 30, 34] is to find the optimum $\boldsymbol{x}^*$ of some objective function f. Throughout this thesis, we consider minimization problems, i.e., we aim to find

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}). \tag{1}$$

Note that any maximization problem h can be framed as a minimization problem by defining $f := -h$. Hence, restricting the analysis to minimization problems does not lose generalizability. The function f is assumed to be expensive to evaluate – substantially limiting the number of evaluations – while providing no gradient information and potentially being noisy. For noisy f, we can only observe $y = f(\boldsymbol{x}) + \varepsilon$ but not $f(\boldsymbol{x})$ alone where ε is distributed according to the noise distribution, often assumed to be a normal distribution.

The Domain. The domain of f, $\mathcal{X}$, is called the *search space*. For purely continuous problems, one often defines lower and upper bounds ($b^{(i)}_{\text{lower}}$ and $b^{(i)}_{\text{upper}}$) for each parameter i so that $\mathcal{X} = \bigtimes_{i=1}^{D} [b^{(i)}_{\text{lower}}, b^{(i)}_{\text{upper}}]$ where D is the *dimensionality* of the problem. Since the domain of each parameter i can be normalized to be in $[0, 1]$ by using min-max scaling

$$\hat{x}_i = \frac{x_i - b^{(i)}_{\text{lower}}}{b^{(i)}_{\text{upper}} - b^{(i)}_{\text{lower}}}, \tag{2}$$

practical implementations usually define $\mathcal{X} = [0, 1]^D$ and consider scaled observations.

In many applications, the search space is more exotic [77, 52, 85, 37]. For instance, a boolean satisfiability problem only has binary parameters, and the search space takes the binary form $\mathcal{X} = \{0, 1\}^D$. Other parameters have an order but are not continuous. For example, an *ordinal* parameter might only take values $[0.5, 1.2, 7]$. Parameters that can only assume finitely many values but do not have an order are called *categorical*. For instance, a parameter with five possible categories has domain $\{1, 2, 3, 4, 5\}$ without loss of generalizability, as any categorical parameter with five categories can be mapped to this domain by renaming the categories. Search spaces can also contain parameters of different types. A typical example is the hyperparameter optimization (HPO) of machine learning (ML) algorithms [48, 104, 110]. For example, for a neural network (NN), the learning rate is continuous, the number of layers is ordinal, and the optimizer type is categorical.

Algorithm 1 Bayesian Optimization

Input: black-box function f, number of design of experiment samples n_{init}

observe f n_{init} times and collect observations $\triangleright\ y_i = f(\boldsymbol{x}_i) + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ $\mathcal{D} \leftarrow \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{\text{init}}, y_{\text{init}})\}$.

while stopping criterion not fulfilled **do**

  fit a surrogate model g on $\mathcal{D}$

  $\boldsymbol{x}_{\text{next}} \leftarrow \arg\max_{\boldsymbol{x}} \alpha(\boldsymbol{x} \mid g)$ $\triangleright$ maximize AF

  $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{x}_{\text{next}}, y_{\text{next}})$ $\triangleright$ update data

Output: $\boldsymbol{x}_i : y_i \in \min \mathcal{D}|_{f(\boldsymbol{x})}$ **or** minimizer of the posterior mean

The Codomain. Similarly, the codomain of f can take various forms. For many problems, $f : \mathcal{X} \to \mathbb{R}$, i.e., the codomain coincides with the reals. For multi-objective problems (e.g, [88, 16, 15]) with real-valued outputs, $f : \mathcal{X} \to \mathbb{R}^{N_{\text{obj}}}$ where N_{obj} is the number of objectives. While many problems have non-real-valued codomains, we restrict the discussion to real-valued codomains for simplicity but note that many concepts also extend to other codomains.

The Surrogate Model. BO models the function f by means of a *probabilistic surrogate* g. Since f is only evaluated at a few discrete points, its values apart from these observations are unknown. If f can only be observed with noise, its exact value is unknown even at the observed locations. Surrogate models allow modeling the uncertainty about f. Given some observations $\mathcal{D}$, the surrogate model can be conditioned on this data such that $f(\boldsymbol{x}) \sim g(\boldsymbol{x} \mid \mathcal{D})$. For noiseless models, $g(\boldsymbol{x} \mid \mathcal{D})$ is deterministic at locations that have previously been observed [34]. We discuss one particularly popular class of surrogate models, Gaussian processes (GPs), in Section 2.2.

The Acquisition Function. The acquisition function (AF) is one of the two main components of BO, with the surrogate model being the other one. Based on the surrogate model, the AF chooses the next point or set of points to evaluate. Typically, AFs is designed to balance exploration and exploitation, i.e., they try to suggest both points from underexplored regions of the search space $\mathcal{X}$ as well as points that are likely to have good value $f(\boldsymbol{x})$ [34]. AFs are denoted $\alpha : \mathcal{X} \to \mathbb{R}$ and are usually maximized. Section 2.4 discusses important AFs.

The Prototypic Bayesian Optimization Algorithm. Having introduced the main components of BO, Algorithm 1 shows a prototypical BO algorithm. The surrogate

model is usually initialized with n_{init} observations, often sampled uniformly at random or using a scrambled Sobol sequence [86]. These initial points are called design of experiment (DoE) points and are used to initialize g with information about f. After "fitting" the surrogate g, which entails the conditioning on the observations and the optimization of the hyperparameters (HPs) of the surrogate – if there are any – the next point $\boldsymbol{x}_{\text{next}}$ is chosen by maximizing the AF α which is conditioned on g. The function f is then evaluated at the proposed location, and the observation is added to the data $\mathcal{D}$. This process is repeated until some stopping criterion is fulfilled. Often, the stopping criterion is defined as a threshold on the total number of function evaluations, but other criteria exist that attempt to detect when the optimization converges [83, 65, 122]. At the end of the optimization, one either returns the point with the best-observed value or the point for which the surrogate predicts the highest mean value. The former strategy is popular for noiseless models as, for the best-observed value, $y_{\text{best}} = f(\boldsymbol{x}_{\text{best}})$. If f is noisy, returning the point the surrogate believes is best is common since the observed function values can vary significantly from the ground truth.

2.2 Gaussian Processes

GPs [92] can be considered a distribution over functions even though, in practice, they only model the distribution of a finite set of points. A GP is defined by a mean function $\mu(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ and a kernel or covariance function $k(\boldsymbol{x}, \boldsymbol{x}') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. With these functions, any finite set $f(\boldsymbol{y})$ of observations of f at locations $\mathbf{X}$ is assumed to be jointly normally distributed, i.e.,

$$p(\boldsymbol{y} \mid \mathbf{X}) = \mathcal{N}(\mu(\mathbf{X}), \Sigma(\mathbf{X}, \mathbf{X})), \tag{3}$$

where

$$\mu(\mathbf{X}) = (\mu(\boldsymbol{x}_1), \dots, \mu(\boldsymbol{x}_N)) \tag{4}$$

and

$$\Sigma(\mathbf{X}, \mathbf{X}) = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}. \tag{5}$$

Often, the data is standardized and the mean function is chosen to be zero ($\mu(\boldsymbol{x}) = 0$), but other mean functions are used if prior information is available [89, 112, 19]. With a mean function $\mu(\boldsymbol{x}) = 0$, Equation (3) simplifies to

$$p(\boldsymbol{y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{0}, \Sigma(\mathbf{X}, \mathbf{X})). \tag{6}$$

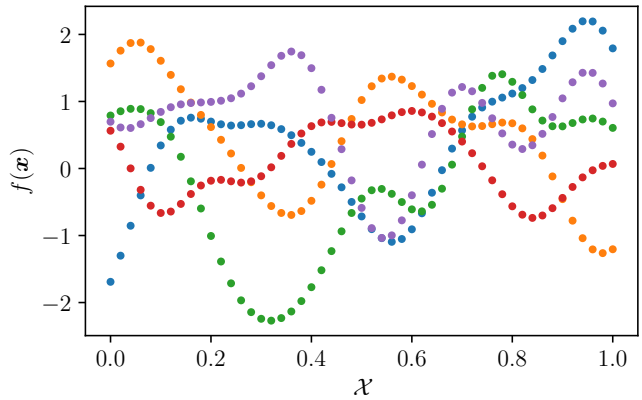


Figure 1: Five realizations drawn from a GP prior. Each realization is a 50-dimensional multivariate Gaussian distribution with mean zero and covariance $\Sigma(\mathbf{X}, \mathbf{X})$ where, in this case, $\mathbf{X} = (0, 0.02, 0.04, \ldots, 0.98)$.

We call Equation (3) the *GP prior* since it models the distribution over f before observing any data. For the remainder of this thesis, we assume $\mu(\boldsymbol{x})$ to be zero unless mentioned otherwise. Figure 1 exemplifies this for a one-dimensional objective function. We draw five realizations from a GP prior with zero mean and a $5/2$-Matérn covariance function on an evenly spaced grid $(0, 0.02, 0.04, \ldots, 0.98)$. Each realization is a multivariate normal distribution with a mean of zero and a 50×50 covariance matrix $\Sigma(\mathbf{X}, \mathbf{X})$.

Since GPs are nothing but "a collection of random variables, any finite number of which have a joint Gaussian distribution" [92], we can apply the standard conditioning rules for normal distributions. In particular, the predictive distribution when conditioning a GP on some observations $\mathcal{D} = (\boldsymbol{y}_{\text{obs}}, \mathbf{X}_{\text{obs}})$ is given by

$$\begin{aligned} p(\boldsymbol{y} \mid \mathbf{X}, \mathcal{D}) = \mathcal{N}(&\Sigma(\mathbf{X}, \mathbf{X}_{\text{obs}})\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}})^{-1}\boldsymbol{y}_{\text{obs}}, \\ &\Sigma(\mathbf{X}, \mathbf{X}) - \Sigma(\mathbf{X}, \mathbf{X}_{\text{obs}})\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}})^{-1}\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X})) \end{aligned} \tag{7}$$

Figure 2 shows five realizations drawn from the GP posterior when conditioning on two observations, marked as black crosses.

Noisy Observations. So far, we implicitly assumed that observations are noiseless. This becomes evident because the realizations in Figure 2 pass *exactly* through the observations – indicating our belief that these observations reflect the ground truth. In practice, many problems can only be observed with noise, i.e., $y = f(\boldsymbol{x}) + \varepsilon$. Often, epsilon is assumed to be homoscedastic and normally distributed, i.e., $\varepsilon \sim$

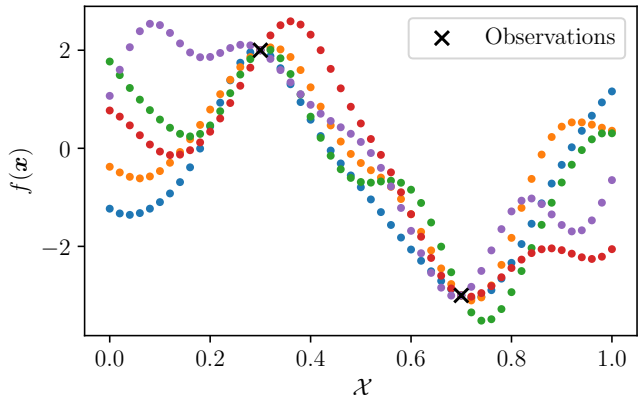


Figure 2: Five realizations drawn from a GP posterior. The GP is conditioned on two observations at 0.3 and 0.7. In the noiseless case, all functions agree on the function values at those locations.

$\mathcal{N}(0, \sigma_n^2)$ and σ_n^2 is independent of $\boldsymbol{x}$. Then, the prior distribution changes to

$$p(\boldsymbol{y} \mid \mathbf{X}) = \mathcal{N}(\mu(\mathbf{X}), \Sigma(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}), \tag{8}$$

and the posterior distribution to

$$\begin{aligned} p(\boldsymbol{y} \mid \mathbf{X}, \mathcal{D}) = \mathcal{N}(&\Sigma(\mathbf{X}, \mathbf{X}_{\text{obs}}) \left(\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}}) + \sigma_n^2 \mathbf{I}\right)^{-1} \boldsymbol{y}_{\text{obs}}, \\ &\Sigma(\mathbf{X}, \mathbf{X}) - \Sigma(\mathbf{X}, \mathbf{X}_{\text{obs}}) \left(\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}}) + \sigma_n^2 \mathbf{I}\right)^{-1} \Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X})), \end{aligned} \tag{9}$$

where $\mathbf{I}$ is the identity matrix [92]. Unless a problem is known to be noiseless, σ_n is usually treated as a HP and optimized jointly with the other HPs of the GP. In more explicit terms, the posterior mean $\mu_{\mathcal{D}}$ and variance $\text{cov}_{\mathcal{D}}(\boldsymbol{x})$ for noisy observations and a mean-zero prior are given by

$$\mu_{\mathcal{D}}(\mathbf{X}) = \Sigma(\mathbf{X}, \mathbf{X}_{\text{obs}}) \left(\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}}) + \sigma_n^2 \mathbf{I}\right)^{-1} \boldsymbol{y}_{\text{obs}}, \tag{10}$$

and

$$\text{cov}_{\mathcal{D}}(\mathbf{X}) = \Sigma(\mathbf{X}, \mathbf{X}) - \Sigma(\mathbf{X}, \mathbf{X}_{\text{obs}}) \left(\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}}) + \sigma_n^2 \mathbf{I}\right)^{-1} \Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}). \tag{11}$$

For a single observation $\boldsymbol{x}$, we further define the variance at $\boldsymbol{x}$:

$$\sigma_{\mathcal{D}}(\boldsymbol{x}) = 1 - \Sigma(\boldsymbol{x}, \mathbf{X}_{\text{obs}}) \left(\Sigma(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}}) + \sigma_n^2 \mathbf{I}\right)^{-1} \Sigma(\mathbf{X}_{\text{obs}}, \boldsymbol{x}). \tag{12}$$

Kernel Functions. The kernel function [99, 92, 78] measures similarity between two points $\boldsymbol{x}$ and $\boldsymbol{x}'$. In the context of this thesis, kernel functions are always assumed to be positive semi-definite (psd), and the terms *kernel function* and *covariance function* are used interchangeably.

The most commonly used type of kernels is *stationary kernels*, i.e., kernels that only depend on the difference between two points [34]:

$$k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}', 0). \tag{13}$$

Examples of non-stationary kernels include linear, polynomial, or periodic kernels [92].

Among the stationary kernels, the radial basis function (RBF) kernel and the Matérn class of kernels are especially popular. The RBF kernel is defined as

$$k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left\{-\frac{r}{2}\right\}, \tag{14}$$

and the most-commonly used $3/2$- and $5/2$-Matérn kernels are defined as

$$k_{\text{Mat}-3/2}(\boldsymbol{x}, \boldsymbol{x}') = \left(1 + \sqrt{3r}\right) + \exp\left\{-\sqrt{3r}\right\} \tag{15}$$

$$k_{\text{Mat}-5/2}(\boldsymbol{x}, \boldsymbol{x}') = \left(1 + \sqrt{5r} + \frac{5r}{3}\right) + \exp\left\{-\sqrt{5r}\right\} \tag{16}$$

where

$$r = \sum_{i=1}^{D} \frac{(x_i - x_i')^2}{\ell_i^2}. \tag{17}$$

Here, $\{\ell_1, \ldots, \ell_D\}$ are HPs – the *length scales* – of the kernel. If all length scales coincide ($\ell_i = \ell_j \,\forall\, i, j$) and Equation 17 can be written as $r = \frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{\ell^2}$, the GP – and, where convenient, the kernel – is called *isotropic* [34]. Otherwise, each parameter has its length scale, a model known as automatic relevance determination (ARD) [18].

If the function values are not standardized, it is convenient to scale a kernel with an additional coefficient σ_f^2, which, in the context of stationary kernels, is called the *function variance* [34] or *output scale* [2].

The above kernels consider real-valued domains, but kernels exist for other domains, such as categorical and mixed spaces [96], strings [77], graphs [85], permutation spaces [21] (see Sections 2.5 and 3.3).

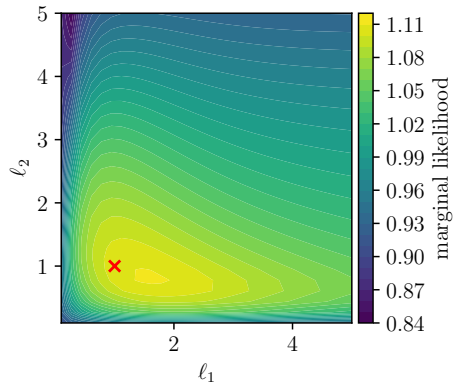


Figure 3: The marginal likelihood surface for a two-dimensional GP conditioned on twenty observations evaluated on an isotropic GP with length scale $\ell = 1$. The marginal likelihood peaks around the true optimum, shown in red, and decays slowly towards longer, and quickly towards short length scales.

2.3 Gaussian Process Fitting.

GPs expose several HPs, denoted by the set $\boldsymbol{\theta}$ which usually contains the length scales $\boldsymbol{\ell}$, the noise variance σ_n^2, and the function variance σ_f^2, i.e., $\boldsymbol{\theta} = \left\{\ell_1, \ldots, \ell_D, \sigma_n^2, \sigma_f^2\right\}$. The performance of a GP is highly dependent on choosing appropriate HP values.

The HPs of a GP are commonly optimized by choosing the model that best explains the data – a process known as maximum-likelihood estimation (MLE) [92, 34]:

$$\boldsymbol{\theta}^*_{\text{MLE}} \in \arg\max_{\boldsymbol{\theta}} \overbrace{p(\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{\theta})}^{\text{marginal likelihood}} \tag{18}$$

$$\begin{aligned} \log p(\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{\theta}) = &\overbrace{-\frac{1}{2}\boldsymbol{y}^\top \left(\Sigma(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} \boldsymbol{y}}^{\text{data fit}} \\ &\underbrace{-\frac{1}{2}\log\left|\Sigma(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right|}_{\text{complexity penalty}} - \frac{n}{2}\log 2\pi, \end{aligned} \tag{19}$$

where n is the number of observations.

The term maximized in Equation (18) is the *marginal likelihood*, which consists of three terms: the *data fit*, *complexity penalty*, and a normalization constant [92]. Figure 3 shows the marginal likelihood of a two-dimensional GP, conditioned on 20 points drawn from a Sobol sequence and evaluated on a sample from two-

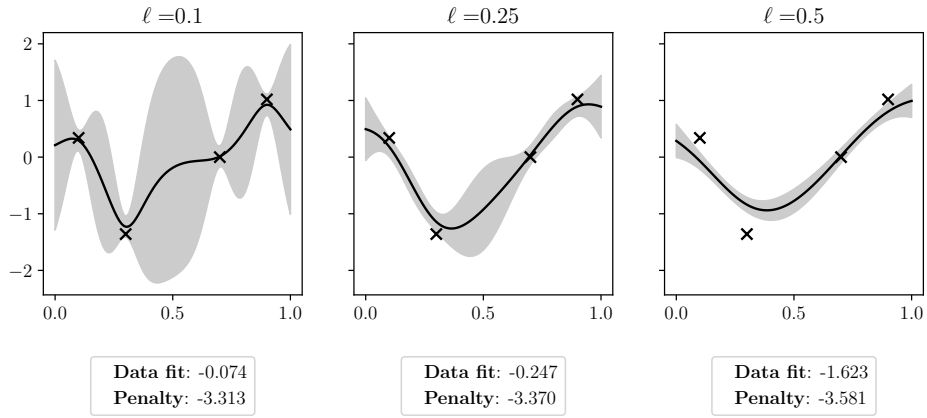


Figure 4: Three GPs with different length scales ($\ell = 0.1, 0.25, 0.5$) fitted to four data points. The data fit and penalty term are shown for each GP. Longer length scales have worse data fit but lower penalty.

dimensional isotropic GP prior with a $5/2$-Matérn kernel with $\ell = 1$ and $\sigma_n^2 = 0.1$. We refer to such realizations drawn from a GP prior as "prior samples". The likelihood is evaluated for different length scales (ℓ_1 and ℓ_2) on a GP which, except for the length scales, has the same HPs as the ground truth GP used to obtain the function values. The red cross shows the ground truth length scales around which the marginal likelihood attains its highest values. Starting from the marginal likelihood mode, it shrinks more slowly in the direction of longer length scales than in the direction of shorter length scales.

Instead of maximizing the marginal likelihood directly, one maximizes the marginal log-likelihood (MLL) (Equation (19)) for better numerical stability in practice [34]. Figure 4 shows GPs with different length scales ($\ell = 0.1, 0.25, 0.5$) fitted to the same four data points. For each GP, the data fit, and penalty terms are shown. The noise term σ_n^2 is set to 0.1 and the kernel is a $5/2$-Matérn kernel. The normalization constant is omitted as it does not change. As the length scale increases, the data fit worsens since the model loses the flexibility to model the data. At the same time, models with longer length scales are penalized less – maximizing the MLL is a trade-off between the two terms.

If one has prior knowledge about the values of one or multiple GP HPs, it is useful to incorporate this prior knowledge into the model fitting. The two main approaches to injecting prior knowledge are maximum a-posteriori estimation (MAP) and the fully

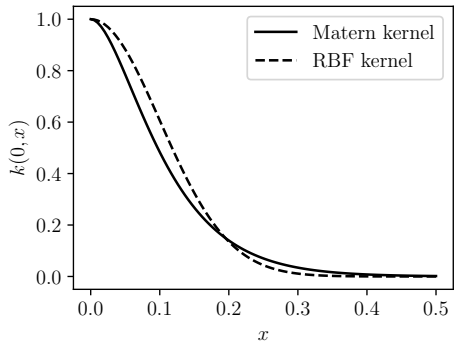


Figure 5: The 5/2-Matérn and RBF kernels for $\ell = 0.1$ in a one-dimensional space.

Bayesian treatment [34, 61]. With MAP, the objective for fitting the GP changes to

$$\boldsymbol{\theta}^*_{\text{MAP}} \in \arg\max_{\boldsymbol{\theta}} \underbrace{p(\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{\theta})}_{\text{evidence}} \cdot \underbrace{p(\boldsymbol{\theta})}_{\text{prior}}, \tag{20}$$

where $p(\boldsymbol{\theta})$ is a prior distribution for the HPs of the GP chosen by the user. Similar to the MLL, Equation 20 is usually optimized in log space, and the objective for fitting the GP becomes the unnormalized log-posterior. The MAP objective can be decomposed into two terms – the evidence and the prior. MAP seeks the mode of the posterior distribution, which, if there is little evidence, corresponds to the prior mode. Here, little evidence means that $p(\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{\theta})$ only changes little with $\boldsymbol{\theta}$. This is the case in high-dimensional spaces with scarce data and moderate length scales since the distance between points is extremely high and $\Sigma(\mathbf{X}, \mathbf{X})$, for stationary kernels, changes little with moderate length scale changes. This becomes evident from Figure 5, showing the Matérn and RBF kernel values for $\ell = 0.1$ in a one-dimensional space. As the distance between x and the origin grows, both values quickly reach zero. In a D-dimensional space, the maximum distance in a unit hypercube is $\sqrt{D}$ so that the values of $\Sigma(\mathbf{X}, \mathbf{X})$ are close to zero everywhere unless a) points are sampled closeby, b) the length scales are very long, or c) there is plenty of data.

`BoTorch` [2] maximizes the MLL with the gradient-based limited-memory Broyden-Fletcher-Goldfarb-Shanno with bound constraints (L-BFGS-B) [125] algorithm starting from the initial parameter values. Before `BoTorch` version `0.12.0`, the function variance σ_f^2 and length scales were initialized with a value of $\log 2$, which resulted from initializing the unconstrained so-called *raw* parameters with a value of 1 and transforming them to positive values with the SoftPlus transformation:

$$\text{Softplus}(\boldsymbol{x}) = \log(1 + \exp\{x\}). \tag{21}$$

In version `0.12.0`, the initial length scale was changed to the mode of the prior proposed by Hvarfner et al. [51], and the default covariance module no longer

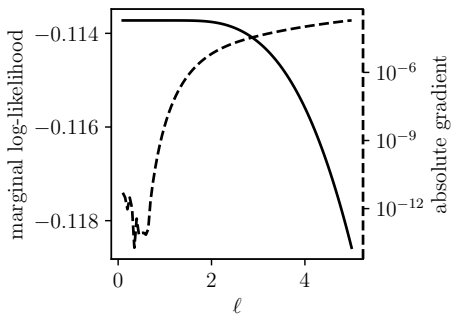


Figure 6: The MLL (solid line, plotted against left y-axis) and its absolute gradient (dashed line, plotted against right y-axis) of a 1000-dimensional GP conditioned on 20 observations evaluated on a GP prior sample with the same hyperparameter setting. For short length scales, the MLL is nearly flat, and the gradients have almost zero absolute value.

models the function variance. The noise variance σ_n^2 is initialized with the mode of a Gamma$(1.1, 0.05)$ prior in `BoTorch` versions $<$ `0.12.0` and with the mode of a LogNormal$(-4, 1)$ prior in version `0.12.0`. If the MLL maximization fails, `BoTorch` samples new values from the prior of each parameter and re-attempts to maximize the MLL.

In high-dimensional spaces and with few observations, the absolute gradients of the MLL are often almost zero. For Figure 6, we fit a 1000-dimensional GP on 20 observations drawn from a Sobol sequence and evaluated on an isotropic prior sample with a $5/2$-Matérn kernel, length scale $\ell = 1$, and $\sigma_n^2 = 0.1$. The MLL is shown in black, and the absolute gradient obtained by finite differences is in red. For low values of ℓ, the MLL is almost flat and the gradient is almost zero. Hence, depending on the starting point for the gradient-based MLL maximization, the optimizer may never escape this initial solution due to *vanishing gradients*. Paper VI discusses in detail how vanishing gradients cause failures in high-dimensional Bayesian optimization (HDBO) algorithms and proposes solutions to avoid these issues.

Fully Bayesian Treatment. MLE and MAP aim to find point estimates of distribution parameters that best explain the data. However, the MLE approach is prone to overfitting and sensitive to the starting conditions for the gradient descent (GD)-based HPO, especially when there are many HPs [61]. An alternative is to include the hyperpriors of the GP HPs in the Bayesian treatment, giving the so-called *fully Bayesian treatment* [61] or *model averaging* [34]. Assume f is noiseless. Then,

$$p(\boldsymbol{y} \mid \mathbf{X}, \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}}) = \int p(\boldsymbol{y} \mid \mathbf{X}, \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}}) \mathrm{d}\boldsymbol{\theta} \tag{22}$$

where

$$p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}}) \propto p(\boldsymbol{y}_{\text{obs}} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{\theta}) p(\boldsymbol{\theta}). \tag{23}$$

Equation (22) can be approximated using Monte-Carlo estimation. However, sampling from the posterior $p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})$ is less trivial and requires Markov-Chain Monte-Carlo (MCMC) methods such as the no u-turn sampler (NUTS) [46]. This renders the fully Bayesian treatment relatively expensive [61].

Instead of approximating $p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})$ with MCMC, we can follow a different approach and use variational inference to find a distribution $q(\boldsymbol{\theta})$ that is close to $p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})$. This is done by minimizing the Kullback-Leibler divergence between $q(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})$:

$$\begin{aligned} q^*(\boldsymbol{\theta}) &= \underset{q(\boldsymbol{\theta}) \in \mathcal{Q}}{\arg\min}\, D_{\text{KL}}(D(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})) && (24) \\ &= \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log p(\boldsymbol{\theta} \mid \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})\right] && (25) \\ &= \underbrace{\mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log p(\boldsymbol{\theta}, \mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})\right]}_{-\text{ELBO}} + \underbrace{p(\mathbf{X}_{\text{obs}}, \boldsymbol{y}_{\text{obs}})}_{\text{const.}} && (26) \end{aligned}$$

where $\mathcal{Q}$ is some family of distributions, e.g., the family of normal distributions parametrized by the mean and variance. The minimization problem in Equation (24) can be approached by maximizing the evidence lower bound (ELBO), which can be rewritten as

$$\text{ELBO} = \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log p(\boldsymbol{y}_{\text{obs}} \mid \boldsymbol{\theta}, \mathbf{X}_{\text{obs}})\right] - D_{\text{KL}}(D(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}))). \tag{27}$$

Choosing a tractable distributional family (e.g., a mean-field approximation [61]), Equation (27) can be optimized with gradient-based approaches. If the number of observations is large, one can resort to an approximation using minibatches [63]. The inference uses Monte-Carlo estimation with samples drawn from q [63].

Similar to MAP, the fully Bayesian treatment assumes the availability of prior information about the HPs $\boldsymbol{\theta}$, which is often unavailable. Nevertheless, in cases where the fully Bayesian treatment can be applied, it often performs better than MAP [61, 27].

Paper VI discusses the MLL in the context of HDBO and shows the gradients of the MLL vanish for high-dimensional GPs and short length scales. Scaling the initial length scale values with D and using MLE is proposed as an alternative to long length scale priors and MAP.

2.4 Acquisition Functions

In the following, we introduce several AFs that are relevant to this thesis and BO in general. Before we do so, we briefly establish the difference between *myopic* and *non-myopic* AFs. In the context of this thesis, all AFs maximize the expected marginal gain of some utility function $u(\mathcal{D})$ given the current observations $\mathcal{D}$ [53]:

$$\alpha(\boldsymbol{x}) = \mathbb{E}_{f(\boldsymbol{x})}\left[u(\mathcal{D} \cup \{\boldsymbol{x}, f(\boldsymbol{x})\}) - u(\mathcal{D})\right]. \tag{28}$$

For instance, expected improvement (EI) maximizes the negative *simple reward* [53, 34]:

$$u(\mathcal{D}) = \max_{(\boldsymbol{x},y)\in\mathcal{D}} -y. \tag{29}$$

AFs that adhere to Equation (28) are called *one-step* AFs and, by some authors, *myopic* AFs [53, 34]. Other authors use a different definition and consider AFs that use point-wise utility functions as *myopic* [123]. For example, under this definition, EI, probability of improvement (PI), and upper-confidence bound (UCB) are myopic whereas entropy-based AFs are non-myopic [123, 34]. However, they all are *one-step* AFs since they only consider the next observation $\boldsymbol{x}$ and hence myopic in the sense of Jiang et al. [53] and Garnett [34].

In the context of this thesis, we use the term *myopic* AFs to refer to AFs that use a point-wise utility function, i.e., we follow the definition of Wilson et al. [123].

Expected Improvement. One of the arguably most popular AFs in BO is EI [56]. We consider the EI only in the noiseless case in which $y = f(\boldsymbol{x})$. It has a closed-form expression and is defined as

$$\alpha_{\mathrm{EI}}(\boldsymbol{x}) = \mathbb{E}_{f(\boldsymbol{x})}\left[\underbrace{\max(y^+ - f(\boldsymbol{x}), 0)}_{=:I(\boldsymbol{x})}\right] \tag{30}$$

$$= \int_{-\infty}^{\infty} I(\boldsymbol{x})\phi(z)\mathrm{d}z \tag{31}$$

$$= \left(\mu_{\mathcal{D}}(\boldsymbol{x}) - y^+\right)\Phi\left(\frac{\mu_{\mathcal{D}}(\boldsymbol{x}) - y^+}{\sigma_{\mathcal{D}}(\boldsymbol{x})}\right) + \sigma_{\mathcal{D}}(\boldsymbol{x}) \cdot \phi\left(\frac{\mu_{\mathcal{D}}(\boldsymbol{x}) - y^+}{\sigma_{\mathcal{D}}(\boldsymbol{x})}\right) \tag{32}$$

where y^+ is the best-observed value so far, and ϕ and Φ are the normal probability density function (PDF) and cumulative density function (CDF), and $\mu_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}$ are the posterior mean and variance at $\boldsymbol{x}$ as defined in Equations (10) and (12).

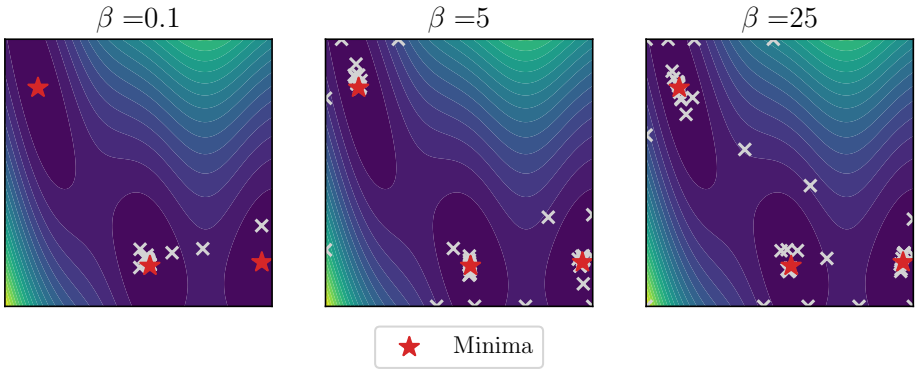


Figure 7: BO of the $2D$-`Branin` function [108] using UCB with varying β as the AF. Low β-values promote a more local behavior, and higher β values are more exploratory.

Probability of Improvement. Instead of considering the expected value of the improvement, the PI [54] only considers the probability that the value at $\boldsymbol{x}$ is better than the best-observed value y^+.

$$\begin{aligned}
\alpha_{\text{PI}}(\boldsymbol{x}) &= \mathbb{P}_{f(\boldsymbol{x})}\left[f(\boldsymbol{x}) < y^+\right] && (33)\\
&= \mathbb{P}_{f(\boldsymbol{x})}\left[y^+ - f(\boldsymbol{x} > 0\right] && (34)\\
&= \mathbb{P}_{f(\boldsymbol{x})}\left[\max(y^+ - f(\boldsymbol{x}), 0) > 0\right] && (35)\\
&= \mathbb{P}_{f(\boldsymbol{x})}\left[I(\boldsymbol{x}) > 0\right] && (36)\\
&= \Phi\left(\frac{\mu_{\mathcal{D}}(\boldsymbol{x}) - y^+}{\sigma_{\mathcal{D}}(\boldsymbol{x})}\right) && (37)
\end{aligned}$$

PI is known to be extremely exploitative [20, 5]. A more exploratory behavior can be induced by not considering the probability of a non-zero improvement but a probability of at least some threshold value $\tau \in [0, 1)$ [34].

Upper-Confidence Bound. The UCB [107] acquisition function explicitly trades off exploitation and exploration:

$$\alpha_{\text{UCB}}(\boldsymbol{x}; \beta) = \mu_{\mathcal{D}}(\boldsymbol{x}) + \beta\sigma_{\mathcal{D}}(\boldsymbol{x}), \tag{38}$$

where β is a parameter controlling the level of exploration. The exploration parameter β is often treated as a hyperparameter, although some regret bounds only hold for certain values of β [107].

Figure 7 shows the behavior of UCB when optimizing the two-dimensional `Branin` function [108] for 50 BO iterations in the bounds $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$

after warm-starting with five DoE points chosen uniformly at random. The `Branin` function has three local minima marked with red stars. Figure 7 only shows the evaluations after the DoE phase. While for $\beta = 0.1$, the method only captures one of the minima, the level of exploration increases with β, and for high β, all three minima are visited.

Thompson Sampling. Motivated by the bandit literature, Thompson sampling (TS) [98] draws a realization from the GP posterior and minimizes it. Since GPs effectively are a joint normal distribution for a finite set of points, TS either searches the minimum on a dense set of points or uses approximations of the GP posterior to allow for the minimization on a continuous domain. The latter can be realized using pathwise conditioning [124]. TS naturally balances exploration and exploitation [23]: If the surrogate exhibits high uncertainty, there is more variation between posterior samples, and the minima of different samples are more likely to be different. As the uncertainty decreases, the minima of the posterior samples converge to the same point.

Information-Theoretic Acquisition Functions. Information-theoretic AFs aim to reduce a particular property of the optimum $\boldsymbol{x}^*$. For entropy search (ES) [41] or predictive entropy search (PES) [43], this property is the location of the optimum. For max-value entropy search (MES) [119], this property is the objective $f(\boldsymbol{x}^*)$ value at $\boldsymbol{x}^*$. Finally, joint entropy search (JES) [49, 114], aims to reduce the uncertainty about both $\boldsymbol{x}^*$ and $f(\boldsymbol{x}^*)$.

Formally, ES [41] is defined as the reduction in differential entropy upon observing a new point $\boldsymbol{x}$:

$$\alpha_{\text{ES}}(\boldsymbol{x}) = \mathrm{H}\left[p(\boldsymbol{x}^* \mid \mathcal{D})\right] - \mathbb{E}_y\left[\mathrm{H}\left[p(\boldsymbol{x}^* \mid \mathcal{D} \cup (\boldsymbol{x}, y)\right]\right] \tag{39}$$

PES is defined equivalently but uses the symmetry of the mutual information to use alternative approximations

$$\alpha_{\text{PES}}(\boldsymbol{x}) = \mathrm{H}\left[p(y \mid \mathcal{D}, \boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}^*}\left[\mathrm{H}\left[p(y \mid \mathcal{D}, \boldsymbol{x}, \boldsymbol{x}^*)\right]\right] \tag{40}$$

MES is defined on the value of the observations:

$$\alpha_{\text{MES}}(\boldsymbol{x}) = \mathrm{H}\left[p(y \mid \mathcal{D}, \boldsymbol{x})\right] - \mathbb{E}_{y^*}\left[\mathrm{H}\left[p(y \mid \mathcal{D}, \boldsymbol{x}, y^*)\right]\right], \tag{41}$$

And JES is defined as

$$\alpha_{\text{JES}}(\boldsymbol{x}) = \mathrm{H}\left[p(y \mid \mathcal{D}, \boldsymbol{x})\right] - \mathbb{E}_{(\boldsymbol{x}^*, f(\boldsymbol{x}^*))}\left[\mathrm{H}\left[p(y \mid \mathcal{D} \cup (\boldsymbol{x}^*, f(\boldsymbol{x}^*)), \boldsymbol{x}, f(\boldsymbol{x}^*))\right]\right]. \tag{42}$$

Entropy-based AFs lack a closed-form expression for the expectation and rely on often expensive approximations.

If one only uses a single y^* for the expectation in Eq. (41), Wang and Jegelka [118] show that MES is equivalent to PI and UCB (for specific parameter choices). This equivalence breaks down if one estimates the expectation with a higher number of samples, leaving open the question of the relationship between non-myopic AFs like MES and myopic strategies when considering the actual expectation in Eq. (41). Paper v establishes a new relationship between the entropy-based MES acquisition function and EI by approximating $p(y \mid \mathcal{D}, \boldsymbol{x}, y^*)$ with variational inference (VI), showing that, in noiseless settings, MES and EI are equivalent if the distributional family is chosen as the exponential distribution.

Knowledge Gradient. Other than EI, the knowledge gradient (KG) AF does not assume that the current best solution has been observed:

$$\alpha_{\mathrm{KG}}(\boldsymbol{x}) = \min_{\boldsymbol{x}' \in \mathcal{X}} \mu(\boldsymbol{x}' \mid \mathcal{D}_n) - \mathbb{E}_y \left[\min_{\boldsymbol{x}' \in \mathcal{X}} \mu\left(\boldsymbol{x}' \mid \mathcal{D}_n \cup (\boldsymbol{x}, y)\right) \right] \tag{43}$$

The KG considers the difference between the current best value according to the surrogate model $\min_{\boldsymbol{x}' \in \mathcal{X}} \mu_n(\boldsymbol{x}')$ and the best value if we make one more observation $\min_{\boldsymbol{x}' \in \mathcal{X}} \mu\left(\boldsymbol{x}' \mid \mathcal{D} \cup (\boldsymbol{x}, y)\right)$. In EI, the best-observed value y^+ is chosen from all observed values in $\mathcal{D}$. In contrast, the KG considers the minimum posterior mean, making it more reliant on an accurate model.

Characterizing Acquisition Functions. Different AFs have different characteristics. For instance, TS is known to prefer exploration [23] and PI is known to operate relatively local [20, 5]. Since the interplay between different AFs or which one to use in a given scenario is often unclear, several works considered portfolios of AFs that select an AF based on some heuristic [5, 45, 14]. Paper iv introduces two methods to quantify the level of exploration of different AFs. Based on these methods, observation traveling salesman distance (OTSD) and observation entropy (OE), paper iv relates an AF's level of exploration with its optimization performance across several benchmarks and creates a taxonomy of AFs which is shown in Figure 8. Our taxonomy confirms findings from the literature, e.g., that PI is relatively exploitative [20, 5], but also establishes missing links – for instance, we show that KG is more explorative than MES. Furthermore, paper iv discusses how RAASP [91] and TRs [29] affect the exploratory behavior and optimization performance of AFs. This provides a profound analysis of the impact of these techniques, which can be

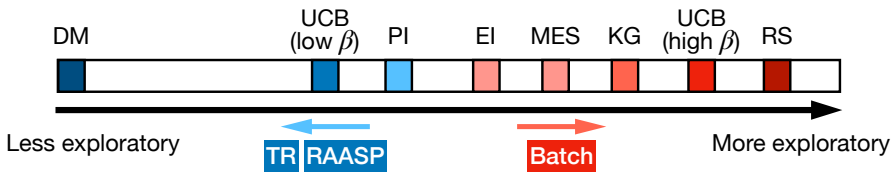


Figure 8: The taxonomy of AFs introduced in paper IV. We compare deterministic selection (DM), UCB with different values of β, PI, EI, MES, KG, and random search (RS) on various synthetic and real-world benchmarks and classify them by their level of exploration, according to OTSD and OE. We further combine each AF with trust regions (TRs), random axis-aligned subspace perturbations (RAASP), and batching. DM is a deterministic selection that always evaluates the same point. RS samples a point from $\mathcal{X}$ uniformly at random.

combined with any AF, and independently studies the effect of RAASP and TRs, which are often only considered jointly. While our results show that RAASP always promotes locality, we sometimes observe that TRs promote exploratory behavior, arguably due to the restarting mechanism of TRs (see Section 3.3).

2.5 Combinatorial Optimization

In applications such as materials science [115, 31, 87, 106, 42, 40, 47], neural architecture search [58, 97], or drug discovery [82, 109], the search space $\mathcal{X}$ is not purely continuous. For example, in drug design, one searches over a discrete set of candidate molecules, and in HPO, the number of nodes in a neural network is an integer-valued parameter. Combinatorial BO is concerned with search spaces where at least one parameter is not real-valued.

In the context of this thesis, we are primarily concerned with three types of combinatorial variables: binary, categorical, and ordinal. Table 1 shows examples of the different types of combinatorial variables. A binary variable always has support $\{0, 1\}$. A categorical variable can take C_i different values. For example, in Table 1, D_1 encodes colors and could have support {"red", "green", "blue", "orange"} so that $C_1 = 4$.

Usually, we assign numerical values to the categories, in which case a realization of a categorical variable resembles an ordinal variable. However, by definition, categorical variables have no order. Hence, they should not be passed as integers to a GP with a standard RBF or Matérn kernel to not "fool" it into modeling an ordinal relationship that does not exist. A common strategy for this problem is to one-hot encode categorical variables [17], i.e., a categorical variable with C_i realizations is represented as a binary vector $\hat{\boldsymbol{x}}$ with length $|\hat{\boldsymbol{x}}| = C_i$. Each category is assigned a position in this vector via its numerical representation, and $\hat{\boldsymbol{x}}$ is set to one at that position and

Table 1: Combinatorial vectors with the different combinatorial variables considered in this thesis. Binary variables always have the same support.

Variable Type	Example
Binary	$\boldsymbol{x} = (0, 1, 1, 0)$
Categorical	$\boldsymbol{x} =$ ("red", "cat")
Ordinal	$\boldsymbol{x} = (1.2, 5, 2.3, 1.2)$

zero everywhere else. For instance, in the example above, "red" would be represented as $\hat{\boldsymbol{x}} = (1, 0, 0, 0)$, "green" as $\hat{\boldsymbol{x}} = (0, 1, 0, 0)$, "blue" as $\hat{\boldsymbol{x}} = (0, 0, 1, 0)$, and "orange" as $\hat{\boldsymbol{x}} = (0, 0, 0, 1)$.

Even though the one-hot encoding potentially drastically increases the dimensionality of the search space, this does not make the problem more challenging since two one-hot encoded vectors $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}'$ representing two realizations of a categorical variable have distance $\|\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}'\|_2 = \sqrt{2}$ if $\hat{\boldsymbol{x}} \neq \hat{\boldsymbol{x}}'$ and 0 otherwise *regardless* of C_i. Hence, the number of categorical variables, not the dimensionality of $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}'$ itself, defines the complexity of the problem.

The one-hot encoded categorical, binary, and ordinal variables can be combined with standard kernels, such as RBF or Matérn kernels. However, specialized kernels and surrogates for combinatorial variables exist. For example, `COMBO` [85] represents combinatorial spaces as graphs and uses a graph diffusion kernel to express similarity between variables, and Baptista and Poloczek [4] use an entirely different surrogate which models quadratic interactions between parameters and use TS as the acquisition function.

2.6 Acquisition Function Maximization

In each BO step, one needs to find the maximum of the AF α (see Line 4 in Algorithm 1). Since the AF has a known form, one can maximize it with gradient-based methods. `BoTorch` [3] initially evaluates α on a dense grid of random samples drawn from a scrambled Sobol sequence [86]. Out of these initial random samples, `BoTorch` then chooses points to start the gradient-based maximization, where points with higher initial acquisition value are more likely to be chosen as starting points (Boltzmann sampling; [1]). `BoTorch` further possesses an optional method that augments the set of initial random samples with points sampled close to points with good function values ("`sample_around_best`") [2]. In particular, `BoTorch` perturbs the 5% top-performing points locally using a Normal distribution, truncated so its support does not extend beyond $\mathcal{X}$ and centered on the respective

top-performing point. Additionally, if $D \geq 20$, `BoTorch` further generates initial random samples with only a subset of the dimensions perturbed,d where each dimension has a probability of $\frac{20}{D}$ of being perturbed. Again, the values of the chosen dimension are drawn from a truncated normal distribution centered on the respective top-performing point. Overall, if calling `optimize_acqf` [2] with m initial random samples and `sample_around_best = True`, `BoTorch` will generate $2m$ initial random samples with m global samples drawn from a scrambled Sobol sequence, $m/2$ samples created around the 5% best-performing points, perturbing all dimensions, and $m/2$ samples created around the 5% best-performing points, perturbing only 20 dimensions in expectation. If $D < 20$, m samples are created around the 5% best-performing points, perturbing all dimensions.

Ament et al. [1] discuss heuristics such as random axis-aligned subspace perturbations (RAASP) as means to improve the AF maximization but argue that they do not solve foundational issues such as vanishing gradients of AFs themselves, motivating the need for `LogEI`. Paper VI discusses RAASP from the perspective of the MLL optimization. The higher likelihood of observations to lie closely together under RAASP also increases the likelihood of the kernel matrix having a non-homogeneous structure, increasing the gradient signal of the MLL and avoiding vanishing gradients at model fitting. Based on this, paper VI highly recommends including it in the AF optimization for high-dimensional problems.

A common approach to maximize the AF in combinatorial settings is to use a local search to optimize α for combinatorial variables [22, 116, 85]. Algorithm 2 shows a simple multi-start local search approach for combinatorial BO similar to the approaches in Deshwal et al. [22], Wan et al. [116], Oh et al. [85]. Note that if $\mathcal{X}$ is high-dimensional, it may not be feasible to create all neighbors of $\boldsymbol{x}_{\text{center}}$ in line 8. In that case, the point returned by Algorithm 2 is not necessarily a local maximizer. While local search only evaluates the acquisition function α at valid locations, it is usually less efficient than gradient-based approaches.

For spaces containing both combinatorial and continuous parameters, it is a common strategy to interleave gradient-based optimization for the continuous parameters and combinatorial optimization for the combinatorial parameters [22, 116]. Here, the number of interleavings is a HP.

Local search and interleaving schemes have the drawback that they usually cannot (fully) leverage efficient gradient-based optimizers, posing an additional challenge for the AF maximization. One workaround is to treat the combinatorial variables like continuous variables during the optimization and to discretize them afterward to evaluate f [17]. However, this approach is problematic because it is unclear whether to add the discretized or the original observation to $\mathcal{D}$ [36]. If one adds the

Algorithm 2 Local Search for Combinatorial Variables

Input: acquisition function α, number of restarts k

randomly sample k initial configurations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathcal{X}$
$\boldsymbol{x}_{\text{best}} \leftarrow \boldsymbol{x}_1$
for all $\boldsymbol{x}_i \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$ **do**
    `improved` $\leftarrow$ `True`
    $\boldsymbol{x}_{\text{center}} \leftarrow \boldsymbol{x}_i$
    while `improved` **do**
        `improved` $\leftarrow$ `False`
        create all neighbors $\boldsymbol{n}_1, \ldots \boldsymbol{n}_j$ of $\boldsymbol{x}_{\text{center}}$ ▷ Or sample a few neighbors if $\mathcal{X}$ is high-dimensional.
        if for any $\boldsymbol{n}_l$: $\alpha(\boldsymbol{n}_l) > \alpha(\boldsymbol{x}_i)$ **then**
            $\boldsymbol{x}_{\text{center}} \leftarrow \boldsymbol{n}_l$
            `improved` $\leftarrow$ `True`
    if $\alpha(\boldsymbol{x}_{\text{center}}) > \alpha(\boldsymbol{x}_{\text{best}})$ **then**
        $\boldsymbol{x}_{\text{best}} \leftarrow \boldsymbol{x}_{\text{center}}$

Output: the point of highest (local) acquisition value $\boldsymbol{x}_{\text{best}}$

discretized observation, one does not add the point of maximum acquisition value, and it cannot be precluded that the next point, after discretization, is the same again. Adding the non-discretized point of maximum acquisition value adds an invalid observation to $\mathcal{D}$. To address these limitations, Garrido-Merchán and Hernández-Lobato [36] propose to round combinatorial parameters before evaluating the AF. While this mitigates the aforementioned modeling issues, optimizing the AF remains challenging since the AF-gradients are now zero for the combinatorial parameters. Daulton et al. [17] propose to optimize instead a real-valued distribution parameter of a distribution from which one can sample a realization of a combinatorial variable. For instance, a realization of a binary variable is sampled from a Bernoulli(β) distribution, and a variant of an acquisition function α is maximized to find the optimal β.

An alternative approach to treat combinatorial as continuous spaces is to use an autoencoder model to transform the combinatorial problem into a continuous one and carry out the optimization in the latent space of the autoencoder [113, 38, 72]. This approach usually requires a significant amount of (unsupervised) data, limiting its applicability in scenarios where such data is unavailable.

Paper III introduces a novel algorithm (`Bounce`) for combinatorial BO in high-dimensional spaces. `Bounce` reduces the dimensionality of the search space by

allowing the representation of multiple combinatorial parameters of the same type in one variable, even if these combinatorial parameters have varying cardinality. This lowers the problem's dimensionality, simplifying the maximization of the AF and improving the sample efficiency. Assuming that representing several parameters with a single parameter still allows finding a minimizer of f (see Section 3), this modeling strategy improves sample efficiency and simplifies the maximization of the AF.

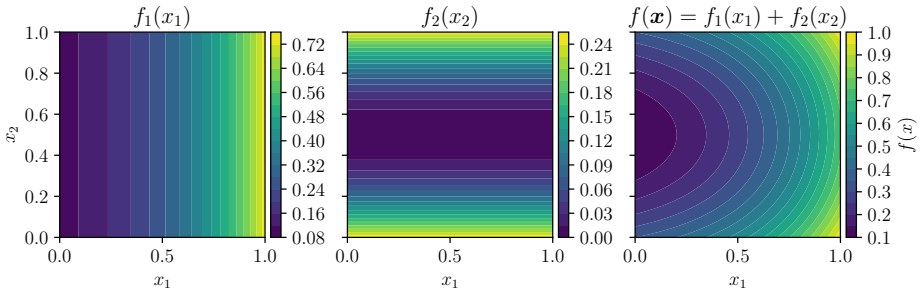


Figure 9: The additive function $f(\boldsymbol{x})$ is composed of two individual functions f_1 and f_2 that only change w.r.t. one of the variables x_1 and x_2.

3 High-Dimensional Bayesian Optimization

For a long time, it was believed that Bayesian optimization (BO) with Gaussian process (GP) surrogate models is limited to 20 dimensions for typical applications and evaluation budgets [30, 76]. The reason for this limitation is the curse of dimensionality (COD), which, with linearly increasing dimensionality, demands exponentially more data to maintain the same level of precision [60]. Therefore, high-dimensional Bayesian optimization (HDBO) refers to problems with more than 20 dimensions, sometimes in the hundreds or thousands.

In the following, we discuss important methods for HDBO.

3.1 Additive Models

One approach is to assume the function is additive (a sum of lower-dimensional components). This allows BO to scale by modeling each component separately [25, 57, 33, 7]. In particular, it is assumed that f can be decomposed as

$$f(\boldsymbol{x}) = \nu + \sum_{i=1}^{N_A} f_i(\boldsymbol{x}_{A_i}), \tag{44}$$

where $\mathcal{A} = \{A_1, \dots, A_{N_A}\}$ contains pairwise disjoint subsets A_i such that $\bigcup_{i=1}^{N_A} A_i = \{1, \dots, D\}$ [57, 33, 7]. Only variables in a subset A_i interact with each other non-additively. Figure 9 shows an example of an additive function that is composed of two functions $f_1(x_1) = \frac{\exp(2x_1)}{10}$ and $f_2(x_2) = \left(x_2 - \frac{1}{2}\right)^2$.

Functions exhibiting such structure can be effectively modeled using N_A base kernels

$k_i\left(\boldsymbol{x}_{A_i}, \boldsymbol{x}'_{A_i}\right)$ [25], such that

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \sum_{i=1}^{N_A} k_i\left(\boldsymbol{x}_{A_i}, \boldsymbol{x}'_{A_i}\right). \tag{45}$$

Additive models alleviate the COD since they avoid the exponentially-increasing data demand: in the extreme case $A_i = \{D_i\}$, the data demand only grows linearly with the number of dimensions: if each dimension can be modeled effectively with k data points, one additional dimension will only require another k data points – the overall data demand is $k \cdot D$. However, the approach hinges on Equation (44) holding, and finding the correct subsets A_i can be challenging [33]. In practice, Equation (44) is unlikely to hold exactly, but one aims at finding a decomposition of $\{1, \ldots, D\}$ so that Equation (44) holds approximately. For instance, Gardner et al. [33] use the Metropolis-Hastings algorithm to, starting from a given decomposition, propose a new decomposition by splitting a subset A_i or merging two subsets A_i and A_j. Ziomek and Ammar [129] circumvent the challenge of learning the decomposition by using randomly chosen additive subsets.

3.2 Subspace Methods

Similar to additive GP models, subspace-based methods make assumptions on f that reduce the complexity of the problem. In particular, while the domain of f is D-dimensional, it is assumed that there exists a low-dimensional *effective subspace* $\mathcal{Z}$ of $\mathcal{X}$ in which the optimization can be carried out. Examples of functions with an effective subspace are the functions f_1 and f_2 in Figure 9. While both functions are defined in two dimensions, they effectively only vary across one of the dimensions (x_1 for the left; x_2 for the center figure). Both functions further have an *axis-aligned* effective subspace since there are dimensions along which they do not change (x_2 for f_1; x_1 for f_2). An axis-aligned subspace is a particularly desirable function property for HDBO since they can be modeled by ignoring the inactive dimensions in Equation (17) as we discuss in Section 3.5. If the effective subspace is not axis-aligned, the GP still needs to model dimensions. Figure 10 shows a function with a one-dimensional non-axis-aligned effective subspace which is obtained by first rotating $\boldsymbol{x}$ by $\frac{\pi}{4}$ with the rotation matrix Θ and subsequently evaluating f_2 from Figure 9 on $(\Theta\boldsymbol{x})_2$.

Formally, we define a function with an effective subspace as follows. Assume the subspace $\mathcal{Z}$ has d_e dimensions and $d_e \ll D$. Then, we assume there exist a mapping $\rho : \mathcal{X} \to \mathcal{Z}$ and a function $\gamma : \mathcal{Z} \to \mathbb{R}$ so that for all $\boldsymbol{x} \in \mathcal{X}$: $(\gamma \circ \rho)(\boldsymbol{x}) = f(\boldsymbol{x})$, i.e., if the mapping ρ was known, one could also carry out the optimization in $\mathcal{Z}$. This assumption can reduce the data demand since the GP only needs to be modeled with

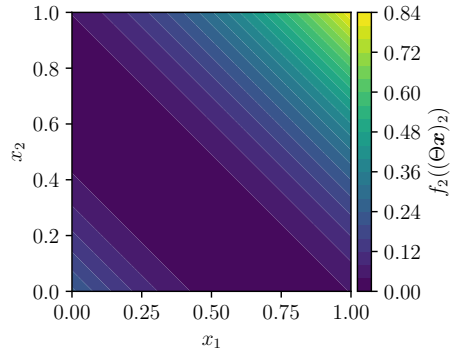


Figure 10: A two-dimensional function with a non-axis-aligned one-dimensional effective subspace. The optimum can be found by only considering the diagonal from the bottom left to the top right. Moving perpendicularly to this diagonal does not change the function value.

d_e dimensions, alleviating the curse of dimensionality and reducing the number of GP hyperparameters (HPs) when using an automatic relevance determination (ARD) kernel. Algorithm 3 shows a generic algorithm for subspace BO.

In the example in Figure 10, the mapping to the low-dimensional subspace was defined as $\rho(\boldsymbol{x}) = (\Theta \boldsymbol{x})_2$. Assume we do not know Θ, but we know that f has a one-dimensional effective subspace. Then, we could aim only to model f_2, learn ρ (or, more specifically, ρ^{-1}), and whenever we want to evaluate a new point, apply ρ^{-1} to map it to the original space.

Linear Embeddings. Aiming to identify ρ with limited data as one usually has available in BO is extremely challenging. Therefore, following a data-agnostic approach with a simple class of models seems a natural choice to address this challenge. Linear subspace methods try to model the inverse operation $\rho^{-1} : \mathcal{Z} \to \mathcal{X}$ as $\tilde{\rho}^{-1}(\boldsymbol{z}) = S\boldsymbol{z}$ where S is a $D \times d$ projection matrix and model the GP in a low-dimensional subspace and $\tilde{\rho}^{-1} : \mathcal{Y} \to \mathcal{X}$ [120]. Here, d is the dimensionality of the subspace $\mathcal{Y}$ chosen by the user. Ideally, d is close to d_e to reflect the effective dimensionality of the problem. In practice, however, d_e is usually unknown, and the user must choose d. Wang et al. [120] show that if $d \geq d_e$, $S_{ij} \sim \mathcal{N}(0, 1)$, and $\mathcal{Y}$ and $\mathcal{X}$ are unconstrained, for every $\boldsymbol{x} \in \mathcal{X}$, $\mathcal{Y}$ contains a point $\boldsymbol{z}$ so that $\boldsymbol{x} = \tilde{\rho}^{-1}(\boldsymbol{z})$, implying that it is sufficient to search in $\mathcal{Y}$. Unfortunately, $\mathcal{X}$ and $\mathcal{Y}$ are not unconstrained. Wang et al. [120]'s `REMBO` ("Random EMbedding Bayesian Optimization") therefore projects $S\boldsymbol{z}$ onto the boundary of $\mathcal{X}$ if it lies outside of $\mathcal{X}$. Letham et al. [64] show that the probability of S to project outside of the bounds of $\mathcal{X}$, assuming the bounds are defined as a d-dimensional hypercube, grows quickly with d, reaching values close to zero even if $D = 1000$ and $d = 4$. This means that

Algorithm 3 Subspace Bayesian Optimization

Input: black-box function f, number of design of experiment samples n_{init}, inverse mapping $\tilde{\rho}^{-1} : \mathcal{Y} \in \mathbb{R}^d \to \mathcal{X} \in \mathbb{R}^D$

1: sample n_{init} points $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_{\text{init}}}$ and collect observations $\mathcal{D} \leftarrow \{(\boldsymbol{z}_1, y_1), \ldots, (\boldsymbol{z}_{n_{\text{init}}}, y_{\text{init}})\}$ ▷ $y_i = f(\tilde{\rho}^{-1}(\boldsymbol{z}_i)) + \varepsilon$

2: **while** stopping criterion not fulfilled **do**

3: fit a surrogate model g on $\mathcal{D}$

4: $\boldsymbol{z}_{\text{next}} \leftarrow \arg\max_{\boldsymbol{z}} \alpha(\boldsymbol{z} \mid g)$ ▷ maximize AF

5: $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{z}_{\text{next}}, y_{\text{next}})$ ▷ update data

Output: $\boldsymbol{z}_i : y_i \in \min \mathcal{D} \mid_{f(\tilde{\rho}^{-1}(\boldsymbol{z}))}$ **or** minimizer of the posterior mean

REMBO introduces significant distortions to $\mathcal{Y}$ as many points have the same function value. Alebo [64] therefore only considers points in $\mathcal{Y}$ that map to a point in $\mathcal{X}$.

Using a ternary projection matrix S only containing values 0 and ± 1 and exactly one non-zero value per row, Nayebi et al. [80] avoid the projection issue. In their construction, the spaces $\mathcal{Y}$ and $\mathcal{X}$ are constrained to $[-1, 1]^d$ and $[-1, 1]^D$ respectively and $\tilde{\rho}^{-1}$ simply propagates the potentially negated value of a dimension in $\mathcal{Y}$ to one or multiple dimensions in $\mathcal{X}$. If the effective subspace is axis-aligned – meaning that the effective subspace can be recovered by simply removing $D - d_e$ dimensions from $\mathcal{X}$ – the HeSBO embedding allows us to analytically determine the probability of the user-chosen subspace to contain any optimum. For this purpose, it is useful to interpret the HeSBO embedding as an assignment of input dimensions (i.e., dimensions of $\mathcal{X}$) to target dimensions (i.e., dimensions of $\mathcal{Y}$). Consider the following HeSBO projection matrix

$$S = \begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix},$$

which maps from a three-dimensional target space $\mathcal{Y}$ to a five-dimensional input space $\mathcal{X}$. For each combination of row and column (i, j), $S_{ij} \neq 0$ can be interpreted as "The i-th input dimension is assigned to the j-th target dimension."

Now, consider an axis-aligned effective subspace with d_e effective dimensions. Clearly, if $d < d_e$, the target space $\mathcal{Y}$ cannot represent any optimum $\boldsymbol{z}^* \in \mathcal{Z}$. However, if $d \geq d_e$, $\mathcal{Y}$ can represent any optimum in $\mathcal{Z}$ if all the effective dimensions are mapped to distinct dimensions of $\mathcal{Y}$. For each input dimension, HeSBO samples a target dimension uniformly at random. This means that a target dimension may

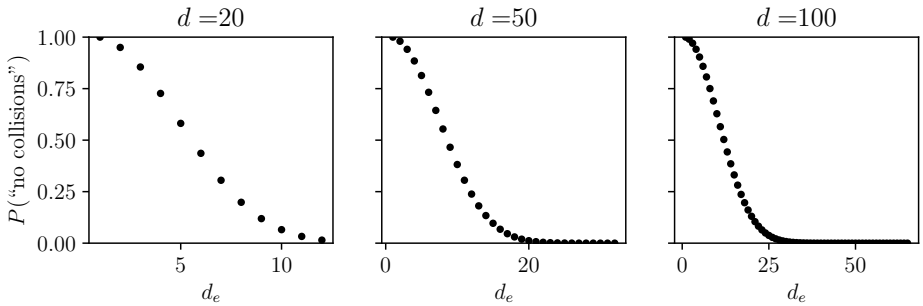


Figure 11: The HeSBO embedding's probability of a collision-free assignment of d_e effective dimensions to d target dimensions. The probability of a collision-free assignment is independent of the input dimensionality D, shrinks with d_e, grows with d, and is only 1 if $d_e = 1$.

not have any input dimensions assigned to it.

We can calculate the probability of all effective dimensions being mapped to distinct target dimensions. Assume we assign the d_e effective dimensions to the d target dimensions. For the first effective dimension, we are free to assign it to any target dimension. Hence, there are d options. For the second dimension, we only have $d - 1$ choices. Therefore, the probability of a collision-free assignment is given by

$$p_{\text{opt}}(d, d_e) = \frac{d}{d} \cdot \frac{d-1}{d} \cdot \ldots \cdot \frac{d-d_e+1}{d} = \frac{d!}{(d-d_e)!d^{d_e}}, \tag{46}$$

matching the expression given by Letham et al. [64].

Figure 11 shows the probabilities of collision-free assignments for embedding and effective dimensionalities $d = (20, 50, 100)$ and $d_e = (1, \ldots, \lfloor 0.75d \rfloor)$. The probability of a collision-free assignment decreases with d_e. At the same time, it increases with d, as seen by looking at the probability of a fixed d_e (e.g., 10) for different d. Surprisingly, the probability of a collision-free assignment of the HeSBO embedding is *independent of the input dimensionality* D. This means that even if the dimensionality of the embedding matches the input dimensionality ($d = D$), the HeSBO embedding risks collisions even though a collision-free assignment can be trivially generated by choosing the projection matrix S as the identity matrix. This is because some target dimensions may not have any, while others may have overproportionally many input dimensions assigned to them. Intuitively, assigning roughly equally many input dimensions to each target dimension should also decrease the risk of assigning two effective dimensions to the same target dimension. Paper I proposes a refinement of the HeSBO embedding with a strictly higher probability of a collision-free assignment, which we refer to as the BAxUS embedding. The BAxUS embedding distributes the input dimensions almost evenly across the target

dimensions. This way, the BAxUS embedding has a strictly higher probability of collision-free assignment, and paper 1 gives theoretical proof that no other embedding of this type can have a higher probability of collision-free assignment than the BAxUS embedding.

Non-data-agnostic methods do not work with random projections but aim to find a suitable projection for optimization. Examples using linear projections are KPLS [8], which uses partial least squares (PLS) to learn a projection suitable for optimization, and Garnett et al. [35], who use active learning to choose points for learning the embedding matrix. Variational autoencoders (VAEs) are often used for non-linear projections due to their structured latent space, which lends itself to optimization [113, 76, 68, 72]. While VAE-based approaches usually optimize the GP HPs and the projection model in two separate phases, deep kernel learning (DKL) [121] jointly optimizes projection and GP HPs. In particular, DKL uses a kernel expression [121]

$$k(\boldsymbol{x}, \boldsymbol{x}' \mid \boldsymbol{\theta}, \boldsymbol{\xi}) = k_{\text{base}}(\psi(\boldsymbol{x} \mid \boldsymbol{\xi}), \psi(\boldsymbol{x}' \mid \boldsymbol{\xi}) \mid \boldsymbol{\theta}), \tag{47}$$

where θ are the HPs of the base kernel (e.g., an radial basis function (RBF) or Matérn kernel), and ψ is a neural network (NN) parametrized by $\boldsymbol{\xi}$. While KPLS conforms to Equation (47), it is not considered DKL since PLS is not a deep-learning method. DKL can be combined with various neural network types, such as deep convolutional neural networks, recurrent neural networks, or graph neural networks, and can therefore be applied to various inputs. However, using a powerful model ψ in the kernel expression can lead to severe overfitting [84], which limits the applicability of DKL. Furthermore, non-data agnostic approaches usually require significant amounts of training data to learn the projection with more complex models such as VAEs, demanding more data than linear models like PLS.

Dictionary-Based Embeddings. Dictionary-based embeddings are linear but are conceptually different enough to be discussed separately. BODi [22] introduced this embedding approach based on the concept of *anchor points*. Anchor points are points in the high-dimensional search space $\mathcal{X}$ that serve as landmarks for other points. Figure 12 exemplifies this for a two-dimensional continuous search space. The figure uses three anchor points to represent the center point. The distance to each anchor point gives one component of the vector representing the center point. If there are more anchor points than $\mathcal{X}$ has dimensions, this can result in a higher-dimensional representation of points than their original representation. Conversely, few anchor points can give a low-dimensional representation.

Figure 12 gives an intuition to the concept of anchor points. However, BODi is designed specifically for combinatorial problems – the anchor points are defined in a binary or categorical space, and the distance measure is chosen as the Hamming

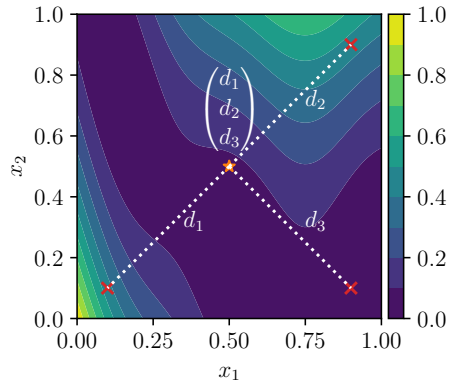


Figure 12: To represent a point in the space, `BODi` uses the distance to pre-defined anchor points. In this example, the point in the center (marked with a star) is represented by the three anchor points (marked with red crosses). The distance to each anchor point is one element of the vector representing the center point.

distance, resulting in an integer-valued subspace $\mathcal{Z}$ as all distances are integer-valued. The dictionary of anchor points is resampled in each BO iteration, resulting in an ever-changing representation of the previously observed points. Interestingly, `BODi` *increases* the problem's dimensionality by using 128 anchor points but only considering problems with up to 60 dimensions. A GP with a standard 5/2-Matérn kernel is then fitted on the 128-dimensional data representation, and the next point is chosen by maximizing the expected improvement (EI) with a local-search approach with restarts.

One key component of `BODi` is the choice of the anchor points. `BODi` uses anchor points from a wide range of *sequencies* – the number of changes from 1 to 0 and vice versa – and Deshwal et al. [22] show that this construction has a crucial impact on the performance compared to other sampling approaches. To create a dictionary of anchor points with diverse sequences, for each anchor point $\boldsymbol{a}$, `BODi` first samples a Bernoulli parameter $\beta \sim \mathcal{U}(0, 1)$ and then samples the value for each dimension of the anchor point as $a_i \sim \text{Bernoulli}(\beta)$.

Paper III identifies that `BODi` benefits from a particular benchmark structure where the optimal solution is attained when all parameters have similar values and shows performance degradations when this structure is absent.

3.3 Trust-Region Methods

Instead of reducing the dimensionality of the GP, trust region (TR)-based approaches focus the search on a subregion of the search space $\mathcal{X}$. Therefore, the GP no longer

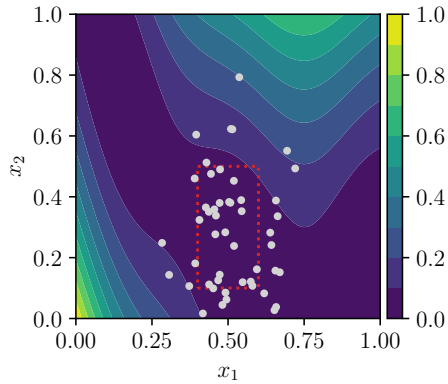


Figure 13: The two-dimensional `Branin` function with normalized bounds and observations. The TR is shown in red. Only points inside of the TR are considered for the next evaluation.

needs to model the entire function f accurately but only a subregion of interest (see Figure 13).

TuRBO. `TuRBO` ("**Tru**st **R**egion **B**ayesian **O**ptimization", Eriksson et al. [29]) introduced TRs to BO, with some key concepts introduced in earlier work [93, 95, 94]. TRs in `TuRBO` are hyper-rectangles of a given volume L^D where L is the *base side length* of the TR. The base side length is not used in practice, but the side lengths of the TR are scaled according to the length scales $\boldsymbol{\ell}$ of the GP surrogate and normalized so the volume remains at L^D:

$$L_i = \frac{\ell_i L}{\sqrt[D]{\prod_{i=1}^{D} \ell_i}}. \tag{48}$$

By this construction, the extent of the TR is smaller for dimensions with short and longer for dimensions with long length scales. The motivation behind this approach is that dimensions with short length scales are more important and, hence, deserve a more fine-grained search. The center of the TR is chosen as the incumbent solution, and the TR is clamped to stay within $\mathcal{X}$ if it would extend outside.

New points to evaluate are chosen by Thompson sampling (TS) (see Section 2.4) based on a set of candidate points chosen from within the TR. `TuRBO` copies the incumbent solution $\boldsymbol{x}^+$ several times and, for each copy, perturbs a subset of the dimensions to create the candidates. This strategy was previously introduced by [93]. The probability of perturbing a dimension is chosen to be $\min\left(1, \frac{20}{D}\right)$ so that at most 20 dimensions are perturbed on average. For each candidate, `TuRBO` generates a scrambled Sobol sequence [86] of the same dimensionality within the TR. It replaces

the values of the chosen dimensions with those of the Sobol sequence. The next point is then selected by evaluating the TS on the candidates and selecting the candidate with the minimum value. Both TRs and the sparse perturbations help `TuRBO` operate locally.

The volume of the TR is increased if `TuRBO` repeatedly fails to make progress and decreases if better solutions are found consecutively. In particular, `TuRBO` initializes a TR with $L = 0.8$ and doubles L if no better solutions are found in D consecutive evaluations. If it finds a better solution three times in a row, L is halved. The value of L cannot exceed 1.6. Suppose L falls below 2^{-7}. In that case, `TuRBO` "restarts" the TR, i.e., it discards previous observations, samples, and evaluates new initial candidates and centers a new TR on the best of those initial candidates.

`TuRBO` further supports batched evaluations and can maintain several TRs in parallel by evaluating the candidates on several TS and TRs, choosing the candidate – or candidates when doing batched evaluations – with minimum value across all candidates and TRs.

Paper 1 leverages TRs to focus the search on subregions of increasingly dimensional subspaces. Other than `TuRBO`, the `BAxUS` algorithm proposed in Paper 1 rarely discards previous observations. Instead, it resizes a TR to its initial size when switching from a lower- to a higher-dimensional space. `BAxUS` only discards observations if it has already reached the full-dimensional input space $\mathcal{X}$, i.e., $\mathcal{Z} = \mathcal{X}$ up to a re-ordering of the dimensions, and the TR falls below the minimum base length.

CASMOPOLITAN. `Casmopolitan` [116] extends `TuRBO` [29] to categorical and categorical-continuous mixed spaces by defining the TR in terms of the Hamming distance between categorical solutions. Consider a problem with three categorical dimensions $D_1 \in \{0, 1, 2\}, D_2 \in \{0, 1\}, D_3 \in \{0, 1, 2, 3\}$ and two points $\boldsymbol{x} = \{0, 1, 3\}, \boldsymbol{y} = \{1, 0, 3\}$. The Hamming distance between these points is two since we need to change the value of $\boldsymbol{x}$ in two places to get $\boldsymbol{y}$. Two points have the maximum Hamming distance of D if they vary in all dimensions; the minimum Hamming distance is zero. Similar to `TuRBO`, only consider points within a certain Hamming distance to the current-best solution as candidates for the next evaluation. `Casmopolitan` can also model mixed spaces by maintaining one TR for the continuous and another for the categorical variables.

`Casmopolitan` uses a Hamming kernel for the categorical variables

$$k_{\text{cat}}(\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}'_{\text{cat}}) = \exp\left(\frac{1}{N_{\text{cat}}} \sum_{i=1}^{N_{\text{cat}}} \frac{\delta\left(\boldsymbol{x}_{\text{cat}_i}, \boldsymbol{x}'_{\text{cat}_i}\right)}{\ell_i}\right), \tag{49}$$

and a mixture of a sum and a product kernel [96]

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x}) = \lambda &\left(k_{\text{cont}}(\boldsymbol{x}_{\text{cont}}, \boldsymbol{x}'_{\text{cont}}) \cdot k_{\text{cat}}(\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}'_{\text{cat}})\right) \\ &+ (1-\lambda)\left(k_{\text{cont}}(\boldsymbol{x}_{\text{cont}}, \boldsymbol{x}'_{\text{cont}}) + k_{\text{cat}}(\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}'_{\text{cat}})\right), \end{aligned} \tag{50}$$

where N_{cat} is the number of categorical variables and

$$\boldsymbol{x}_{\text{cat}} = \left(\boldsymbol{x}_i : \delta_{[D_i \text{ is categorical}]}\right) \tag{51}$$

$$\boldsymbol{x}_{\text{cont}} = \left(\boldsymbol{x}_i : \delta_{[D_i \text{ is continuous}]}\right). \tag{52}$$

Similar to `BAxUS` (Paper I), the `Bounce` algorithm proposed in Paper III operates in subspaces of increasing dimensionality. While `BAxUS` only considers continuous space, `Bounce` considers combinatorial or mixed spaces. `Bounce` follows `Casmopolitan` in using TRs for combinatorial spaces, i.e., it maintains two separate base lengths L^{comb} and L^{cont} for combinatorial and continuous variables, respectively. In contrast to `Casmopolitan` and `TuRBO`, which shrink or expand the base lengths by constant factors, `Bounce` uses flexible TR shrinkage and growth factors. This is because `Bounce` is designed to allocate a sample budget to a subspace that is proportional to that subspace's dimensionality. However, `TuRBO` and `Casmopolitan` have to fail a fixed number of times in finding a better solution to shrink the TR base length. Furthermore, the base length is only shrunk by a constant factor, so the minimum number of function evaluations `TuRBO` and `Casmopolitan` require to re-start a TR is relatively high. To enable `Bounce` to allocate an arbitrary positive number of function evaluations for a subspace, it shrinks or expands the TR base length after each function evaluation by a factor designed to reach the minimum base length in the budget allocated to the current subspace. This more flexible TR management and improved length scale priors let `Bounce` outperform `BAxUS` even in the purely continuous case.

3.4 Monte-Carlo Tree Search

Monte-Carlo tree search (MCTS)-based methods represent regions of the search space or subsets of variables in a tree structure to simplify the problem of minimizing f.

Similar to TRs, space partitioning methods narrow down the search space. Wang et al. [117] and Li et al. [66] use a tree-representation and MCTS to partition the space. At every tree level, they separate the points corresponding to that level into two clusters using k-means on the feature vector $(\boldsymbol{x}, y)$. These clusters give the nodes in the binary tree, where each level has one "good" and one "bad" node. The average function values of the associated points give the specific value of a node. The node's values and number of visits are used to select the node to visit next. The clusters of each inner node are trained to train an support vector machine (SVM) to obtain a boundary between the good and bad regions. When maximizing α, only points inside a region defined by repeatedly bounding the space as one follows a path in the tree are considered.

The difference between Wang et al. [117]'s `LA-MCTS` ("Latent Action Monte-Carlo Tree Search") and Li et al. [66]'s `HiBO` ("Hierarchical Bayesian Optimization") is the way the region to visit is selected. `LA-MCTS` strictly follows a classical MCTS approach that trades off a node's value and the number of times it has been visited, whereas `HiBO` uses a node's value and its number of visits to determine the *sampling potential* which is used to modify the acquisition function α. Nodes with a lower sampling potential receive a lower acquisition value than before the adjustment, whereas the acquisition value is boosted for nodes with high sampling potential.

Instead of partitioning the space, `MCTS-VS` ("Monte-Carlo Tree Search Variable Selection") [105] uses MCTS to separate "good" from "bad" variables. Each variable receives a score computed using the function values of queries involving this point. Tree nodes consist of subsets of variables that are optimized over when deciding which point to evaluate next. A "fill-in" strategy chooses the values for the remaining variables.

Paper II introduces the `GTBO` algorithm, which uses *group testing* [24] to identify active variables for sparse axis-aligned problems, i.e., problems where only a subset of the parameters have a considerable impact on the objective value and the other parameters can be varied without changing the function value. While other methods, such as `BAxUS` (paper I), `Bounce` (paper III), or `SAASBO` [27], also leverage this assumption, `GTBO` differs in that it returns a clear-cut decision whether a variable is active or not. Based on this result, `GTBO` assigns different length scale priors to active and inactive variables, considerably improving the sample complexity as evidenced by the empirical results. Furthermore, `GTBO` gives valuable insights into which parameters matter for a problem and which do not.

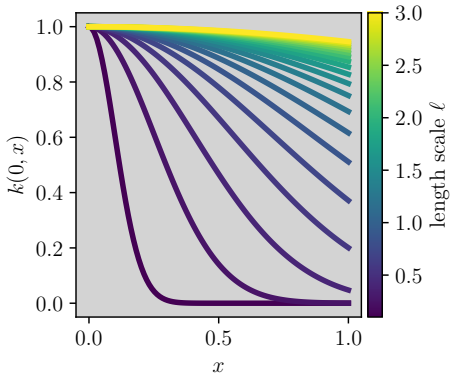


Figure 14: RBF kernel values for different distances x from the origin and different length scales ℓ between 0.1 and 3. The kernel values change quickly for short length scale values but little for long length scale values.

3.5 Long Length Scales

When using a stationary kernel, an alternative way of "deactivating" a parameter is to set its length scale to a very high value. Consider the r term from Equation 17 which is used in the RBF, 3/2-, and 5/2-Matérn kernels. The maximum contribution of a parameter to r is $1/\ell_i^2$ if $\mathcal{X}$ is normalized to a unit hypercube. If ℓ_i is chosen low, the contribution of parameter i changes quickly. Figure 14 illustrates this. With a short length scale ($\ell = 0.1$, bottom-most curve), the kernel value quickly drops from 1 to 0 as the distance from x to the origin increases. A dimension with a low associated length scale value would significantly impact the overall kernel contribution, which is the product of the individual contributions in the case of an RBF kernel. Conversely, a dimension with a long length scale does not change the value of the kernel significantly, as its value remains close to 1 – the neutral element for multiplication.

An alternative way of thinking about long length scales is that they fight the curse of dimensionality since kernels with long length scales consider points "similar" even if they are distant in space. Hence, the longer the length scales, the less data is required to cover the space. Hvarfner et al. [51] leverage this idea. They use a length scale prior mode scales with $\sqrt{D}$ to counteract increasing distances. In their initial preprint, Xu and Zhe [127] posit a uniform $\mathcal{U}(10^{-3}, 30)$ prior on the length scales and sample initial length scale values from this prior. This results in average length scale values of around 15, which is sufficiently long to avoid vanishing gradients for a wide range of values of D. In a revised version, they independently identified vanishing gradients as a challenge in HDBO and propose to scale the initial value of the length scale with $\sqrt{D}$ before the GP fitting [128].

Eriksson and Jankowiak [27] use a length scale prior with a high mass on long length scale values to model the assumption that only a few parameters are "active" in high-dimensional spaces. In particular, they sample a shrinkage parameter $\tau \sim \mathcal{HC}(0.1)$ from a half-Cauchy distribution and, subsequently, $\sqrt{\ell}^{-1} \sim \mathcal{HC}(\tau)$, resulting in length scales with a mean value of approximately 6.217 and a median value of approximately 3.1 under the prior. With sufficient evidence, their method `SAASBO` ("Sparse Axis-Aligned Subspace BO") can yield shorter length scales, and the number of dimensions with short length scales typically increases during the optimization as more observations are gathered. Hence, `SAASBO` is designed to discover an axis-aligned subspace automatically.

Examples of sparse axis-aligned functions are f_1 and f_2 in Figure 9. Both functions do not change for one of the parameters. Eriksson and Jankowiak [27] model such inactive parameters with long length scales, i.e., they assume that the function is mostly flat for that parameter (like f_1 is flat for x_2 in Figure 9). The `GTBO` algorithm proposed in paper 11 takes another approach and first learns a dichotomy of active and inactive dimensions. Other than `SAASBO` [27], which learns active dimensions during the optimization, `GTBO` defines separate hyperpriors for active and inactive dimensions that are not changed during the optimization. For problems exhibiting a sparse axis-aligned subspace, `GTBO` achieves state-of-the-art (SOTA) BO performance.

3.6 Vanishing Gradients

Many acquisition functions (AFs) in BO are defined in terms of the posterior mean and variance. Examples are EI (Equation (32)), probability of improvement (PI) (Equation (37)), and upper-confidence bound (UCB) (Equation (38)). Furthermore, those AFs usually only vary moderately with changing GP posterior. Consequently, many AFs are flat where the GP posterior is flat. Figure 15 shows the normalized AF value (top row) and the absolute gradient (bottom row) of the `LogEI`, UCB ($\beta = 0.1$), and PI AFs. The AF values are normalized to [0,1] with min-max scaling for each AF individually. The gradients are not normalized and are plotted on a log scale. The `LogEI` AF has been proposed by Ament et al. [1]. It has the same set of optimizers as EI but is a numerically more stable variant and is less likely to exhibit vanishing gradients in regions where the AF is nearly flat. The top left panel in Figure 15 shows the exponentiated `LogEI` values – the gradients are obtained on the regular `LogEI` AF. The AF values are obtained on a two-dimensional GP with an isotropic $5/2$-Matérn kernel, length scale $\ell = 2.5 \cdot 10^{-2}$, and noise term $\sigma_n^2 = 10^{-4}$ conditioned on five points drawn from a Sobol sequence and evaluated on a GP prior sample with the same hyperparameter configuration. The gradients are

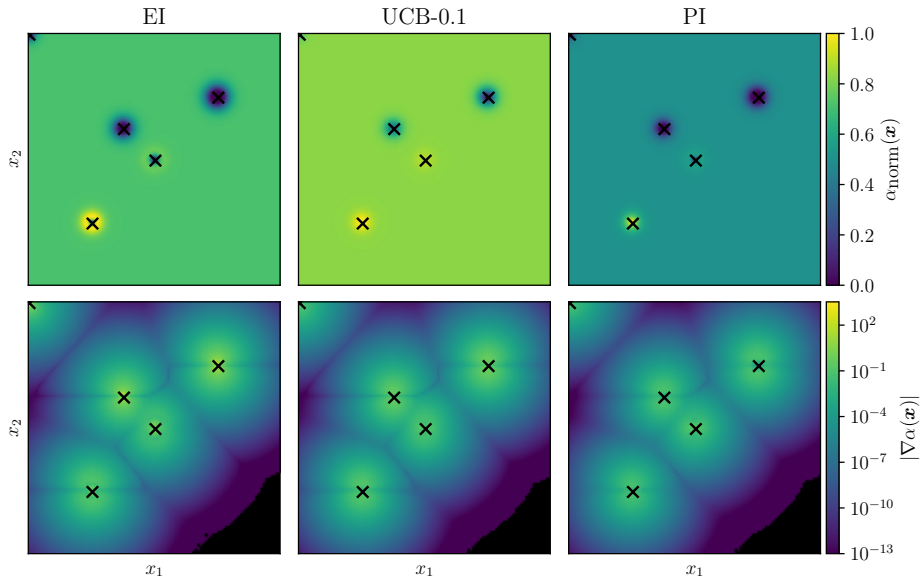


Figure 15: The acquisition surface (top row) and its absolute gradients (bottom row) for different AFs (columns). The AF values are computed for a GP conditioned on five points shown marked with black crosses. The values of each AF are individually normalized to 0-1; the values in the bottom row are not normalized. For all three AFs, the surface quickly flattens out as the distance to previous observations increases.

obtained numerically using finite differences. Values in black correspond to values of zero. The five points the GP is conditioned on are plotted in black.

The AF values are almost equal everywhere for all AFs except close to the five observations, as can be seen in the top row of Figure 15. Close to an observed point, the AF value is either smaller or higher than this 'default value', depending on whether f has shown bad or good performance for that point. The gradients behave similarly for the three different AFs: They attain their highest values close to previous observations and quickly vanish as the distance to an observation increases. Vanishing gradients are problematic for BO due to the multi-start gradient descent (GD)-based optimization of the AF. If the starting point for the AF maximization falls into an area where the AF exhibits vanishing gradients, it will not be further optimized – the optimization stagnates. While increasing the number of starting points helps in low-dimensional spaces, this is infeasible in high-dimensional spaces. Assume you divide the unit hypercube representing the search space $\mathcal{X}$ into smaller hypercubes, each with a side length $\frac{1}{2}$ and you want to place a starting point in each of these hypercubes. Then, for a 100-dimensional space, you need $2^{100} \approx 1.268 \cdot 10^{30}$ starting points.

Thus, if the GP's length-scale hyperparameters are poorly initialized in a high-dimensional problem, the GP may never learn an appropriate model of f. This, in turn, can cause the BO process to perform suboptimally or fail entirely.

Paper VI shows that vanishing gradients of the AF continue to pose a challenge for HDBO, despite the existence of a numerically more stable variant of EI. While Figure 15 gives an intuition for a low-dimensional problem with extremely short length scales, paper VI shows that this problem exists for real-world scenarios with considerably longer length scales.

4 Contributions

This thesis advances the field of Bayesian optimization (BO) by making high-dimensional Bayesian optimization (HDBO) more practical, quantifying exploration to design new acquisition functions (AFs), and strengthening the understanding of simple BO setups in high dimensions. In particular, we answer the research questions outlined in Section 1.2 as follows.

4.1 RQ 1: Understanding and Addressing Failure Modes in High-Dimensional Bayesian Optimization

The connection between acquisition functions for Bayesian optimization. The AF and the surrogate model are the cornerstones of any BO algorithm. A bad AF can cause poor optimization performance even if the surrogate model is fitted accurately. Good AFs should balance the exploration-exploitation trade-off (EETO), which dictates that previously unobserved areas should be visited to learn more about the black-box function, but also areas known to yield good function values to improve over the current observation.

Paper IV contributes two quantitative measures of exploration (observation traveling salesman distance (OTSD) and observation entropy (OE)) that capture how "global" or "local" a BO algorithm's search is. These measures can reveal when an algorithm switches from global exploration to local refinement (something previous analyses approximated only indirectly), and they enable a new taxonomy of AF behaviors linked to performance. For instance, in Eriksson et al. [29, Figure 6, middle-left panel], the authors aim to show how their method, `TuRBO` (see Section 3.3), initially exhibits an exploratory behavior at the beginning and a local behavior at the end of the optimization. This is done by measuring the *trajectory arc length* of a trust region (TR) center, i.e., the cumulative distance the center of the TR has traveled. Here, the OTSD or OE would give a clearer picture of the algorithm's behavior since they not only consider the center of a TR but all points observed up to each BO iteration. Similarly, in Hvarfner et al. [51, Figure 20], the authors assess the distance of new points from the incumbent observation to analyze how Gaussian process (GP) kernel priors affect locality. However, this method falls short in scenarios where, for instance, points from two high-performing regions are frequently selected by the AF. Both methods are used to create a taxonomy of the explorative behavior of various AFs, connecting an AF's level of exploration to its empirical performance and studying cases in which an AF's behavior changes.

Paper V introduces variational entropy search (VES), a novel AF for BO that

bridges the gap between myopic and non-myopic AFs in BO. In particular, VES is derived from a new information-theoretic interpretation of the popular expected improvement (EI) acquisition function. Using variational inference (VI) on the exponential distribution to estimate the distribution of the optimal value in the max-value entropy search (MES) acquisition function, paper V shows that this variational approximation of MES is equivalent to EI, uncovering a new connection between these popular AFs. This connection is surprising (since conventional wisdom held EI and information-theoretic approaches to be distinct [123]), and it leads to a new AF, VES, which often outperforms both EI and MES in practice. Besides proving the theoretical equivalence between EI and this "variational" variant of MES, paper V also shows this equivalence empirically. Replacing the distributional family in the VI step with the more expressive Gamma distribution yields the proposed VES AF.

Failure modes and their counter-measures for high-dimensional spaces. Two works recently questioned the tribal knowledge that BO does not perform well on most applications with considerably more than 20 parameters. While both works delivered approaches to an explanation, several questions remained that paper VI answers.

Paper VI identifies two failure modes of HDBO, vanishing gradients of the AF and the marginal log-likelihood (MLL), which occur in high dimensions and models with overly short length scales. Previous works solved these issues, although potentially unaware, by either employing a long length scale prior or changing the length scale initialization strategy. Based on the numerical failure modes of HDBO, we propose a dimensionality-scaled length scale initialization strategy designed to avoid these issues and show that this strategy suffices to compete with the state-of-the-art (SOTA) without having to utilize length scale hyperpriors, which should only be used based on prior knowledge. In other words, using a prior (maximum a-posteriori estimation (MAP)) stabilizes the hyperparameter estimates (lower variance) but at the cost of biasing them, whereas pure maximum-likelihood estimation (MLE) is noisy but not systematically skewed. In comparison, MAP has higher bias but lower variance. We further identify random axis-aligned subspace perturbations (RAASP) as an important method for enhancing HDBO. Finally, we use our insights to empirically derive design recommendations for HDBO algorithms and validate our choices. Note that Xu et al. [128] independently also identified vanishing gradients of the MLL as a challenge for HDBO and proposed to use MLE with a initial length scale value scaled with $\sqrt{D}$ before the GP fitting.

4.2 RQ 2: Practical High-Dimensional Bayesian Optimization

Papers I, II, and III present contributions to this research question.

RQ 2.1: Optimizing problems of unknown effective dimensionality. Paper I introduces `BAxUS`, a novel HDBO algorithm that leverages a family of nested subspaces to optimize problems with thousands of dimensions and removes the need to estimate the dimensionality of the effective subspace. `BAxUS` starts the optimization in a very low-dimensional space and increases the dimensionality of the subspace over time to eventually model all problem parameters independently. Observations from lower-dimensional subspaces are carried over to higher-dimensional subspaces to maintain sample efficiency. `BAxUS` further proposes a novel method for constructing the embedding, which ensures that all dimensions in the embedding model have an almost equal number of parameters. Paper I proves that this construction is the optimal method for constructing projections of this type in terms of the worst-case probability of containing an optimum, improving over previously proposed strategies.

Paper III follows up on `BAxUS` and proposes `Bounce`, an algorithm for practical HDBO in combinatorial and mixed spaces. `Bounce` extends the approach of nested subspaces to combinatorial variables by proposing strategies for jointly representing several combinatorial parameters of differing cardinalities with a single variable. This allows for the optimization of high-dimensional combinatorial spaces in nested subspaces. `Bounce` further improves the choice of several hyperparameters (HPs) and thus also improves over `BAxUS` for purely continuous problems. Furthermore, `Bounce` supports parallel function evaluations using a novel TR management strategy. Finally, `Bounce` identifies previously unknown properties of two other combinatorial BO algorithms that let them perform considerably better on problems with a simple structure than on arbitrarily structured combinatorial problems. Paper III empirically verifies that the performance of the methods degrades once a benchmark no longer exhibits this structure.

RQ 2.2: Learning and using relevant problem parameters for high-dimensional Bayesian optimization. Paper II addresses the problem of HDBO by extending the mathematical framework of *adaptive group testing (GT)*, which assumes binary function values, to real-valued functions. The proposed method, `GTBO`, consists of two phases. In the first phase, the GT procedure identifies problem parameters relevant to the optimization. The success of this phase is contingent on an axis-aligned structure of the problem, i.e., some parameters are assumed not to affect the function value. After identifying the relevant parameters, `GTBO` places a prior

promoting long length scales on the inactive and a prior promoting short length scales on the active parameters, allowing the method to leverage the insights of the GT phase.

5 Conclusions and Future Work

Optimizing expensive black-box functions is a notoriously difficult burden in many fields. Bayesian optimization (BO) can help lift this burden as its sample efficiency makes optimizing such functions feasible. However, this promise often only holds for problems with a few tunable parameters, and the curse of dimensionality (COD) poses a serious obstacle to solving problems even of moderate dimensionality. Therefore, high-dimensional Bayesian optimization (HDBO) has been considered as one of the "holy grails" of the field [120, 80]. This thesis approaches this holy grail from different directions.

First, it introduces several practical methods that scale to hundreds or thousands of dimensions under the effective subspace assumption (papers I, II, and III). Besides improving state-of-the-art (SOTA) optimization performance, these approaches contribute to several innovations. `BAxUS` (paper I) introduces nested embeddings to dynamically increase the dimensionality of the subspace for subspace BO. `Bounce` (paper III) extends this concept to combinatorial spaces, providing a data-agnostic embedding for combinatorial BO. Finally, `GTBO` (paper II) explicitly learns the active variables of a problem and then uses this information to increase sample efficiency. As a byproduct, practitioners gain important insights into a problem. These three methods are valuable tools for practical HDBO problems and establish new concepts that will spark future work.

Second, this thesis introduces new methods to study the behavior of BO algorithms. Observation traveling salesman distance (OTSD) and observation entropy (OE) allow us to quantify the level of exploration of a BO method, which often is of particular interest to understand an algorithm's behavior. These methods will be useful beyond that and allow dynamically switching the acquisition functions (AFs) or controlling the BO loop in other ways. Paper IV deepens our understanding of AFs by proposing a taxonomy of AFs based on their level of exploration and relating this to their empirical performance. Variational entropy search (VES) (paper V) further bridges the gap between myopic and non-myopic AFs by establishing an equivalence between expected improvement (EI) and max-value entropy search (MES) when using a variational approximation. In many cases, VES improves over the SOTA and can improve most BO setups.

Finally, this thesis analyzes the basic building blocks of any HDBO algorithm and shows how these can break in unexpected ways. Paper VI shows that constant initialization schemes cause vanishing gradients of the marginal log-likelihood (MLL) and the AF, causing the optimization to get stuck even if numerically stable EI variants are used. This can cause failures on HDBO problems, even if they are

amenable to long length scales, and hence could, in principle, be solved if only the MLL optimizer were able to learn the correct length scales.

Nevertheless, HDBO is far from being solved, and there remain open questions and research directions.

Learning the Effective Dimensionality. Several approaches using data-agnostic random embeddings to leverage the assumption of an effective subspace for BO require the definition of a subspace dimensionality d [120, 80, 64]. Paper I introduces `BAxUS`, which removes this limitation by building on a family of nested subspaces of increasing dimensionality. `BAxUS` considers subspaces of dimensionality of approximately two up to the full dimensionality of the problem, which can lie in the hundreds or thousands. `Bounce` (Paper III) operates similarly, albeit considering combinatorial spaces and improving several heuristic components of `BAxUS`. While eventually covering the full-dimensional search space is important for `BAxUS`' and `Bounce`'s theoretical guarantees, it potentially overshoots the effective dimensionality significantly. This reduces sample efficiency in later optimization stages as `BAxUS` and `Bounce` might have to model many irrelevant parameters. Once a higher-dimensional subspace no longer covers additional regions of the effective subspace, increasing the dimensionality only harms performance as it requires modeling irrelevant dimensions. Future work should aim to find this critical subspace dimensionality. This will provide an upper bound on the effective dimensionality and potentially drastically improve sample efficiency as `BAxUS` and `Bounce`, in late optimization stages, need to model many irrelevant dimensions.

Budget-Aware Optimization. Most BO strategies do not consider the budget. One-step AFs always choose the next point, assuming that the next evaluation will be the last, and even non-myopic AFs, such as entropy-based methods, choose the point that reduces uncertainty most, assuming there is only one remaining function evaluation. Multi-step AFs [53, 32, 126] assume there are more remaining evaluations but are limited to very short horizons, typically less than 10 evaluations. Hence, other strategies that consider the budget are needed.

Paper IV introduces two methods for quantifying the level of exploration of AFs for BO and creates a taxonomy of existing AFs. While more exploratory AFs, such as entropy-based methods, can help to learn more about the function, they can be wasteful if the computational budget is extremely limited. Contrarily, "greedy" AFs such as EI or probability of improvement (PI) can be useful for small budgets but often fail to explore unknown regions, which one may want to learn more about if the budget allows. Following up on the insights from paper IV, future research

should connect the choice of the AF with the budget. In settings with high budgets, one may favor more exploratory AFs to learn the function holistically and increase the chance of finding the global minimum. In low-budget settings, one may exploit a local minimum as much as possible and thus favor a greedy AF. The methods proposed in paper IV can guide the choice of the AF in these different settings.

Data-Dependent Initialization. Paper VI highlights that vanishing gradients in both the MLL and the AF are a critical failure mode in HDBO, especially in early stages with few observations. To mitigate this, paper VI proposes initializing the length scales of Gaussian process (GP) kernels proportionally to the problem dimensionality. This design choice counters the overly short length scales of standard initialization schemes, which result in nearly zero kernel entries and, consequently, flat optimization surfaces. Other works [51, 127] similarly advocate for priors promoting long length scales, suggesting that such scaling significantly improves robustness in high-dimensional settings.

However, current strategies are static and agnostic to the growing amount of data. As the number of observations increases, more of the kernel matrix becomes non-zero, and gradient flow naturally improves. Initialization schemes that only consider a problem's dimensionality may thus become suboptimal in later optimization stages. Future work should explore adaptive initialization strategies that respond dynamically to the state of the data, e.g., by shrinking the length scales as the posterior becomes more confident or as the effective dimensionality narrows.

References

[1] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected Improvements to Expected Improvement for Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] BoTorch Authors. BoTorch: Bayesian Optimization in PyTorch. `https://github.com/pytorch/botorch`, 2024. Accessed: 2025-03-12.

[3] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[4] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. *International Conference on Machine Learning*, pages 462–471. PMLR, 2018.

[5] Carolin Benjamins, Elena Raponi, Anja Jankovic, Koen van der Blom, Maria Laura Santoni, Marius Lindauer, and Carola Doerr. PI is back! Switching Acquisition Functions in Bayesian Optimization. *2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*, 2022.

[6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24. Curran Associates, Inc., 2011.

[7] Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

[8] Mohamed Amine Bouhlel, Nathalie Bartoli, Abdelkader Otsmane, and Joseph Morlier. Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53:935–952, 2016.

[9] Benjamin Burger, Phillip M Maffettone, Vladimir V Gusev, Catherine M Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M Alston, Buyi Li, Rob Clowes, et al. A mobile robotic chemist. *Nature*, 583(7815):237–241, 2020.

[10] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian Gait Optimization for Bipedal Locomotion.

Learning and Intelligent Optimization, pages 274–290. Springer International Publishing, 2014.

[11] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[12] A. Candelieri, R. Perego, and F. Archetti. Bayesian Optimization of Pump Operations in Water Distribution Systems. *Journal of Global Optimization*, 71 (1):213–235, May 2018.

[13] Zachary Cosenza, Raul Astudillo, Peter Frazier, Keith Baar, and David E Block. Multi-Information Source Bayesian Optimization of Culture Media for Cellular Agriculture. *Biotechnology and Bioengineering*, 2022.

[14] Alexander I Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, et al. HEBO Pushing The Limits of Sample-Efficient Hyperparameter Optimisation. *Journal of Artificial Intelligence Research*, 74: 1269–1349, 2022.

[15] Samuel Daulton, Sait Cakmak, Maximilian Balandat, Michael A Osborne, Enlu Zhou, and Eytan Bakshy. Robust Multi-Objective Bayesian Optimization Under Input Noise. *International Conference on Machine Learning*, pages 4831–4866. PMLR, 2022.

[16] Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective Bayesian optimization over high-dimensional search spaces. *Uncertainty in Artificial Intelligence*, pages 507–517. PMLR, 2022.

[17] Samuel Daulton, Xingchen Wan, David Eriksson, Maximilian Balandat, Michael A. Osborne, and Eytan Bakshy. Bayesian Optimization over Discrete and Mixed Spaces via Probabilistic Reparameterization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[18] David J. C. MacKay and Radford M. Neal. Automatic Relevance Determination for Neural Networks. Technical report, Cambridge University, 1994. Technical Report in preparation.

[19] George De Ath, Jonathan E Fieldsend, and Richard M Everson. What do you Mean? The Role of the Mean Function in Bayesian Optimisation. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1623–1631, 2020.

[20] George De Ath, Richard M Everson, Alma AM Rahat, and Jonathan E Fieldsend. Greed is Good: Exploration and Exploitation Trade-offs in Bayesian Optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1):1–22, 2021.

[21] Aryan Deshwal, Syrine Belakaria, Janardhan Rao Doppa, and Dae Hyun Kim. Bayesian optimization over permutation spaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6515–6523, 2022.

[22] Aryan Deshwal, Sebastian Ament, Maximilian Balandat, Eytan Bakshy, Janardhan Rao Doppa, and David Eriksson. Bayesian Optimization over High-Dimensional Combinatorial Spaces via Dictionary-based Embeddings. *International Conference on Artificial Intelligence and Statistics*, pages 7021–7039. PMLR, 2023.

[23] Bach Do, Taiwo Adebiyi, and Ruda Zhang. Epsilon-greedy Thompson sampling to Bayesian optimization. *Journal of Computing and Information Science in Engineering*, 24(12), 2024.

[24] Robert Dorfman. The detection of defective members of large populations. *The Annals of mathematical statistics*, 14(4):436–440, 1943.

[25] David K Duvenaud, Hannes Nickisch, and Carl Rasmussen. Additive Gaussian Processes. *Advances in neural information processing systems*, 24, 2011.

[26] Adel Ejjeh, Leon Medvinsky, Aaron Councilman, Hemang Nehra, Suraj Sharma, Vikram Adve, Luigi Nardi, Eriko Nurvitadhi, and Rob A. Rutenbar. HPVM2FPGA: Enabling True Hardware-Agnostic FPGA Programming. *Proceedings of the 33rd IEEE International Conference on Application-specific Systems, Architectures, and Processors*, 2022.

[27] David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

[28] David Eriksson and Matthias Poloczek. Scalable Constrained Bayesian Optimization. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 730–738. PMLR, 13–15 Apr 2021.

[29] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5496–5507, 2019.

[30] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[31] Peter I. Frazier and Jialei Wang. *Bayesian Optimization for Materials Design*, pages 45–75. Springer International Publishing, Cham, 2016. ISBN 978-3-319-23871-5.

[32] Peter I Frazier, Warren B Powell, and Savas Dayanik. A Knowledge-Gradient Policy for Sequential Information Collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

[33] Jacob Gardner, Chuan Guo, Kilian Weinberger, Roman Garnett, and Roger Grosse. Discovering and exploiting additive structure for Bayesian optimization. *International Conference on Artificial Intelligence and Statistics*, pages 1311–1319, 2017.

[34] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[35] Roman Garnett, Michael A. Osborne, and Philipp Hennig. Active learning of linear embeddings for Gaussian processes. *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 230–239, Arlington, Virginia, USA, 2014. AUAI Press. ISBN 9780974903910.

[36] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian Optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.

[37] Miguel González-Duque, Richard Michael, Simon Bartels, Yevgen Zainchkovskyy, Søren Hauberg, and Wouter Boomsma. A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences. *arXiv preprint arXiv:2406.04739*, 2024.

[38] Antoine Grosnit, Rasul Tutunov, Alexandre Max Maraval, Ryan-Rhys Griffiths, Alexander I Cowen-Rivers, Lin Yang, Lin Zhu, Wenlong Lyu, Zhitang Chen, Jun Wang, et al. High-Dimensional Bayesian Optimisation with Variational Autoencoders and Deep Metric Learning. *arXiv preprint arXiv:2106.03609*, 2021.

[39] Florian Hase, Loïc M Roch, Christoph Kreisbeck, and Alán Aspuru-Guzik. Phoenics: a Bayesian optimizer for chemistry. *ACS central science*, 4(9):1134–1145, 2018.

[40] Florian Häse, Matteo Aldeghi, Riley J Hickman, Loïc M Roch, and Alán Aspuru-Guzik. Gryffin: An algorithm for Bayesian optimization of categorical

variables informed by expert knowledge. *Applied Physics Reviews*, 8(3):031406, 2021.

[41] Philipp Hennig and Christian J Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[42] Henry C Herbol, Weici Hu, Peter Frazier, Paulette Clancy, and Matthias Poloczek. Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization. *npj Computational Materials*, 4(1):1–7, 2018.

[43] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. *Advances in Neural Information Processing Systems*, 27, 2014.

[44] José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space. *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1470–1479. PMLR, 06–11 Aug 2017.

[45] Matthew Hoffman, Eric Brochu, and Nando de Freitas. Portfolio Allocation for Bayesian Optimization. *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 327–336, 2011.

[46] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[47] Zak E Hughes, Michelle A Nguyen, Jialei Wang, Yang Liu, Mark T Swihart, Matthias Poloczek, Peter I Frazier, Marc R Knecht, and Tiffany R Walsh. Tuning Materials-Binding Peptide Sequences toward Gold-and Silver-Binding Selectivity with Bayesian Optimization. *ACS nano*, 15(11):18260–18269, 2021.

[48] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.

[49] Carl Hvarfner, Frank Hutter, and Luigi Nardi. Joint Entropy Search for Maximally-Informed Bayesian Optimization. *Advances in Neural Information Processing Systems*, 35:11494–11506, 2022.

[50] Carl Hvarfner, Danny Stoll, Artur Souza, Luigi Nardi, Marius Lindauer, and Frank Hutter. PiBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization. *International Conference on Learning Representations*, 2022.

[51] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian Optimization Performs Great in High Dimensions. Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20793–20817. PMLR, 21–27 Jul 2024.

[52] Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets Riemannian manifolds in robot learning. *Conference on Robot Learning*, pages 233–246. PMLR, 2020.

[53] Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33:18039–18049, 2020.

[54] Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21:345–383, 2001.

[55] Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. *MOPTA 2008 Conference (20 August 2008)*, 2008.

[56] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455, 1998.

[57] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. *International conference on machine learning (ICML)*, pages 295–304, 2015.

[58] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018.

[59] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th*

International Conference on Artificial Intelligence and Statistics, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 20–22 Apr 2017.

[60] Mario Köppen. The curse of dimensionality. *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8, 2000.

[61] Vidhi Lalchand and Carl Edward Rasmussen. Approximate inference for fully Bayesian Gaussian process regression. *Symposium on Advances in Approximate Bayesian Inference*, pages 1–12. PMLR, 2020.

[62] Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in Bayesian optimization with applications in aerospace engineering. *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[63] Felix Leibfried, Vincent Dutordoir, ST John, and Nicolas Durrande. A Tutorial on Sparse Gaussian Processes and Variational Inference. *arXiv preprint arXiv:2012.13962*, 2020.

[64] Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional Bayesian optimization. *Advances in Neural Information Processing Systems 33*, NeurIPS, 2020.

[65] Shuang Li, Ke Li, and Wei Li. "Why Not Looking backward?" A Robust Two-Step Method to Automatically Terminate Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36:43435–43446, 2023.

[66] Wenxuan Li, Taiyi Wang, and Eiko Yoneki. HiBO: Hierarchical Bayesian Optimization via Adaptive Search Space Partitioning. *arXiv preprint arXiv:2410.23148*, 2024.

[67] Daniel J Lizotte, Tao Wang, Michael H Bowling, Dale Schuurmans, et al. Automatic Gait Optimization With Gaussian Process Regression. *IJCAI*, volume 7, pages 944–949, 2007.

[68] Xiaoyu Lu, Javier Gonzalez, Zhenwen Dai, and Neil D. Lawrence. Structured Variationally Auto-encoded Optimization. Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3273–3281. PMLR, 2018.

[69] Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

[70] Hauke F Maathuis, Roeland De Breuker, and Saullo G Castro. High-Dimensional Bayesian Optimisation with Large-Scale Constraints-An Application to Aeroelastic Tailoring. *AIAA SCITECH 2024 Forum*, page 2012, 2024.

[71] Arun Mannodi-Kanakkithodi and Maria KY Chan. Data-driven design of novel halide perovskite alloys. *Energy & Environmental Science*, 15(5):1930–1949, 2022.

[72] Natalie Maus, Haydn Thomas Jones, Juston Moore, Matt Kusner, John Bradshaw, and Jacob R. Gardner. Local latent space bayesian optimization over structured inputs. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[73] Matthias Mayr, Faseeh Ahmad, Konstantinos I. Chatzilygeroudis, Luigi Nardi, and Volker Krüger. Skill-based Multi-objective Reinforcement Learning of Industrial Robot Tasks with Planning and Knowledge Integration. *CoRR*, abs/2203.10033, 2022.

[74] Jonas Mockus. The application of Bayesian methods for seeking the extremum. *Towards global optimization*, 2:117, 1998.

[75] Jonas Mockus. The Bayesian approach to global optimization. *System Modeling and Optimization: Proceedings of the 10th IFIP Conference New York City, USA, August 31–September 4, 1981*, pages 473–481. Springer, 2005.

[76] Riccardo Moriconi, Marc Peter Deisenroth, and KS Sesh Kumar. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109:1925–1943, 2020.

[77] Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. BOSS: Bayesian Optimization over String Spaces. *Advances in neural information processing systems*, 33:15476–15486, 2020.

[78] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[79] Luigi Nardi, David Koeplinger, and Kunle Olukotun. Practical design space exploration. *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 347–358. IEEE, 2019.

[80] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian Optimization in Embedded Subspaces. *Proceedings of the 36th*

International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research (PMLR)*, pages 4752–4761, 09–15 Jun 2019.

[81] Diana M Negoescu, Peter I Frazier, and Warren B Powell. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[82] Diana M. Negoescu, Peter I. Frazier, and Warren B. Powell. The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[83] Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Regret for Expected Improvement over the Best-Observed Value and Stopping Condition. *Asian conference on machine learning*, pages 279–294. PMLR, 2017.

[84] Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. The Promises and Pitfalls of Deep Kernel Learning. *Uncertainty in Artificial Intelligence*, pages 1206–1216. PMLR, 2021.

[85] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian Optimization using the Graph Cartesian Product. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[86] Art B Owen. Scrambling Sobol' and Niederreiter–-Xing Points. *Journal of complexity*, 14(4):466–489, 1998.

[87] Daniel Packwood. *Bayesian Optimization for Materials Science*. Springer, 2017.

[88] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A Flexible Framework for Multi-Objective Bayesian Optimization using Random Scalarizations. *Uncertainty in Artificial Intelligence*, pages 766–776. PMLR, 2020.

[89] Maitreyee Sharma Priyadarshini, Oluwaseun Romiluyi, Yiran Wang, Kumar Miskin, Connor Ganley, and Paulette Clancy. Pal 2.0: a physics-driven bayesian optimization framework for material discovery. *Materials Horizons*, 11(3):781–791, 2024.

[90] Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778, 2018.

[91] Bahador Rashidi, Kerrick Johnstonbaugh, and Chao Gao. Cylindrical Thompson sampling for high-dimensional Bayesian optimization.

International Conference on Artificial Intelligence and Statistics, pages 3502–3510. PMLR, 2024.

[92] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006.

[93] Rommel G Regis. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers & Operations Research*, 38(5):837–853, 2011.

[94] Rommel G Regis. Trust regions in Kriging-based optimization with expected improvement. *Engineering optimization*, 48(6):1037–1059, 2016.

[95] Rommel G Regis and Christine A Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

[96] Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. *International Conference on Machine Learning*, pages 8276–8285. PMLR, 2020.

[97] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable Neural Architecture Search via Bayesian Optimisation with Weisfeiler-Lehman Kernels. *International Conference on Learning Representations*, 2021.

[98] Daniel Russo and Benjamin Van Roy. Learning to Optimize Via Posterior Sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[99] Bernhard Schölkopf and Alexander J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2002.

[100] Artur M Schweidtmann, Adam D Clayton, Nicholas Holmes, Eric Bradford, Richard A Bourne, and Alexei A Lapkin. Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. *Chemical Engineering Journal*, 352:277–282, 2018.

[101] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. *First Conference on Automated Machine Learning (Main Track)*, 2022.

[102] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[103] Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590 (7844):89–96, 2021.

[104] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.

[105] Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte Carlo Tree Search based Variable Selection for High Dimensional Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022.

[106] Artur Souza, Leonardo B Oliveira, Sabine Hollatz, Matt Feldman, Kunle Olukotun, James M Holton, Aina E Cohen, and Luigi Nardi. DeepFreak: Learning crystallography diffraction patterns with automated machine learning. *arXiv preprint arXiv:1904.11834*, 2019.

[107] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *Proceedings of the International Conference on Machine Learning*, 2010.

[108] Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets, optimization test problems. `https://www.sfu.ca/~ssurjano/optimization.html`. Accessed: 2024-09-01.

[109] Hampus Gummesson Svensson, Esben Jannik Bjerrum, Christian Tyrchan, Ola Engkvist, and Morteza Haghir Chehreghani. Autonomous drug design with multi-armed bandits. *2022 IEEE International Conference on Big Data (Big Data)*, pages 5584–5592. IEEE, 2022.

[110] Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter, and Michael A Osborne. Raiders of the lost architecture: Kernels for bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011*, 2014.

[111] Lorillee Tallorin, JiaLei Wang, Woojoo E Kim, Swagat Sahu, Nicolas M Kosa, Pu Yang, Matthew Thompson, Michael K Gilson, Peter I Frazier, Michael D Burkart, et al. Discovering de novo peptide substrates for enzymes using machine learning. *Nature communications*, 9(1):1–10, 2018.

[112] Petru Tighineanu, Lukas Grossberger, Paul Baireuther, Kathrin Skubch, Stefan Falkner, Julia Vinogradska, and Felix Berkenkamp. Scalable Meta-Learning

with Gaussian Processes. *International Conference on Artificial Intelligence and Statistics*, pages 1981–1989. PMLR, 2024.

[113] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:11259–11272, 2020.

[114] Ben Tu, Axel Gandy, Nikolas Kantas, and Behrang Shafei. Joint Entropy Search for Multi-objective Bayesian Optimization. *Advances in Neural Information Processing Systems*, 35:9922–9938, 2022.

[115] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials Discovery*, 4:18–21, 2016.

[116] Xingchen Wan, Vu Nguyen, Huong Ha, Binxin Ru, Cong Lu, and Michael A. Osborne. Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces. *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10663–10674. PMLR, 18–24 Jul 2021.

[117] 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 (NeurIPS)*, 33:19511–19522, 2020.

[118] Zi Wang and Stefanie Jegelka. Max-value Entropy Search for Efficient Bayesian Optimization. *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

[119] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[120] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *Journal of Artificial Intelligence Research (JAIR)*, 55:361–387, 2016.

[121] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep Kernel Learning. *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[122] James Wilson. Stopping Bayesian Optimization with Probabilistic Regret Bounds. *Advances in Neural Information Processing Systems*, 37:98264–98296, 2024.

[123] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for Bayesian optimization. *Advances in neural information processing systems*, 31, 2018.

[124] James T Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Pathwise Conditioning of Gaussian Processes. *Journal of Machine Learning Research*, 22(105):1–47, 2021.

[125] Stephen J Wright. Numerical Optimization, 2006.

[126] Jian Wu and Peter Frazier. Practical two-step lookahead Bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[127] Zhitong Xu and Shandian Zhe. Standard Gaussian Process is All You Need for High-Dimensional Bayesian Optimization. *arXiv preprint arXiv:2402.02746*, 2024.

[128] Zhitong Xu, Haitao Wang, Jeff M. Phillips, and Shandian Zhe. Standard Gaussian Process is All You Need for High-Dimensional Bayesian Optimization. *The Thirteenth International Conference on Learning Representations*, 2025.

[129] Juliusz Krzysztof Ziomek and Haitham Bou Ammar. Are Random Decompositions all we need in High Dimensional Bayesian Optimisation? *International Conference on Machine Learning*, pages 43347–43368. PMLR, 2023.

Chapter 1

Scientific publications

Author contributions

Table 1.1: Overview of contributions in each paper included in the thesis.

Paper	Concept	Implementation	Evaluation	Writing
I	◕	●	◕	◕
II	◑	◑	◑	◑
III	◕	●	●	◕
IV	◑	◕	◑	◑
V	◔	◑	◕	◑
VI	◕	●	●	◕

The dark portion of the circle represents the amount of work and responsibilities assigned to Leonard Papenmeier for each individual step.

- ◔ Leonard Papenmeier was a minor contributor to the work
- ◑ Leonard Papenmeier was a contributor to the work
- ◕ Leonard Papenmeier did most of the work
- ● Leonard Papenmeier did almost all of the work

Co-authors are abbreviated as follows:

Table 1.2: Author Acronyms

Acronym	Name
LP	Leonard Papenmeier
LN	Luigi Nardi
MP	Matthias Poloczek
SB	Stephen Becker
NC	Nuojin Cheng
EH	Erik Hellsten
CH	Carl Hvarfner

Paper I: Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces

MP proposed the initial research idea. LP designed the method in collaboration with LN and MP. LP implemented the method, ran the empirical evaluation, and wrote the paper with help from LN and MP.

Paper II: High-dimensional Bayesian Optimization with Group Testing

CH proposed the main concept. LP and EH designed and implemented the method with input from CH. LP and EH primarily wrote the paper, with contributions from CH and LN.

Paper III: Bounce: Reliable High-Dimensional Bayesian Optimization for Combinatorial and Mixed Spaces

LP mainly designed the method in consultation with MP and LN. LP implemented the method and did the empirical evaluation. LP wrote large parts of the paper with feedback and suggestions from MP and LN.

Paper IV: Exploring Exploration in Bayesian Optimization

LP, NC, SB, and LN developed the concept. LP implemented large parts of the method and was mainly responsible for the empirical evaluation with contributions by NC. LP and NC wrote large parts of the paper, with contributions from SB and LN.

Paper v: A Unified Framework for Entropy Search and Expected Improvement in Bayesian Optimization

NC proposed the concept and implemented the initial version. LP improved the implementation and performed large parts of the empirical evaluation. NC and LP wrote the paper, with contributions from SB and LN.

Paper vi: Understanding High-Dimensional Bayesian Optimization

LP proposed the concept, which was refined in consultation with LN and MP. LP implemented the method, ran the empirical evaluation, and mainly wrote the paper with support from LN and MP.

Paper 1

Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces

Advances in Neural Information Processing Systems 35, NeurIPS 2022

Leonard Papenmeier
Lund University

Luigi Nardi
Lund University

Matthias Polozcek
Amazon

Abstract

Recent advances have extended the scope of Bayesian optimization (BO) to expensive-to-evaluate black-box functions with dozens of dimensions, aspiring to unlock impactful applications, for example, in the life sciences, neural architecture search, and robotics. However, a closer examination reveals that the state-of-the-art methods for high-dimensional Bayesian optimization (HDBO) suffer from degrading performance as the number of dimensions increases or even risk failure if certain unverifiable assumptions are not met. This paper proposes `BAxUS` that leverages a novel family of nested random subspaces to adapt the space it optimizes

over to the problem. This ensures high performance while removing the risk of failure, which we assert via theoretical guarantees. A comprehensive evaluation demonstrates that BAxUS achieves better results than the state-of-the-art methods for a broad set of applications.

1 Introduction

The optimization of expensive-to-evaluate black-box functions where no derivative information is available has found many applications, for example, in chemical engineering [31, 61, 28, 59, 11], materials science [69, 23, 52, 65, 30, 29, 32], aerospace engineering [43, 39], hyperparameter optimization [62, 38, 5, 33], neural architecture search [36, 58], vehicle design [34, 14], hardware design [49, 19], drug discovery [51], robotics [41, 13, 54, 12, 46], and the life sciences [66, 60, 18]. Here increasing the number of dimensions (or parameters) of the optimization problem usually allows for better solutions. For example, by exposing more process parameters of a chemical reaction, we obtain a more granular control of the process; for a design task, we may optimize a larger number of design decisions jointly; in robotics, we gain access to more sophisticated control policies.

A series of breakthroughs have recently pushed the envelope of high-dimensional Bayesian optimization and facilitated a wider adoption in science and engineering. The key challenge for further scaling is the so-called curse of dimensionality. The complexity of the task of finding an optimum grows exponentially with the number of dimensions [7, 22]. Recently, methods that rely on local 'trust regions' have gained popularity. They usually achieve good performance for problems with up to a couple of dozen input parameters. However, we observe that their performance degrades for higher-dimensional problems. This is not surprising, given that trust regions have a smaller volume but still the full dimensionality of the problem. Other state-of-the-art methods suppose the existence of a low-dimensional active subspace and enjoy great scalability if they find such a space. The caveat is that its existence is usually not known for practical applications. Moreover, the user needs to 'guess' a good upper bound on its dimensionality to enjoy a good sample efficiency.

In this work, we propose a theoretically-founded approach for high-dimensional Bayesian optimization, BAxUS (**B**ayesian optimization with **a**daptively e**x**panding s**u**bspace**s**), that reliably achieves a high performance on a comprehensive set of applications. BAxUS utilizes a family of nested embedded spaces to increase the dimensionality of the domain that it optimizes over as it collects more data. As a byproduct, BAxUS can leverage an active subspace, if it exists, without requiring the user to 'guess' its dimensionality. BAxUS is based on a novel random linear subspace

embedding that enables a more efficient optimization and has strong theoretical guarantees. We make the following contributions:

1. We develop BAxUS that reliably achieves excellent solutions on a broad set of high-dimensional tasks, outperforming the state-of-the-art.
2. We present a novel family of nested random embeddings that has the following properties: a) The BAxUS embedding provides a larger worst-case guarantee for containing a global optimum than the HeSBO embedding proposed by [50]. b) The BAxUS embedding is an optimal *sparse embedding*, as defined in Def. 1. c) Its probability of containing an optimum converges to the one of the HeSBO embedding as the input dimensionality $D \to \infty$.
3. We conduct a comprehensive evaluation on a representative collection of benchmarks that demonstrates that BAxUS outperforms the state-of-the-art methods.

The remainder of this paper is structured as follows. Section 2 states the problem and discusses related work. Section 3 presents the BAxUS algorithm and the corresponding embedding. Section 4 evaluates BAxUS on a variety of benchmarks. We give concluding remarks in Section 5.

2 Background

The task is to find a minimizer

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}),$$

where $\mathcal{X} = [-1, +1]^D$. The objective function $f : \mathcal{X} \to \mathbb{R}$ is an expensive-to-evaluate black-box function. Hence the number of function evaluations needed to find an optimizer is crucial. Evaluations may be subject to observational noise, i.e., $f(\boldsymbol{x}_i) + \varepsilon_i$, with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. This work focuses on *scalable high-dimensional Bayesian optimization*, where the *input dimensionality* D is in the hundreds, and the sampling budget may comprise a thousand or more function evaluations.

Linear Embeddings. A successful approach for HDBO is to assume the existence of an *active subspace* [17], i.e., there exist a space $\mathcal{Z} \subseteq \mathbb{R}^{d_e}$, with $d_e \leq D$ and a function $g : \mathcal{Z} \to \mathbb{R}$, such that for all $\boldsymbol{x}$: $g(T\boldsymbol{x}) = f(\boldsymbol{x})$ where $T \in \mathbb{R}^{d_e \times D}$ is

a projection matrix projecting $\boldsymbol{x}$ onto $\mathcal{Z}$ and d_e is the *effective dimensionality* of the problem. In practice, both d_e and $\mathcal{Z}$ are unknown.

`REMBO` (Random Embedding BO) [73] and `HeSBO` (Hashing-enhanced subspace BO) [50] try to capture this active subspace by a randomly chosen linear subspace. Therefore, they generate a random projection matrix $S^\top$ that maps from a d-dimensional subspace $\mathcal{Y} \in \mathbb{R}^d$ with $d \ll D$ (the *target space*) to $\mathcal{X}$. We call d the *target dimensionality*. For `REMBO`, each entry in $S^\top$ is normally distributed. `REMBO` uses a heuristic to determine a hyperrectangle in $\mathcal{Y}$ that it optimizes over. Note that the bounded domain may not contain a point that maps to an optimizer of f, a risk aggravated by distortions introduced by the projection. Binois [6], Binois et al. [8, 9] proposed ideas to mitigate the issue. `HeSBO`'s random projection assigns one target dimension and sign (± 1) to each input dimension. This embedding is inspired by the count-sketch algorithm [15] for estimating frequent items in data streams. The sparse projection matrix $S^\top$ is binary except for the signs, and each row has exactly one non-zero entry. Even though this embedding avoids `REMBO`'s distortions, as the authors proved, it has a lower probability of containing the optimum [40]. `Alebo` [40] uses a Mahalanobis kernel and imposes linear constraints on the acquisition function to avoid projecting outside of $\mathcal{X}$.

Non-linear embeddings. Several works use autoencoders to learn non-linear spaces for optimization, trading in sample efficiency. Tripp et al. [68] change the training objective of a variational autoencoder (VAE) [37] to make the target space more suitable for optimization. They give higher weight to better-performing points when training the VAE and show that this improves optimization. Moriconi et al. [47] incorporate the training of an autoencoder directly into the likelihood maximization of a Gaussian process (GP) surrogate. The computational cost is cubic in the number of samples and the input dimension. Lu et al. [42] and Maus et al. [45] use autoencoders to learn embeddings of highly structured input spaces such as kernels or molecules. Other approaches include partial least squares (PLS) [10] or sliced inverse regression [16].

High-Dimensional BO in the Input Space. A popular approach to make HDBO in the input space feasible is trust regions (TRs) [56, 53, 22, 75]. The `TuRBO` algorithm [22] optimizes over bounded TRs instead of the global space, adapting their side lengths and the center points during the optimization process. By restricting function evaluations to trust regions, `TuRBO` addressed the problem of over-exploration; see [22] for details. Note that the TRs have full input dimensionality, which may impact `TuRBO`'s ability to scale to very large dimensions. Nonetheless, `TuRBO` set a new state-of-the-art by scaling to dozens on input dimensions and thousands

of function evaluations. Wan et al. [70] extended the idea of TRs to categorical and mixed spaces by using the Hamming distance to define the TR boundaries. SAASBO [20] uses sparse priors on the GP length scales which seems particularly valuable if the active subspace is axis-aligned. Indeed, SAASBO can outperform TuRBO on certain benchmarks [20]. The cost of inference scales cubically with the number of function evaluations; thus, SAASBO is not expected to scale beyond small sampling budgets, which is confirmed by our experiments. Another line of research relies on the assumption that the input space has an additive structure [35, 24, 72, 48]. Additive GPs rely on computationally expensive sampling methods to learn a decomposition of the input variables, which limits the scalability of such methods to problems of moderate dimensionalities and sampling budgets [50, 22]. Wang et al. [71] combined the meta-level algorithm LA-MCTS with TuRBO to improve optimization performance by learning a hierarchical space partition.

In Sect. 4 we evaluate the performances of TuRBO, SAASBO, Alebo, and HeSBO. Moreover, we study the popular CMA-ES [26] and random search [4].

3 The BAxUS Algorithm

Wang et al. [73] showed that the REMBO embedding contains an optimum in the target space with probability one if $d \geq d_e$ and if there are no bounds on the target and input spaces, i.e., $\mathcal{Y} = \mathbb{R}^d$ and $\mathcal{X} = \mathbb{R}^D$. For $d < d_e$, it is in general impossible to represent an optimum in $\mathcal{X}$ for arbitrary f because S projects to a d-dimensional subspace in $\mathcal{X}$. We call the probability of a target space to contain the optimum the *success probability*. For $d \geq d_e$, there is a positive success probability that increases with d [40, 73]. The main problem is to set d sufficiently small to avoid the detrimental effects of the curse of dimensionality, while keeping it as large as necessary to achieve a high probability that $\mathcal{Y}$ contains an optimum.

In practice, the active subspace and its dimensionality are usually unknown. The performance of methods such as REMBO [73], HeSBO [50], and Alebo [40] depends on choosing d such that the success probability is high. Therefore, they implicitly rely on guessing the effective dimensionality d_e appropriately. We argue that choosing the target dimensionality is problematic in many practical applications. If chosen too small, the subspace cannot represent f sufficiently well. If it is chosen too large, the curse of dimensionality slows down the optimization.

The proposed algorithm, BAxUS, operates on target spaces of increasing dimensionality while preserving previous observations. Let d_{init} be the initial target dimensionality and m the total evaluation budget. BAxUS starts with a d_{init}-

Algorithm 1.1 `BAxUS`

Input: b: new bins per dimension split, D: input dimension, m_D: # evaluations by which the input dimension is reached.

Output: minimizer $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$.

1: $d_0 \leftarrow$ initial target dimensionality of the subspace given by Eq. (1.4).
2: Compute initial projection matrix $S^\top : \mathcal{Y} \to \mathcal{X}$ by `BAxUS` embedding for D and d_0.
3: Sample initial data $\mathcal{D} = \{(\boldsymbol{y}_1, f(S^\top \boldsymbol{y}_1)), \ldots\}$ and fit the GP surrogate.
4: $n \leftarrow 0$
5: **while** evaluation budget not exhausted **do**
6: $\quad L \leftarrow L_{\text{init}}$ ▷ Initialize the trust region
7: $\quad$ Calculate number of accepted "failures" as described in Sec. 3.4: $\tau^s_{\text{fail}} \leftarrow \max\left(1, \min\left(\left\lfloor \frac{m^s_i}{k} \right\rfloor, d_n\right)\right)$
8: $\quad$ **while** $L > L_{\min}$ **and** evaluation budget not exhausted **do**
9: $\quad\quad$ Find $\boldsymbol{y}$ by Thompson sampling in TR, evaluate $f(S^\top \boldsymbol{y})$, and add to $\mathcal{D}$.
10: $\quad\quad$ Re-fit the GP hyperparameters.
11: $\quad\quad$ Adjust trust region, see Section 3.2 for details.
12: $\quad$ **if** $d_n < D$ **then**
13: $\quad\quad$ $d_{n+1} \leftarrow \min(d_n \cdot (b+1), D)$.
14: $\quad\quad$ Increase $S^\top$ by Algorithm 1.2. ▷ See Appendix D
15: $\quad$ **else**
16: $\quad\quad$ Re-sample initial data and discard previous observations, $d_{n+1} \leftarrow d_n$.
17: $\quad$ $n \leftarrow n + 1$
18: **Return** $S^\top(\arg\min_{\boldsymbol{y}\in\mathcal{D}} f(S^\top \boldsymbol{y}))$ or $S^\top(\arg\min_{\boldsymbol{y}\in\mathcal{D}} \mathbb{E}_n[f(S^\top \boldsymbol{y})])$. ▷ Return the best observation in case of observations without noise, or the best point according to posterior mean in case of noisy observations.

dimensional embedding that is increased over time until, after $m_D \leq m$ evaluations, it roughly reaches the input dimensionality D. With this strategy, we can leverage the efficiency of BO in low-dimensional spaces while guaranteeing to find an optimum in the limit. Increasing the target dimensionality is enabled by a novel embedding, which lets us carry over observations from previous, lower-dimensional target spaces into more high-dimensional target spaces. We further use a TR-based approach based on Eriksson et al. [22] to carry out optimization for high target dimensions effectively. `BAxUS` uses a GP surrogate [55] to model the function in the target space. Algorithm 3.1 gives the pseudocode for `BAxUS`. In Appendix B, we prove global convergence for `BAxUS`. We will now present the different components in detail.

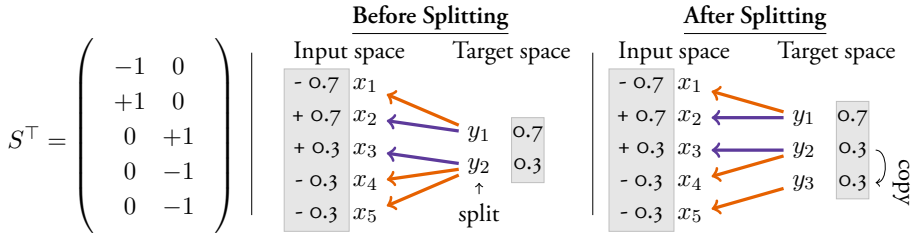


Figure 1.1: Observations are kept when increasing the target dimensionality. We give an example of the splitting method for $D = 5$ and $d = 2$. The first target dimension y_1 has two contributing input dimensions, x_1 and x_2. y_2 has three contributing input dimensions, x_3, x_4, and x_5. By $S^\top$, a point $(0.7, 0.3)^\top$ in the target space is mapped to $(-0.7, +0.7, +0.3, -0.3, -0.3)^\top$ in the input space. Assigning the fifth input dimension to a new target dimension and copying the function values from the second target dimension does not change the observation in the input space. The new $S^\top$ is not shown but has one additional column with -1 in the last row, and the last row of the second column is set to 0.

3.1 The sparse BAxUS subspace embedding

The BAxUS embedding uses a sparse projection matrix to map from $\mathcal{Y}$ to $\mathcal{X}$. The number of non-zero entries in this matrix is equal to the input dimensionality D. Another embedding with this property is the HeSBO embedding [50]. Given the D and a target dimensionality d, HeSBO samples a target dimension in $\{1, \dots, d\}$ and a sign $\{\pm 1\}$ for each input dimension uniformly at random. Conversely, each target dimension has a set of signed contributing input dimensions. We call the set of contributing input dimensions to a target dimension a *bin*. These relations implicitly define the embedding matrix $S \in \{0, \pm 1\}^{d \times D}$, where each column has exactly one non-zero entry [15]. In the HeSBO embedding, the number of contributing input dimensions varies between 0 and D.

The interpretation of contributing input dimensions allows for an intuitive way to refine the embedding, which is shown in Figure 1.1. We update the embedding matrix such that contributing input dimensions of the target dimension are re-assigned to the current bin and b new bins. We then say that we *split* the corresponding target dimension. Importantly, this type of embedding allows for retaining observations (see Figure 1.1). Assume for example, that y_i is the dimension to be split. The contributing input dimensions are re-assigned to y_i and three new target dimensions y_j, y_k, and y_l (here, $b = 3$); the observations can be retained by copying the value of the coordinate y_i to the coordinates y_j, y_k, and y_l. Thus, the observations are contained in the old and in the new target space. Algorithm 1.2 describes the procedure in detail.

In the BAxUS embedding, we force each bin of a target dimension to have roughly

the same number of contributing input dimensions: the bin sizes differ by at most one. First, we create a random permutation of the input dimensions $1, \ldots, D$. The list of input dimensions is split into $\min(d, D)$ individual bins. If d does not divide D, not all bins can have the same size. We split the permutation of input dimensions such that the i-th bin has size $\lceil D/d \rceil$, if $i + d\lfloor D/d \rfloor \leq D$, and $\lfloor D/d \rfloor$ otherwise. The first bins have one additional element with this construction if d does not divide D. We further randomly assign a sign to each input dimension. The sign of the input dimensions and their assignment to target dimensions then implicitly define $S^\top$ (see Figure 1.1). We now show that the `BAxUS` embedding has a strictly larger worst-case success probability than the `HeSBO` embedding. We establish the following two definitions.

Definition 1 (Sparse embedding matrix). A matrix $S \in \{0, \pm 1\}^{d \times D}$ is a *sparse embedding matrix* if and only if each column in S has exactly one non-zero entry [74].

We formalize the event of "recovering an optimum" [40] as follows.

Definition 2 (Success of a sparse embedding). A success of a random sparse embedding is the event $Y^* =$ "All d_e active input dimensions are mapped to distinct target dimensions."

It is important to note that the definition of a success is sufficient but not necessary for the embedding to contain a global optimum. For example, if the origin is a global optimum, then both embeddings contain it with probability one. In that sense, the above definition provides a *worst-case guarantee*. We refer to Definition 4 in Appendix A for a formal definition of a sparse function. In Theorem 1, we give the worst-case success probability of the `BAxUS` embedding. All proofs have been deferred to Appendix A. Note that other than in the count-sketch algorithm [15], our hashing function is not pairwise independent. However, this does not affect our theoretical analysis.

Theorem 1 (Worst-case success probability of the `BAxUS` embedding). *Let D be the input dimensionality and $d \geq d_e$ the dimensionality of the embedding. Let $\beta_{small} = \lfloor \frac{D}{d} \rfloor$ and $\beta_{large} = \lceil \frac{D}{d} \rceil$ be the small and large bin sizes. Then the probability of Y^* (see Definition 2) for the `BAxUS` embedding is*

$$p_B(Y^*; D, d, d_e) = \frac{\sum_{i=0}^{d_e} \binom{d(1+\beta_{small})-D}{i} \binom{D-d\beta_{small}}{d_e-i} \beta_{small}^i \beta_{large}^{d_e-i}}{\binom{D}{d_e}}. \tag{1.1}$$

Figure 1.2 shows the worst-case success probabilities of the `BAxUS` and `HeSBO` embeddings for three different settings of D. The worst-case success probability of

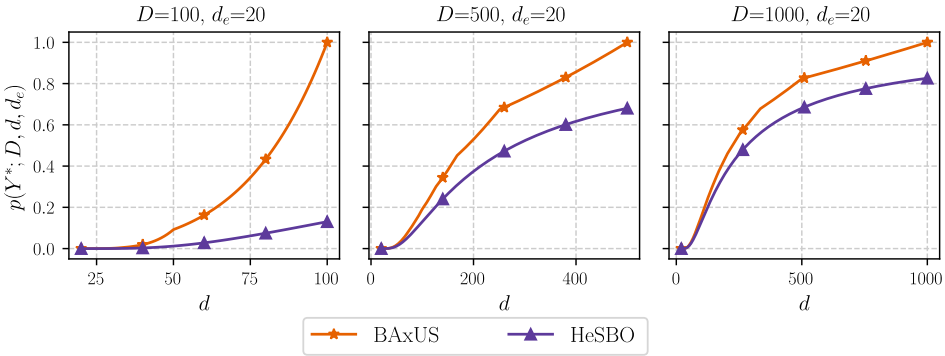


Figure 1.2: The worst-case guarantees for the success probabilities $p(Y^*; D, d, d_e)$ of the `BAxUS` and `HeSBO` embeddings for different input dimensionalities, D=100 (left), D=500 (center), D=1000 (right), as a function of the target dimensionality d. The effective dimensionality is d_e=20. The `BAxUS` embedding has a higher worst-case success probability than the `HeSBO` embedding. The improvement is large for input dimensionalities in the low hundreds and still substantial for 1000D tasks. In accordance with the theoretical analysis, the difference vanishes as the input dimension D grows.

the `HeSBO` embedding is given by $p_H(Y^*; d, d_e) = d!/((d - d_e)!d^{d_e})$ (see Letham et al. [40] and Appendix A.3). It is independent of D but is shown for varying d-ranges on the x-axis, therefore the probabilities seem to change between the different subplots. The `BAxUS` embedding ensures that the worst-case success probability is one for $d = D$. Discontinuities in the curve of the `BAxUS` embedding occur due to the unequal bin sizes in the `BAxUS` embedding's worst-case success probability. The difference between the two embeddings in Figure 1.2 is particularly striking when d_e is high: for example, for $d_e = 20$ `HeSBO` requires $d = 1000$ to reach a worst-case success probability of approximately 0.8, whereas the `BAxUS` embedding has a success probability of 1 as soon as $d = D$. For finite D, `HeSBO`'s worst-case success probability is smaller than `BAxUS`' success probability.

Corollary 1. *For $D \to \infty$, the worst-case success probability of the `BAxUS` embedding is*

$$\lim_{D \to \infty} p_B(Y^*; D, d, d_e) = \frac{d!}{(d - d_e)!d^{d_e}}, \tag{1.2}$$

and hence matches `HeSBO`'s worst-case success probability $p_H(Y^*; d, d_e)$.

We show that the `BAxUS` embedding is optimal among sparse embeddings.

Corollary 2. *With the same input, target, and effective dimensionalities (D, d, and d_e), no sparse embedding has a higher worst-case success probability than the `BAxUS` embedding.*

3.2 Trust-region approach

Similar to TuRBO [22], BAxUS operates in trust regions (TRs). TRs are hyper-rectangles in the input space. Their shape is determined by their *base side length* L and the GP length scales. The side length for each dimension is proportional to the corresponding length scale of the GP kernel fitted to the data. The idea of this construction is that length scales indicate how quickly the function changes along the associated dimension. Thus, the TR is rescaled accordingly. The volume of a TR is shrunk when TuRBO fails τ_{fail} consecutive times to make progress, i.e., to find a better solution. If the algorithm consecutively makes progress for $\tau_{\text{success}} = 3$ times, it expands the TR. It restarts when the base side length L of the current TR falls below a threshold $L_{\min} := 2^{-7}$. In that case, it discards all observations for the TR and initializes a new TR on new samples. TRs enable TuRBO to focus on regions of the space close to the incumbent, i.e., the current best solution found by the algorithm. To choose the next evaluation point, TuRBO uses Thompson sampling (TS) [67], i.e., it draws a realization of the GP on a set of candidate locations in the TR and then selects a point of minimum sampled value.

TRs are an essential component of BAxUS because the target dimensionality usually grows exponentially during a run of the algorithm. We use the same hyperparameter settings as TuRBO [22] with the following modifications. First, we change the criterion for when to restart a TR. Instead of restarting a TR when it becomes too small, we increase the target dimensionality by splitting each target dimension into several new bins unless BAxUS has already reached the input dimensionality. In this case, we reset the TR base side length to the initial value and re-initialize the algorithm with a new random set of initial observations. By resetting the base side length, we also avoid convergence to a particular local minimum as the TR covers large regions of the space again. TuRBO solves this problem by allowing for multiple parallel TRs. Secondly, we change the number of accepted "failures" τ_{fail}, such that BAxUS can roughly reach the input dimensionality in a fixed number of evaluations as described in Section 3.4.

3.3 Splitting Strategy

Starting in a low-dimensional embedded space, BAxUS successively grows the target dimensionality to increase the probability of containing an optimum. By Corollary 2, it is optimal to keep the number of contributing input dimensions in the target bins as equal as possible. At each splitting point, the target dimensionality grows exponentially. The number of splits required to reach some input dimensionality D is logarithmic in D. BAxUS uses a larger evaluation budget in each split. Suppose

that the algorithm starts in target dimensionality d_{init}. Then, after k splits, the target dimensionality is $d_k = d_{\text{init}}(b+1)^k$.

3.4 Controlling the Number of Accepted Failures

BAxUS needs to be able to reach high target dimensionalities to find a global optimum. As described in Section 3.2, BAxUS increases the target dimensionality when the TR base side length falls below the minimum threshold. For this to happen, the TR base length needs to be halved at least $k = \left\lfloor \log_{\frac{1}{2}} \frac{L_{\text{min}}}{L_{\text{init}}} \right\rfloor$ times. Halving occurs if BAxUS consecutively fails τ_{fail} times in finding a better function value. If, similarly to TuRBO, we set the number of accepted "failures" τ_{fail} to the current target dimensionality of the TR, we get the lower bound $k \cdot \tau_{\text{fail}}$ on the number of function evaluations spent in that target dimensionality. This bound does not scale with the input dimensionality D of the problem, i.e., the maximum target dimensionality is independent of D for a fixed evaluation budget.

To enable BAxUS to reach any desired target dimensionality for the fixed evaluation budget, we scale down τ_{fail} dependent on D, i.e., we adjust the lower bound. We choose to make it dependent on D as we are guaranteed that the target space contains all global optima if the final target space corresponds to the input space $\mathcal{X}$ (see Appendix B). In contrast to imposing a hard limit on the number of function evaluations in a target dimensionality, scaling down the number of accepted "failures" has the advantage that we do not restrain BAxUS in cases where it finds better function values. The idea is to choose τ_{fail} dependent on the current target dimensionality d_i *and* such that BAxUS can reach any desired target dimensionality.

We calculate the number of splits n required to reach D by

$$D \approx d_{\text{init}} \cdot (b+1)^n \Rightarrow n = \left\lfloor \log_{b+1} \frac{D}{d_{\text{init}}} \right\rceil, \tag{1.3}$$

with $\lfloor \cdot \rceil$ indicating rounding to the nearest integer. The minimum evaluation budget for a split is then found by multiplying m_D with the "weight" of each target dimensionality. We assign each split i a *split budget* m_i^s that is proportional to d_i, such that $\sum_{i=0}^{n} m_i^s = m_D$, where m_D is the budgeted number of function evaluations until D would be reached under the above assumptions:

$$m_i^s = \left\lfloor m_D \frac{d_i}{\sum_{k=0}^{n} d_k} \right\rceil = \left\lfloor \frac{b \cdot m_D \cdot d_i}{d_{\text{init}}((b+1)^{n+1} - 1)} \right\rceil.$$

Finally, we set the number τ_{fail}^i of accepted "failures" for the i-th target dimensionality d_i such that (1) it adheres to its *split budget* in the event that it never

obtains a better function value, (2) it is not larger than if we would use TuRBO's choice d_i, and (3) it is at least 1:

$$\tau_{\text{fail}}^i = \max\left(1, \min\left(\left\lfloor \frac{m_i^s}{k} \right\rfloor, d_i\right)\right).$$

Setting the Initial Target Dimension. Due to the rounding in Eq. (1.3) and the exponential growth of the target dimensionality, the final target dimensionality $d_{\text{init}} \cdot (b+1)^n$ might differ considerably from the input dimensionality D. This is undesirable as we might not reach D before depleting the evaluation budget, or we might overestimate the evaluation budget for the final target dimensionality d_n. Therefore, we set the initial target dimensionality such that the final target dimensionality is as close to D as possible:

$$d_{\text{init}} = \underset{i \in \{1,\dots,b\}}{\arg\min} \left| i \cdot (b+1)^n - D \right|, \tag{1.4}$$

where n is given by Eq. (1.3). We point out that $1 \leq d_{\text{init}} \leq b$. An alternative to adjusting d_{init} would be to fix the initial d_0 and adjust the growth factor b.

4 Experimental Evaluation

In this section, we evaluate the performance of BAxUS on a 388D hyperparameter optimization task, a 124D design problem, and a collection of tasks that exhibit an active subspace. The BAxUS code is available at `https://github.com/LeoIV/BAxUS`.

The Experimental Setup. We benchmark against TuRBO [22] with one and five trust regions, SAASBO [20], Alebo [40], random search [4], CMA-ES [26], and HeSBO [50], using the implementations provided by the respective authors with their settings, unless stated otherwise. For CMA-ES, we use the PyCma [27] implementation. For HeSBO and Alebo, we use the Ax implementation [1]. To show the effect of different choices of d, we run HeSBO and Alebo with $d = 10$ and $d = 20$. We observed that Alebo and SAASBO are constrained by their high runtime and memory consumption. The available hardware allowed up to 100 function evaluations for SAASBO and 500 function evaluations for Alebo for each individual run. Larger sampling budgets or higher target dimensions for Alebo resulted in out-of-memory errors. We point out that limited scalability was expected for these two methods, whereas the other methods scaled to considerably larger budgets, as required for scalable BO. We initialize each optimizer, including BAxUS, with ten

initial samples and BAxUS with $b = 3$ and $m_D = 1000$ and run 20 repeated trials. Plots show the mean performance with one standard error.

The Benchmarks. We evaluate the selected algorithms on six benchmarks that differ considerably in their characteristics. Following Wang et al. [73], we augment the Branin2 and Hartmann6 functions with additional dummy dimensions that have no influence on the function value. We use the 388D Svm benchmark and the 124D soft-constraint version of the Mopta08 benchmark proposed in [20]. We set a budget of 1000 evaluations for Mopta08, Branin2, and Hartmann6 and of 2000 evaluations for the other benchmarks. Moreover, we stopped for Branin2 and Hartmann6 when the simple regret dropped below .001. We show results on additional noise-free benchmarks in Appendices C.2 and C.3. We also tested the algorithms on the 300D Lasso-High and the 1000D Lasso-Hard benchmarks from `LassoBench` [60]. These benchmarks have an effective dimensionality of 5% of the input dimensionality, i.e., the Lasso-High and Lasso-Hard benchmarks have 15 and 50 effective dimensions, respectively. To study the robustness to observational noise, we also tested on noisy variants of Lasso-Hard and Lasso-High.

4.1 Experimental Results

We begin with the six noise-free benchmarks. Fig. 1.3 summarizes the performances. On Mopta08, a 124D vehicle design problem, SAASBO initially makes slightly faster progress than BAxUS. We suspect that this benchmark has high effective dimensionality, such that BAxUS first needs to adapt the target dimensionality to make further progress. On the 388D Svm benchmark, BAxUS adapts to the appropriate target dimensionality where it can reach good function values faster than TuRBO and CMA-ES. For this benchmark, Eriksson and Jankowiak [20] reported that SAASBO learned three active dimensions. Yet, the fact that Alebo and HeSBO seem to stagnate after a few hundred evaluations, while BAxUS, TuRBO, and CMA-ES find better solutions, indicates that optimizing more of the 385 kernel length scales of the Svm benchmark allows for better solutions. On the 500D Hartmann6, SAASBO performs best, closely followed by BAxUS. Alebo and HeSBO are competitive initially but converge to suboptimal solutions. HeSBO, BAxUS, SAASBO, Alebo all find excellent solutions on the Branin benchmark, with the latter algorithms converging faster.

Next, we examine the performances on the 1000D Lasso-Hard and the 300D Lasso-High that exhibit active subspaces, here without observational noise. BAxUS achieves considerably better solutions than all state-of-the-art methods. We also note that TuRBO and CMA-ES perform better than SAASBO and Alebo. While one may

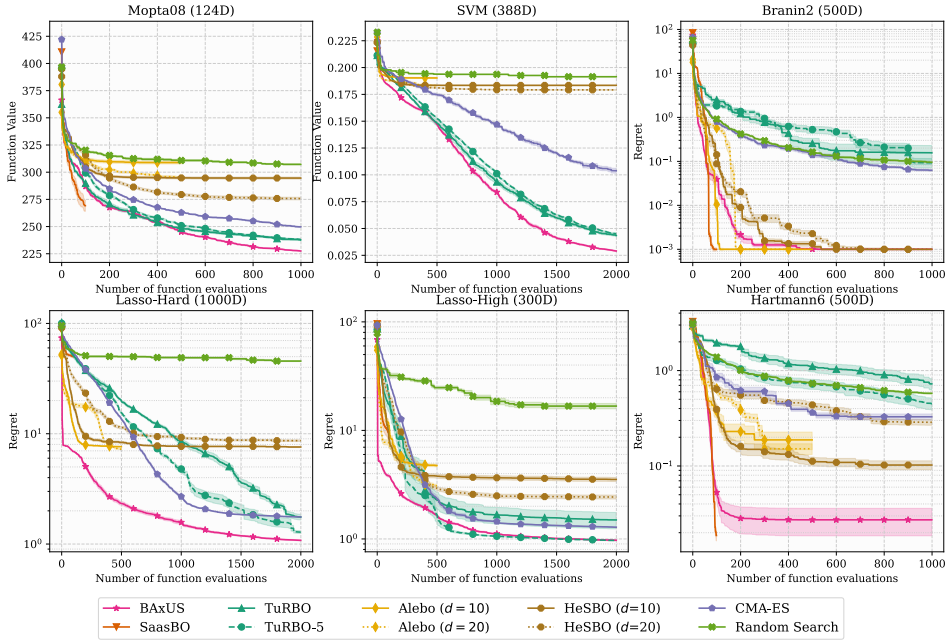


Figure 1.3: Top row: 124D Mopta08 (l): `BAxUS` obtains the best solutions, followed by `TuRBO` and `CMA-ES`. 388D Svm (c): `BAxUS` outperforms the other methods from the start. 500D Branin (r): `SAASBO`, `BAxUS`, `Alebo`, and `HeSBO` find an optimum; `SAASBO` and `Alebo` converge fastest. Bottom row: 100D Lasso-Hard (l) and 300D Lasso-High (c): `BAxUS` outperforms the baselines. `SAASBO`, `Alebo`, and `HeSBO` struggle. 500D Hartmann6 (r): `SAASBO` performs best, closely followed by `BAxUS`. The other methods show only slow progress or stagnate.

expect `BAxUS` to outperform `TuRBO` and `CMA-ES` on these tasks with high input dimensions, it is surprising that `SAASBO`, `HeSBO`, and `Alebo` are not able to benefit from the present active subspace. Here `BAxUS`'s strategy to adaptively expand the nested subspace is superior. Another crucial observation is that performances of `BAxUS` vary only slightly across runs. Thus, `BAxUS` is robust despite the stochastic construction of the embedding. Across the broad collection of benchmarks, `BAxUS` is the only method to consistently achieve high performance.

Noisy Benchmarks. We evaluate the algorithms also for tasks with observational noise. Fig. 1.4 summarizes the results. We observe that `BAxUS` achieves considerably better solutions for any number of observations than the competitors. Moreover, we note that the performances of `SAASBO`, `CMA-ES`, and `HeSBO` ($d = 20$) degrade considerably on the Lasso-High task compared to the noise-free formulation of the task studied above. `BAxUS`' performance is equally strong as for the noise-free case and keeps making progress after 1000 observations.

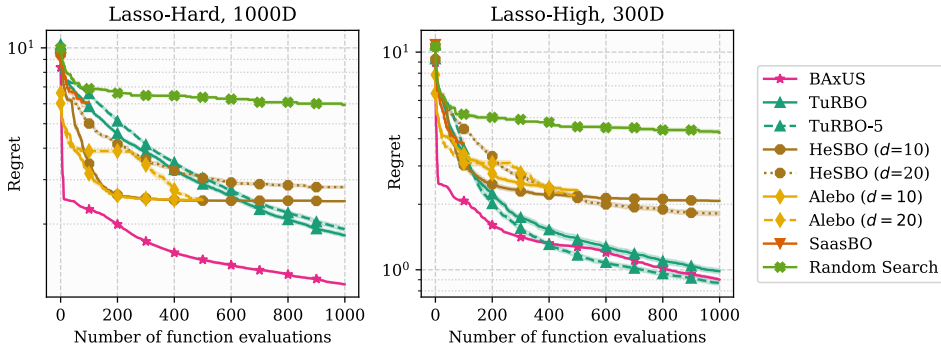


Figure 1.4: `BAxUS` outperforms the SOTA and in particular proves to be robust to observational noise on the 1000D Lasso-Hard (l) and the 300D Lasso-High (r). Note that `CMA-ES` performs considerably worse than on the noise-free versions of the benchmarks.

`BAxUS` Embedding Ablation Study. To investigate whether the proposed family of nested random subspaces contributes to the superior performance of `BAxUS`, we replaced the new embedding with a similar family of nested `HeSBO` embeddings. The results show that the proposed embedding provides a significant performance gain. Due to space constraints, the results were moved to Appendix C.1.

5 Discussion

High-dimensional Bayesian optimization is aspiring to unlock impactful applications broadly in science and industry. However, state-of-the-art methods suffer from limited scalability or, in some cases, require practitioners to 'guess' certain hyperparameters that critically impact the performance. This paper proposes `BAxUS` that works out-of-the-box and achieves considerably better performance for high-dimensional problems, as the comprehensive evaluation shows. A key idea is to scale up the dimensionality of the target subspace that the algorithm optimizes over. We apply a simple strategy that we find to work well across the board. However, we expect substantial headroom in tailoring this strategy to specific applications, either using domain expertise or a more sophisticated data-driven approach that, for example, learns a suitable target space. Moreover, future work will explore extending `BAxUS` to structured domains, particularly the combinatorial spaces common in materials sciences and drug discovery.

Societal impact. Bayesian optimization has recently become a popular tool for tasks in drug discovery [51], chemical engineering [31, 61, 28, 59, 11], materials science [69, 23, 52, 65, 30, 29], aerospace engineering [43, 2, 39], robotics [41, 13, 54, 12, 46], and

many more. This speaks to the progress that Bayesian optimization has made in becoming a robust and reliable 'off-the-shelf solver.' However, this promise is not yet fulfilled for the newer field of high-dimensional Bayesian optimization that allows optimization over hundreds of 'tunable levers.' The abovementioned applications benefit from incorporating more such levers in the optimization: it allows for more detailed modeling of an aerospace design or a more granular control of a chemical reaction, to give some examples. The evaluation shows that the performance of state-of-the-art methods degenerates drastically for such high dimensions if the application does not meet specific requirements. Adding insult to the injury, such requirements as the dimensionality of an active subspace cannot be determined beforehand.

The proposed algorithm achieves a robust performance over a broad collection of tasks and thus will become a 'goto' optimizer for practitioners in other fields. Therefore, we released the `BAxUS` code.

Acknowledgments

Luigi Nardi was supported in part by affiliate members and other supporters of the Stanford DAWN project — Ant Financial, Facebook, Google, Intel, Microsoft, NEC, SAP, Teradata, and VMware. Leonard Papenmeier and Luigi Nardi were partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Luigi Nardi was partially supported by the Wallenberg Launch Pad (WALP) grant Dnr 2021.0348. The computations were also enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at LUNARC, partially funded by the Swedish Research Council through grant agreement no. 2018-05973. We would like to thank Erik Hellsten from Lund University and Eddy de Weerd from the University of Twente for valuable discussions.

A Theoretical Foundation for the BAxUS Embedding

For convenience, we re-state Definition 1 and Definition 2 from Section 3.

Definition 1 (**Sparse embedding matrix**). A matrix $S \in \{0, \pm 1\}^{d \times D}$ is a *sparse embedding matrix* if and only if each column in S has exactly one non-zero entry [74].

Definition 2 (**Success of a sparse embedding**). A success of a random sparse embedding is the event $Y^* =$ "All d_e active input dimensions are mapped to distinct target dimensions."

We introduce the following two definitions.

Definition 3 (**Optima-preserving sparse embedding**)**.** A sparse embedding matrix is *optima-preserving* if each target dimension (i.e., each column in S) contains at most one active input dimension.

Definition 4 (**Sparse function / function with an active subspace**)**.** Let $\mathcal{X} = [-1, 1]^D$. A function $f : \mathcal{X} \to \mathbb{R}$ has an *active subspace* (or *effective subspace* [73]), if there exist a subspace (i.e., a space $\mathcal{Z} \subseteq \mathbb{R}^{d_e}$, with $d_e \leq D$ where $d_e \in \mathbb{N}_{++}$ is the effective dimensionality and $\mathbb{N}_{++} = \mathbb{N} \backslash \{0\}$) and a projection matrix $S^\top \in \mathbb{R}^{D \times d_e}$, such that for any $\boldsymbol{x} \in \mathcal{X}$ there exists a $\boldsymbol{z} \in \mathcal{Z}$ so that $f(\boldsymbol{x}) = f(S^\top \boldsymbol{z})$ and d_e is the smallest integer with this property. The function is called *sparse* if it has an active subspace and $S^\top$ is a sparse embedding matrix and $\mathcal{Z} = [-1, 1]^{d_e}$.

A.1 Proof of Theorem 1

We prove the worst-case success probability for the BAxUS embedding.

Theorem 1 (**Worst-case success probability of the BAxUS embedding**). Let D be the input dimensionality and $d \geq d_e$ the dimensionality of the embedding. Let $\beta_{\text{small}} = \left\lfloor \frac{D}{d} \right\rfloor$ and $\beta_{\text{large}} = \left\lceil \frac{D}{d} \right\rceil$ be the small and large bin sizes. Then the probability of Y^* (see Definition 2) for the BAxUS embedding is

$$p_B(Y^*; D, d, d_e) = \frac{\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}})-D}{i} \binom{D - d\beta_{\text{small}}}{d_e - i} \beta_{\text{small}}^i \beta_{\text{large}}^{d_e - i}}{\binom{D}{d_e}}. \tag{1.5}$$

Proof. The assignment of input dimensions to target dimensions and the signs of the input dimensions fully define the BAxUS embedding. Note that the signs do not affect $p_B(Y^*; D, d, d_e)$ because they only correspond to "flipping" the input dimension in the target space, and our construction ensures that the value ranges are symmetric to the origin.

An assignment is optima-preserving if and only if it is possible to find a point in $\mathcal{Y}$ that maps to an optimum in $\mathcal{X}$ for any f. The "only if" is true because f is assumed to be sparse with an active subspace with d_e active dimensions. This means that the optima in $\mathcal{X}$ only change their function values along the d_e active dimensions. Suppose it is possible to find a point in $\mathcal{Y}$ that maps to an arbitrary optimum in $\mathcal{X}$. In that case, the assignment is optima-preserving because it can individually adjust all the d_e active dimensions in $\mathcal{X}$. However, this generally requires each active input dimension to be mapped to a distinct target dimension (note that we require being able to represent the optimum for *any* f). Otherwise, there would be at least two active input dimensions that cannot be changed independently. Therefore, the probability of Y^* equals the probability of an optima-preserving assignment.

As all assignments are equally likely under the construction, the probability of an assignment being optima-preserving is equal to the number of possible optima-preserving assignments divided by the total number of assignments. There are $\binom{D}{d_e}$ ways of distributing the d_e active dimensions across the D positions, giving the denominator in Eq. (1.5).

Let us first assume that $\beta_{\text{small}} = \beta_{\text{large}} = \beta$, i.e., all target dimensions have the same number of input dimensions and d divides D. We refer to this case as the *balanced case*. There are $\binom{d}{d_e}$ ways of distributing the d_e active dimensions across the d different target dimensions. Given one active dimension, there are β ways in which this dimension can map to the target dimension. Therefore, for the balanced case, the worst-case success probability is given by

$$p_B(Y^*; D, d, d_e) = \frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}}. \tag{1.6}$$

Next, we generalize Eq. (1.6) for cases where d does not divide D. We refer to this case as the *near-balanced case*. In that case, there are two bin sizes: β_{small} and β_{large} with $\beta_{\text{large}} = \beta_{\text{small}} + 1$. There are $d\beta_{\text{large}} - D$ small bins (i.e., bins with bin size β_{small}) and $D - d\beta_{\text{small}}$ large bins: $D - d\beta_{\text{small}}$ gives the number of input dimensions that would not be covered if all bins were small. Since β_{small} and β_{large} differ by 1, this also gives the number of bins that have to be large. Conversely, if we only had large bins, we would cover $d\beta_{\text{large}} - D$ too many input dimensions. Therefore, we need $D - d\beta_{\text{small}}$ large and $d\beta_{\text{large}} - D$ small bins.

We consider all ways of distributing the d_e active dimensions across the the $d\beta_{\text{large}} - D$ small and $D - d\beta_{\text{small}}$ large bins so that there is at most one active dimension in each bin. Recall that this number gives the numerator in Eq. (1.5). For a conflict-free assignment, if i active dimensions are mapped to small bins, then $d_e - i$ active dimensions must be assigned to large bins. There are $\binom{d(1+\beta_{\text{small}})-D}{i}\binom{D-d\beta_{\text{small}}}{d_e-i}$ such

assignments. Here we use that $1 + \beta_{\text{small}} = \beta_{\text{large}}$ holds for the near-balanced case. Recall that each small bin has β_{small} locations and that each large bin has β_{large} locations that an active dimension can be assigned to. Because $0 \leq i \leq d_e$ by construction, the number of assignments that result in an optima-preserving embedding is

$$\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}}) - D}{i} \binom{D - d\beta_{\text{small}}}{d_e - i} \beta_{\text{small}}^{i} \beta_{\text{large}}^{d_e - i}.$$

Note that we leverage the facts $\binom{0}{0} = 1$, $\binom{0}{x} = 0$ for all $x \geq 1$, $\binom{y}{x} = 0$ if $x > y \geq 0$, and $\binom{x}{0} = 1$ for all x, thus the sum is well defined. Recall that we already showed that the denominator is $\binom{D}{d_e}$. Therefore, Eq. (1.5) gives the success probability in the near-balanced case.

It is easy to see that Eq. (1.5) is equivalent to the near-balanced formulation in Eq. (1.6) when d divides D. When d divides D, $\beta_{\text{small}} = \beta_{\text{large}} = \beta$, $d(1 + \beta_{\text{small}}) - D = d$, and $D - d\beta_{\text{small}} = 0$. Therefore, the worst-case success probability for the near-balanced case is given by

$$\begin{aligned} p_B(Y^*; D, d, d_e) &= \frac{\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}})-D}{i} \binom{D-d\beta_{\text{small}}}{d_e-i} \beta_{\text{small}}^{i} \beta_{\text{large}}^{d_e-i}}{\binom{D}{d_e}} \\ &\overset{\beta_{\text{small}}=\beta_{\text{large}}}{=} \frac{\sum_{i=0}^{d_e} \binom{d}{i} \binom{0}{d_e-i} \beta^i \beta^{d_e-i}}{\binom{D}{d_e}} \\ &= \frac{\beta^{d_e} \sum_{i=0}^{d_e} \binom{d}{i} \binom{0}{d_e-i}}{\binom{D}{d_e}} = \frac{\beta^{d_e} \binom{d}{d_e}}{\binom{D}{d_e}} \end{aligned}$$

where the last equality is true because the sum is zero unless $i = d_e$.

□

A.2 Proof of Corollary 2

We prove the optimality of the `BAxUS` embedding in terms of the worst-case success probability.

Corollary 2. With the same input, target, and effective dimensionalities (D, d, and d_e), no sparse embedding has a higher worst-case success probability than the `BAxUS` embedding.

Proof. By Definition 1, an embedding matrix $S \in \{0, \pm 1\}^{D \times d}$ is sparse if each row in S has exactly one non-zero entry. Such an embedding can always be interpreted as

disjoint sets of signed input dimensions assigned to different target dimensions: For the n-th input dimension, find the column with the non-zero. The respective column gives the target dimension; the entry in the matrix itself gives the sign. Conversely, each target dimension has a set of contributing input dimensions, and we call the set of input dimensions mapping to a target dimension a "bin". The sign of the input dimensions does not influence the success probability as it does not influence the ability of an embedding to contain the optimum.

We will prove that the `BAxUS` embedding is optimal, i.e., every other sparse embedding has a worst-case success probability that is lower or equal. We start by giving the worst-case success probability for arbitrary bin sizes.

Let β_n be the bin size of the n-th bin. By Definition 2, a success is guaranteed if each bin contains at most one active input dimension. Therefore, the worst-case success probability for arbitrary bin sizes has to consider the number of cases where each bin contains at most one active input dimension and the number of bins containing one active input dimension is equal to the number of active input dimensions d_e. In a bin of size β_i, the active input dimension can lie in β_i different locations. Bins not containing an active dimension do not contribute to the worst-case success probability.

We suppose w.l.o.g. that $D \geq d$. Thus, every target dimension has at least one input dimension. For each n from 1 to d, let the value i_n indicate whether the n-th bin (or target dimension) contains an active dimension ($i_n = 1$) or not ($i_n = 0$). The indicator variable $\mathbb{1}_{(\sum_{n=1}^{d} i_n)=d_e}$ ensures that only cases where exactly d_e bins contain an active input dimension are counted. Note that $\sum_{i_1=0}^{1}\sum_{i_2=0}^{1}\cdots\sum_{i_d=0}^{1}\mathbb{1}_{(\sum_{n=1}^{d} i_n)=d_e} = \binom{d}{d_e}$. For each case where the d_e active dimensions are assigned to d_e out of d disjoint bins, the term $\prod_{n=1}^{d}\beta_n^{i_n}$ accounts for the locations in which the active dimension can lie in the n-th bin. Other cases do not contribute to the worst-case success probability. The exponent ensures that only bins containing an active dimension contribute to the denominator.

Then the worst-case success probability for arbitrary bin sizes is given by

$$p_{\text{general}}(Y^*; D, d, d_e) = \frac{\overbrace{\sum_{i_1=0}^{1}\sum_{i_2=0}^{1}\cdots\sum_{i_d=0}^{1}\mathbb{1}_{(\sum_{n=1}^{d} i_n)=d_e}}^{=\binom{d}{d_e}}\prod_{n=1}^{d}\beta_n^{i_n}}{\binom{D}{d_e}}, \tag{1.7}$$

with $\beta_n > 0$, $\sum_{n=1}^{d}\beta_n = D$, and $d \geq d_e$:

As in Theorem 1, the denominator of Eq. (1.7) gives all ways of assigning d_e active dimensions to D input dimensions.

We now prove that any sparse embedding has a worst-case success probability that is less or equal to the worst-case success probability of the `BAxUS` embedding.

Let β_{small}, β_{large}, and $p_B(Y^*; D, d, d_e)$ as in Theorem 1. Then,

$$p_{\text{general}}(Y^*; D, d, d_e) \leq p_B(Y^*; D, d, d_e) \tag{1.8}$$

$$= \frac{\overbrace{\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}})-D}{i} \binom{D-d\beta_{\text{small}}}{d_e-i}}^{=\binom{d}{d_e}} \beta_{\text{small}}^{i} \beta_{\text{large}}^{d_e-i}}{\binom{D}{d_e}}. \tag{1.9}$$

We refer the reader to the proof of Theorem 1 for an explanation of the binomial coefficients. The fact that $\sum_{i=0}^{d_e} \binom{d(1+\beta_{\text{small}})-D}{i} \binom{D-d\beta_{\text{small}}}{d_e-i} = \binom{d}{d_e}$ can be seen by noting that $(d(1+\beta_{\text{small}})-D) + (D - d\beta_{\text{small}}) = d$ and applying Vandermonde's convolution [25].

We will now prove that if d divides D, then the product in the numerator of Eq. (1.7) is maximized if all the factors are the same, i.e., $\beta = \frac{D}{d}$. We will then show that if d does not divide D, the integer-solution of maximal value is attained for $\beta_{\text{large}} - \beta_{\text{small}} = 1$.

First case (d **divides** D) We now show that the following holds for the term $\prod_{n=1}^{d} \beta_n^{i_n}$ in the numerator of Eq. (1.7): $\prod_{n=1}^{d} \beta_n^{i_n} \leq \beta^{d_e}$. The numerator in Eq. (1.7) can also be written as $e_{d_e}(\beta_1, \ldots, \beta_d)$ where

$$\begin{aligned} e_{d_e}(\beta_1, \ldots, \beta_d) &= \sum_{i_1 < i_2 < \ldots < i_{d_e}} \beta_{i_1} \beta_{i_2} \ldots \beta_{i_{d_e}} \\ &= \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_d=0}^{1} \mathbb{1}_{(\sum_{n=1}^{d} i_n) = d_e} \prod_{n=1}^{d} \beta_n^{i_n} \end{aligned}$$

is the d_e-th elementary symmetric function of $\beta_1, \ldots, \beta_d$ [3]. Maclaurin's inequality [3] states that

$$\frac{e_1(\beta_1, \ldots, \beta_d)}{\binom{d}{1}} \geq \sqrt{\frac{e_2(\beta_1, \ldots, \beta_d)}{\binom{d}{2}}} \geq \ldots \tag{1.10}$$

$$\geq \sqrt[d_e]{\frac{e_{d_e}(\beta_1, \ldots, \beta_d)}{\binom{d}{d_e}}} \geq \ldots \geq \sqrt[d]{\frac{e_d(\beta_1, \ldots, \beta_d)}{\binom{d}{d}}}. \tag{1.11}$$

In particular,

$$\frac{e_1(\beta_1, \ldots, \beta_d)}{\binom{d}{1}} = \frac{\sum_{i=1}^{d} \beta_i}{d} = \frac{D}{d} = \beta \tag{1.12}$$

holds. Taking Eq. (1.11) and Eq. (1.12) to the power d_e and multiplying by $\binom{d}{d_e}$, we obtain

$$\beta^{d_e} \binom{d}{d_e} \geq e_{d_e}(\beta_1, \ldots, \beta_d) = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_d=0}^{1} \mathbb{1}_{(\sum_{n=1}^{d} i_n) = d_e} \prod_{n=1}^{d} \beta_n^{i_n}, \tag{1.13}$$

with equality if and only if $\beta_i = \beta_j$ for $i, j \in \{1, \ldots, n\}$ [3]. Therefore, the product in the numerator of Eq. (1.7) is maximized if all factors are equal.

Second case (d **does not divide** D) However, if d does not divide D, then β is no integer which is not feasible in our setting. The d_e-th elementary symmetric function $e_{d_e}(\boldsymbol{\beta})$ (see Eq. (1.13)) is known to be *Schur-concave* if $\beta_i \geq 0$ holds for all i [44]. This condition is met by β. We use the following definition of Marshall et al. [44]: A function $f : \mathbb{R}^d \to \mathbb{R}$ is called Schur-concave if $\boldsymbol{\gamma} \prec \boldsymbol{\beta}$ implies $f(\boldsymbol{\gamma}) \geq f(\boldsymbol{\beta})$. Here, $\boldsymbol{\gamma} \prec \boldsymbol{\beta}$ means that $\boldsymbol{\beta}$ *majorizes* $\boldsymbol{\gamma}$, i.e.,

$$\sum_{i=1}^{k} \boldsymbol{\gamma}_i^{\downarrow} \leq \sum_{i=1}^{k} \boldsymbol{\beta}_i^{\downarrow} \quad \text{for all } k \in \{1, \ldots, d\}, \quad \text{and} \quad \sum_{i=1}^{d} \boldsymbol{\gamma}_i = \sum_{i=1}^{d} \boldsymbol{\beta}_i,$$

where $\boldsymbol{\gamma}^{\downarrow}$ and $\boldsymbol{\beta}^{\downarrow}$ are the vectors of all elements in $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ in descending order [44].

We now show that there is no integer solution $\boldsymbol{\gamma}$ such that there is a near-balanced solution that majorizes $\boldsymbol{\gamma}$.

For some near-balanced assignment $\boldsymbol{\beta}$ of small and large bins to the d target dimensions, consider the vector

$$\boldsymbol{\beta}^{\downarrow} = \left(\underbrace{\overbrace{\left\lceil \frac{D}{d} \right\rceil}^{=\beta_{\text{large}}}, \ldots, \left\lceil \frac{D}{d} \right\rceil}_{D - d\beta_{\text{small}} \text{ many}}, \underbrace{\overbrace{\left\lfloor \frac{D}{d} \right\rfloor}^{=\beta_{\text{small}}}, \ldots, \left\lfloor \frac{D}{d} \right\rfloor}_{d\beta_{\text{large}} - D \text{ many}} \right)$$

of bin sizes in decreasing order. For any other BAxUS embedding given by some permutation $\boldsymbol{\beta}'$ of $\boldsymbol{\beta}$, it holds that $\boldsymbol{\beta}'^{\downarrow} = \boldsymbol{\beta}^{\downarrow}$. Note that for any assignment $\boldsymbol{\gamma} = \{\gamma_1, \dots, \gamma_d\}$ of bin sizes over the d target dimensions, it has to hold that

$$\sum_{i=1}^{d} \boldsymbol{\gamma}_i = D;\ \boldsymbol{\gamma}_i \geq 0,\ i \in \{1, \dots, d\};\ \boldsymbol{\gamma} \in \mathbb{N}_0^d.$$

By assumption, since we are in the near-balanced case, $\beta_{\text{small}} = \beta_{\text{large}} - 1$.

Assume there exists an assignment of bin sizes $\boldsymbol{\gamma}$ that is not a permutation of $\boldsymbol{\beta}$ such that $\boldsymbol{\gamma} \prec \boldsymbol{\beta}$, i.e.,

$$\sum_{i=1}^{k} \boldsymbol{\gamma}_i^{\downarrow} \leq \sum_{i=1}^{k} \boldsymbol{\beta}_i^{\downarrow} \quad \text{for all } k \in \{1, \dots, d\}, \tag{1.14}$$

and

$$\sum_{i=1}^{d} \boldsymbol{\gamma}_i = \sum_{i=1}^{d} \boldsymbol{\beta}_i, \tag{1.15}$$

and

$$\exists j : \boldsymbol{\gamma}_j^{\downarrow} < \boldsymbol{\beta}_j^{\downarrow}. \tag{1.16}$$

Let $\boldsymbol{\kappa}$ denote the (non-empty) set of such indices. Because the elements of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ both sum up to D, it has to hold for all $\kappa \in \boldsymbol{\kappa}$ that

$$\sum_{i=1, i\neq\kappa}^{d} \boldsymbol{\gamma}_i^{\downarrow} > \sum_{i=1, i\neq\kappa}^{d} \boldsymbol{\beta}_i^{\downarrow}. \tag{1.17}$$

Remember that $\boldsymbol{\beta}$ only contains elements of sizes β_{small} and β_{large} with $\beta_{\text{large}} = \beta_{\text{small}} - 1$. Then, Eq. (1.17) can only hold if either 1) $\boldsymbol{\gamma}$ contains more elements of size β_{large} than $\boldsymbol{\beta}$ or 2) if it contains at least one element that is larger than β_{large}, the largest element in $\boldsymbol{\beta}$.

Both cases lead to a contradiction. In the first case,

$$\sum_{i=1}^{D-d\beta_{\text{small}}+1} \boldsymbol{\gamma}_i^{\downarrow} > \sum_{i=1}^{D-d\beta_{\text{small}}+1} \boldsymbol{\beta}_i^{\downarrow} \Rightarrow \boldsymbol{\gamma} \not\prec \boldsymbol{\beta}$$

since at least the first $D - d\beta_{\text{small}} + 1$ elements of $\boldsymbol{\gamma}^{\downarrow}$ are $\left\lceil \frac{D}{d} \right\rceil$ but only the first $D - d\beta_{\text{small}}$ elements of $\boldsymbol{\beta}^{\downarrow}$ are $\left\lceil \frac{D}{d} \right\rceil$ and the $D - d\beta_{\text{small}} + 1$-th element of $\boldsymbol{\beta}^{\downarrow}$ is $\left\lceil \frac{D}{d} \right\rceil - 1$.

In the second case, $\boldsymbol{\gamma} \not\prec \boldsymbol{\beta}$ because $\boldsymbol{\gamma}_1^{\downarrow} > \boldsymbol{\beta}_1^{\downarrow}$. It follows that no such $\boldsymbol{\gamma}$ exists. Therefore, the BAxUS embedding has a maximum worst-case success probability among sparse embeddings. □

A.3 Proof of Corollary 1

Corollary 1. For $D \to \infty$, the worst-case success probability of the BAxUS embedding is

$$\lim_{D\to\infty} p_B(Y^*; D, d, d_e) = \frac{d!}{(d-d_e)!d^{d_e}}, \tag{1.18}$$

and hence matches HeSBO's worst-case success probability $p_H(Y^*; d, d_e)$.

Proof. By Corollary 1, the following holds for arbitrary $1 \leq d_e \leq d \leq D$ where $d, d_e, D \in \mathbb{N}_{++}$:

$$\frac{d!}{(d-d_e)!d^{d_e}} \leq p_B(Y^*; D, d, d_e),$$

because $\frac{d!}{(d-d_e)!d^{d_e}}$ is HeSBO's worst-case success probability and hence less or equal to the worst-case success probability of BAxUS.

Furthermore, by the proof of Corollary 1,

$$p_B(Y^*; D, d, d_e) \leq \frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}},$$

because $\frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}}$ is larger or equal the worst-case success probability of any sparse embedding, among which BAxUS is the embedding with maximum worst-case success probability and $\frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}} = p_B(Y^*; D, d, d_e)$ if and only if $\beta_{\text{small}} = \beta_{\text{large}}$, i.e., d divides D.

In summary, we have

$$\frac{d!}{(d-d_e)!d^{d_e}} \leq p_B(Y^*; D, d, d_e) \leq \frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}}.$$

We now show that, for fixed d and d_e, the sequences $\frac{d!}{(d-d_e)!d^{d_e}}$ and $\frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}}$ converge to the same point as $D \to \infty$. We consider $\lim_{\beta\to\infty}$, which is equivalent to $\lim_{D\to\infty}$ as $\beta = \frac{D}{d}$ and d is fixed. Note, that we can consider $\beta = \frac{D}{d}$ even though

it is not a valid success probability when d does not divide D, since we are only interested in bounding the true success probability. Then,

$$\lim_{\beta\to\infty} \frac{\beta^{d_e}\binom{d}{d_e}}{\binom{D}{d_e}} = \lim_{\beta\to\infty} \frac{\beta^{d_e} d!(D-d_e)!}{D!(d-d_e)!} \tag{1.19}$$

$$= \lim_{\beta\to\infty} \frac{\beta^{d_e} d!(\beta d-d_e)!}{(\beta d)!(d-d_e)!} \tag{1.20}$$

$$= \lim_{\beta\to\infty} \beta^{d_e} \frac{d!}{(d-d_e)!} \frac{(\beta d-d_e)!}{(\beta d)!} \tag{1.21}$$

Applying Stirling's approximation [25] to the numerator and the denominator of the last factor, we obtain

$$= \lim_{\beta\to\infty} \beta^{d_e} \frac{d!}{(d-d_e)!} \frac{\sqrt{2\pi(\beta d-d_e)}\left(\frac{\beta d-d_e}{e}\right)^{\beta d-d_e}}{\sqrt{2\pi\beta d}\left(\frac{\beta d}{e}\right)^{\beta d}} \frac{r(\beta d-d_e)}{r(\beta d)} \tag{1.22}$$

$$= \lim_{\beta\to\infty} \beta^{d_e} \frac{d!}{(d-d_e)!} \sqrt{\frac{\beta d-d_e}{\beta d}} e^{d_e} \frac{(\beta d-d_e)^{\beta d-d_e}}{(\beta d)^{\beta d}} \frac{r(\beta d-d_e)}{r(\beta d)} \tag{1.23}$$

$$= \lim_{\beta\to\infty} \beta^{d_e} \frac{d!}{(d-d_e)!} \sqrt{\frac{\beta d-d_e}{\beta d}} e^{d_e} \left(\frac{\beta d-d_e}{\beta d}\right)^{\beta d} \cdot \frac{1}{(\beta d-d_e)^{d_e}} \frac{r(\beta d-d_e)}{r(\beta d)} \tag{1.24}$$

$$= \lim_{\beta\to\infty} \frac{d!}{(d-d_e)!} \sqrt{\frac{\beta d-d_e}{\beta d}} e^{d_e} \left(\frac{\beta d-d_e}{\beta d}\right)^{\beta d} \cdot \frac{\beta^{d_e}}{(\beta d-d_e)^{d_e}} \frac{r(\beta d-d_e)}{r(\beta d)} \tag{1.25}$$

$$= \lim_{\beta\to\infty} \frac{d!}{(d-d_e)!} \underbrace{\sqrt{\frac{\beta d-d_e}{\beta d}}}_{\to 1} e^{d_e} \underbrace{\left(\frac{\beta d-d_e}{\beta d}\right)^{\beta d}}_{\to e^{-d_e}} \underbrace{\left(\frac{1}{d}\right)^{d_e}}_{=d^{-d_e}} \cdot \underbrace{\left(\frac{\beta d}{\beta d-d_e}\right)^{d_e}}_{\to 1} \underbrace{\frac{r(\beta d-d_e)}{r(\beta d)}}_{\to 1} \tag{1.26}$$

$$= \frac{d!}{(d-d_e)!d^{d_e}} \tag{1.27}$$

where the following holds for the error term $r(x)$ of the Stirling approximation [57]:

$$\exp\left(\frac{1}{12x+1}\right) \leq r(x) \leq \exp\left(\frac{1}{12x}\right).$$

Then, $\frac{r(\beta d - d_e)}{r(\beta d)} \to 1$ for $\beta \to \infty$ holds since

$$\frac{r(\beta d - d_e)}{r(\beta d)} \leq \exp\left(\frac{1}{12(\beta d - d_e)} - \frac{1}{12\beta d + 1}\right) \tag{1.28}$$

$$= \exp\left(\frac{12d_e + 1}{12^2\beta^2 d^2 + 12\beta d - 12^2\beta d d_e - 12d_e}\right) \tag{1.29}$$

$$= \exp\left(\frac{d_e + \frac{1}{12}}{\beta d(12\beta d + 1 - 12d_e) - d_e}\right), \tag{1.30}$$

and

$$\frac{r(\beta d - d_e)}{r(\beta d)} \geq \exp\left(\frac{1}{12(\beta d - d_e) + 1} - \frac{1}{12\beta d}\right) \tag{1.31}$$

$$= \exp\left(\frac{12d_e - 1}{12^2\beta^2 d^2 + 12\beta d - 12^2\beta d d_e - 12\beta d_e}\right) \tag{1.32}$$

$$= \exp\left(\frac{d_e - \frac{1}{12}}{\beta d(12\beta d + 1 - 12d_e) - d_e}\right), \tag{1.33}$$

which both go to 1 as $\beta \to \infty$.

Hence, `BAxUS`' worst-case success probability is bounded from below and above by sequences that converge to the same point as $D \to \infty$. The squeeze theorem (e.g., [63]) implies

$$\lim_{D\to\infty} p_B(Y^*; D, d, d_e) = \frac{d!}{(d-d_e)!d^{d_e}}.$$

□

B Consistency of `BAxUS`

We prove the global convergence of function values for `BAxUS`. The proof idea is similar to Eriksson and Poloczek [21] but relaxes the assumption of a unique global minimizer. By construction, f is sparse (see Definition 4), i.e., there exists a set of dimensions of f that do not influence the function value. Thus, an optimal solution

stays optimal regardless of how inactive dimensions are set. This is why we must relax the assumption of a unique global minimizer in the input space. Instead, we assume a unique global minimizer in the active subspace $\boldsymbol{z}^* \in \mathcal{Z}$ that can map to arbitrarily many minimizers in the input space. Note that this assumption covers the case when the target space corresponds to the input space, i.e., $d = D$.

Theorem 2 (**BAxUS consistency**). *With the following definitions:*

D1. $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ *is a sequence of points of decreasing function value;*

D2. $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ *is a minimizer in* $\mathcal{X}$*;*

and under the following assumptions:

A1. D *is finite;*

A2. f *is observed without noise;*

A3. f *is sparse and bounded in* $\mathcal{X}$*, i.e.,* $\exists\, C \in \mathbb{R}_{++}$ *s.t.* $|f(\boldsymbol{x})| < C \,\forall\, \boldsymbol{x} \in \mathcal{X}$*;*

A4. At least one of the minimizers $\boldsymbol{x}_i^*$ *lies in a continuous region with positive measure;*

A5. Once **BAxUS** *reached the input dimensionality* D*, the initial points* $\{\boldsymbol{x}_i\}_{i=1}^{n_{init}}$ *after each TR restart for* **BAxUS** *are chosen such that* $\forall\delta \in \mathbb{R}_{++}$ *and* $\boldsymbol{x} \in \mathcal{X}$*,* $\exists\, \nu(\boldsymbol{x},\delta) > 0$*:* $\mathbb{P}\left(\exists i : ||\boldsymbol{x} - \boldsymbol{x}_i||_2 \leq \delta\right) \geq \nu(\boldsymbol{x},\delta)$*, i.e., the probability that at least one point in* $\{\boldsymbol{x}_i\}_{i=1}^{n_{\text{init}}}$ *ends up in a ball centered at* $\boldsymbol{x}$ *with radius* δ *is at least* $\nu(\boldsymbol{x},\delta)$*;*

$f(\boldsymbol{x}_k)$ *converges to* $f(\boldsymbol{x}^*)$ *with probability 1.*

Proof. We first show that BAxUS must eventually arrive at an embedding equivalent to the input space. By Assumption A1, the number of accepted "failures" (i.e., the number of times BAxUS needs to fail in finding a better solution until the TR base length is shrunk) is always finite since it is always bounded by the target dimension ($\forall i\, \tau_{\text{fail}}^i \leq d_i$) which is at most equal to D ($\forall i\, d_i \leq D$). By the facts that BAxUS considers any sampled point an improvement only if it improves over the current best solution by at least some constant $\gamma \in \mathbb{R}_{++}$ and that f is bounded (Assumption A3), BAxUS can only perform a finite number of function evaluations without increasing the target dimensionality of its embedding.

Once BAxUS reaches D, it behaves like TuRBO [22] for which Eriksson and Poloczek [21] proved global convergence assuming a unique global minimizer. For the case

$d_e < D$, we notice that multiple minima in the input space occur due to inactive dimensions that do not influence the function value.

The remainder of our proof is based on the convergence theorem for global search by Solis and Wets [64], which proves convergence of function values for random search with possibly multiple minima. By considering the sequence

$$\left\{ \boldsymbol{x}_i' \in \underset{\hat{\boldsymbol{x}} \in \{\boldsymbol{x}_k\}_{k=1}^{i}}{\arg\min} \; f(\hat{\boldsymbol{x}}) \right\}_{i=1}^{\infty}$$

of points of decreasing function values where $\{\boldsymbol{x}_k\}_{k=1}^{i}$ are the observations up to the i-th function evaluation, Definition D1 is satisfied. Additionally, by the fact that, at each TR restart, `BAxUS` performs random restarts with uniform probability on $\mathcal{X}$, `BAxUS` satisfies the assumptions of the theorem by Solis and Wets [64].

The theorem by Solis and Wets [64] states that for a sequence $\{\boldsymbol{x}_k\}_{k=1}^{\infty}$ of sampling points with $\varepsilon \in \mathbb{R}_{++}$,

$$\begin{aligned} &\lim_{k \to \infty} \mathbb{P}\left[\boldsymbol{x}_k \in R_\varepsilon\right] = 1 \\ &R_\varepsilon = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) < \alpha + \varepsilon\} \\ &\alpha = \inf\{t : v(\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) < t) > 0\} \end{aligned}$$

where R_ε is the set of ε-optimal function values, α is the *essential infimum*, and v is the Lebesgue measure. Note that the essential infimum α is equal to the minimum if the minimizer lies in a continuous region of positive measure, i.e., $\alpha = f(\boldsymbol{x}_i^*)$. By Assumption A4 and by letting $\varepsilon \to 0$, $f(\boldsymbol{x}_k)$ converges to $f(\boldsymbol{x}_i^*)$. □

C Additional Empirical Evaluations

C.1 Ablation Study for the `BAxUS` Embedding

We conduct an ablation study to investigate the difference between the `BAxUS` and `HeSBO` embeddings. We run `TuRBO` of Eriksson et al. [22] in an embedded subspace with the two different embeddings. We use a version of Ackley10 (ten active dimensions, i.e., $d_e = 10$), where we shift the optimum away from the origin with a uniformly random vector $\boldsymbol{\delta} \in [-32.768, 32.768]^{d_e}$ with $\delta_i \sim \mathcal{U}(-32.768, 32.768)$. The function we optimize is then

$$f_{\text{ShiftedAckley10}}(\boldsymbol{x}) = f_{\text{Ackley10}}(\boldsymbol{x} + \boldsymbol{\delta}).$$

We adjust the boundaries of the search space such that $f_{\text{ShiftedAckley10}}$ is evaluated on the domain $\mathcal{X} = [-32.768, 32.768]^{d_e}$. The reason for shifting the optimum is that the original Ackley function has its optimum at the origin. In that case, any sparse embedding contains this optimum, even if all the active input dimensions are mapped to the same target dimension.

We add 10 dummy dimensions, such that $D = 30$ and set $d = 20$. With this problem-setting, the `BAxUS` and `HeSBO` embeddings have a probability of approximately 0.27 and 0.07 of containing the optimum, respectively.

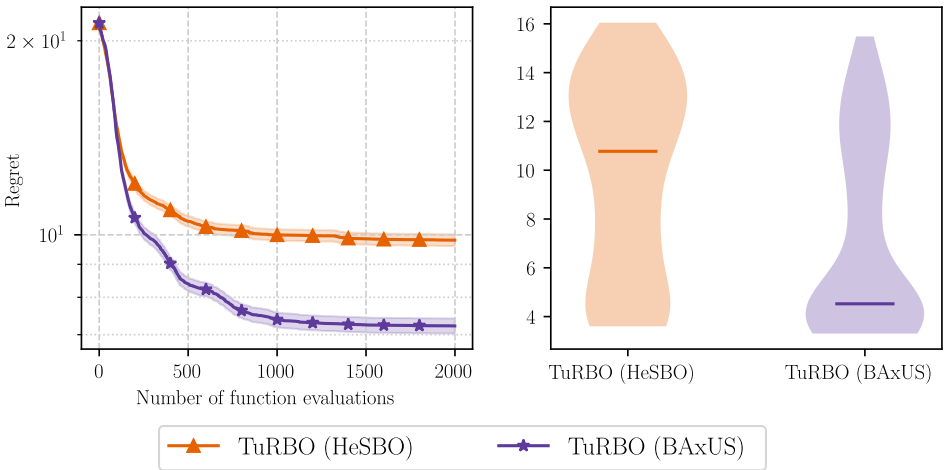


Figure 1.5: Left: the `BAxUS` embedding gives better optimization performance on the shifted Ackley10 function: `TuRBO` in embedded subspaces of the `BAxUS` and `HeSBO` embeddings. The `BAxUS` embedding has a higher probability to contain the optimum. Right: the distribution of the final incumbents (lower the better). The horizontal bars show the median.

The left side of Figure 1.5 shows the incumbent mean for `TuRBO` in the two different embedded subspaces. The shaded regions show one standard error. `TuRBO` in an `BAxUS` embedding has significantly better optimization performance than in a `HeSBO` embedding. The right side of Figure 1.5 shows the distributions of the final incumbents and their median. The `BAxUS` embedding leads to a significantly lower median and only rarely a similarly bad embedding as the `HeSBO` method when combined with `TuRBO`.

We perform a two-sided Wilcoxon rank-sum statistical test to check the difference between the best observed function values for the two embeddings. The difference is significant with $p \approx 0.00001$.

The performance difference between the two embeddings depends on the

characteristics of the function and the different dimensionalities, the input dimensionality D, the target dimensionality d, and the effective dimensionality d_e. For problems with few active dimensions and many input dimensions, the BAxUS and HeSBO embeddings become more similar (see Figure 1.2). However, by Corollary 2, the BAxUS embedding is, in expectation and in terms of the worst-case success probability, better than the HeSBO embedding for arbitrary sparse functions.

For functions with an optimum at the origin, both embeddings contain that optimum regardless of d: Even if all active input dimensions are mapped to the same target dimension, the optimum in the input space can be reached by "setting" this particular target dimension to zero.

TuRBO with BAxUS embedding vs. BAxUS. We compare the simple idea of running TuRBO in a BAxUS embedding of fixed target dimensionality with the BAxUS algorithm described in Section 3. We run this simple approach for 11 different target dimensionalities d $(2, 10, 20, \ldots, 100)$ on the Lasso-Hard benchmark and show the results with a sequential color map in Figure 1.6. Only the first $d = 2$-dimensional embedding achieves the same initial speedup as BAxUS, which is expected as BAxUS starts in a similarly low-dimensional initial embedding. However, the fixed embedding cannot explore the input space sufficiently and has the worst final solution. High-dimensional fixed embeddings have more freedom in exploring the input space; however, they suffer from slower initial optimization performance.

BAxUS has the same initial speedup as the two-dimensional fixed embedding but can explore the space further by increasing the dimensionality of its embedding.

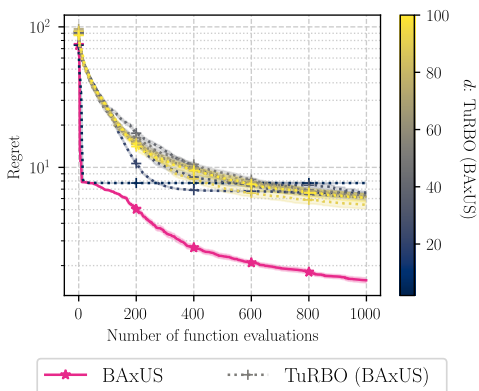


Figure 1.6: An evaluation of BAxUS and TuRBO with BAxUS embeddings of different target dimensionalities on Lasso-Hard: We run TuRBO with the BAxUS embedding for fixed target dimensionalities $d = 2, 10, 20, \ldots, 100$ and compare to BAxUS.

Summing up, we observe that BAxUS achieves a better performance than TuRBO with a fixed embedding dimensionality.

C.2 Evaluation on an Additional Lasso Benchmark

In addition to the synthetic Lasso-High and Lasso-Hard benchmarks studied in Section 4, we evaluate BAxUS on the Lasso-DNA benchmark from LassoBench [60]. The Lasso-DNA benchmark is a biomedical classification task, taking binarized DNA sequences as input [60].

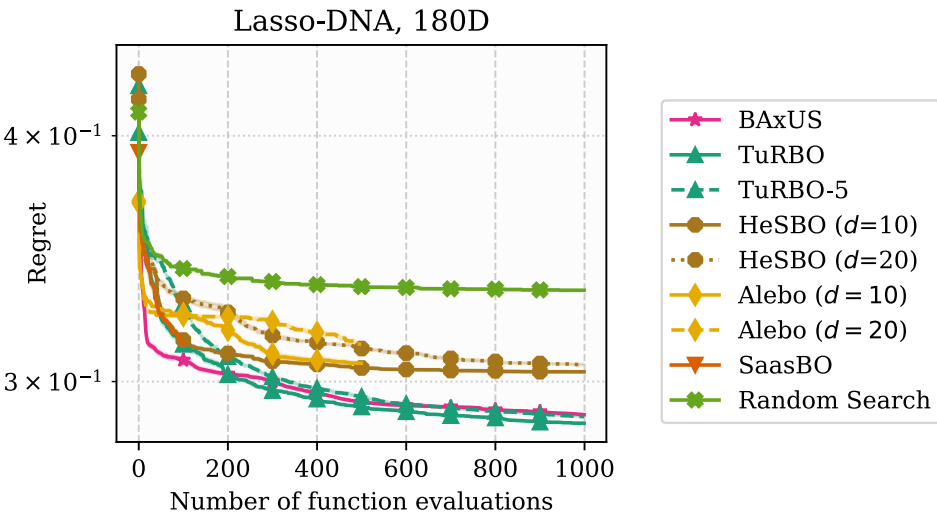


Figure 1.7: BAxUS and baselines on Lasso-DNA. As before, BAxUS makes considerable progress in the beginning and converges faster than TuRBO and CMA-ES.

Figure 1.7 shows the mean performance of BAxUS on the Lasso-DNA. Each line shows the incumbent mean; the shaded regions around the lines show one standard error. We see the same qualitative behavior as discussed in Section 4: BAxUS reaches a good initial solution faster than any other method.

After a worse start, TuRBO finds slightly better solutions than BAxUS.

C.3 Evaluation on Additional MuJoCo Benchmarks

We evaluate BAxUS with the same baselines as in Section 4. We use the implementation of [71][1], in particular we use the Gym environments Ant, Swimmer, Half-Cheetah, Hopper, Walker 2D, and Humanoid 2D, all in version 2. For the 6392-dimensional Humanoid benchmark, we limit the target dimensionality of BAxUS to 1000 dimensions to keep the split budgets sufficiently large. For the other benchmarks, we do not limit the target dimensionality. Due to the high variance between runs, we ran all methods for 50 different runs.

We summarize the results in Fig. 1.8. We observe that BAxUS obtains equal or better solutions than the competitors on four out of six benchmarks. On the 120-dimensional Walker benchmark, BAxUS is the clear winner, followed by TuRBO and CMA-ES. On the 888-dimensional Ant benchmark, HeSBO finds the best solutions, followed by BAxUS that outperforms TuRBO and CMA-ES. For the 102-dimensional Half-Cheetah, TuRBO produces the best solutions, followed by CMA-ES and BAxUS; here, the subspace-based approaches (Alebo and HeSBO) find significantly worse solutions. For the 6392-dimensional Humanoid 2D, CMA-ES obtains the best solutions, followed by BAxUS, Alebo, and HeSBO.

D The Nested Family of Random Embeddings

We describe the method for increasing the target dimensionality under the retention of the observations. Suppose that we have collected n observations and are in target dimension d when Algorithm 1.2 is invoked. Algorithm 1.2 loops over the target dimensions $1, \dots, d$. For each target dimension, the contributing input dimensions are randomly re-assigned to new bins of given sizes. This can, for example, be realized by first randomly permuting the list of contributing input dimensions, and then dividing the list into $b + 1$ chunks (bins). If the number of contributing input dimensions is less than $b + 1$ (remember that b is the number of *new* bins), then it is not possible to re-assign the contributing input dimensions to $b + 1$ bins. Therefore, we re-assign the contributing input dimensions to $\hat{b} = \min(b, l_s - 1)$ new bins, where l_s is the number of contributing input dimensions to the s-th target dimension. This also ensures that the target dimension never grows larger than D in the BAxUS embedding. We evenly distribute the l_s contributing input dimensions across the $\hat{b}$ bins by again using the BAxUS embedding. This gives a smaller (in terms of number of rows) projection matrix $\tilde{S}^\top$ which we finally use to update $S^\top$:

[1] https://github.com/facebookresearch/LA-MCTS/blob/main/example/mujuco/functions.py, last accessed: 06/10/2022

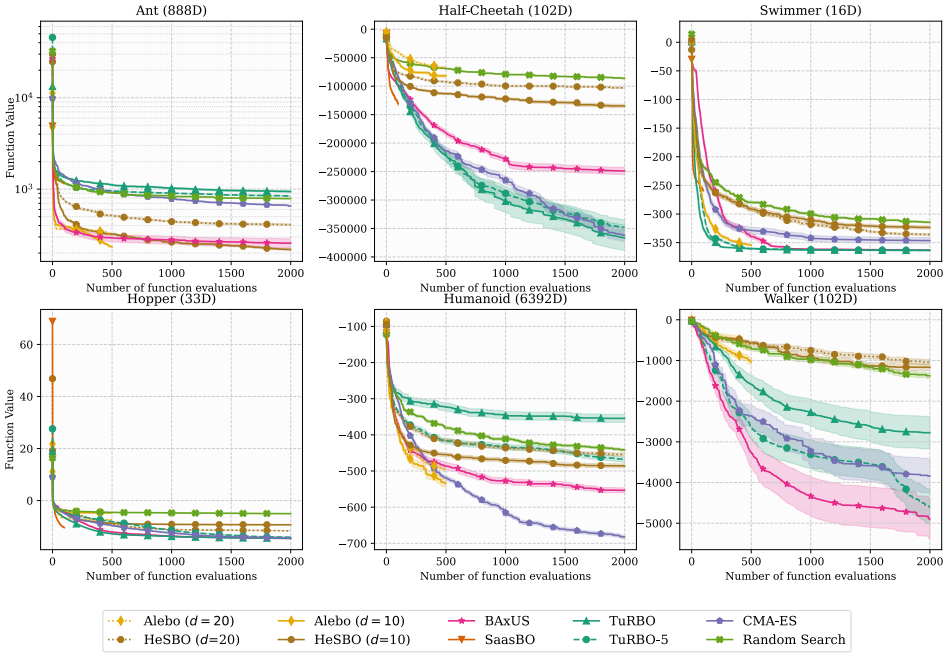


Figure 1.8: An evaluation of `BAxUS` and other methods on high-dimensional test problems of MuJoCo.

Algorithm 1.2 Observation-preserving embedding increase

Input: transposed embedding matrix $S^\top$, number of new bins per latent dimension b, observed points $Y \in [-1, 1]^{n \times d}$.

Output: updated transposed embedding matrix $S^\top$ and updated observation matrix Y

for $s \in \{1, \dots, d\}$ **do**

$\boldsymbol{D}^s \leftarrow$ contributing input dimensions of s-th latent dimension of the current embedding

$l_s \leftarrow |\boldsymbol{D}^s|$

$\hat{b} \leftarrow \min(b, l_s - 1)$ ▷ If $l_s - 1 < b$, we can at most create $l_s - 1$ new bins.

Copy and append s-th column of Y $\hat{b}$ times at the end of Y.

Add $\hat{b}$ zero columns at the end of $S^\top$.

$\boldsymbol{\sigma} \leftarrow$ signs of dimensions $\in \boldsymbol{D}^s$.

$\tilde{S}^\top \leftarrow$ `Baxus-Embedding`$(l_s, \hat{b} + 1)$ ▷ Re-assign input dims. equally[2], $\tilde{S}^\top \in \{0, \pm 1\}^{l_s \times \hat{b}+1}$

for $i \in \{1, \dots, l_s\}, j \in \{1, \dots, \hat{b} + 1\}$ **do**

if $\tilde{S}^\top_{ij} \neq 0$ **then**

if $j > 1$ **then** ▷ Move values that fall into new bins to end of $S^\top$.

$S^\top_{\boldsymbol{D}^s_i, \hat{d}-\hat{b}-1+j} \leftarrow \boldsymbol{\sigma}_i$ ▷ $\hat{d}$: columns of $S^\top$

$S^\top_{\boldsymbol{D}^s_i, s} \leftarrow 0$ ▷ Set value in "old" column to zero.

Return $S^\top$ and Y.

E Additional Details on the Implementation and the Empirical Evaluation

We benchmark against `SAASBO`, `TuRBO`, `HeSBO`, `Alebo`, and `CMA-ES`:

- For `SAASBO`, we use the implementation from [20] (`https://github.com/martinjankowiak/saasbo`, license: none, last accessed: 05/09/2022).
- For `TuRBO`, we use the implementation from [22] (`https://github.com/uber-research/TuRBO`, license: `Uber`, last accessed: 05/09/2022).
- For `HeSBO` and `Alebo`, we use the implementation from [40] (`https://github.com/facebookresearch/alebo`, license: `CC BY-NC 4.0`, last accessed: 05/09/2022).
- For the Lasso benchmarks, we use the implementation from [60] (`https://github.com/ksehic/LassoBench`, license: `MIT` and `BSD-3-Clause`, last accessed: 05/09/2022).

We use `GPyTorch` (version 1.8.1) to train the GP with the following setup: We place a top-hat prior on the Gaussian likelihood noise, the signal variance, and the length scales of the Matérn 5/2 automatic relevance determination (ARD) kernel. The interval for the noise is $[0.005, 0.2]$, for the signal variance $[0.05, 20]$, and for the lengthscales $[0.005, 10]$.

We evaluate on the synthetic Branin2[3] and Hartmann6[4] functions. Since we augment the function with dummy dimensions, we use the same domain for x_1 and x_2, namely $[-5, 15]^D$ for Branin2 and $[0, 1]^D$ for Hartmann6.

Similar to `TuRBO`, we sample a $\min(100 d_n, 5000)$-element Sobol sequence on which we minimize the posterior sample. To maximize the marginal log-likelihood of the GP, we sample 100 initial hyperparameter configurations. The ten best samples are further optimized using the Adam optimizer for 50 steps.

We ran the experiments for approximately 15,000 core hours on Intel Xeon Gold 6130 CPUs provided by a compute cluster.

[2]Equally means that all $\hat{b}+1$ bins have roughly the same number of contributing input dimensions. The number of contributing input dimensions to the different bins differ by at most 1.

[3]See `https://www.sfu.ca/~ssurjano/branin.html`, last accessed: 05/09/2022

[4]See `https://www.sfu.ca/~ssurjano/hart6.html`, last accessed: 05/09/2022

References

[1] Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Benjamin Letham, Ashwin Murthy, and Shaun Singh. Ae: A domain-agnostic platform for adaptive experimentation. *Conference on Neural Information Processing Systems*, pages 1–8, 2018.

[2] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. *International Conference on Machine Learning*, pages 462–471. PMLR, 2018.

[3] Edwin F Beckenbach and Richard Bellman. *Inequalities*, volume 30. Springer Science & Business Media, 2012.

[4] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(2), 2012.

[5] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24. Curran Associates, Inc., 2011.

[6] Mickaël Binois. *Uncertainty quantification on Pareto fronts and high-dimensional strategies in Bayesian optimization, with applications in multi-objective automotive design*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2015.

[7] Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

[8] Mickaël Binois, David Ginsbourger, and Olivier Roustant. A warped kernel improving robustness in Bayesian optimization via random embeddings. *International Conference on Learning and Intelligent Optimization (LION)*, pages 281–286. Springer, 2015.

[9] Mickaël Binois, David Ginsbourger, and Olivier Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of global optimization*, 76(1):69–90, 2020.

[10] Mohamed Amine Bouhlel, Nathalie Bartoli, Rommel G. Regis, Abdelkader Otsmane, and Joseph Morlier. Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method. *Engineering Optimization*, 50(12):2038–2053, 2018.

[11] Benjamin Burger, Phillip M Maffettone, Vladimir V Gusev, Catherine M Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M Alston, Buyi Li, Rob Clowes, et al. A mobile robotic chemist. *Nature*, 583(7815):237–241, 2020.

[12] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian Gait Optimization for Bipedal Locomotion. *Learning and Intelligent Optimization*, pages 274–290. Springer International Publishing, 2014.

[13] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[14] A. Candelieri, R. Perego, and F. Archetti. Bayesian Optimization of Pump Operations in Water Distribution Systems. *Journal of Global Optimization*, 71 (1):213–235, May 2018.

[15] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding Frequent Items in Data Streams. Peter Widmayer, Stephan Eidenbenz, Francisco Triguero, Rafael Morales, Ricardo Conejo, and Matthew Hennessy, editors, *Automata, Languages and Programming*, pages 693–703, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45465-6.

[16] Jingfan Chen, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. Semi-supervised Embedding Learning for High-dimensional Bayesian Optimization. *arXiv preprint arXiv:2005.14601*, 2020.

[17] Paul G. Constantine. *Active Subspaces*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015.

[18] Zachary Cosenza, Raul Astudillo, Peter Frazier, Keith Baar, and David E Block. Multi-Information Source Bayesian Optimization of Culture Media for Cellular Agriculture. *Biotechnology and Bioengineering*, 2022.

[19] Adel Ejjeh, Leon Medvinsky, Aaron Councilman, Hemang Nehra, Suraj Sharma, Vikram Adve, Luigi Nardi, Eriko Nurvitadhi, and Rob A. Rutenbar. HPVM2FPGA: Enabling True Hardware-Agnostic FPGA Programming. *Proceedings of the 33rd IEEE International Conference on Application-specific Systems, Architectures, and Processors*, 2022.

[20] David Eriksson and Martin Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 493–503. PMLR, 27–30 Jul 2021.

[21] David Eriksson and Matthias Poloczek. Scalable Constrained Bayesian Optimization. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 730–738. PMLR, 13–15 Apr 2021.

[22] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5496–5507, 2019.

[23] Peter I. Frazier and Jialei Wang. *Bayesian Optimization for Materials Design*, pages 45–75. Springer International Publishing, Cham, 2016. ISBN 978-3-319-23871-5.

[24] Jacob Gardner, Chuan Guo, Kilian Weinberger, Roman Garnett, and Roger Grosse. Discovering and exploiting additive structure for Bayesian optimization. *International Conference on Artificial Intelligence and Statistics*, pages 1311–1319, 2017.

[25] Ronald L Graham, Donald E Knuth, Oren Patashnik, and Stanley Liu. Concrete mathematics: a foundation for computer science. *Computers in Physics*, 3(5):106–107, 1989.

[26] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. *Proceedings of IEEE International Conference on Evolutionary Computation (ICEC)*, pages 312–317, 1996.

[27] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. Last access: 05/09/2022. License: BSD-3-Clause.

[28] Florian Hase, Loïc M Roch, Christoph Kreisbeck, and Alán Aspuru-Guzik. Phoenics: a Bayesian optimizer for chemistry. *ACS central science*, 4(9):1134–1145, 2018.

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

[30] Henry C Herbol, Weici Hu, Peter Frazier, Paulette Clancy, and Matthias Poloczek. Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization. *npj Computational Materials*, 4(1):1–7, 2018.

[31] José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space. *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1470–1479. PMLR, 06–11 Aug 2017.

[32] Zak E Hughes, Michelle A Nguyen, Jialei Wang, Yang Liu, Mark T Swihart, Matthias Poloczek, Peter I Frazier, Marc R Knecht, and Tiffany R Walsh. Tuning Materials-Binding Peptide Sequences toward Gold-and Silver-Binding Selectivity with Bayesian Optimization. *ACS nano*, 15(11):18260–18269, 2021.

[33] Carl Hvarfner, Danny Stoll, Artur Souza, Luigi Nardi, Marius Lindauer, and Frank Hutter. PiBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization. *International Conference on Learning Representations*, 2022.

[34] Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. *MOPTA 2008 Conference (20 August 2008)*, 2008.

[35] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. *International conference on machine learning (ICML)*, pages 295–304, 2015.

[36] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018.

[37] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[38] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 20–22 Apr 2017.

[39] Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in Bayesian optimization with applications in aerospace engineering. *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[40] Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-Examining Linear Embeddings for High-Dimensional Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1546–1558, 2020.

[41] Daniel J Lizotte, Tao Wang, Michael H Bowling, Dale Schuurmans, et al. Automatic Gait Optimization With Gaussian Process Regression. *IJCAI*, volume 7, pages 944–949, 2007.

[42] Xiaoyu Lu, Javier Gonzalez, Zhenwen Dai, and Neil D. Lawrence. Structured Variationally Auto-encoded Optimization. Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3273–3281. PMLR, 2018.

[43] Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

[44] Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: theory of majorization and its applications*, volume 143. Springer, 1979.

[45] Natalie Maus, Haydn Thomas Jones, Juston Moore, Matt Kusner, John Bradshaw, and Jacob R. Gardner. Local latent space bayesian optimization over structured inputs. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[46] Matthias Mayr, Faseeh Ahmad, Konstantinos I. Chatzilygeroudis, Luigi Nardi, and Volker Krüger. Skill-based Multi-objective Reinforcement Learning of Industrial Robot Tasks with Planning and Knowledge Integration. *CoRR*, abs/2203.10033, 2022.

[47] Riccardo Moriconi, Marc Peter Deisenroth, and K. S. Sesh Kumar. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109(9):1925–1943, Sep 2020.

[48] Mojmir Mutny and Andreas Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[49] Luigi Nardi, David Koeplinger, and Kunle Olukotun. Practical design space exploration. *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 347–358. IEEE, 2019.

[50] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian Optimization in Embedded Subspaces. *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research (PMLR)*, pages 4752–4761, 09–15 Jun 2019.

[51] Diana M. Negoescu, Peter I. Frazier, and Warren B. Powell. The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[52] Daniel Packwood. *Bayesian Optimization for Materials Science*. Springer, 2017.

[53] Giulia Pedrielli and Szu Hui Ng. G-STAR: A new kriging-based trust region method for global optimization. *2016 Winter Simulation Conference (WSC)*, pages 803–814. IEEE, 2016.

[54] Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778, 2018.

[55] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006.

[56] Rommel G Regis. Trust regions in Kriging-based optimization with expected improvement. *Engineering optimization*, 48(6):1037–1059, 2016.

[57] Herbert Robbins. A remark on stirling's formula. *The American mathematical monthly*, 62(1):26–29, 1955.

[58] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable Neural Architecture Search via Bayesian Optimisation with Weisfeiler-Lehman Kernels. *International Conference on Learning Representations*, 2021.

[59] Artur M Schweidtmann, Adam D Clayton, Nicholas Holmes, Eric Bradford, Richard A Bourne, and Alexei A Lapkin. Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. *Chemical Engineering Journal*, 352:277–282, 2018.

[60] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. *First Conference on Automated Machine Learning (Main Track)*, 2022.

[61] Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590 (7844):89–96, 2021.

[62] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.

[63] Houshang H Sohrab. *Basic real analysis*, volume 231. Springer, 2003.

[64] Francisco J. Solis and Roger J-B. Wets. Minimization by Random Search Techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.

[65] Artur Souza, Leonardo B Oliveira, Sabine Hollatz, Matt Feldman, Kunle Olukotun, James M Holton, Aina E Cohen, and Luigi Nardi. DeepFreak: Learning crystallography diffraction patterns with automated machine learning. *arXiv preprint arXiv:1904.11834*, 2019.

[66] Lorillee Tallorin, JiaLei Wang, Woojoo E Kim, Swagat Sahu, Nicolas M Kosa, Pu Yang, Matthew Thompson, Michael K Gilson, Peter I Frazier, Michael D Burkart, et al. Discovering de novo peptide substrates for enzymes using machine learning. *Nature communications*, 9(1):1–10, 2018.

[67] William R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3/4): 285–294, 1933.

[68] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 11259–11272, 2020.

[69] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials Discovery*, 4:18–21, 2016.

[70] Xingchen Wan, Vu Nguyen, Huong Ha, Binxin Ru, Cong Lu, and Michael A. Osborne. Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces. *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10663–10674. PMLR, 18–24 Jul 2021.

[71] 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 (NeurIPS)*, 33:19511–19522, 2020.

[72] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[73] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *Journal of Artificial Intelligence Research (JAIR)*, 55:361–387, 2016.

[74] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[75] Jinzhu Zhou, Zhanbiao Yang, Yu Si, Le Kang, Haitao Li, Mei Wang, and Zhiya Zhang. A trust-region parallel Bayesian optimization method for simulation-driven antenna design. *IEEE Transactions on Antennas and Propagation*, 69(7): 3966–3981, 2020.

Paper II

High-dimensional Bayesian Optimization with Group Testing

Submitted to the Fourth International Conference on Automated Machine Learning

Erik Hellsten* Lund University

Carl Hvarfner* Lund University

Leonard Papenmeier* Lund University

Luigi Nardi Lund University

Abstract

Bayesian optimization (BO) is an effective method for optimizing expensive-to-evaluate black-box functions, but its susceptibility to the curse of dimensionality limits its applicability to high-dimensional problems. The assumption of an axis-aligned active subspace, where few dimensions have a significant impact on the objective, motivated several algorithms for high-dimensional BO. However, the validity of this assumption is rarely verified, and the assumption is rarely exploited to its full extent. We propose a group testing (GT) approach to identify active variables to facilitate efficient optimization in these domains. The proposed algorithm, Group Testing Bayesian Optimization (`GTBO`), first runs a testing phase where groups of variables are systematically selected and tested on whether they influence the

*Equal contribution.

objective. To that end, we extend the well-established GT theory to functions over continuous domains. In the second phase, GTBO guides optimization by placing more importance on the active dimensions. By leveraging the axis-aligned subspace assumption, GTBO is competitive against state-of-the-art methods on benchmarks satisfying the assumption of axis-aligned subspaces. Furthermore, for a given application, GTBO helps discover the active variables, enhancing practitioners' understanding and explainability of the problem.

1 Introduction

Noisy and expensive-to-evaluate black-box functions occur in many practical optimization tasks, including material design [55], hardware design [35, 19], hyperparameter tuning [28, 43, 12], and robotics [10, 6, 34]. BO is an established framework that allows optimization of such problems in a sample-efficient manner [47, 22]. Despite its many advantages, BO faces challenges with the curse of dimensionality, limiting its effectiveness in high-dimensional domains like robotics [10], joint neural architecture search and hyperparameter optimization [5], drug discovery [37], chemical engineering [9], and vehicle design [27].

In recent years, efficient approaches have been proposed to tackle the limitations of BO in high dimensions. Many of these approaches assume the existence of a low-dimensional *active subspace* of the input domain that has a significantly larger impact on the optimization objective than its complement [52, 30]. Often, the active subspace is further assumed to be axis-aligned [36, 20, 49, 39, 40], i.e., only a set of all considered variables impact the objective. The validity of this simplifying assumption does not always hold in real-world applications, and as a consequence, several approaches relax this assumption to reduce the risk of failure [20, 39, 40]. Those relaxations, however, come at a cost: firstly, they negatively affect sample efficiency for problems with axis-aligned subspaces, and secondly, they dilute the insights into which variables are relevant to the application at hand. Instead, we aim to leverage those assumptions more strongly, yielding better performance and stronger insights when they hold.

Knowing the active dimensions of a problem yields additional insight into the application, informing the user which problem parameters deserve more attention. When the active subspace is axis-aligned, finding the active dimensions can be framed as a feature selection problem. A straightforward approach is first to learn the active dimensions using a dedicated feature selection approach and subsequently optimize over the learned subspace. We propose to initially find the active dimensions using an information-theoretic approach built around the well-established theory of group

testing [18]. Group testing is the problem of finding several active elements within a larger set by iteratively testing groups of elements. We develop the theory needed to transition noisy GT, which otherwise only allows binary observations, to support evaluations of continuous black-box functions. This enables GT in BO and other applications, such as feature selection for regression problems. The contributions of this work are:

1. We extend the theory of group testing to feature importance analysis in a continuous setting tailored towards Gaussian process (GP) modeling.
2. We introduce Group Testing Bayesian Optimization (`GTBO`), a BO method that, based on the assumption of axis-aligned active subspaces, leverages the activeness information obtained from the preceding GT phase to guide the optimization.
3. We demonstrate that `GTBO` is competitive against state-of-the-art high-dimensional methods and reliably identifies active dimensions with high probability when the underlying assumptions hold.

2 Background

2.1 High-Dimensional Bayesian Optimization

We aim to find a minimizer $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ of the black-box function $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$, over the D-dimensional input space $\mathcal{X} = [0,1]^D$. We assume that f can only be observed point-wise and that the observation is perturbed by noise, $y(\boldsymbol{x}) = f(\boldsymbol{x}) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, where σ_n^2 is the *noise variance*. We further assume f to be expensive to evaluate, so the number of function evaluations is limited. In this work, we consider problems of high dimensionality D, where only d_e dimensions are *active*, and the other $D - d_e$ dimensions are *inactive*. Here, inactive means that the function value changes only marginally along the inactive dimensions compared to the active dimensions to the extent that satisfactory optimization performance can be obtained by considering the active dimensions alone. We refer the reader to Frazier [22] for an in-depth introduction to BO.

A popular remedy to the curse of dimensionality is trust regions (TRs) [41, 42], where, instead of reducing the dimensionality, one optimizes over a hyper-rectangle in input space. This makes the algorithm more local to counteract the over-exploration exhibited by traditional BO in high dimensions. One successful approach in this category is `TuRBO` [20]. Even though `TuRBO` operates in the full input dimensionality and might not scale to arbitrarily high-dimensional problems, it has shown remarkable performance in several applications. `Casmopolitan` [51] extends `TuRBO` to mixed and combinatorial spaces.

2.2 Low-Dimensional Subspace Bayesian Optimization

Using linear embeddings is a common approach when optimizing high-dimensional functions that contain a low-dimensional active subspace. REMBO [52] shows that a random embedded subspace with at least the same dimensionality as the active subspace is guaranteed to contain an optimum if the subspace is unbounded. However, BO usually requires a bounded search space, and REMBO suffers from projecting outside of this search space. HeSBO [36] uses a sparse projection matrix to avoid points outside the search space. BAxUS [39] and Bounce [40] use an HeSBO-like embedding [36] but allow the dimensionality of the target space to grow over time. This ensures that the optimum can eventually be found but leads to BAxUS and Bounce optimizing over high-dimensional spaces in later optimization stages. Alebo [29] presents another remedy to shortcomings in the search space design of REMBO. In particular, bounds from the original space are projected into the embedded space, and the kernel in the embedded space is adjusted to preserve distances from the original space. Notably, methods that rely on random embeddings require the user to provide a guess on the effective dimensionality of the problem, which might be challenging for real-world applications.

Axis-aligned active subspaces. One common assumption to tackle high-dimensional problems is that the active subspace is axis-aligned, i.e., a subspace that can be obtained by removing the inactive dimensions. This is equivalent to the assumption of active and inactive dimensions. SAASBO [20] sets up on this assumption by adding a strong sparsity-inducing prior to the hyperparameters of the GP model, prioritizing fewer active dimensions unless the data strongly suggests otherwise. VS-BO [48] actively identifies relevant variables in a problem, similar to our approach. However, it relies on a heuristic tied to a specific surrogate model (GP). It lacks a clear-cut decision on variable relevance due to ongoing variable importance estimation during optimization. MCTS-VS [49] uses a Monte Carlo tree search to select the active dimensions dynamically. It relies on randomly chosen sets of active and inactive dimensions, making finding the "correct" active dimensions difficult if the number of active dimensions is high. Our work also leverages the axis-aligned assumption, but it differs in how we identify the active dimensions.

Active subspace learning. In this paper, we resolve to a more direct approach, where we learn the active subspace explicitly. This is frequently denoted by *active subspace learning*. A common approach is to divide the optimization into two phases. The first phase involves selecting points and analyzing the structure to find the subspace. An optimization phase then follows on the subspace that was identified. The initial

phase can also be used alone to gain insights into the problem. One of the more straightforward approaches is to look for linear trends using methods such as *principal component analysis* [50] or *partial least squares* [8]. Djolonga et al. [17] use low-rank matrix recovery with directional derivatives with finite differences to find the active subspace. If gradients are available, the active subspace is spanned by the eigenvectors of the matrix $C \coloneqq \int_{\mathcal{X}} \nabla f(x)(\nabla f(x))^T dx$ with non-zero eigenvalues. This is used by Constantine et al. [14] and Wycoff et al. [54] to show that C can be estimated in closed form for GP regression. We refer to the survey by Binois and Wycoff [7] for a more in-depth introduction to active subspace learning. Notably, large parts of the active subspace learning literature yield non-axis-aligned subspaces. This is disadvantageous for problems with axis-aligned subspaces, as the information about the active dimensions is diluted.

2.3 Group Testing

Group testing (GT, Aldridge et al. [1]) is a methodology for identifying elements with some low-probability characteristic of interest by jointly evaluating groups of elements. GT was initially developed to test for infectious diseases in larger populations but has later been applied in quality control [15], communications [53], molecular biology [4, 38], pattern matching [33, 13], and machine learning [56].

Group testing can be subdivided into two paradigms: *adaptive* and *non-adaptive*. In adaptive GT, tests are conducted sequentially, and previous results can influence the selection of subsequent groups, whereas, in the non-adaptive setting, the complete testing strategy is provided up-front. A second distinction is whether test results are perturbed by evaluation noise. In the noisy setting, there is a risk that testing a group with active elements would show a negative outcome and vice versa. Our method presented in Section 3 can be considered an adaptation of noisy adaptive GT [45].

Cuturi et al. [15] present a *Bayesian Sequential Experimental Design* approach for binary outcomes, which at each iteration selects groups that maximize one of two criteria: the first one is the mutual information between the elements' probability of being active, $\boldsymbol{\xi}$, in the selected group and the observation. The second is the area under the marginal encoder's curve (AUC). As the distribution over the active group $p(\boldsymbol{\xi})$ is a 2^n-dimensional vector, it quickly becomes impractical to store and update. Consequently, they propose using a sequential Monte Carlo (SMC) sampler [16], representing the posterior probabilities by several weighted particles.

3 Group Testing for Bayesian Optimization

Our proposed method, `GTBO`, leverages the assumption of axis-aligned active subspaces by explicitly identifying the active dimensions. This gives the user additional insight into the problem and improves sample efficiency by focusing the optimization on the active dimensions. This section describes how we adapt the GT methodology to find active dimensions in as few evaluations as possible. Subsequently, we use the information to set strong priors for the GP length scales, providing the surrogate model with the knowledge about which features are active.

Noisy adaptive group testing. The underlying assumption is that a population of n elements exists, each of which either possesses or lacks a specific characteristic. We refer to the subset of elements with this characteristic as the active group, considering the elements belonging to this group as active. We let the random variable (RV) ξ_i denote whether the element i is active ($\xi_i = 1$), or inactive ($\xi_i = 0$), similar to Cuturi et al. [15] who studied binary outcomes. The state of the whole population can be written as the random vector $\boldsymbol{\xi} = \{\xi_1, \dots, \xi_n\} \in \{0, 1\}^n$.

We aim to uncover each element's activeness by performing repeated group tests. We write $\boldsymbol{g}$ as a binary vector $\boldsymbol{g} = \{g_1, \dots, g_n\} \in \{0, 1\}^n$, such that $g_i = 1$ signifies that element i belongs to the group. In noisy GT, the outcome of testing a group is a random event described by the RV $A(\boldsymbol{g}, \boldsymbol{\xi}) \in \{0, 1\}$. A common assumption is that the probability distribution of $A(\boldsymbol{g}, \boldsymbol{\xi})$ only depends on whether group $\boldsymbol{g}$ contains any active elements, i.e., $\boldsymbol{g}^\intercal \boldsymbol{\xi} \geq 1$. In this case, one can define the sensitivity $p(A(\boldsymbol{g}, \boldsymbol{\xi}) = 1 \mid \boldsymbol{g}^\intercal \boldsymbol{\xi} \geq 1)$ and specificity $p(A(\boldsymbol{g}, \boldsymbol{\xi}) = 0 \mid \boldsymbol{g}^\intercal \boldsymbol{\xi} = 0)$ of the test setup.

As we assume the black-box function f to be expensive to evaluate, we select groups $\boldsymbol{g}_t$ to learn as much as possible about the distribution $\boldsymbol{\xi}$ while limiting the number of iterations to $t = 1 \dots T$, which subsequently limits the number of function evaluations. We note that the group $\boldsymbol{g}_i$ is not a RV, but is selected as part of GT iteration i.

We can identify the active variables by modifying only a few variables in the search space and observing how the objective changes. Intuitively, if the function value remains approximately constant after perturbing a subset of variables from the default configuration, it suggests that these variables are inactive. On the contrary, if a specific dimension i is included in multiple subsets and the output changes significantly upon perturbation of those subsets, this suggests that dimension i is highly likely to be active.

Unlike in the traditional GT problem, where outcomes are binary, we work with continuous, real-valued function observations. To evaluate how a group of variables affects the objective function, we first evaluate a *default* configuration in the center of the search space, $\boldsymbol{x}_{\text{def}}$, and then vary the variables in the group and study the difference. We use the group notation $\boldsymbol{g}_t \in \{0,1\}^D$ as a binary indicator denoting which variables we change in iteration t. Similarly, we reuse the notation that the RV $\boldsymbol{\xi}$ denotes the active dimensions, and the true state is denoted by $\boldsymbol{\xi}^*$.

The new configuration to evaluate is selected as

$$\boldsymbol{x}_t = \boldsymbol{x}_{\text{def}} \oplus (\boldsymbol{g}_t \otimes \boldsymbol{u}_t), \tag{2.1}$$

where $\boldsymbol{u}_t \in [0,1]^D$ is drawn from $\mathcal{U}(\mathbf{0},\mathbf{1})$ until each active dimension has a distance of at least 0.4 to $\boldsymbol{x}_{\text{def}}$, $\oplus$ is element-wise addition, and $\otimes$ is element-wise multiplication. Note that a point $\boldsymbol{x}_t$ is always associated with a group $\boldsymbol{g}_t$ that determines along which dimensions $\boldsymbol{x}_t$ differs from the default configuration. For the newly obtained configuration $\boldsymbol{x}_t$, we must assess whether $|f(\boldsymbol{x_t}) - f(\boldsymbol{x}_{\text{def}})| \gg 0$, which would indicate that the group $\boldsymbol{g}_t$ contains active dimensions, i.e., $\boldsymbol{g}_t^\intercal \boldsymbol{\xi}^* \geq 1$. However, as we generally do not have access to the true values $f(\boldsymbol{x}_{\text{def}})$ or $f(\boldsymbol{x}_t)$ due to observation noise, we use an estimate $\hat{f}(\boldsymbol{x})$.

Since f can only be observed with Gaussian noise of unknown variance σ_n^2, there is always a non-zero probability that a high difference in function value occurs between $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{def}}$ even if group $\boldsymbol{g}$ contains no active dimensions. Therefore, we take a probabilistic approach, which relies on two key assumptions:

1. $Z_t \coloneqq \hat{f}(\boldsymbol{x}_t) - \hat{f}(\boldsymbol{x}_{\text{def}}) \sim \mathcal{N}(0, \sigma_n^2)$ if $\boldsymbol{g}_t^\intercal \boldsymbol{\xi} = 0$, i.e., function values follow the noise distribution if the group $\boldsymbol{g}_t$ contains no active dimensions.
2. $Z_t \coloneqq \hat{f}(\boldsymbol{x}_t) - \hat{f}(\boldsymbol{x}_{\text{def}}) \sim \mathcal{N}(0, \sigma^2)$ if $\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1$, i.e., function values are drawn from a zero-mean Gaussian distribution with the function-value variance if the group $\boldsymbol{g}_t$ contains active dimensions.

The first assumption follows from the assumption of Gaussian observation noise and an axis-aligned active subspace. The second assumption follows from a GP prior assumption on f, under which $\hat{f}(x_t)$ is normally distributed. As we are only interested in the change from $f(\boldsymbol{x}_{\text{def}})$, we assume this distribution to have mean zero.

We estimate the noise variance, σ_n^2, and function-value variance, σ^2, based on an assumption on the maximum number of active variables. First we evaluate f at the default configuration $\boldsymbol{x}_{\text{def}}$. We then split the dimensions into several roughly equally sized bins. For each bin, we evaluate f on the default configuration perturbed along the direction of all variables in that bin and compare the result with the default value.

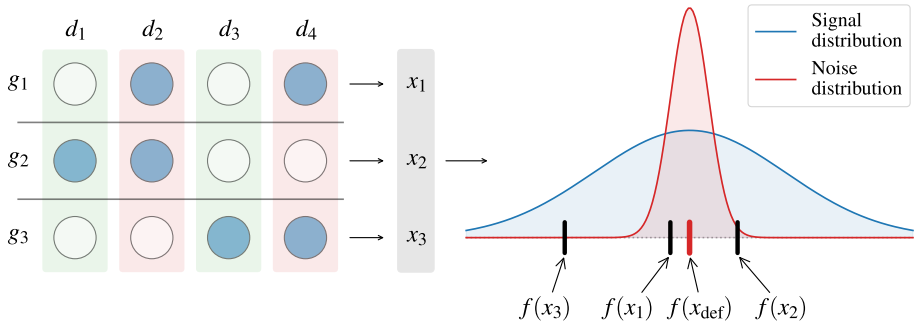


Figure 2.1: `GTBO` assumes an axis-aligned subspace. A point x_1 that only varies along inactive dimensions (d_2 and d_4) obtains a similar function value as the default configuration (x_{def}). Points x_2 and x_3 that vary along active dimensions (d_1 and d_3) have a higher likelihood under the signal distribution than under the noise distribution.

We then estimate the function variance as the empirical variance among the `max_act` largest such differences and the noise variance as the empirical variance among the rest. Here, `max_act` represents the assumed maximum number of active dimensions. If the assumption holds, there can be no active dimensions in the noise estimate, which is more sensitive to outliers. It must be an upper bound, as the method is more sensitive to estimating the noise from active dimensions than vice versa. An example of this is shown in Appendix D.

Under Assumptions 1 and 2, the distribution of Z_t depends only on whether $\boldsymbol{g}_t$ contains active variables. Given the probability distribution over population states $p(\boldsymbol{\xi})$, the probability that $\boldsymbol{g}_t$ contains any active elements is

$$p(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1) = \sum_{\boldsymbol{\xi} \in \{0,1\}^D} \delta_{\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1} p(\boldsymbol{\xi}). \tag{2.2}$$

We exemplify this in Fig. 2.1. Here, three groups are tested sequentially, out of which the second and third contain active variables. The three corresponding configurations, x_1, x_2, and x_3, give three function values shown on the right-hand side. As observing $f(x_1)$ is more likely under the noise distribution, g_1 has a higher probability of being inactive. Similarly, as $f(x_2)$ and $f(x_3)$ are more likely to be observed under the signal distribution, g_2 and g_3 are more likely to be active.

Estimating the group activeness probability. Equation (2.2) requires summing over 2^D possible activity states, which, for higher-dimensional functions, becomes prohibitively expensive. Instead, we use an SMC sampler with M particles $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}$ and particle weights $\{\omega_1, \ldots, \omega_M\}$. Each particle $\boldsymbol{\xi}_k \in \{0,1\}^D$

represents a possible ground truth. We follow the approach presented in Cuturi et al. [15] and use a modified Gibbs kernel for discrete spaces [31]. We then estimate the probability $p(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1)$ of a group $\boldsymbol{g}_t$ to be active by

$$\hat{p}(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1) = \sum_{k=1}^{M} \omega_k \delta_{\boldsymbol{g}_t^\intercal \boldsymbol{\xi}_k \geq 1}. \tag{2.3}$$

Choice of new groups. We choose new groups to maximize the information obtained about $\boldsymbol{\xi}$ when observing Z_t. This can be achieved by maximizing their mutual information (MI). Under Assumptions 1 and 2, we can write the MI as

$$\begin{aligned}
I(\boldsymbol{\xi}, Z_t) &= H(Z_t) - H(Z_t|\boldsymbol{\xi}) && (2.4)\\
&= H(Z_t) - \sum_{\bar{\boldsymbol{\xi}} \in \{0,1\}^D} p(\bar{\boldsymbol{\xi}}) H(Z_t|\boldsymbol{\xi} = \bar{\boldsymbol{\xi}}) && (2.5)\\
&= H(Z_t) - [p(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1) H(Z_t|\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1) \\
&\quad + p(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} = 0) H(Z_t|\boldsymbol{g}_t^\intercal \boldsymbol{\xi} = 0)] && (2.6)\\
&= H(Z_t) - \frac{1}{2}[p(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} = 0) \log(2\sigma_n^2 \pi e) \\
&\quad + p(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1) \log(2\sigma^2 \pi e)]. && (2.7)
\end{aligned}$$

Since Z_t is modeled as a Gaussian mixture model (GMM), its entropy $H(Z_t)$ has no known closed-form expression [25], but can be approximated using Monte Carlo:

$$H(Z_t) = \mathbb{E}[-\log p(Z_t)] \approx -\frac{1}{N} \sum_{i=1}^{N} \log p(z_t^i), \tag{2.8}$$

and $z_t^i \sim \mathcal{N}(0, \sigma^2)$ with probability $\hat{p}(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} \geq 1)$ and $z_t^i \sim \mathcal{N}(0, \sigma_n^2)$ with probability $\hat{p}(\boldsymbol{g}_t^\intercal \boldsymbol{\xi} = 0)$.

Maximizing the mutual information. `GTBO` optimizes the MI using a multi-start forward-backward algorithm [44]. First, several initial groups are generated by sampling from the prior and the posterior over $\boldsymbol{\xi}$. Then, elements are greedily added for each group in a *forward phase* and removed in a subsequent *backward phase*. In the forward phase, we incrementally include the element that results in the greatest MI increase. Conversely, in the backward phase, we eliminate the element that contributes the most to MI increase. Each phase is continued until no further elements are added or removed from the group. Finally, the group with the largest MI is returned.

Batch evaluations. If the black-box function can be run in parallel, we greedily select additional groups by running the forward-backward algorithm again, excluding already selected groups. For high-dimensional problems there are frequently several distinct groups which each yields close to optimal MI. We continue adding groups to evaluate until we have reached a user-specified upper limit or until the MI of new groups drops below a threshold.

Updating the activeness probability. Once we have selected a new group $\boldsymbol{g}_t$ and observed the corresponding function value z_t, we update our estimate of $\hat{p}(\boldsymbol{\xi}_k)$ for each particle k:

$$\hat{p}^t(\boldsymbol{\xi}_k) \propto \hat{p}^{t-1}(\boldsymbol{\xi}_k) p(z_t|\boldsymbol{\xi}_k) \tag{2.9}$$

$$\propto \hat{p}^{t-1}(\boldsymbol{\xi}_k) \begin{cases} p(z_t|\boldsymbol{g}_t^\intercal \boldsymbol{\xi}_k \geq 1) & \text{if } \boldsymbol{g}_t^\intercal \boldsymbol{\xi}_k \geq 1 \\ p(z_t|\boldsymbol{g}_t^\intercal \boldsymbol{\xi}_k = 0) & \text{if } \boldsymbol{g}_t^\intercal \boldsymbol{\xi}_k = 0, \end{cases} \tag{2.10}$$

where $p(z_t|\boldsymbol{g}_t^\intercal \boldsymbol{\xi}_k = 0)$ and $p(z_t|\boldsymbol{g}_t^\intercal \boldsymbol{\xi}_k \geq 1)$ are Gaussian likelihoods. Assuming that the probabilities of dimensions to be active are independent, the prior probability is given by $\hat{p}^0(\boldsymbol{\xi}_k) = \prod_{i=1}^D q_i^{\boldsymbol{\xi}_{k,i}} (1-q_i)^{1-\boldsymbol{\xi}_{k,i}}$ where q_i is the prior probability for the i-th dimension to be active. As we represent the probability distribution $\hat{p}^0(\boldsymbol{\xi})$ by a point cloud, any prior distribution can be used to insert prior knowledge. We use the same SMC sampler as Cuturi et al. [15].

The GTBO algorithm. With the individual parts defined, we present the complete procedure for `GTBO`. `GTBO` iteratively selects and evaluates groups for T iterations or until convergence. We consider it to have converged when the posterior marginal probability for each variable $\hat{p}^t(\xi_i)$ lies in $[0, C_{\text{lower}}] \cup [C_{\text{upper}}, 1]$, for some convergence thresholds C_{lower} and C_{upper}. More details on the GT phase can be found in Algorithm 2.1 in Appendix A.

Subsequently, their marginal posterior distribution decides which variables are selected to be active. A variable i is considered active if its marginal is larger than some threshold, $\hat{p}_i^t(\boldsymbol{\xi}) \geq \eta$. Once we have deduced which variables are active, we perform BO using the remaining sample budget. To strongly focus on the active subspace, we use short lengthscale priors for the active variables and long lengthscale priors for the inactive variables. We use a GP with a Matérn$-5/2$ kernel as the surrogate model and `qLogNoisyExpectedImprovement` [3] as the acquisition function. The BO phase is initialized with data sampled during the feature selection phase. Several points are sampled throughout the GT phase that only differ marginally in the active subspace. Such duplicates are removed to facilitate the fitting of the GP.

4 Computational Experiments

In this section, we showcase the performance of the proposed methodology, both for finding the relevant dimensions and for the subsequent optimization. We compare state-of-the-art frameworks for high-dimensional BO on several synthetic and real-life benchmarks. GTBO outperforms previous approaches on the tested real-world and synthetic benchmarks. In Section 4.2, we study the sensitivity of GTBO to external traits of the optimization problem, such as noise-to-signal ratio and the number of active dimensions. The efficiency of the GT phase is tested against other feature analysis algorithms in Appendix B and the GTBO wallclock times are presented in Table 5.6 in Appendix E. The code for GTBO is available at `https://github.com/gtboauthors/gtbo`.

4.1 Experimental Setup

We test GTBO on four synthetic benchmark functions, Branin2, Levy in 4 dimensions, Hartmann6, and Griewank in 8 dimensions, which we extend with inactive "dummy" dimensions [52, 20, 39] as well as two real-world benchmarks: the 124D soft-constraint version of the Mopta08 benchmark [20], and the 180D LassoDNA benchmark [46]. We add significant observation noise for the synthetic benchmarks, but the inactive dimensions are truly inactive. In contrast, the real-world benchmarks do not exhibit observation noise, but all dimensions have at least a marginal impact on the objective function. Note that the noisy synthetic benchmarks are considerably more challenging for GTBO than their noiseless counterparts.

Since the search space center is a decent solution for LassoDNA, GTBO chooses a default configuration for each GT repetition uniformly at random. To not give BAxUS a similar advantage, we subtract a random offset from the search space bounds, which we add again before evaluating the function. This ensures that BAxUS cannot always represent the near-optimal origin.

To evaluate the BO performance, we benchmark against TuRBO [21] with one and five trust regions, SAASBO [20], CMA-ES [23], HeSBO [36], and BAxUS [39] using the implementations and settings provided by the authors, unless stated otherwise. We compare against random search, i.e., choose points in the search space uniformly at random.

We use the pycma implementation for CMA-ES [24] and the Ax implementation for Alebo [2]. To show the effect of different choices of the target dimensionality d, we run Alebo with $d = 10$ and $d = 20$. We observed that Alebo and SAASBO are constrained by their high runtime and memory consumption. The available

hardware allowed up to 100 evaluations for SAASBO and 300 evaluations for Alebo for each run. Larger sampling budgets or higher target dimensions for Alebo resulted in out-of-memory errors. We note that limited scalability was expected for these two methods, whereas the other methods scaled to considerably larger budgets, as required for scalable BO. We initialize each optimizer with ten initial samples and BAxUS with $b = 3$ and $m_D = 1000$ and run ten repeated trials. Plots show the mean logarithmic regret for synthetic benchmarks and the mean function value for real-world benchmarks. The shaded regions indicate one standard error.

Unless stated otherwise, we run GTBO with 10 000 particles for the SMC sampler, the prior probability of being active, $q = 0.05$, and 3 initial groups for the forward-backward algorithm. When estimating the function signal and noise variance, we set the assumed maximum number of active dimensions, max_act, to $\sqrt{D}$. The threshold to be considered active after the GT phase, η, is set to 0.5, and the lower and upper convergence thresholds, C_{lower} and C_{upper}, are $5 \cdot 10^{-3}$ and 0.9. We run all experiments with a log-normal $\mathcal{LN}(7, 1)$ length scale prior to the inactive dimensions. If a benchmark is known to have strictly active and inactive parameters, this prior can be chosen more aggressively to "switch off" the inactive dimensions. We use a $\mathcal{LN}(0, 1)$ prior for the active variables, resulting in significantly shorter length scales. In the GT phase, we use batch evaluation with a maximum of 5 groups in each batch and a maximum MI drop of 1%. Note that we still count the number of evaluations, not the number of batches, towards the budget. An analysis of the impact of some core hyperparameters is presented in Appendix D. The experiments are run on Intel Xeon Gold 6130 machines using two cores.

4.2 Performance of the Group Testing

Before studying GTBO's overall optimization performance in high-dimensional settings, we analyze the performance of the GT procedure. In Fig. 2.2, we show the evolution of the average marginal probability of being active over the iterations for the different dimensions. The truly active dimensions are plotted in green, and the inactive ones are in blue squares. For all the problems, GTBO correctly classifies all active dimensions during all runs within 39–112 iterations. Across ten runs, GTBO misclassifies 6 out of 1180 inactive variables to be active once each, for a false positive rate of 0.05%. The change in the number of active variables over the iterations is shown in Appendix C.

Sensitivity analysis. We explore the sensitivity of GTBO to the output noise and problem size by evaluating it on the Levy4 synthetic benchmark extended to 100 dimensions, with a noise standard deviation of 0.1, and varying the properties of

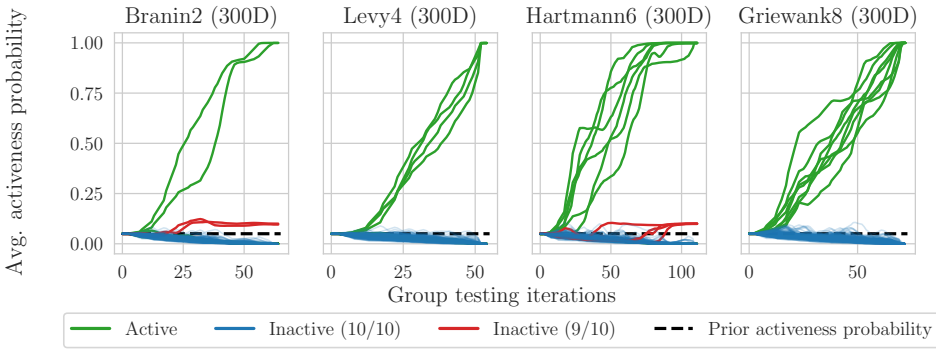


Figure 2.2: Evolution of the average marginal probability of being active across ten repetitions. Each line represents one dimension; active dimensions are colored green, and inactive dimensions are blue. In the few cases where `GTBO` finds inactive variables to be active, the lines are emphasized in red. The last iteration marks the end of the longest GT phase across all runs. All active dimensions are identified in all runs. 6 out of 1180 inactive dimensions are incorrectly classified as active once in ten runs across the benchmarks, implying a false positive rate of slightly above 0.05%.

interest. In Fig. 2.3, we show how the percentage of correctly predicted variables evolves with the number of tests t for different functional properties. Correctly classified is defined here as having a probability of less than 1% if inactive or above 90% if active. `GTBO` shows to be robust to lower noise but suffers from very high noise levels. As expected, higher function dimensionality and number of active dimensions increases the time until convergence. Note that the signal and noise variance estimates build on the assumption that there are a maximum of $\sqrt{D}$ active dimensions, which does not hold with 32 active dimensions.

4.3 Optimization of Real-World and Synthetic Benchmarks

We show that identifying the relevant variables can drastically improve optimization performance. Fig. 2.5 shows the performance of `GTBO` and competitors on the real-world benchmarks, Fig. 2.4 on the synthetic benchmarks. The results show the incumbent function value for each method, averaged over ten repeated trials. We plot the true average incumbent function values on the noisy benchmarks without observation noise.

Note that `Griewank` has its optimum in the center of the search space. To not gain an unfair advantage, we run `GTBO` with a non-standard default away from the optimum. However, the optimum being in the center means that all linear projections will contain the optimum, which boosts the projection-based methods `Alebo` and `BAxUS` as it allows them to represent the optimum regardless of the embedding choice.

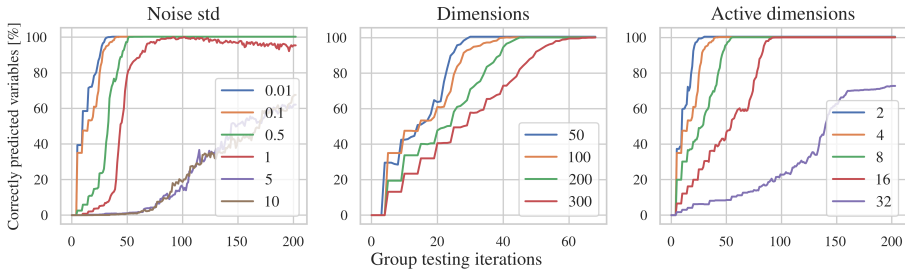


Figure 2.3: Sensitivity analysis for GTBO. The average percentage of correctly classified variables is displayed for increasing GT iterations. The percentage is ablated for (left) various levels of output noise, (middle) number of total dimensions, and (right) number of effective dimensions. Each legend shows the configurations of the respective parameter.

Figure 2.5 shows GTBO's performance on the 124D Mopta08 and 180D LassoDNA benchmarks. On Mopta08, GTBO performs like a random search during the GT phase but quickly outperforms the other methods once the BO phase begins. This behavior suggests that several dimensions of the Mopta08 benchmark have negligible impact on the optimization objective and highlights GTBO's ability to leverage the activeness information for benchmarks with inactive dimensions efficiently.

On LassoDNA, GTBO again reaches similar performance levels as the random search and improves significantly when the BO phase starts, outperforming CMA-ES after 1 000 function evaluations. However, the overall performance is not on par with other BO methods. Perhaps the fact that both variable selection methods, MCTS-VS and GTBO, fail to reach the same performance levels as TuRBO and BAxUS suggests that most dimensions in LassoDNA are active. While this benchmark is suspected to violate our axis-aligned assumptions, GTBO still shows reasonable performance and outperforms MCTS-VS by a large margin.

Overall, GTBO first identifies relevant variables, followed by a sharp drop when the optimization phase starts, indicating that knowing the active dimensions drastically speeds up optimization. In the real-world Mopta08 application where the dimensions detected as inactive have negligible impact on the objective, GTBO outperforms state-of-the-art methods despite the delayed onset of the BO phase. However, GTBO can suffer in cases where the axis-alignment assumption does not hold, as shown by the LassoDNA results.

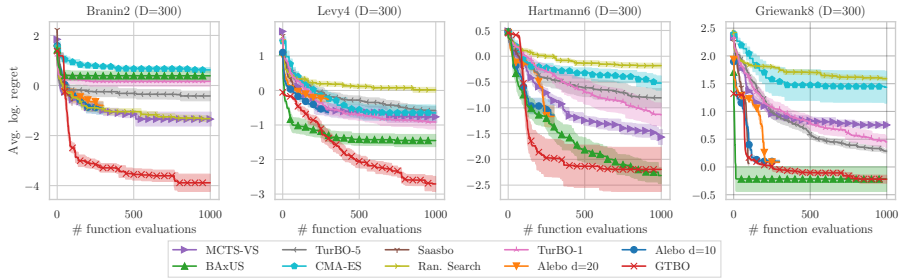


Figure 2.4: `GTBO` finds active dimensions and optimizes efficiently on synthetic noisy benchmarks (`Branin2`, `Levy4`, `Hartmann6`, and `Griewank8`).

5 Discussion

Optimizing expensive-to-evaluate high-dimensional black-box functions is a challenge for applications in industry and academia. We propose `GTBO`, a novel algorithm that focuses the Bayesian optimization on variables found relevant in a preceding group testing phase. `GTBO` explicitly exploits the structure of a sparse axis-aligned subspace to reduce the complexity of an application in high dimensions and is the first method to adapt group testing, in which it aims to find infected individuals by conducting pooled tests, to Bayesian optimization. It differs from `SAASBO` [20] in that it yields a clear-cut decision on which variables are active or inactive and from `Alebo` [30] or `BAxUS` [39] in that it does not rely on random projections to identify relevant variables. Similarly to `MCTS-VS` [49], our method explicitly identifies the set of relevant variables; however, `GTBO` is the first method to use a more principled approach to learn them by employing Group Testing principles and theory.

`GTBO` quickly detects active and inactive variables and shows robust optimization performance in synthetic and real-world settings. Furthermore, the GT phase yields a set of relevant dimensions, which allows users to learn something fundamental about their application. For example, on the `Mopta08` benchmark, the user learns which shape parameters minimize vehicle mass under some constraints [27] `GTBO` robustly uses the activeness information so that it can still optimize efficiently even if the inactive dimensions have a marginal impact on the objective function.

In future work, we will fuse the GT and BO phases so that the non-default values in a group test guide the optimization. This will further increase the sample efficiency of `GTBO`.

For problems where inactive dimensions do not affect the objective function, `GTBO` can be used to identify the active dimensions and then only optimize on those. This further improves sample efficiency and is advantageous for applications where

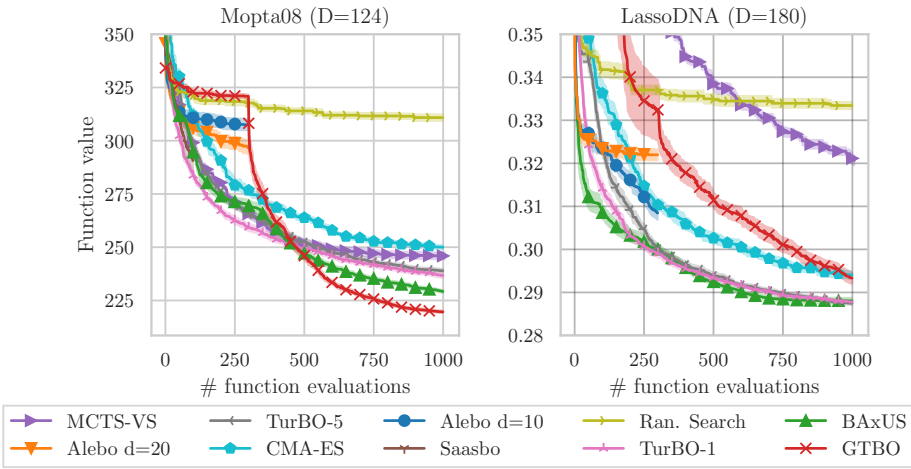


Figure 2.5: `GTBO` outperforms competitors in real-world experiments. Notably, the performance on `Mopta08` increases significantly after the GT phase at iteration 300, suggesting that the dimensions found during the GT phase are highly relevant. For `LassoDNA`, the performance is worse for both `GTBO` and `MCTS-VS`, indicating that the assumption of an axis-aligned subspace is violated.

optimizing over a larger set of dimensions incurs additional costs [32].

Limitations. `GTBO` relies on the assumption that an application has several irrelevant parameters. If this assumption is unmet, the method might under-perform or waste a fraction of the evaluation budget to identify all variables as relevant. Furthermore, `GTBO` cannot exploit problems with a low-dimensional subspace that is not axis-aligned, and it is unclear how many tests are required for `GTBO`'s GT phase to converge as it is challenging to prove bounds in the non-asymptotic regime [15].

Broader Impact

This paper presents basic research with the goal of advancing the fields of Machine Learning and optimization. There are many potential societal impacts of our work, none of which should be specifically highlighted here.

A The GTBO Algorithm

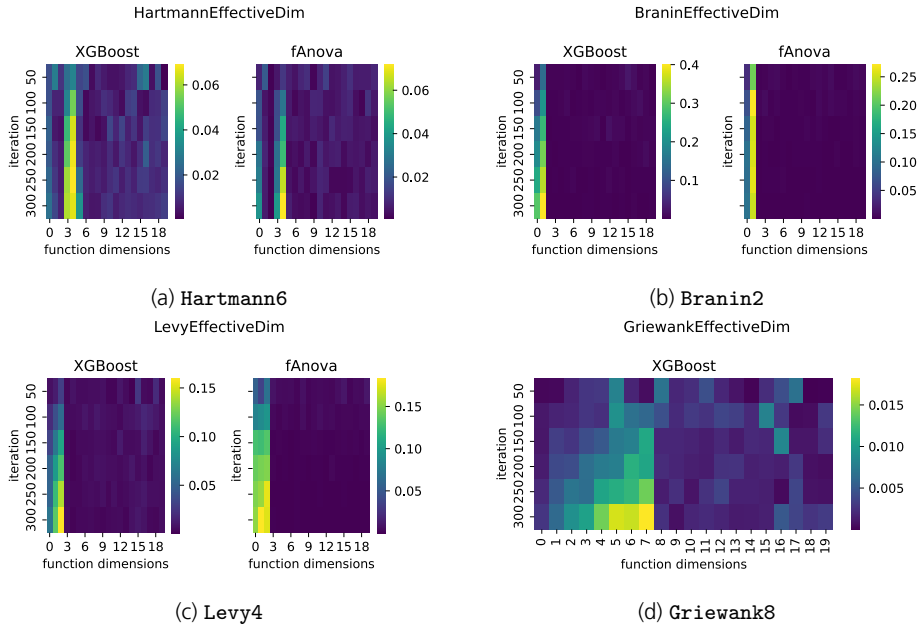
This section describes the GT phase of the GTBO algorithm in additional detail.

Algorithm 2.1 Group testing phase

Input: black-box function $f : \mathcal{X} \to \mathbb{R}$, number of default configuration evaluations n_{def}, number of group tests T, number of particles M, prior distribution $p^0(\boldsymbol{\xi})$

Output: estimate of active dimensions $\boldsymbol{\gamma}$, posterior distribution $\hat{p}^T$

for $j \in \{1 \dots n_{\text{def}}\}$ **do**
  $y_{\text{def}}^{(i)} = y(\boldsymbol{x}_{\text{def}})$
$\hat{f}(\boldsymbol{x}_{\text{def}}) \leftarrow \frac{1}{n_{\text{def}}} \sum_{i=1}^{n_{\text{def}}} y_{\text{def}}^{(i)}$
split the dimensions into $3\lfloor\sqrt{D}\rfloor$ bins B
for $j \in \{1 \dots |B|\}$ **do**
  $y_{\text{bin}}^{(j)} = y(\boldsymbol{x}_{\text{def}} \oplus \alpha b_j)$ ▷ a random perturbation along the dimensions in bin b_j
sort (ascending) $|y_{\text{bin}}^{(j)} - \hat{f}(\boldsymbol{x}_{\text{def}})|$
$\hat{\sigma}_n^2 \leftarrow \texttt{var}(|y_{\text{bin}}^{(1)} - \hat{f}(\boldsymbol{x}_{\text{def}})|, \dots, |y_{\text{bin}}^{(2\lfloor\sqrt{D}\rfloor)} - \hat{f}(\boldsymbol{x}_{\text{def}})|)$
$\hat{\sigma}^2 \leftarrow \texttt{var}(|y_{\text{bin}}^{(2\lfloor\sqrt{D}\rfloor+1)} - \hat{f}(\boldsymbol{x}_{\text{def}})|, \dots, |y_{\text{bin}}^{(3\lfloor\sqrt{D}\rfloor)} - \hat{f}(\boldsymbol{x}_{\text{def}})|)$
$\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_M \sim \hat{p}^0(\boldsymbol{\xi})$
$\omega_1, \dots, \omega_M \leftarrow \frac{1}{M}$ ▷ initial particle weights
for all $t \in \{1, \dots, T\}$ **do**
  $\boldsymbol{g}^* \leftarrow \texttt{maximize_mi}(\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_M)$ ▷ find a group that maximizes MI
  $\boldsymbol{x}_t \leftarrow$ create using Eq. (2.1) and $\boldsymbol{g}^*$
  $z_t \leftarrow f(\boldsymbol{x}_t) + \varepsilon - \hat{f}(\boldsymbol{x}_{\text{def}})$
  $(\boldsymbol{\xi}_i, \omega_i)_{i\in[M]} \leftarrow \texttt{resample}(z_t, (\boldsymbol{\xi}_i, \omega_i)_{i\in[M]})$
  $\hat{p}^t \leftarrow \texttt{marginal}((\boldsymbol{\xi}_i, \omega_i)_{i\in[M]})$ ▷ get marginals
$\boldsymbol{\gamma} \leftarrow (\delta_{\hat{p}_1^T(\boldsymbol{\xi})\geq\eta}, \dots, \delta_{\hat{p}_D^T(\boldsymbol{\xi})\geq\eta})$ ▷ check which dimensions are active



(a) `Hartmann6` (b) `Branin2`

(c) `Levy4` (d) `Griewank8`

Figure 2.6: `XGBoost` and `fAnova` feature importance analysis on the 100-dimensional version of different synthetic benchmarks, averaged over 20 repetitions. Only the first 20 dimensions are shown.

B Comparison with Feature Importance Algorithms

We compare the performance of the GT phase with the established feature importance analysis methods `XGBoost` [11] and `fAnova` [26]. Since `fAnova`'s stability degrades with increasing dimensionality, we run these methods on the 100-dimensional version of the synthetic benchmarks: `Branin2` (noise std 0.5), `Griewank8` (noise std 0.5), `Levy4` (noise std 0.1), and `Hartmann6` (noise std 0.01).

Figure 2.6a shows the results of `fAnova` and `XGBoost` on the 100-dimensional version of `Hartmann6` with added output noise. Per our results, both methods flag the third dimension as not more important than the added input dimensions (dimensions 7-100 with no impact on the function value). Additionally, `fAnova` seems to "switch off" the second dimension.

On `Branin2` (Fig. 2.6b), both methods detect the correct dimensions (the first and second dimensions). Furthermore, all other dimensions have zero importance, and the methods find the correct partitioning earlier than for `Hartmann6`. Similarly to `Hartmann6`, both methods fail to detect an active dimension (the fourth dimension).

On `Griewank8`, `fAnova` does not terminate gracefully. Therefore, we only discuss

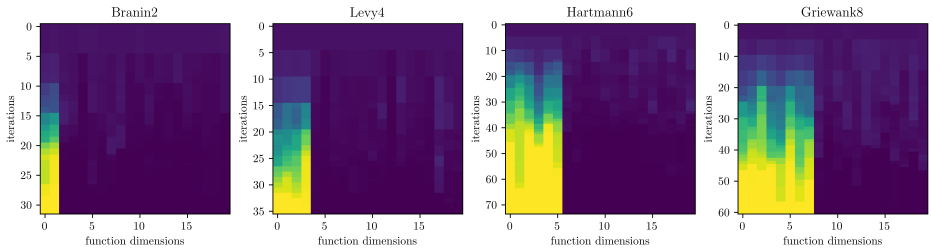


Figure 2.7: GTBO-marginals of the first 20 dimensions., averaged over ten repetitions. If the GT phase ends early, the last marginals are repeated to match the length of the longest GT phase.

XGBoost for Griewank8. After 300 iterations, XGBoost only detects six dimensions reliably as active. The other two dimensions are determined to be not more important than the added dimensions. The marginals found by GTBO are shown in Fig. 2.7. Compared to conventional feature importance analysis methods, GTBO detects all active dimensions with high probability.

C Number of Active Variables

Here, we show the average number of active dimensions throughout the GT phase. Given that the acceptance threshold of 0.5 is much higher than the initial probability of acceptance of 0.05, dimensions once considered active are rarely later considered inactive again, resulting in a close to monotonically increasing number of active dimensions.

D Hyperparameter Analysis

To measure the impact of different hyperparameters in GTBO, we run an analysis on Griewank8 in 100 dimensions. In Fig. 2.9, we focus on the maximum batch size, initial probability of being active, the assumed number of active dimensions for estimating the signal and noise ratio, and the number of particles in the SMC sampler. The iteration at which all 20 repetitions have converged for a single setting is marked with a circle. Increasing the maximum batch size does not greatly impact the GT performance. The initial probability is important, but both 0.05 and 0.10 perform well; the very small or large values perform worse. The assumed number of active dimensions, set to $\sqrt{D}$ in the paper, is highly important because it must be larger than the actual number of active dimensions. Otherwise, active dimensions will be used to estimate the noise, severely hampering the performance. We see this

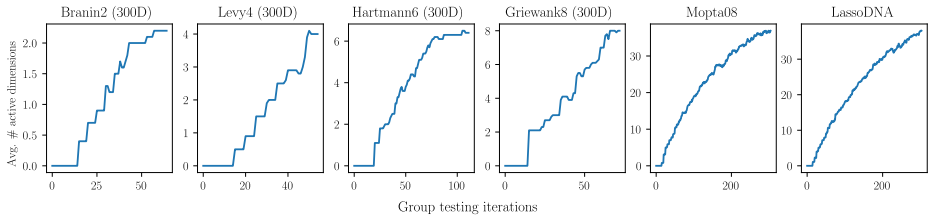


Figure 2.8: Evolution average number of active variables during the GT phase (10 repetitions). The synthetic benchmarks find the correct number of active variables, whereas the real-world benchmarks find a significantly higher number.

for the performance of the assumed active dimension of five, which is less than the true active dimensionality 8. Lastly, the number of particles in the SMC sampler is irrelevant for this benchmark, but it becomes more important for harder benchmarks.

We also study the impact of the parameters of the lengthscale LogNormal priors for inactive dimensions in Fig. 2.10. We see that the performance increases for longer lengthscales. For the synthetic benchmarks, the inactive dimensions have absolutely no impact on the objective function, and as such, it is expected that de-emphasizing the importance of those variables would be beneficial.

E Run Times

In this section, we show the average run of `GTBO`. Note that the SMC resampling is a part of the GT phase, and as such, the total algorithm time is the GT time plus the BO time. We do batch evaluations in the GT phase described in Section 3, with a maximum of five groups tested before resampling. This significantly reduces the SMC resample time.

Table 2.3: Average `GTBO` runtimes. Group testing time is on the same order of magnitude as the time allocated towards BO. For BO, the $\mathcal{O}(D^2)$ complexity of Quasi-Newton-based acquisition function optimization dominates the runtime.

Benchmark	GT time [h]	SMC resample time [h]	BO time [h]
`Branin2` (300D)	1.94	0.583	9.31
`Levy4` (300D)	2.33	0.603	10.8
`Hartmann6` (300D)	3.29	1.14	14.1
`Griewank8` (300D)	2.41	0.846	9.08
`Mopta08` (124D)	5.70	4.74	7.95
`LassoDNA` (180D)	8.92	7.06	10.2

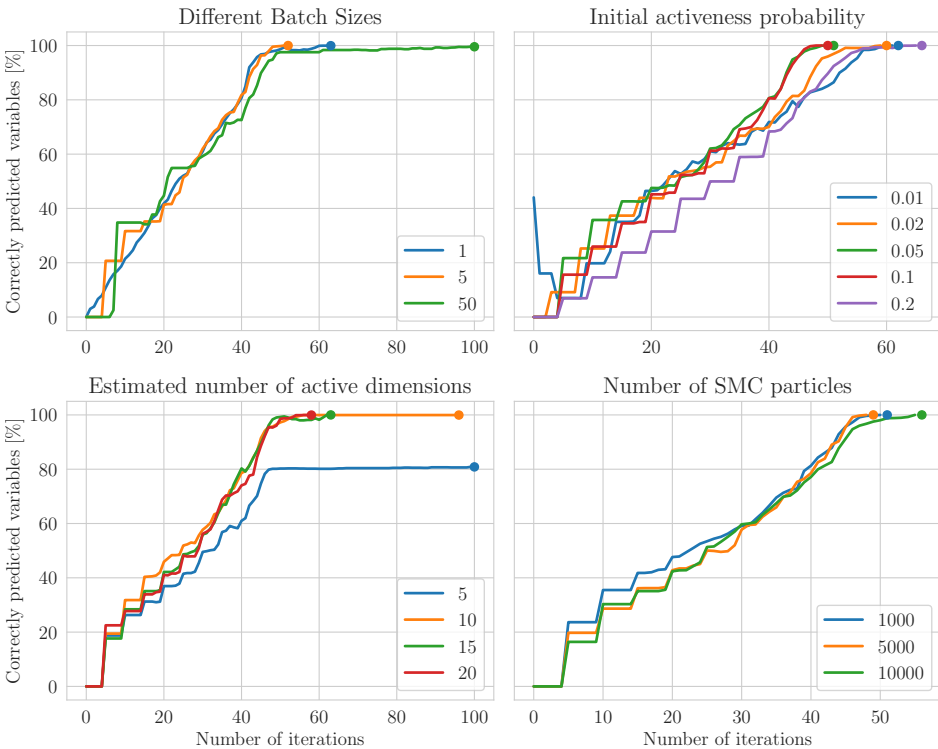


Figure 2.9: The percentage of correctly classified dimensions after an increasing number of iterations for different hyperparameter settings for Griewank8 (100D). The final dot on each line indicates that all runs have converged.

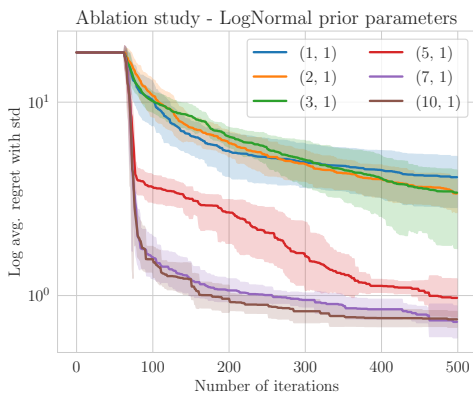


Figure 2.10: The log average regret of GTBO for different values of the LogNormal prior for Griewank8 (100D).

References

[1] Matthew Aldridge, Oliver Johnson, Jonathan Scarlett, et al. Group testing: an information theory perspective. *Foundations and Trends® in Communications and Information Theory*, 15(3-4):196–392, 2019.

[2] Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Benjamin Letham, Ashwin Murthy, and Shaun Singh. Ae: A domain-agnostic platform for adaptive experimentation. *Conference on Neural Information Processing Systems*, pages 1–8, 2018.

[3] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[4] DJ Balding, WJ Bruno, DC Torney, and E Knill. A comparative survey of non-adaptive pooling designs. *Genetic mapping and DNA sequencing*, pages 133–154. Springer, 1996.

[5] Archit Bansal, Danny Stoll, Maciej Janowski, Arber Zela, and Frank Hutter. JAHS-Bench-201: A Foundation For Research On Joint Architecture And Hyperparameter Search. *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[6] Felix Berkenkamp, Andreas Krause, and Angela Schoellig. Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics. *Machine Learning*, 06 2021. doi: 10.1007/s10994-021-06019-1.

[7] Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

[8] Mohamed Amine Bouhlel, Nathalie Bartoli, Abdelkader Otsmane, and Joseph Morlier. Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53:935–952, 2016.

[9] Benjamin Burger, Phillip M Maffettone, Vladimir V Gusev, Catherine M Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M Alston, Buyi Li, Rob Clowes, et al. A mobile robotic chemist. *Nature*, 583(7815):237–241, 2020.

[10] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian gait optimization for bipedal locomotion. *Learning and*

Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers 8, pages 274–290. Springer, 2014.

[11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[12] Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian Optimization in AlphaGo. *CoRR*, abs/1812.06855, 2018.

[13] Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. Pattern matching with don't cares and few errors. *Journal of Computer and System Sciences*, 76(2):115–124, 2010.

[14] Paul G Constantine, Armin Eftekhari, and Michael B Wakin. Computing active subspaces efficiently with gradient sketching. *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 353–356. IEEE, 2015.

[15] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Noisy Adaptive Group Testing using Bayesian Sequential Experimental Design. *CoRR*, abs/2004.12508, 2020.

[16] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[17] Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional gaussian process bandits. *Advances in neural information processing systems*, 26, 2013.

[18] Robert Dorfman. The detection of defective members of large populations. *The Annals of mathematical statistics*, 14(4):436–440, 1943.

[19] Adel Ejjeh, Leon Medvinsky, Aaron Councilman, Hemang Nehra, Suraj Sharma, Vikram Adve, Luigi Nardi, Eriko Nurvitadhi, and Rob A. Rutenbar. HPVM2FPGA: Enabling True Hardware-Agnostic FPGA Programming. *Proceedings of the 33rd IEEE International Conference on Application-specific Systems, Architectures, and Processors*, 2022.

[20] David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

[21] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5496–5507, 2019.

[22] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[23] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

[24] Nikolaus Hansen, yoshihikoueno, ARF1, Kento Nozawa, Luca Rolshoven, Matthew Chan, Youhei Akimoto, brieglhostis, and Dimo Brockhoff. CMA-ES/pycma: r3.2.2, March 2022.

[25] Marco F Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D Hanebeck. On entropy approximation for Gaussian mixture random vectors. *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 181–188. IEEE, 2008.

[26] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. *International conference on machine learning*, pages 754–762. PMLR, 2014.

[27] Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. *MOPTA 2008 Conference (20 August 2008)*, 2008.

[28] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31, 2018.

[29] B. Letham, K. Brian, G. Ottoni, and E. Bakshy. Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis*, 2018.

[30] Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional Bayesian optimization. *Advances in Neural Information Processing Systems 33*, NeurIPS, 2020.

[31] Jun S Liu. Peskun's theorem and a modified discrete-state Gibbs sampler. *Biometrika*, 83(3), 1996.

[32] Sulin Liu, Qing Feng, David Eriksson, Benjamin Letham, and Eytan Bakshy. Sparse bayesian optimization. *International Conference on Artificial Intelligence and Statistics*, pages 3754–3774. PMLR, 2023.

[33] Anthony J Macula and Leonard J Popyack. A group testing method for finding patterns in data. *Discrete applied mathematics*, 144(1-2):149–157, 2004.

[34] Matthias Mayr, Carl Hvarfner, Konstantinos Chatzilygeroudis, Luigi Nardi, and Volker Krueger. Learning skill-based industrial robot tasks with user priors. *IEEE 18th International Conference on Automation Science and Engineering*, 2022. URL `https://arxiv.org/abs/2208.01605`.

[35] Luigi Nardi, David Koeplinger, and Kunle Olukotun. Practical design space exploration. *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 347–358. IEEE, 2019.

[36] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian Optimization in Embedded Subspaces. *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research (PMLR)*, pages 4752–4761, 09–15 Jun 2019.

[37] Diana M Negoescu, Peter I Frazier, and Warren B Powell. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[38] Hung Q Ngo and Ding-Zhu Du. A survey on combinatorial group testing algorithms with applications to DNA library screening. *Discrete mathematical problems with medical applications*, 55:171–182, 2000.

[39] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[40] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Bounce: a Reliable Bayesian Optimization Algorithm for Combinatorial and Mixed Spaces. *arXiv preprint arXiv:2307.00618*, 2023.

[41] Giulia Pedrielli and Szu Hui Ng. G-STAR: A new kriging-based trust region method for global optimization. *2016 Winter Simulation Conference (WSC)*, pages 803–814. IEEE, 2016.

[42] Rommel G Regis. Trust regions in Kriging-based optimization with expected improvement. *Engineering optimization*, 48(6):1037–1059, 2016.

[43] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. *arXiv preprint arXiv:2006.07556*, 2020.

[44] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[45] Jonathan Scarlett. Noisy adaptive group testing: Bounds and algorithms. *IEEE Transactions on Information Theory*, 65(6):3646–3661, 2018.

[46] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. *First Conference on Automated Machine Learning (Main Track)*, 2022.

[47] Bobak Shahriari, Alexandre Bouchard-Côté, and Nando Freitas. Unbounded bayesian optimization via regularization. *Artificial intelligence and statistics*, pages 1168–1176. PMLR, 2016.

[48] Yihang Shen and Carl Kingsford. Computationally Efficient High-Dimensional Bayesian Optimization via Variable Selection. *AutoML Conference 2023*, 2023.

[49] Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte Carlo Tree Search based Variable Selection for High Dimensional Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022.

[50] Doniyor Ulmasov, Caroline Baroukh, Benoit Chachuat, Marc Peter Deisenroth, and Ruth Misener. Bayesian optimization with dimension scheduling: Application to biological systems. *Computer Aided Chemical Engineering*, volume 38, pages 1051–1056. Elsevier, 2016.

[51] Xingchen Wan, Vu Nguyen, Huong Ha, Binxin Ru, Cong Lu, and Michael A. Osborne. Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces. *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10663–10674. PMLR, 18–24 Jul 2021.

[52] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *Journal of Artificial Intelligence Research (JAIR)*, 55:361–387, 2016.

[53] Jack Wolf. Born again group testing: Multiaccess communications. *IEEE Transactions on Information Theory*, 31(2):185–191, 1985.

[54] Nathan Wycoff, Mickael Binois, and Stefan M Wild. Sequential learning of active subspaces. *Journal of Computational and Graphical Statistics*, 30(4):1224–1237, 2021.

[55] Yichi Zhang, Daniel W Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1):1–13, 2020.

[56] Yingbo Zhou, Utkarsh Porwal, Ce Zhang, Hung Q Ngo, XuanLong Nguyen, Christopher Ré, and Venu Govindaraju. Parallel feature selection inspired by group testing. *Advances in neural information processing systems*, 27, 2014.

Paper III

Bounce: Reliable High-Dimensional Bayesian Optimization for Combinatorial and Mixed Spaces

Advances in Neural Information Processing Systems 36, NeurIPS 2023

Leonard Papenmeier
Lund University

Luigi Nardi
Lund University

Matthias Poloczek
Amazon

Abstract

Impactful applications such as materials discovery, hardware design, neural architecture search, or portfolio optimization require optimizing high-dimensional black-box functions with mixed and combinatorial input spaces. While Bayesian optimization has recently made significant progress in solving such problems, an in-depth analysis reveals that the current state-of-the-art methods are not reliable. Their performances degrade substantially when the unknown optima of the function do not

have a certain structure. To fill the need for a reliable algorithm for combinatorial and mixed spaces, this paper proposes `Bounce` that relies on a novel map of various variable types into nested embeddings of increasing dimensionality. Comprehensive experiments show that `Bounce` reliably achieves and often even improves upon state-of-the-art performance on a variety of high-dimensional problems.

1 Introduction

Bayesian optimization (BO) has become a 'go-to' method for optimizing expensive-to-evaluate black-box functions [27, 84] that have numerous important applications, including hyperparameter optimization for machine learning models [9, 27], portfolio optimization in finance [7], chemical engineering and materials discovery [26, 38, 37, 33, 69, 15, 40, 72, 57, 13, 82], hardware design [51, 22, 36], or scheduling problems [42]. These problems are challenging for a variety of reasons. Most importantly, they may expose hundreds of tunable parameters that allow for granular optimization of the underlying design but also lead to high-dimensional optimization tasks and the 'curses of dimensionality' [63, 10]. Typical examples are drug design [53, 77] and combinatorial testing [55]. Moreover, real-world applications often have categorical or ordinal tunable parameters, in addition to the real-valued parameters that BO has traditionally focused on [71, 27, 10]. Recent efforts have thus extended BO to combinatorial and mixed spaces. `Casmopolitan` of Wan et al. [80] uses trust regions (TRs) to accommodate high dimensionality, building upon prior work of Eriksson et al. [25] for continuous spaces. `COMBO` of Oh et al. [56] constructs a surrogate model based on a combinatorial graph representation of the function. Recently, Deshwal et al. [21] presented `BODi` that employs a novel type of dictionary-based embedding and showed that it outperforms the prior work. However, the causes for `BODi`'s excellent performance are not yet well-understood and require a closer examination. Moreover, the ability of methods for mixed spaces to scale to higher dimensionalities trails behind BO for continuous domains. In particular, Papenmeier et al. [60] showed that nested embeddings allow BO to handle a thousand input dimensions, thus outperforming vanilla TR-based approaches and raising the question of whether similar performance gains are feasible for combinatorial domains.

In this work, we assess and improve upon the state-of-the-art in combinatorial BO. In particular, we make the following contributions:

1. We conduct an in-depth analysis of two state-of-the-art algorithms for combinatorial BO, `COMBO` [56] and `BODi` [21]. The analysis reveals that their performances often degrade considerably when the optimum of the optimization

problem does not exhibit a particular structure common for synthetic test problems.

2. We propose `Bounce` (**B**ayesian **o**ptimization **u**sing **i**ncreasingly high-dimensional **c**ombinatorial and continuous **e**mbeddings), a novel high-dimensional Bayesian optimization (HDBO) method that effectively optimizes over combinatorial, continuous, and mixed spaces. `Bounce` leverages parallel function evaluations efficiently and uses nested random embeddings to scale to high-dimensional problems.
3. We provide a comprehensive evaluation on a representative collection of combinatorial, continuous, and mixed-space benchmarks, demonstrating that `Bounce` is on par with or outperforms state-of-the-art methods.

2 Background and Related Work

Bayesian Optimization. Bayesian optimization aims to find the global optimum $\boldsymbol{x}^* \in \mathcal{X}$ of a black-box function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is the D-dimensional search space or *input space*. Throughout this paper, we consider minimization problems, i.e., we aim to find $\boldsymbol{x}^* \in \mathcal{X}$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. The search space $\mathcal{X}$ may contain variables of different types: continuous, categorical, and ordinal. We denote the number of continuous variables in $\mathcal{X}$ by n_{cont} and the number of combinatorial variables by $n_{\text{comb}} = n_{\text{cat}} + n_{\text{ord}} = D - n_{\text{cont}}$, where we denote the number of categorical variables by n_{cat} and the number of ordinal variables by n_{ord}.

Combinatorial Domains. Extending BO to combinatorial spaces is challenging, for example, because the acquisition function is only defined at discrete locations or the dimensionality of the space grows drastically when using one-hot encoding for categorical variables. Due to its numerous applications, combinatorial BO has received increased attention in recent years. `BOCS` [6] handles the exponential explosion of combinations by only modeling lower-order interactions of combinatorial variables and imposing a sparse prior on the interaction terms. `COMBO` [56] models each variable as a graph and uses the graph-Cartesian product to represent the search space. We revisit `COMBO`'s performance on categorical problems in Appendix H.1. `CoCaBo` [68] combines multi-armed bandits and BO to allow for optimization in mixed spaces. It uses two separate kernels for continuous and combinatorial variables and proposes a weighted average of a product and a sum kernel to model mixed spaces. Liu and Wang [47] show that certain, possibly combinatorial functions can be modeled by parametric function approximators such as neural networks or random forests. As a general method to optimize the acquisition

function with gradient-based methods, Daulton et al. [19] propose probabilistic reparametrization.

High-Dimensional Continuous Spaces. Subspace-based methods are primarily used for continuous spaces. Wang et al. [85] propose `REMBO` for HDBO in continuous spaces using Gaussian random projection matrices. `REMBO` suffers from distortions and projections outside the search domains that the corrections of Binois et al. [11, 12] address. The `HeSBO` algorithm of Nayebi et al. [52] avoids the need for corrections by using the `CountSketch` embedding [87]. `Alebo` of Letham et al. [46] builds upon `REMBO`, learning suitable corrections of distortions. `TuRBO` [25] is a method that operates in the full-dimensional input space $\mathcal{X}$, relying on trust region (TR) to focus the search on promising regions of the search space. `BAxUS` of Papenmeier et al. [60] combines the trust region approach of `TuRBO` with the random subspace idea of `HeSBO`. `BAxUS` uses a novel family of *nested* random subspaces that exhibit better theoretical guarantees than the `CountSketch` embedding. While `BAxUS` handled a $1000D$ problem, it only considers continuous problems and cannot leverage parallel function evaluations. `GTBO` [35] assumes the existence of an axis-aligned active subspace. The algorithm first identifies "active" variables and optimizes in the full-dimensional space by placing separate strong length scale priors onto active and inactive variables. Another line of recent approaches employs Monte-Carlo tree search (MCTS) to reduce the complexity of the problem. Wang et al. [83] use MCTS to learn a partitioning of the continuous search space to focus the search on promising regions in the search space. Song et al. [75] use a similar approach, but instead of learning promising regions in the search space, they assume an axis-aligned active subspace and use MCTS to select important variables. Linear embeddings and random linear embeddings [85, 46, 52, 60, 14] require little or no training data to construct the embedding but assume a linear subspace.

Combinatorial High-Dimensional Domains. These works optimize black-box functions defined over a combinatorial or mixed space with dozens of input variables. Thebelt et al. [78] use a tree-ensemble kernel to model the Gaussian process (GP) prior and derive a formulation of the upper-confidence bound (UCB) that allows it to be optimized globally and to incorporate constraints. `RDUCB` [88] relies on random additive decompositions of the GP kernel to model the correlation between variables. `Casmopolitan` [80] follows `TuRBO` in using TRs to focus the search on promising regions of the search space and uses the Hamming distance to model TRs for combinatorial variables. For mixed spaces, `Casmopolitan` uses interleaved search and models continuous and categorical variables with two separate TRs. Kim et al. [44] use a random projection matrix to approach combinatorial problems in a continuous embedded subspace. When evaluating a point, their approach

projects the continuous candidate point to the high-dimensional search space and then rounds to the next feasible combinatorial solution. Deshwal et al. [20] propose two algorithms for *permutation spaces*, which occur in problems such as compiler optimization [36] and pose special challenges due to the superexponential explosion of solutions. `BODi` [21] proposes an embedding type based on a dictionary of anchor points in the search space. The pairwise Hamming distances between the point and each anchor point $\boldsymbol{a}_i$ in the dictionary represent a point in the search space. The anchor points in the dictionary change at each iteration of the algorithm. They are sampled from the search space to cover a wide range of 'sequencies', i.e., the number of changes from 0 to 1 (and vice versa) in the binary vector. The authors hypothesize that the diverse sampling procedure leads to `BODi`'s remarkable performance in combinatorial spaces with up to 60 dimensions. To our knowledge, `BODi` is the only other method combining embeddings and combinatorial spaces. We show in Section 4.6 that `BODi`'s reported good performance relies on an artificial structure of the optimizer $\boldsymbol{x}^*$ and that its performance degrades considerably when this structure is violated.

3 The `Bounce` Algorithm

To overcome the aforementioned challenges in HDBO for real-world applications, we propose `Bounce`, a new algorithm for continuous, combinatorial, and mixed spaces. `Bounce` uses a GP [65] surrogate in a lower-dimensional subspace, the *target space*, that is realized by partitioning input variables into 'bins', the so-called *target dimensions*. `Bounce` only bins variables of the same type (categorical, binary, ordinal, and continuous). When selecting new points to evaluate, `Bounce` sets all input variables within the same bin to a single value. It thus operates in a subspace of lower dimensionality than the input space and, in particular, maximizes the acquisition function in a subspace of lower dimensionality. `Bounce` iteratively refines its subspace embedding by splitting bins into smaller bins, allowing for a more granular optimization at the expense of higher dimensionality. Note that by splitting up bins, `Bounce` asserts that observations taken in earlier subspaces are contained in the current subspace; see Papenmeier et al. [60] for details. Thus, `Bounce` operates in a series of nested subspaces. It uses a novel TR management to leverage batch parallelism efficiently, improving over the single point acquisition of `BAxUS` [60].

The Nested Subspaces. To model the GP in low-dimensional subspaces, `Bounce` leverages `BAxUS`' family of nested random embeddings [60]. In particular, `Bounce` employs the sparse count-sketch embedding [87] in which each input dimension is assigned to exactly one target dimension. When increasing the target dimensionality,

Algorithm 3.1 The `Bounce` algorithm

Input: initial target dimensionality d_{init}, evaluation budget m, batch size B, evaluation budget to input dimensionality m_D, # new bins added per dimension b, number design of experiment (DoE) points n_{init}

Output: optimizer $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x})$.

$i \leftarrow 0$, $d \leftarrow d_{\text{init}}$
$b \leftarrow \textsc{AdjustBins}(b, d_{\text{init}}, m_D)$ ▷ Section 3
$m_i \leftarrow \left\lfloor \frac{b \cdot m_D \cdot d_{\text{init}}}{d_{\text{init}} \cdot (1-(b+1)^{k+1})} \right\rceil$
$S \leftarrow \textsc{InitialEmbedding}(d_0, n_{\text{cont}}, n_{\text{cat}}, n_{\text{comb}}, n_{\text{bin}})$ ▷ Section 3.1
$\mathcal{D} \leftarrow \{(\boldsymbol{z}_k, f(S^{-1}(\boldsymbol{z}_k)))\}_{k \in n_{\text{init}}}$ ▷ Sample and evaluate initial points.
for $j = 1, \ldots, m$ **do**
    $L_{\text{cont}} \leftarrow 0.8$, $L_{\text{comb}} \leftarrow \min(40, n_{\text{comb}})$
    while $L_{\text{cont}} > L_{\text{min}}^{\text{cont}} \wedge L_{\text{comb}} > L_{\text{min}}^{\text{comb}}$ **do**
        Find B candidates according to Sec. 3.2
        Evaluate f at B and update $\mathcal{D}$: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{z}_k, f(S^{-1}(\boldsymbol{z}_k)))\}_{k \in B}$
        Update L_{cont} and L_{comb} ▷ Section 3
    if $d < D$ **then**
        $i \leftarrow i + 1$ ▷ Increase index for target dimensionality.
        $S \leftarrow \textsc{IncreaseEmbedding}(S, b)$ ▷ Section 3.1
        $d \leftarrow$ # target variables in S
        $m_i \leftarrow \left\lfloor \frac{b \cdot m_D \cdot d}{d_{\text{init}} \cdot (1-(b+1)^{k+1})} \right\rceil$
    else
        Reset $\mathcal{D}$ by resampling and evaluating new initial points
        Resample S, reset L_{cont} and L_{comb}, $j \leftarrow j + n_{\text{init}}$

`Bounce` creates b new bins for every existing bin and re-distributes the input dimensions that had previously been assigned to that bin across the now $b + 1$ bins. `Bounce` allocates an individual evaluation budget m_i to the current target space $\mathcal{X}_i$ that is proportional to the dimensionality of $\mathcal{X}_i$. When the budget for the current target space is depleted, and `Bounce` has not found a better solution, `Bounce` will increase the dimension of the target space until it reaches the input space of dimensionality D. Let d_{init} denote the dimensionality of the first target space, i.e., the random embedding that `Bounce` starts with. Then `Bounce` has to increase the target dimension $\lceil \log_{b+1} D/d_{\text{init}} \rceil =: k$-times to reach the input dimensionality D. After calculating k, `Bounce` re-sets the split factor b such that the distance between the predicted final target dimensionality $d_k = d_{\text{init}} \cdot (b+1)^k$ and the input dimensionality D is minimized: $b = \lfloor \log_k(D/d_{\text{init}}) - 1 \rceil$, where $\lfloor x \rceil$ denotes the integer closest to x. This ensures that the predetermined evaluation budget for each subspace will be approximately proportional to its dimensionality.

This contrasts to BAxUS [60] that uses a constant split factor b and adjusts the initial target dimensionality d_{init}. The evaluation budget m_i for the i-th subspace $\mathcal{X}_i$ is $m_i \coloneqq \left\lfloor \frac{b \cdot m_D \cdot d_i}{d_{\text{init}} \cdot (1-(b+1)^{k+1})} \right\rceil$, where m_D is the budget until D is reached and b is the maximum number of bins added per split.

Trust Region Management. Bounce follows TuRBO [25] and Casmopolitan [80] in using trust regions (TRs) to efficiently optimize over target spaces of high dimensionality. TRs allow focusing on promising regions of the search space by restricting the next points to evaluate to a region centered at the current best function value [25]. TR-based methods usually expand their TR if they find better points and conversely shrink it if they fail to make progress. If the TR falls below the threshold given by the *base length*, TuRBO and Casmopolitan restart with a fresh TR elsewhere. Casmopolitan [80] uses different base lengths for combinatorial and continuous variables. For combinatorial variables, the distance to the currently best function value is defined in terms of the Hamming distance, and the base length is an integer. For continuous variables, Casmopolitan defines the base length in terms of the Euclidean distance, i.e., a real number. Similarly, Bounce has separate base lengths $L_{\text{min}}^{\text{cont}}$ and $L_{\text{min}}^{\text{comb}}$ for continuous and combinatorial variables but does not fix the factor by which the TR volume is increased or decreased upon successes or failures. Instead, the factor is adjusted dynamically so that the evaluation budget m_i for the current target space $\mathcal{X}_i$ is adhered to. This design is crucial to enable batch parallelism, as we describe next.

Batch Parallelism. We allow Bounce to efficiently evaluate batches of points in parallel by using a scalable TR management strategy and q-expected improvement (qEI) [81, 86, 5] as the acquisition function for batches of size $B > 1$. When Bounce starts with a fresh TR, we sample n_{init} initial points to initialize the GP, using a Sobol sequence for continuous variables and uniformly random values for combinatorial variables.

The TR management strategy of Bounce differs from previous strategies [66, 25, 80, 60] in that it uses a dynamic factor to determine the TR base length. Recall that Bounce shrinks the TR if it fails to make progress and starts a fresh TR if the TR falls below the threshold given by the base length. Bounce's rule is based on the idea that the minimum admissible TR base length should be reached when the current evaluation budget is exhausted. If one employed the strategies of TuRBO [25], Casmopolitan [80], or BAxUS [60] for larger batch sizes B and Bounce's nested subspaces, then one would spend a large part of the evaluation budget in early target spaces. For example, suppose a continuous problem, the common values for the

initial, minimum, and maximum TR base length, and the constant shrinkage factor of [66]. Then, such a method has to shrink the TR base length at least seven times (i.e., evaluate f $7B$-times) before it would increase the dimensionality of the target space. Thus, the method would risk depleting its budget before reaching a target space suitable for the problem. On the other hand, we will see that `Bounce` chooses an evaluation budget that is smaller in low-dimensional target spaces and higher for later target spaces of higher dimensionality. Considering a 1000-dimensional problem with an evaluation budget of 1000, it uses only $3, 12$, and 47 samples for the first three target spaces of dimensionalities 2, 8, and 32.

`Bounce`'s strategy permits flexible TR shrinkage factors and base lengths, i.e., TR base lengths to vary within the range $[L_{\min}, L_{\max}]$. This allows `Bounce` to comply with the evaluation budget m_i for the current target space $\mathcal{X}_i$. Suppose that `Bounce` has evaluated j batches of B points each since it last increased the dimensionality of the target space, and let L_j denote the current TR base length. Observe that hence $m_i - jB$ evaluations remain for $\mathcal{X}_i$. Then `Bounce` sets the TR base length to $L_{j+1} \coloneqq \lambda_j^{-B} L_j$ with $\lambda_j < 1$, if it found a new best point whose objective value improves upon the incumbent by at least ε. We call this a 'success'. Otherwise, `Bounce` observes a 'failure' and sets $L_{j+1} \coloneqq \lambda_j^{+B} L_j$. The rationale of this rule is that if the algorithm is in iteration j and only observes failures subsequently, then we apply this factor $(m_i - jB)$-times, which is the remaining number of function evaluations in the current subspace $\mathcal{X}_i$. Hence, the last batch of the i-th target space $\mathcal{X}_i$ will have the minimum TR base length, and `Bounce` will increase the target dimensionality afterward. If the TR is expanded upon a 'success', we need to adjust λ_j not to use more than the allocated number of function evaluations in a target space. At each iteration, we therefore set adjustment factor $\lambda_j = (L_{\min}/L_j)^{1/(m_i - jB)}$. Note that λ_j remains unchanged under this rule unless the TR expanded in the previous iteration.

The Kernel Choice. To harvest the sample efficiency of a low-dimensional target space, we would like to combine categorical variables into a single bin, even if they vary in the number of categories. This is not straightforward. For example, note that the popular one-hot encoding of categorical variables would give rise to multiple binary input dimensions, which would not be compatible with the above strategy of binning variables to form nested subspaces. `Bounce` overcomes these obstacles and allows variables of the same type to share a representation in the target space. We provide the details in Sect. 3.1.

For the GP model, we use the `CoCaBo` kernel [68]. In particular, we model the continuous and combinatorial variables with two separate $5/2-$Matérn kernels where we use automatic relevance determination (ARD) for the continuous variables and

share one length scale for all combinatorial variables. Following Ru et al. [68], we use a mixture of the sum and the product kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2(\rho k_{\text{cmb}}(\boldsymbol{x}_{\text{cmb}}, \boldsymbol{x}'_{\text{cmb}}) k_{\text{cnt}}(\boldsymbol{x}_{\text{cnt}}, \boldsymbol{x}'_{\text{cnt}}) + (1-\rho)(k_{\text{cmb}}(\boldsymbol{x}_{\text{cmb}}, \boldsymbol{x}'_{\text{cmb}}) + k_{\text{cnt}}(\boldsymbol{x}_{\text{cnt}}, \boldsymbol{x}'_{\text{cnt}}))),$$

where $\boldsymbol{x}_{\text{cnt}}$ and $\boldsymbol{x}_{\text{cmb}}$ are the continuous and combinatorial variables in $\boldsymbol{x}$, respectively, and σ_f^2 is the signal variance. The trade-off parameter $\rho \in [0, 1]$ is learned jointly with the other hyperparameters *via* likelihood maximization. See Appendix G.2 for additional details.

Algorithm 3.1 gives a high-level overview of `Bounce`. In Appendix A, we prove that `Bounce` converges to the global optimum under mild assumptions. We now explain the different components of `Bounce` in detail.

3.1 The Subspace Embedding of Mixed Spaces

`Bounce` supports mixed spaces of four types of input variables: categorical, ordinal, binary, and continuous variables. We discuss binary and categorical variables separately because we model them differently. The proposed embedding maps only variables of a single type to each 'bin', i.e., no target dimension of the embedding has variables of different types. Thus, target dimensions are homogeneous in this regard. Note that the number of target dimensions of each type is implied by the current bin size of the embedding that may grow during the execution. The proposed embedding can handle categorical or ordinal input variables that differ in the number of discrete values they can take.

Continuous Variables. As common in BO, we suppose that each continuous variable takes values in a bounded interval. Thus, we may normalize each interval to $[-1, 1]$. The embedding of continuous variables, i.e., input dimensions, follows `BAxUS` [60]: each input dimension D_i is associated with a random sign $s_i \in \{-1, +1\}$ and one or multiple input dimensions can be mapped to the same target dimension of the low-dimensional embedded subspace. Recall that `Bounce` works on the low-dimensional subspace and thus decides an assignment v_j for every target dimension d_j of the embedding. Then, all input variables mapped to this particular target dimension are set to this value v_j.

Binary Variables. Binary dimensions are represented by the values -1 and $+1$. Each input dimension D_i is associated with a random sign $s_i \in \{-1, +1\}$,

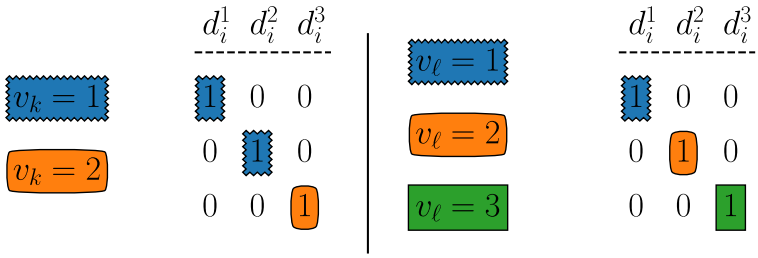


Figure 3.1: The mapping (or binning) of categorical and ordinal variables. Suppose that variable v_k has two categories and that v_ℓ has three categories. Both are mapped to the target dimension d_i that has cardinality $3 = \max\{2, 3\}$. While the mapping of v_ℓ to d_i is a straightforward bijection, v_k has fewer categories than d_i. Thus, the mapping of v_k to d_i repeats label 1. Ordinal variables are mapped similarly.

and the subspace embedding may map one or more binary input dimensions to the same binary target dimension. While the embedding for binary and continuous dimensions is similar, `Bounce` handles binary dimensions differently when optimizing the acquisition function.

Categorical and Ordinal Variables. Categorical variables that differ in the number of categories may be mapped to the same target dimension (bin). Suppose that the categorical variables $v_1, \ldots, v_\ell$ with cardinalities $c_1, \ldots, c_\ell$ are mapped to a single bin that is associated with the target dimension d_j of the subspace embedding. Then d_j is of categorical type and has $\max\{c_i \mid 1 \leq i \leq \ell\} =: c_{\max}$ distinct categories, that is, its cardinality is given by the maximum cardinality of any variable mapped to it. Thus, the bin d_j can represent every category of these input variables.

Suppose that `Bounce` assigns the category $k \in \{1, \ldots, c_{\max}\}$ to the categorical bin (target dimension) d_j. We transform this label to a categorical assignment to each input variable $v_1, \ldots, v_\ell$, setting $v_i = \lceil k \cdot (c_i / c_{\max}) \rceil$. Recall that `Bounce` may split up bins, i.e., target dimensions, to increase the dimensionality of its subspace embedding. In such an event, every derived bin inherits the cardinality of the parent bin. This allows us to retain any observations the algorithm has taken up to this point. Analogously to the random sign for binary variables, we randomly shuffle the categories before the embedding. This reduces the risk of `Bounce` being biased towards a specific structure of the optimizer (see Appendix E).

We treat ordinal variables as categorical variables whose categories correspond to the discrete values the ordinal variable can take. For the sake of simplicity, we suppose here that an ordinal variable v_i has range $\{1, 2, \ldots, c_i\}$ and $c_i \geq 2$ for all $i \in \{1, \ldots, \ell\}$. Figure 3.1 shows examples of the binning of categorical and ordinal variables.

3.2 Maximization of the Acquisition Function

We use expected improvement (EI) [43] for batches of size $B = 1$ and qEI [81, 86, 5] for larger batches. We optimize the EI using gradient-based methods for continuous problems and local search for combinatorial problems. We interleave gradient-based optimization and local search for functions defined over a mixed space; see Appendix G.1 for details.

4 Experimental Evaluation

We evaluate `Bounce` empirically on various benchmarks whose inputs are combinatorial, continuous, or mixed spaces. The evaluation comprises the state-of-the-art algorithms `BODi` [21], `Casmopolitan` [80], `COMBO` [56], `SMAC` [41], and `RDUCB` [88], using code provided by the authors. We also report `Random Search` [8] as a baseline. For categorical problems, `COMBO`'s implementation suffers from a bug explained in Appendix H.2. We report the results for `COMBO` with the correct benchmark implementation as "`COMBO` (fixed)".

The Experimental Setup. We initialize every algorithm with five initial points. The plots show the performances of all algorithms averaged over 50 repetitions except `BODi`, which has 20 repetitions due to resource constraints caused by its high memory demand. The shaded regions give the standard error of the mean. We use common random seeds for all algorithms and for randomizing the benchmark functions. We run all methods for 200 function evaluations unless stated otherwise. The `Labs` (Section 4.1) and `MaxSat125` (Section 4.2) benchmarks are run for 500 evaluations.

The Benchmarks. The evaluation uses seven established benchmarks [21]: 53D `SVM`, 50D `LABS`, 125D `ClusterExpansion` [3, 4], 60D `MaxSAT60` [56, 21], 25D `PestControl`, 53D `Ackley53`, and 25D `Contamination` [39, 6, 56]. Due to space constraints, we moved the results for the `MaxSAT60`, `Contamination`, and `Ackley53` benchmarks to Appendix B.1. For each benchmark, we report results for the originally published formulation and for a modification where we move the optimal point to a random location. The randomization procedure is fixed for each benchmark for all algorithms and repetitions. For binary problems, we flip each input variable independently with probability 0.5. For categorical problems, we randomly permute the order of the categories. We motivate this randomization in Section 4.6.

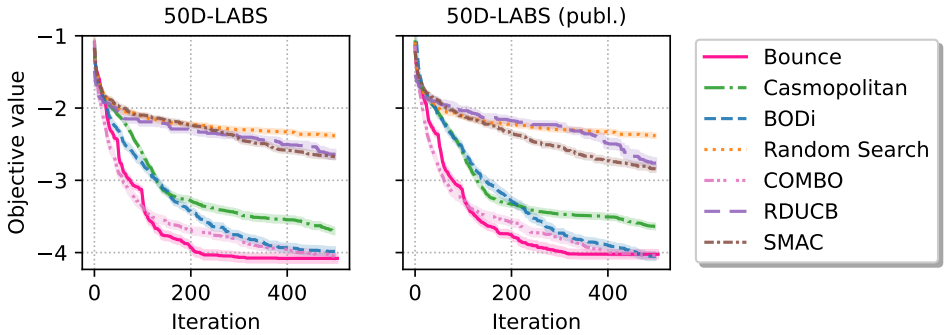


Figure 3.2: The 50D low-autocorrelation binary sequence problem. `Bounce` finds the best solutions, followed by `COMBO`.

4.1 50D Low-Autocorrelation Binary Sequences (LABS)

`LABS` has $n = 50$ binary dimensions. It has important applications in communications engineering and mathematics; see Packebusch and Mertens [58] for details. `LABS` is a hard combinatorial problem and currently solved *via* exhaustive search. The goal is to find a sequence $\boldsymbol{x} \in \{-1, +1\}^n$ with a maximum merit factor $F(\boldsymbol{x}) = \frac{n^2}{2E(\boldsymbol{x})}$, where $E(\boldsymbol{x}) = \sum_{k=1}^{n-1} C_k^2(\boldsymbol{x})$ and $C_k(\boldsymbol{x}) = \sum_{i=1}^{n-k} x_i x_{i+k}$ for $k = 0, \ldots, n-1$ are the autocorrelations of $\boldsymbol{x}$ [58]. Figure 3.2 summarizes the performances. We observe that `Bounce` outperforms all other algorithms on the benchmark's original and randomized versions.

4.2 Industrial Maximum Satisfiability: 125D `ClusterExpansion` benchmark

MaxSat is a notoriously hard problem for which various approximations and exact (exponential time) algorithms have been developed; see Hansen and Jaumard [32], Poloczek and Williamson [61] for an overview. We evaluate `Bounce` and the other algorithms on the 125-dimensional `ClusterExpansion` benchmark, a real-world MaxSAT instance with many applications in materials science [2]. Unlike the `MaxSAT60` benchmark (see Appendix B.1), `ClusterExpansion` is not a crafted benchmark, and its optimum has no synthetic structure [3, 1]. We treat the MaxSat problems as black-box problems; hence, algorithms do not have access to the clauses, and we cannot use the usual algorithms.

Figure 3.3 shows the total negative weight of the satisfied clauses as a function of evaluations. We cannot plot regret curves since the optimum is unknown [4]. We observe that `Bounce` finds better solutions than all other algorithms. `BODi` is the only

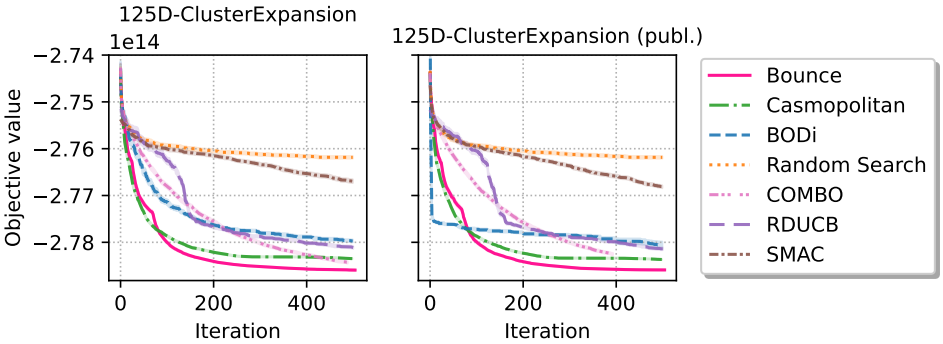


Figure 3.3: The 125D weighted `ClusterExpansion` maximum satisfiability problem. We plot the total negative weight of clauses. `Bounce` produces the best assignments.

algorithm for which we observe sensitivity to the location of the optimal assignment: for the published version of the benchmark, `BODi` quickly jumps to a moderately good solution but fails to make further progress.

4.3 25D Categorical Pest Control

`PestControl` is a more complex version of the `Contamination` benchmark and has 25 categorical variables with five categories each [56]. The task is to select one out of five actions $\{1, 2, \ldots, 5\}$ at each of 25 stations to minimize the objective function that combines total cost and a measure of the spread of the pest. We note the setting $\boldsymbol{x} = (5, 5, \ldots, 5)$ achieves a good value of 12.57, while the best value found in our evaluation is 12.07 is $\boldsymbol{x} = (5, 5, \ldots, 5, 1)$ and thus has a Hamming distance of one. The random seed used in our experiments is zero. Figure 3.4 summarizes the performances of the algorithms. `Bounce` is robust to the location of the global optimum and consistently obtains the best solutions. In particular, the performances of `COMBO` and `BODi` depend on whether the optimum has a certain structure. We discuss this issue in detail in Appendix H.1.

4.4 `SVM` – a 53D AutoML task

In the `SVM` benchmark, we optimize over a mixed space with 50 binary and 3 continuous parameters to tune an ε-support vector regression (SVR) model [73]. The 50 binary parameters determine whether to include or exclude an input feature from the dataset. The 3 continuous parameters correspond to the regularization parameter C, the kernel width γ, and the ε parameter of the ε-SVR model [73]. Its root mean squared error on a held-out dataset gives the function value. Figure 3.5

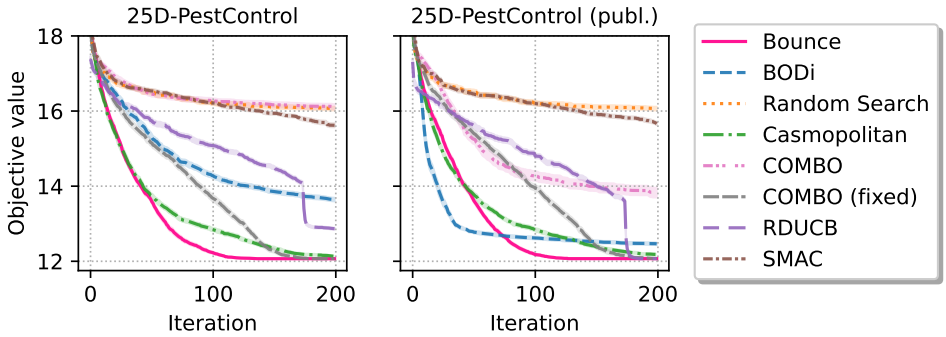


Figure 3.4: The 25D categorical pest control problem. `Bounce` obtains the best solutions, followed by `Casmopolitan`. `BODi`'s performance degrades significantly when shuffling the order of categories.

summarizes the performances of the algorithms. We observe that `Bounce`, `BODi`, and `Casmopolitan` achieve comparable solutions. `BODi` performs slightly worse if the ordering of the categories is shuffled and slightly better if the optimal assignment to all binary variables is one. `COMBO` does not support continuous variables and thus was omitted.

4.5 `Bounce`'s efficacy for batch acquisition

We study the sample efficiency of `Bounce` when it selects a batch of B points in each iteration to evaluate in parallel. Figure 3.6 shows the results for $B = 1, 3, 5, 10,$ and 20, where `Bounce` was run for $\min(2000, 200 \cdot B)$ function evaluations. We configure `Bounce` to reach the input dimensionality after 100 evaluations for $B = 1, 3, 5$ and after $25B$ for $B = 10, 20$. We observe that `Bounce` leverages parallel function evaluations effectively: it obtains a comparable function value at a considerably smaller number of iterations, thus saving wall-clock time for applications with time-consuming function evaluations. We also studied batch acquisition for continuous problems and found that `Bounce` also provides significant speed-up. Due to space constraints, we deferred the discussion to Appendix C.

4.6 The sensitivity of `BODi` and `COMBO` to the location of the optima

The empirical evaluation reveals that the performances of `BODi` [21] and `COMBO` [56] are sensitive to the location of the optima. Both methods degrade on at least one benchmark when the optimum is moved to a randomly chosen point. This is particularly unexpected for categorical variables where moving the optimum to a

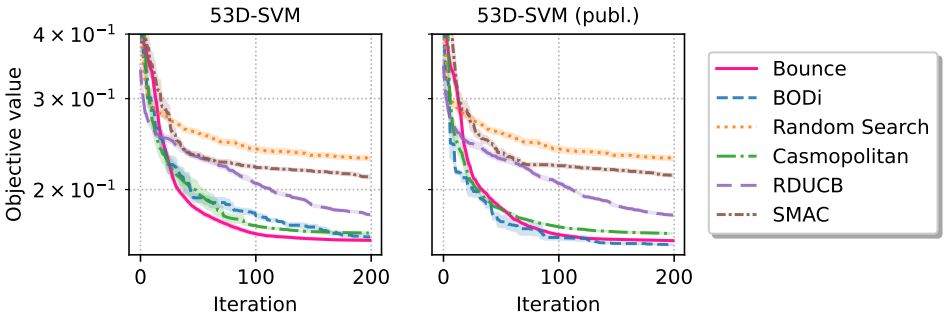


Figure 3.5: The 53-dimensional `SVM` benchmark. `Bounce`, `BODi`, and `Casmopolitan` achieve comparable solutions.

random location is equivalent to shuffling the labels of the categories of each variable. Such a change of representation should not affect the performance of an algorithm.

`BODi` is more susceptible to the location of the optimizer than `COMBO`. The performance of `COMBO` degrades only on the categorical `PestControl` benchmark, whereas `BODi` degrades on five out of seven benchmarks. Looking closer, we observe that `BODi`'s performance degradation is particularly large for synthetic benchmarks like `Ackley53` and `MaxSAT60`, where setting all variables to the same value is optimal. Figure 3.7 summarizes the effects of moving the optimum on `BODi`. Due to space constraints, we moved the details and a discussion of categorical variables to the appendix. Similarly, setting all binary variables of the `SVM` benchmark to one produces a good objective value. It is not surprising, given that the all-one assignment corresponds to including all features previously selected for the benchmark because of their high importance.

We show in Appendix H.1 that `BODi` adds a point to its dictionary with zero Hamming distance to an all-zero or all-one solution, with a probability that increases with the dictionary size. Deshwal et al. [21, p. 7] reported that `BODi`'s performance 'tends to improve' with the size of the dictionary. Moreover, `BODi` samples a new dictionary in each iteration, eventually increasing the chance of having such a point in its dictionary. Thus, we hypothesize that `BODi` benefits from having a near-optimal solution in its dictionary, likely for all-zero or all-one solutions. For `COMBO`, Figure 3.4 shows that the performance on `PestControl` degrades substantially if the labels of the categories are shuffled. Then `COMBO`'s sample-efficiency becomes comparable to `Random Search`.

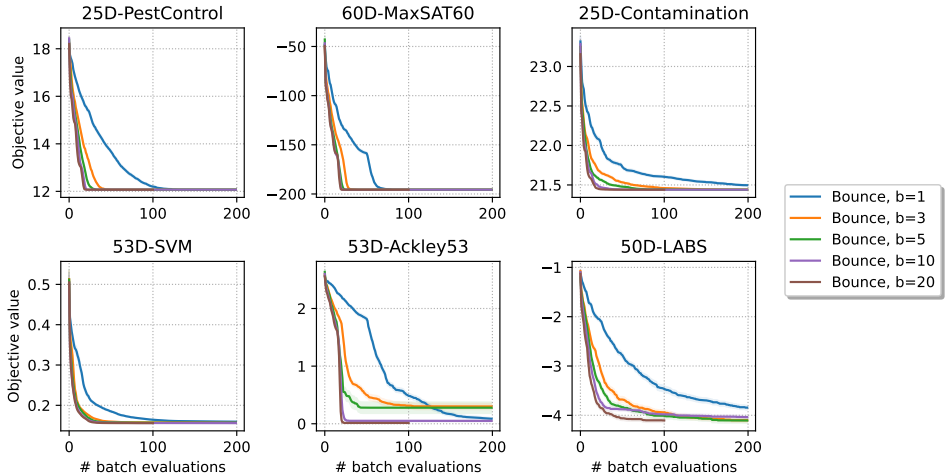


Figure 3.6: `Bounce` benefits from the batch acquisition that allows parallelizing function evaluations. We show the best function values obtained after each batch for batch sizes $1, 3, 5, 10$, and 20.

5 Discussion

BO in combinatorial spaces has many exciting and impactful applications. Its applicability to real-world problems, such as `LABS` that defy a closed-form solution, makes it a valuable tool for practitioners. Our empirical evaluation reveals that state-of-the-art methods fail to provide good solutions reliably. In particular, it finds that `BODi` and `COMBO`, which performed best in recent publications, are sensitive to the location of the optimizer. We identified design flaws in `BODi` and an implementation bug in `COMBO` as the root causes of the performance degradations.

The proposed `Bounce` algorithm is reliable for high-dimensional black-box optimization in combinatorial, continuous, and mixed spaces. The empirical evaluation demonstrates that `Bounce` reliably outperforms the state-of-the-art on a diverse set of problems. Using a novel TR management strategy, `Bounce` leverages parallel evaluations of the objective function to improve its performance. We anticipate headroom by tailoring the modeling of combinatorial objects, e.g., arising in the search for peptides or materials discovery [79, 28, 59, 76, 37, 34]. Here it seems particularly interesting to incorporate prior belief on the importance of decision variables while maintaining the overall scalability. Moreover, extending the present work to black-box constraints [30, 24], multiple objectives, and multiple information sources [62, 31, 18] will considerably expand the applicable use cases.

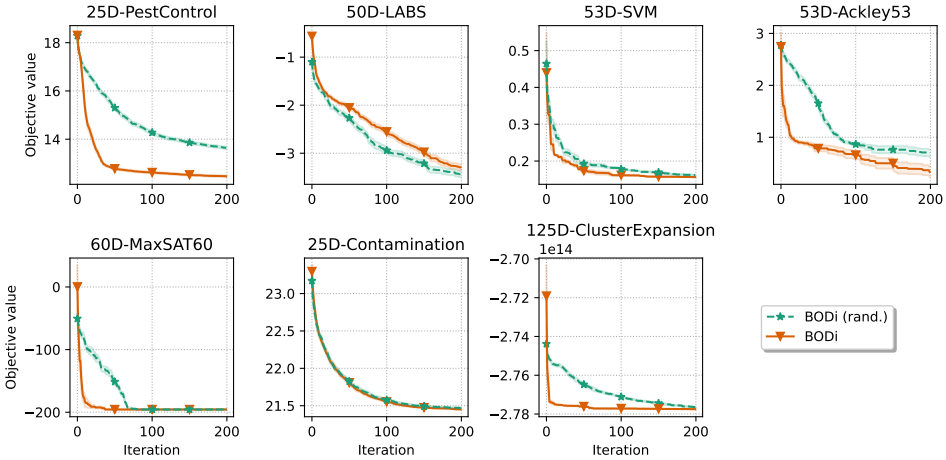


Figure 3.7: `BODi`'s performance degrades on five out of seven benchmarks when randomizing the location of the optimal solution: `BODi` on the default version (orange) of the benchmark and on the modified version (green, dashed) where the optimum was moved.

Limitations. `Bounce` is not designed to handle noisy evaluations of the objective function. While it seems straightforward to extend `Bounce` to handle noisy evaluations, e.g., by using a Gaussian process with a noise term and acquisition functions that account for noise [71], we leave this for future work. Moreover, in applications where the categorical or ordinal variables vary substantially in the number of values they can take, there may be better ways to 'bin' them.

Societal impact. Bayesian optimization has recently gained wide-spread popularity for tasks in drug discovery [53], chemical engineering [38, 72, 33, 69, 15], materials discovery [79, 28, 59, 76, 37, 34], aerospace engineering [49, 6, 45], robotics [48, 17, 64, 16, 50], and many more. This highlights the Bayesian optimization community's progress toward providing a reliable 'off-the-shelf optimizer.' However, this promise is not yet fulfilled for the newer domain of mixed-variable Bayesian optimization that allows optimization over hundreds of 'tunable levers', some of which are discrete, while others are continuous. This domain is of particular relevance for the tasks above. `Bounce`'s ability to incorporate more such levers in the optimization significantly impacts the above practical applications, allowing for more granular control of a chemical reaction or a processing path, to give some examples. The empirical evaluation shows that the performance of state-of-the-art methods is highly sensitive to the location of the unknown global optima and often degenerates drastically, thus putting practitioners at risk. The proposed algorithm `Bounce`, however, achieves robust performance over a broad collection of tasks and thus will become a 'goto' optimizer for practitioners in other fields. Therefore, we open-source

the `Bounce` code.[5]

Acknowledgments

Leonard Papenmeier and Luigi Nardi were partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Luigi Nardi was partially supported by the Wallenberg Launch Pad (WALP) grant Dnr 2021.0348 and by affiliate members and other supporters of the Stanford DAWN project — Ant Financial, Facebook, Google, Intel, Microsoft, NEC, SAP, Teradata, and VMware. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at the Chalmers Centre for Computational Science and Engineering (C3SE) and the National Supercomputer Centre at Linköping University, partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

[5] `https://github.com/LeoIV/bounce`

A Consistency of Bounce

In this section, we prove the consistency of the Bounce algorithm. The proof is based on Papenmeier et al. [60] and Eriksson and Poloczek [24].

Theorem 3 (Bounce consistency). *With the following definitions*

Def. 1. $(\boldsymbol{x}_k)_{k=1}^{\infty}$ *is a sequence of points of decreasing function values;*
Def. 2. $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}}$ *is a minimizer of* f *in* $\mathcal{X}$*;*

and under the following assumptions:

Ass. 1. D *is finite;*
Ass. 2. f *is observed without noise;*
Ass. 3. The range of f *is bounded in* $\mathcal{X}$*, i.e.,* $\exists C \in \mathbb{R}_{++}$ *s.t.* $|f(\boldsymbol{x})| < C \ \forall \boldsymbol{x} \in \mathcal{X}$*;*
Ass. 4. For at least one of the minimizers $\boldsymbol{x}_i^*$ *the (partial) assignment corresponding to the continuous variables lies in a (continuous) region with positive measure;*
Ass. 5. One Bounce *reached the input dimensionality* D*, the continuous elements of the initial points* $\{\boldsymbol{x}_{cont_i}\}_{n=1}^{n_{init}}$ *after each TR restart are chosen*

(a) uniformly at random for continuous variables; and
(b) such that every realization of the combinatorial variables has positive probability;

then the Bounce *algorithm finds a global optimum with probability 1, as the number of samples* N *goes to* ∞*.*

Proof. The range of f is bounded per Assumption 3, and Bounce only considers a function evaluation a 'success' if the improvement over the current best solution exceeds a certain constant threshold. Bounce can only have a finite number of 'successful' evaluations because the range of f is bounded per Assumption 3. For the sake of a contradiction, we suppose that Bounce does not obtain an optimal solution as its number of function evaluations $N \to \infty$. Thus, there must be a sequence of failures, such that the TRs in the current target space, i.e., the current subspace, will eventually reach its minimum base length. Recall that in such an event, Bounce increases the target dimension by splitting up the 'bins', thus creating a subspace of $(b+1)$-times higher dimensionality. Then Bounce creates a new TR that again experiences a sequence of failures that lead to another split, and so on. This series of events repeats until the embedded subspace eventually equals the input space and thus has dimensionality D. See lines $12-16$ in Algorithm 3.1 in Sect. 3.

Still supposing that `Bounce` does not find an optimum in the input space, there must be a sequence of failures such that the side length of the TR again falls below the set minimum base length, now forcing a restart of `Bounce`. Recall that at every restart, `Bounce` samples a fresh set of initial points uniformly at random from the input space; see line 18 in Algorithm 3.1. Therefore, with probability 1, a random sample will eventually be drawn from any subset $\mathcal{Y} \subseteq \mathcal{X}$ with positive Lebesgue measure ($\nu(\mathcal{Y}) > 0$):

$$1 - \lim_{k \to \infty} (1 - \mu(\mathcal{Y}))^k = 1, \tag{3.1}$$

where μ is the uniform probability measure of the sampling distribution that `Bounce` employs for initial data points upon restart [74].

Let

$$\alpha = \inf \{t : \nu [x \in \mathcal{X} \mid f(x) < t] > 0\} \tag{3.2}$$

denote the essential infimum of f on $\mathcal{X}$ with ν being the Lebesgue measure [74].

Following Solis and Wets [74], we define the optimality region, i.e., the set of points whose function value is larger by at most ε than the essential infimum:

$$R_{\varepsilon,M} = \{x \in \mathcal{X} \mid f(x) < \alpha + \varepsilon\} \tag{3.3}$$

with $\varepsilon > 0$ and $M < 0$. Because of Ass. 4, at least one optimal point lies in a region of positive measure that is continuous for the continuous variables. Therefore, we have that $\alpha = f(\boldsymbol{x}^*)$. Note that this is also the case if the domain of f only consists of combinatorial variables (Ass. 5). Then, $R_{\varepsilon,M} = \{\boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) < f(\boldsymbol{x}^*) + \varepsilon\}$.

Let $(\boldsymbol{x}_k^\star)_{k=1}^\infty$ denote the sequence of best points that `Bounce` discovers with $\boldsymbol{x}_k^\star$ being the best point up to iteration k. This sequence satisfies Def. 1 by construction. Note that $\boldsymbol{x}_k^\star \in R_{\varepsilon,M}$ implies that $\boldsymbol{x}_{k'}^\star \in R_{\varepsilon,M}$ for all $k' \geq k+1$ [74] because observations are noise-free. Then,

$$\begin{aligned} \mathbb{P}[\boldsymbol{x}_k^\star \in R_{\varepsilon,M}] &= 1 - \mathbb{P}[\boldsymbol{x}_k^\star \in \mathcal{X} \setminus R_{\varepsilon,M}] && (3.4) \\ &\geq 1 - (1 - \mu(R_{\varepsilon,M}))^k, && (3.5) \end{aligned}$$

and,

$$1 \geq \lim_{k \to \infty} \mathbb{P}[\boldsymbol{x}_k^\star \in R_{\varepsilon,M}] \geq \underbrace{1 - \lim_{k \to \infty} (1 - \mu(R_{\varepsilon,M}))^k}_{=1, \text{ Eq. (3.1)}} = 1, \tag{3.6}$$

i.e., $\boldsymbol{x}_k^\star$ eventually falls into the optimality region [74]. By letting $\varepsilon \to 0$, $\boldsymbol{x}_k^\star$ converges to the global optimum with probability 1 as $k \to \infty$.

□

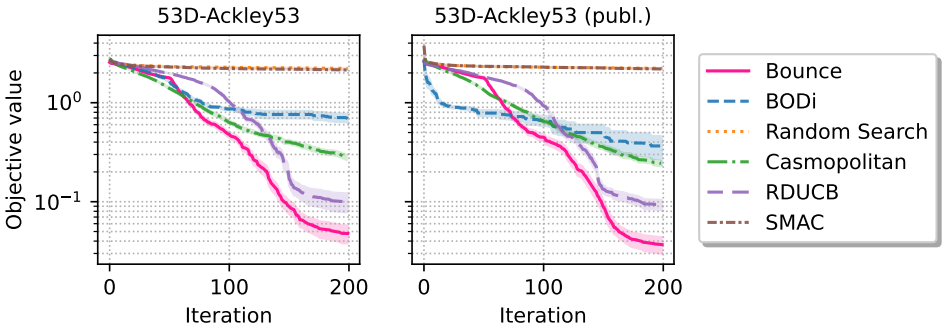


Figure 3.8: `Bounce` the other algorithms on the synthetic `Ackley53` benchmark function. `Bounce` outperforms all other algorithms and quickly finds excellent solutions. `BODi`'s performance degrades upon randomization.

B Additional Experiments

We compare `Bounce` to the other algorithms on three additional benchmark problems: `Ackley53` and `MaxSAT60` [21]. Moreover, we run two additional studies to investigate the performance of `Bounce` further. First, we run `Bounce` on a set of continuous problems from Papenmeier et al. [60] to showcase the performance and scalability of `Bounce` on purely continuous problems. We then present a "low-sequency" version of `Bounce` to showcase how such a version can outperform its competitors on the original benchmarks by introducing a bias towards low-sequency solutions.

B.1 `Bounce` and other algorithms on additional benchmarks

The synthetic `Ackley53` benchmark function

`Ackley53` is a 53-dimensional function with 50 binary and three continuous variables. Wan et al. [80] discretized 50 continuous variables of the original Ackley function, requiring these variables to be either zero or one. This benchmark was designed such that the optimal value of 0.0 is at the origin $\boldsymbol{x} = (0, \ldots, 0)$. Here, we perturb the optimal assignment of combinatorial variables by flipping each binary variable with probability $1/2$. Figure 3.8 summarizes the performances of the algorithms. `Bounce` outperforms all other algorithms and proves to be robust to the location of the optimum point. `Casmopolitan` is a distanced runner-up. `BODi` initially outperforms `Casmopolitan` on the published benchmark version but falls behind later.

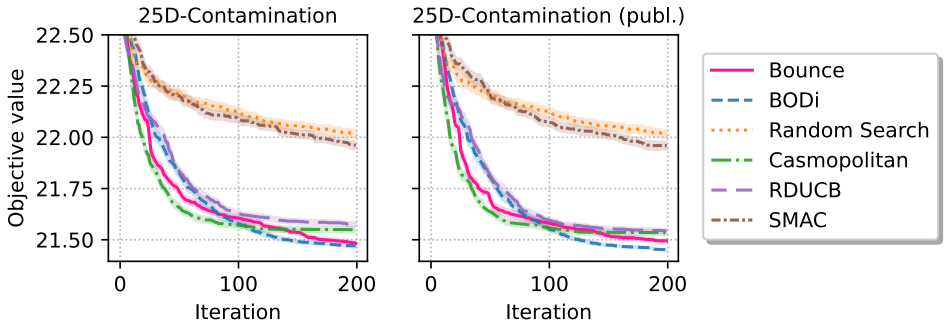


Figure 3.9: `Bounce` and the other algorithms on the 25-dimensional contamination problem. `Bounce` performs on par with `Casmopolitan` and `BODi` on both versions of the benchmark.

Contamination control

The `Contamination` benchmark models a supply chain with 25 stages [39]. At each stage, a binary decision is made whether to quarantine food that has not yet been contaminated. Each such intervention is costly, and the goal is to minimize the number of contaminated products and prevention cost [6, 56]. Figure 3.9 shows the performances of the algorithms.

`Bounce`, `Casmopolitan`, and `BODi` all produce solutions of comparable objective value. `Bounce` and `Casmopolitan` find better solutions than `BODi` initially, but after about 100 function evaluations, the solutions obtained by the three algorithms are typically on par.

The `MaxSAT60` benchmark

`MaxSAT60` is a 60-dimensional, weighted instance of the Maximum Satisfiability (MaxSAT) problem. MaxSAT is a notoriously hard combinatorial problem that cannot be solved in polynomial time (unless $\mathbb{P} = \mathbb{NP}$). The goal is to find a binary assignment to the variables that satisfies clauses of maximum total weight. For every i in $\{1, 2, \ldots, d\}$, this benchmark has one clause of the form x_i with a weight of 1 and 638 clauses of the form $\neg x_i \vee \neg x_j$ with a weight of 61. Following Oh et al. [56], Deshwal et al. [21], Wan et al. [80], we normalize these weights to have zero mean and unit standard deviation. This normalization causes the one-variable clauses to have a *negative weight*, i.e., the function value improves if such a clause is not satisfied, which is atypical behavior for a MaxSAT problem. Since the clauses with two variables are satisfied for $x_i = x_j = 0$ and the clauses with one variable of negative weights are never satisfied for $x_i = 0$, the normalized benchmark version

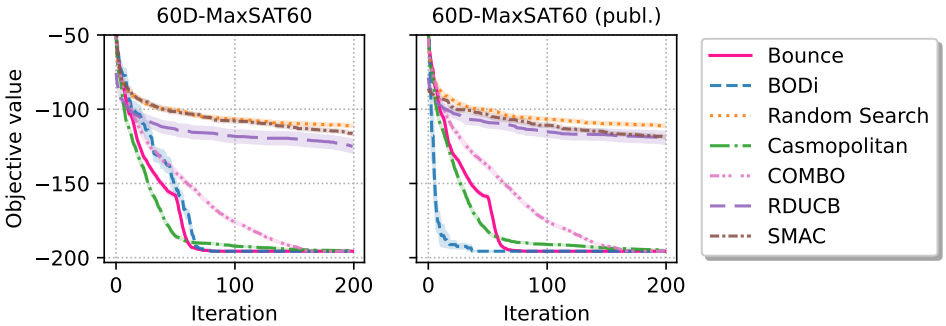


Figure 3.10: `Bounce` and other algorithms on the 60-dimensional weighted maximum satisfiability problem. `Bounce` is the first to find an optimal solution (left). On the published version (right), `Bounce` comes in second after `BODi`.

has a global optimum at $\boldsymbol{x}^* = (0, \ldots, 0)$ by construction. The problem's difficulty is finding an assignment for variables such that *all* two-variable clauses are satisfied and *as many* one-variable clauses as possible are not captured by normalized weights.

Figure 3.10 summarizes the performances of the algorithms. The general version that attains the global optimum for a randomly selected binary assignment is shown on the left. The special case where the global optimum is set to the all-zero assignment is shown on the right.

We observe that `Bounce` requires the smallest number of samples to find an optimal assignment in general, followed by `BODi` and `Casmopolitan`. Only in the special case where the optimum is the all-zero assignment, `BODi` ranks first, confirming the corresponding result in Deshwal et al. [21].

C An evaluation of Bounce on continuous problems

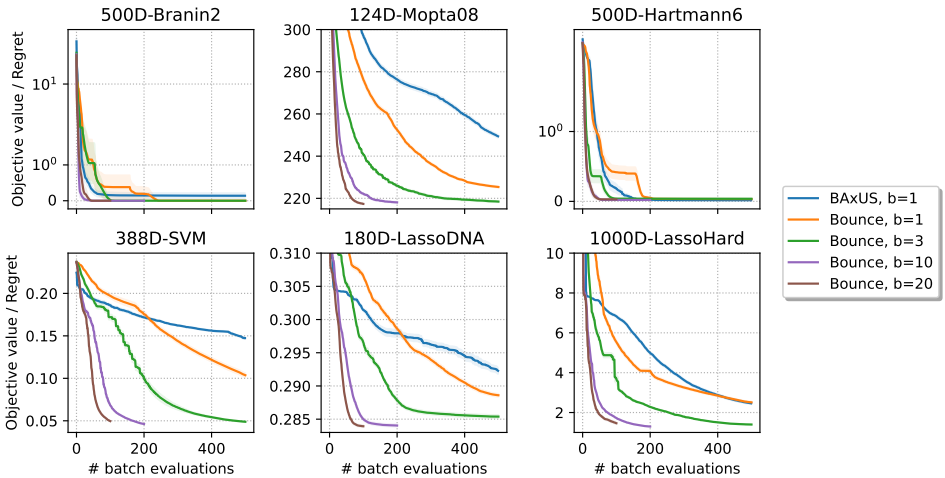


Figure 3.11: Bounce on continuous problems with different batch sizes (plotted in terms of batch evaluations).

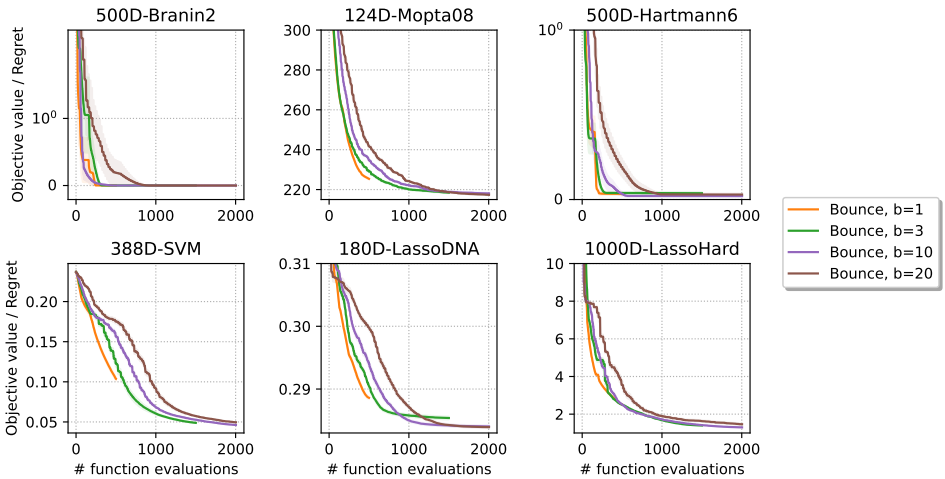


Figure 3.12: Bounce on continuous problems with different batch sizes (plotted in terms of function evaluations).

To showcase the performance and scalability of Bounce, we run it on a set of continuous problems from Papenmeier et al. [60]. The 124-dimensional Mopta08 benchmark is a constrained vehicle optimization problem. We adopt the soft-constrained version from Eriksson and Jankowiak [23]. The 388-dimensional SVM problem [23] concerns the classification performance with an SVR on the slice

localization dataset. The 180-dimensional `LassoDNA` benchmark [70] is a sparse regression problem on a real-world dataset, and the 1000-dimensional `LassoHard` benchmark optimizes over a synthetic dataset. The 500-dimensional `Branin2` and `Hartmann6` problems are versions of the 2- and 6-dimensional benchmark problems where additional dimensions with no effect on the function value were added.

We set the number of function evaluations to $\max(2000, 500B)$ for a batch size of B and configure `Bounce` such that it reaches the input dimensionality after 250 function evaluations for $B \neq 20$ and 500 otherwise. Figures 3.11 and 3.12 show the simple regret for the synthetic `Branin2` and `Hartmann6` problems, and the best function value obtained after a given number of batch (Figure 3.11) or function (Figure 3.12) evaluations for the remaining problems: `Mopta08`, `SVM`, `LassoDNA`, and `LassoHard`.

We observe that `Bounce` always benefits from more parallel function evaluations. The difference between smaller batch sizes, such as $B = 1$ and $B = 3$ or $B = 3$ and $B = 10$, is more remarkable than between larger batch sizes, like $B = 10$ and $B = 20$. Parallel function evaluations prove especially effective on `SVM` and `LassoDNA`. Here, the optimization performance improves drastically. We conclude that a small number of parallel function evaluations already helps to considerably increase the optimization performance.

On the synthetic `Branin2` and `Hartmann6` problems, `Bounce` quickly converges to the global optimum. Here, we see that a larger number of parallel function evaluations also helps in converging to a better solution.

In Figure 3.11, we also compare to `BAxUS` by Papenmeier et al. [60], which does not support parallel function evaluations and is therefore run with a batch size of 1. We observe that `Bounce` with a batch size of 1 outperforms `BAxUS` on all benchmarks except for `Hartmann6`, showcasing `Bounce`'s overall performance improvements even for continuous problems and single-element batches.

D Batched evaluations on mixed and combinatorial problems

In addition to Figure 3.6, which shows the best objective value in relation to the number of batches, Figure 3.13 shows the objective value in terms of function evaluations.

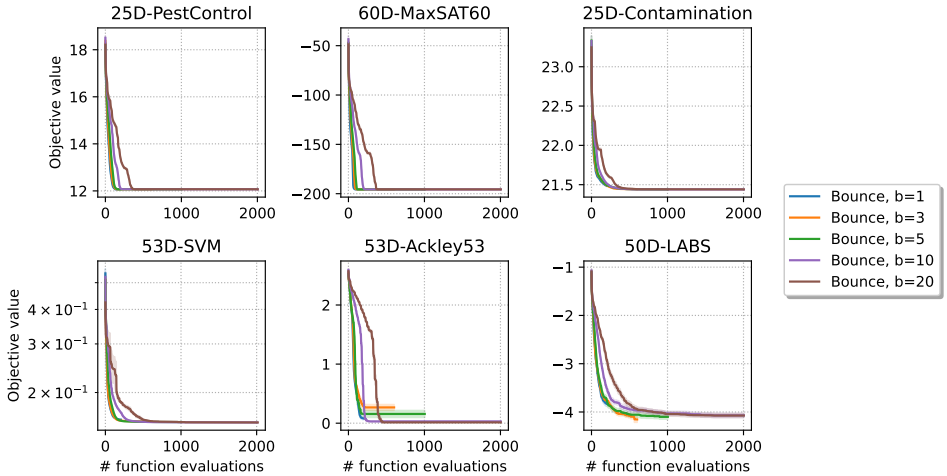


Figure 3.13: `Bounce` on mixed and combinatorial problems with different batch sizes (plotted in terms of function evaluations).

The figure confirms that `Bounce` leverages parallel function values efficiently.

E Low-sequency version of `Bounce`

We show how we can bias `Bounce` towards low-sequency solutions. A binary vector has low sequency if there are few changes from 0 to 1 or vice versa. Similarly, a solution to a categorical or ordinal problem has low sequency if there are few changes from one category or level to another. We remove the random signs (for binary and continuous variables) and the random offsets (for categorical and ordinal variables) from the `Bounce` embedding. We conduct this study to show a) that `Bounce` can outperform `BODi` on the unmodified versions of the benchmark problems if we introduce a similar bias towards low-sequency solutions and b) that the random signs empirically show to remove biases towards low-sequency solutions. However, we want to emphasize that the results of this section are not representative of the performance of `Bounce` on arbitrary real-world problems. Nevertheless, if one knows that the problem has a low-sequency structure, then `Bounce` can be configured to exploit this structure and outperform `BODi`.

Figure 3.14 shows the results of the low-sequency version of `Bounce` on the original benchmarks from Section 4. We observe that `Bounce` outperforms `BODi` and the other algorithms on the unmodified versions of the benchmark problems. This shows that `Bounce` can outperform `BODi` on the unmodified version of the benchmarks if we introduce a similar bias towards low-sequency solutions.

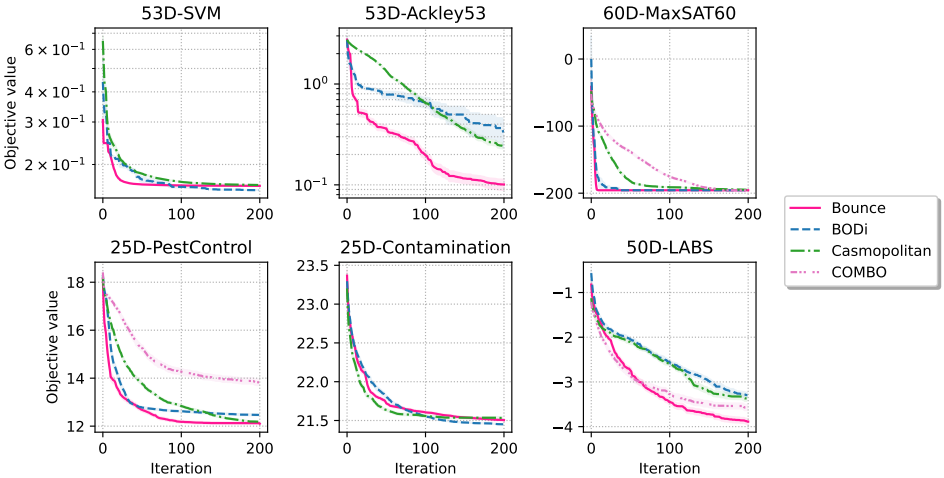


Figure 3.14: 'Low-sequency' version of `Bounce` on the original benchmarks from Section 4: with a bias towards low-sequency solutions, `Bounce` outperforms `BODi` on the original versions of the benchmark problems.

Figure 3.15 shows the results of the low-sequency version of `Bounce` on the flipped benchmarks from Section 4. The low-sequency version of `Bounce` is robust towards the randomization of the optimal point.

F Effect of trust-region management

`Bounce` uses a novel trust-region management that differs from previous approaches in that it allows arbitrary trust region base lengths in $[L_{\min}, L_{\max}]$. This strategy allows `Bounce` to efficiently leverage parallel function evaluations, which we refer to as batch acquisition.

We compare `Bounce`'s trust-region management strategy with the strategy employed by `BAxUS` [60] for purely continuous benchmarks and an adapted version of `Casmopolitan`'s [80] strategy for mixed or discrete-space benchmarks. For the latter, the difference is that in `Bounce` we reduce the failure tolerance as described by Papenmeier et al. [60]: We first calculate the number of times the TR base length needs to be reduced to reach the minimum TR base length as $k = \left\lfloor \log_{1.5^{-1}} \frac{L_{\min}}{L_{\text{init}}} \right\rfloor$, and then find the failure tolerance for the i-th target space by $\tau^i_{\text{fail}} = \max\left(1, \min\left(\left\lfloor \frac{m^s_i}{k} \right\rfloor\right)\right)$, where m^s_i is the budget for the i-th target space [60]. This change in regard to `Casmopolitan` is necessary because `Casmopolitan`'s failure tolerance of 40 [80] would cause the algorithm to spend a large part of the evaluation budget in initial target spaces of low dimensionality.

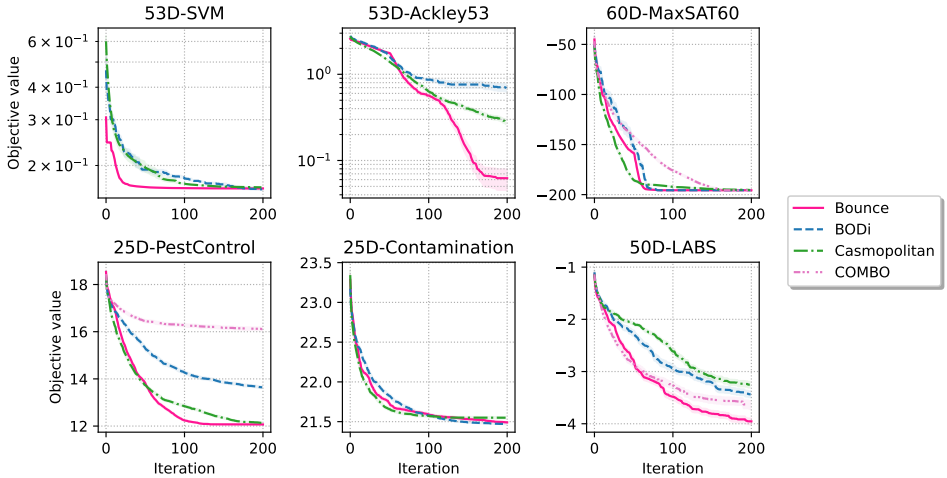


Figure 3.15: "Low-sequency" version of `Bounce` on the modified benchmarks from Section 4.

We show the effect of this strategy on discrete and mixed-space benchmarks in Figure 3.16. Figure 3.17 shows the effect on continuous benchmarks. All experiments are replicated 50 times. We observe that the proposed trust-region management method not only enables efficient batch parallelism but also improves the performance for single-function evaluations when compared to the respective baseline for the TR management that we stated above. The TR management proposed here usually provides better solutions at the same number of batches than the respective baseline TR management. There are a few exceptions. On `Labs`, the previous TR management strategy outperforms `Bounce`'s strategy by a small margin for batch sizes 5, 10, and 20, and on `Ackley53`, the previous strategy converges to a better solution for a batch size of three, five, and ten.

On the continuous benchmarks (Figure 3.17), we observe that the new TR management strategy also improves the performance for single function evaluations with the exception of `Branin2`. Similar to the mixed and combinatorial-space benchmarks, large batch sizes (10 and 20) bring little advantage compared to a batch size of 5. For `Hartmann6`, the previous strategy with a batch size of 20 performs worse than 3.

G Implementation Details

We implement `Bounce` in Python using the `BoTorch` [5] and `GPyTorch` [29] libraries.

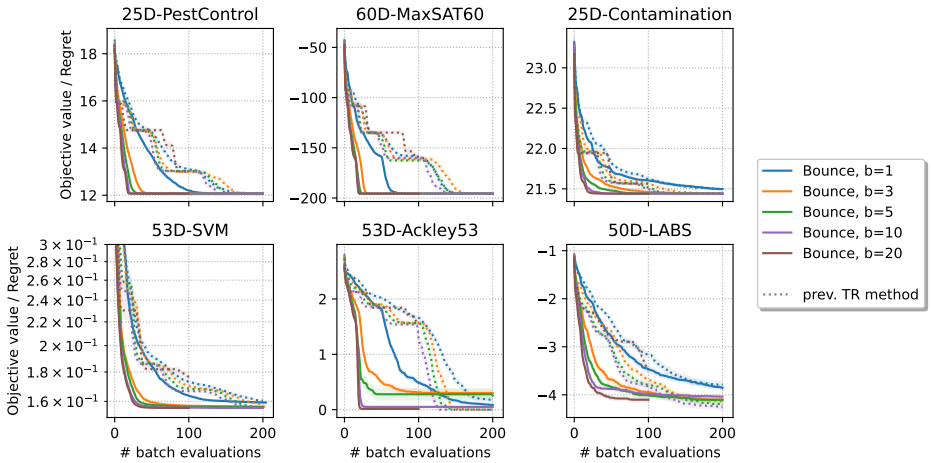


Figure 3.16: Effect of the proposed trust region management on the discrete and mixed benchmarks. Here, the baseline is the TR management of `Casmopolitan`. We see that often, even large batch sizes for the old TR management strategy do not outperform the proposed new strategy with a batch size of 1.

We employ a $\Gamma(1.5, 0.1)$ prior on the lengthscales of both kernels and a $\Gamma(1.5, 0.5)$ prior on the signal variance. We further use a $\Gamma(1.1, 0.1)$ prior on the noise variance.

Motivated by Wan et al. [80] and Eriksson et al. [25], we use an initial trust region baselength of 40 for the combinatorial variables, and 0.8 for the continuous variables. We maintain two separate TR shrinkage and expansion parameters (γ_{cmb} and γ_{cnt}) for the combinatorial and continuous variables, respectively such that each TR base length reaches its respective minimum of 1 and 2^{-7} after a given number of function evaluations. When `Bounce` finds a better or worse solution, we increase or decrease both TR base lengths.

We use the author's implementations for `COMBO`[6], `BODi`[7], `RDUCB`[8], `SMAC`[9], and `Casmopolitan`[10]. We use the same settings as the authors for `COMBO`, `RDUCB`, `SMAC`, and `BODi`. For `Casmopolitan`, we use the same settings as the authors for benchmarks reported in Wan et al. [80] and set the initial trust region base length to 40 otherwise.

Due to its high-memory footprint, we ran `BODi` on NVidia A100 80GB GPUs for

[6] `https://github.com/QUVA-Lab/combo`, unspecified license, last access: 2023-05-04

[7] `https://github.com/aryandeshwal/bodi`, no license provided, last access: 2023-05-04

[8] `https://github.com/huawei-noah/HEBO/tree/master/RDUCB`, `MIT` license, last access: 2023-10-20

[9] `https://github.com/automl/pysmac`, `AGPL-3.0` license, last access 2023-10-20

[10] `https://github.com/xingchenwan/casmo`, `MIT` license, last access: 2023-05-04

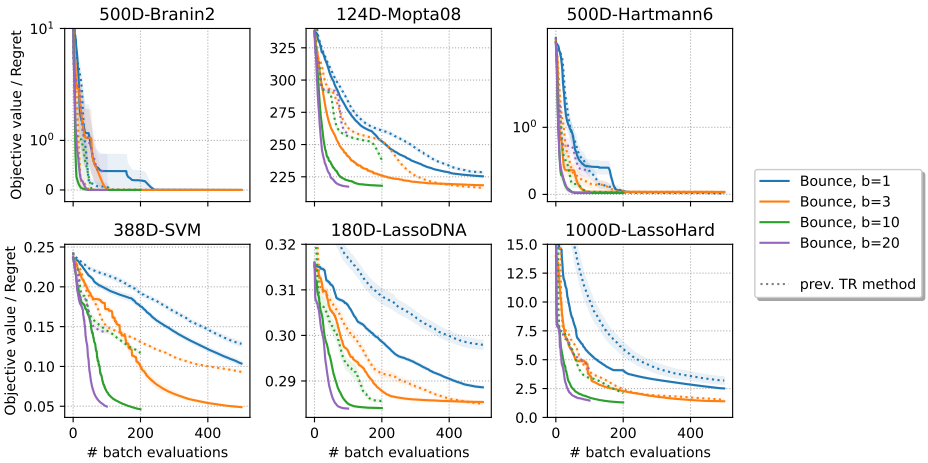


Figure 3.17: Effect of the proposed trust-region management on the continuous benchmarks. Here, the baseline is the TR management of `BAxUS`. We observe that larger batches help to improve performance for both methods. The new strategy, however, outperforms the baseline in almost all cases with the exception of `LassoDNA` with a batch size of 3.

300 GPU/h. We ran `Bounce` on NVidia A40 GPUs for 2,000 GPU/h. We ran the remaining methods for 20,000 GPU/h on one core of Intel Xeon Gold 6130 CPUs with 60GB of memory.

G.1 Optimization of the acquisition function

We use different strategies to optimize the acquisition function depending on the type of variables present in a problem.

Continuous problems. For purely continuous problems, we follow a similar approach as `TuRBO` [25]. In particular, we use the lengthscales of the GP posterior to shape the TR. We use gradient descent to optimize the acquisition function within the TR bounds with 10 random restarts and 512 raw samples. For a batch size of 1, we use analytical EI. For larger batch sizes, we use the `BoTorch` implementation of qEI [86, 67, 5].

Binary problems. Similar to Wan et al. [80], we use discrete TRs centered on the current best solution. A discrete TR describes all solutions with a certain Hamming distance to the current best solution.

We use a local search approach to optimize the acquisition function for all problems

with a combinatorial search space of only binary variables: When starting the optimization, we first create a set of $\min(5000, \max(2000, 200 \cdot d_i))$ random solutions. The choice of the number of random solutions is based on `TuRBO` [25]. For each candidate, we first draw L_i indices uniformly at random from $\{1, \dots, d_i\}$ without replacement, where L_i is the TR length at the i-th iteration. We then sample d_i values in $\{0, 1\}$ and set the candidate at the sampled indices to the sampled values. All other values are set to the values of the current best solution. Note that this construction ensures that each candidate solution lies in the TR bounds of the current best solution. We add all neighbors (i.e., points with a Hamming distance of 1) of the current best solution to the set of candidates. This is inspired by `BODi` [21]. We find the 20 candidates with the highest acquisition function value and use local search to optimize the acquisition function within the TR bounds: At each local search step, we create all direct neighbors that do not coincide with the current best solution or would violate the TR bounds. We then move the current best solution to the neighbor with the highest acquisition function value. We repeat this process until the acquisition function value does not increase anymore. Finally, we return the best solution found during local search.

Categorical problems. We adopt the approach for binary problems, i.e., we first create a set of random solutions with the same size as for purely binary problems and start the local search on the 20 best initial candidates.

Suppose the number of categorical variables of the problem is smaller or equal to the current TR length. In that case, we sample, for each candidate and each categorical variable, an index uniformly at random from $\{1, 2, \dots, |v_i|\}$ where $|v_i|$ is the number of values of the i-th categorical variable. We then set the candidate at the sampled index to 1 and all other values to 0.

If the number of categorical variables of the problem is larger than the current TR length L_i, we first sample L_i categorical variables uniformly at random from $[d_i]$ without replacement. For each initial candidate and each sampled categorical variable, we sample an index uniformly at random, for which we set the categorical variable to 1 and all other values to 0. The values for the variables that were not sampled are set to the values of the current best solution.

As for the binary case, we add all neighbors of the current best solution to the set of candidates, and we sample the 20 candidates with the highest acquisition function value.

We then use multi-start local search to optimize the acquisition function within the TR bounds while neighbors are created by changing the index of one categorical variable. Again, we repeat until convergence and return the best solution found

during the local search.

Ordinal problems. The construction for ordinal problems is similar to the one for categorical problems.

Suppose the number of ordinal variables of the problem is smaller or equal to the current TR length. In that case, we sample an ordinal value uniformly at random to set the ordinal variable for each candidate and each ordinal variable. Otherwise, we choose as many ordinal variables as each candidate's current TR length and sample an ordinal value uniformly at random to set the ordinal variable. We add all neighbors of the current best solution, all solutions where the distance to the current best solution is 1 for one ordinal variable, to the set of candidates. We then sample the 20 candidates with the highest acquisition function value and use local search to optimize the acquisition function within the TR bounds. In the local search, we increment or decrement the value of a single ordinal variable.

Mixed problems. Mixed problems are effectively handled by treating every variable type separately. Again, we create a set of initial random solutions where the values for the different variable types are sampled according to the abovementioned approaches. This can lead to solutions outside the TR bounds. We remove these solutions and find the 20 best candidates only across the solutions within the TR bounds.

When optimizing the acquisition function, we differentiate between continuous and combinatorial variables. We optimize the continuous variables by gradient descent with the same settings as purely continuous problems. When optimizing, we fix the values for the combinatorial values.

We use local search to optimize the acquisition function for the combinatorial variables. In this step, we fix the values for the continuous variables and only optimize the combinatorial variables. We create the neighbors by creating neighbors within Hamming distance of 1 for each combinatorial variable type and then combining these neighbors. Again, we run a local search until convergence.

We do five interleaved steps, starting with the continuous variables and ending with the combinatorial variables.

G.2 Kernel choice

We use the `CoCaBo` kernel [68] with one global lengthscale for the combinatorial and ARD for the continuous variables:

$$
\begin{aligned}
k(\boldsymbol{x}, \boldsymbol{x}') &= \sigma_f^2(\rho k_{\text{cmb}}(\boldsymbol{x}_{\text{cmb}}, \boldsymbol{x}'_{\text{cmb}}) k_{\text{cnt}}(\boldsymbol{x}_{\text{cnt}}, \boldsymbol{x}'_{\text{cnt}}) \\
&\quad + (1-\rho)(k_{\text{cmb}}(\boldsymbol{x}_{\text{cmb}}, \boldsymbol{x}'_{\text{cmb}}) + k_{\text{cnt}}(\boldsymbol{x}_{\text{cnt}}, \boldsymbol{x}'_{\text{cnt}}))) \\
k_{\text{cmb}}(\boldsymbol{x}_{\text{cmb}}, \boldsymbol{x}'_{\text{cmb}}) &= \left(1 + \frac{\sqrt{5} r_{\text{cmb}}}{\ell_{\text{cmb}}} + \frac{5 r_{\text{cmb}}^2}{3\ell_{\text{cmb}}^2}\right) \exp\left(-\frac{\sqrt{5} r_{\text{cmb}}}{\ell_{\text{cmb}}}\right) \\
r_{\text{cmb}} &= \|\boldsymbol{x}_{\text{cmb}} - \boldsymbol{x}'_{\text{cmb}}\| \\
k_{\text{cnt}}(\boldsymbol{x}_{\text{cnt}}, \boldsymbol{x}'_{\text{cnt}}) &= \left(1 + \sqrt{5} r_{\text{cnt}} + 5 r_{\text{cnt}}^2\right) \exp\left(-\sqrt{5} r_{\text{cnt}}\right) \\
r_{\text{cnt}} &= \sqrt{\sum_{i \in [d];\, d_i \text{ continuous}} \frac{(x_i - x'_i)^2}{\ell_{\text{cnt},i}^2}}
\end{aligned}
$$

where $\boldsymbol{x}_{\text{cnt}}$ and $\boldsymbol{x}_{\text{cmb}}$ are the continuous and combinatorial variables in $\boldsymbol{x}$, respectively, i.e.,

$$
\begin{aligned}
\boldsymbol{x}_{\text{cmb}} &= (x_i : \ d_i \text{ is combinatorial}) \\
\boldsymbol{x}_{\text{cnt}} &= (x_i : \ d_i \text{ is continuous})
\end{aligned}
$$

and $\rho \in [0, 1]$ is a tradeoff parameter learned jointly with the other hyperparameters during the likelihood maximization. Here, ℓ_{cmb} and $\ell_{\text{cnt},i}$ are the lengthscale hyperparameters.

H Additional Analysis of `BODi` and `COMBO`

H.1 Analysis of `BODi`

Binary problems. `BODi` by Deshwal et al. [21] uses a dictionary of anchor points $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m)$ to encode a candidate point $\boldsymbol{z}$. In particular, the i-th entry of the m-dimensional embedding $\phi_{\boldsymbol{A}}(\boldsymbol{z})$ is obtained by computing the Hamming-distance between $\boldsymbol{z}$ and $\boldsymbol{a}_i$. Notably, Deshwal et al. [21] choose the dimensionality of the embedding m as 128, which is larger than the dimensionality of the objective functions themselves.

The sampling procedure for dictionary elements $\boldsymbol{a}_i$ is chosen to yield a wide range of *sequencies*. The sequency of a binary string is defined as the number of times the string

changes from 0 to 1 and vice versa. Deshwal et al. [21] propose two approaches to generate the dictionary elements: (i) using binary wavelets, and (ii) by first drawing a Bernoulli parameter $\theta_i \sim \mathcal{U}(0,1)$ for each $i \in [m]$ and then drawing a binary string $\boldsymbol{a}_i$ from the distribution $\mathcal{B}(\theta_i)$. The latter approach is their default method.

We prove that `BODi` generates an all-zeros (or, by a similar symmetry argument, all-ones) representer point with a probability that is significantly higher than 2^{-D}. Moreover, we claim without proof that a similarly increased probability holds for points with low Hamming distance to all-zeros or all-ones. This is consistent with the intention of `BODi` to sample points of diverse sequency and not necessarily an issue.

However, we claim without proof that having such points in the dictionary substantially increases the probability that `BODi` evaluates the all-zeros (and, by symmetry, the all-ones) points. That hypothesis is consistent with our observation in Section 4.6 that `BODi` has a much higher chance of finding good or optimal solutions when they are near the all-zero point.

Deshwal et al. [21] choose the dictionary to have $m = 128$ dictionary elements. Given a Bernoulli parameter θ_i, the probability that the i-th dictionary point $\boldsymbol{a}_i$ is a point of sequency zero is given by $\theta_i^D + (1-\theta_i)^D$:

$$\mathbb{P}(\text{“zero sequency”} \mid \theta_i) = \prod_{i=1}^{m} \theta_i^D + (1-\theta_i)^D \tag{3.7}$$

Then, since θ_i follows a uniform distribution, the overall probability for a point of zero sequency is given by

$$\begin{aligned}
\mathbb{P}(\text{“zero sequency”}) &= 1 - \underbrace{\prod_{i=1}^{m} \int_0^1 \left(1 - \theta_i^D - (1-\theta_i)^D\right) \underbrace{p(\theta_i)}_{=1} \, d\theta_i}_{\text{prob. of } m \text{ times not zero sequency}} && (3.8)\\
&= 1 - \prod_{i=1}^{m} \left(\theta_i - \frac{\theta_i^{D+1}}{D+1} + \frac{(1-\theta_i)^{D+1}}{D+1} \right) \Bigg|_{\theta_i=1} && (3.9)\\
&= 1 - \left(1 - \frac{2}{D+1}\right)^m, && (3.10)
\end{aligned}$$

i.e., the probability of at least one dictionary element being of sequency zero is $1 - \left(1 - \frac{2}{D+1}\right)^m$. The probability of `BODi`'s dictionary to contain a zero-sequency point increases with the number of dictionary elements m and decreases with the function dimensionality D (see Figure 3.18).

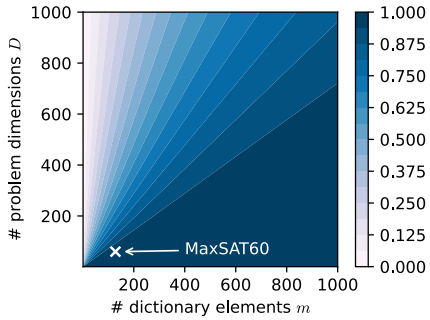


Figure 3.18: Probabilities of `BODi` to contain a zero-sequency solution for different choices of the dictionary size m and the function dimensionality D.

For instance, for the 60-dimensional `MaxSAT60` benchmark, the probability that at least one dictionary element is of sequency zero is $1 - \left(1 - \frac{2}{60+1}\right)^{128} \approx 0.986$ (see Figure 3.18).

Note that at least one point $\boldsymbol{z}^*$ has a probability of $\leq 1/2^d$ to be drawn. The probability of the dictionary containing that $\boldsymbol{z}^*$ is less than or equal to $1 - \left(1 - 1/2^d\right)^m$ which is already less than 0.01 for $d = 14$ and $m = 128$. In Section 4, we show that randomizing the optimal point structure leads to performance degradation for `BODi`. We hypothesize this is due to the reduced probability of the dictionary containing the optimal point after randomization.

Categorical Problems. We calculate the probability that `BODi` contains a vector in its dictionary where all elements are the same. For categorical problems, `BODi` first samples a vector $\boldsymbol{\theta}$ from the $\tau_{\max}$-simplex $\Delta^{\tau_{\max}}$ for each vector $\boldsymbol{a}_i$ in the dictionary, with $\tau_{\max}$ being the maximum number of categories across all categorical variables of a problem. We assume that all variables have the same number of categories as is the case for the benchmarks in Deshwal et al. [21]. Let τ be the number of categories of the variables. For each element in $\boldsymbol{a}_i$, `BODi` draws a value from the categorical distribution with probabilities $\boldsymbol{\theta}$. While line 7 in Algorithm 5 in Deshwal et al. [21] might suggest that the elements in $\boldsymbol{\theta}$ are shuffled for every element in $\boldsymbol{a}_i$, we observe that $\boldsymbol{\theta}$ remains fixed based on the implementation provided by the authors[11]. The random resampling of elements from $\boldsymbol{\theta}$ is probably only used for benchmarks where the number of realizations differs between categorical variables.

Then, for a fixed $\boldsymbol{\theta}$, the probability that all D elements in $\boldsymbol{a}_i$ for any i are equal to

[11] See `https://github.com/aryandeshwal/bodi/blob/aa507d34a96407b647bf808375b5e162ddf1066bodi/categorical_dictionary_kernel.py#L18`

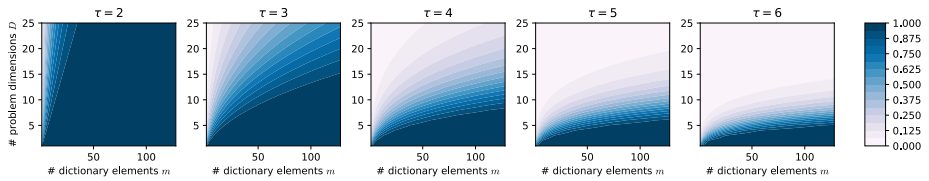


Figure 3.19: Probabilities of `BODi`'s dictionary to contain at least one categorical point where each category has the same value. The probability increases with the number of dictionary elements m but decreases with the number of categories τ and the number of problem dimensions D.

some fixed value $t \in \{1, 2, \ldots, \tau\}$ is given by θ_t^d. The probability that, for any of the m dictionary elements, all D elements in $\boldsymbol{a}_i$ are equal to some fixed value $t \in \{1, 2, \ldots, \tau\}$ is given by

$$\mathbb{P}(\text{“all one specific category”}) = 1 - \prod_{i=1}^{m} \int (1 - \theta_t^D) p(\theta_t) d\theta_t. \tag{3.11}$$

We note that $\boldsymbol{\theta}$ follows a Dirichlet distribution with $\boldsymbol{\alpha} = \mathbf{1}$ [54]. Then, θ_t is marginally Beta$(1, \tau - 1)$-distributed [54]. With that, Eq. (3.11) becomes

$$\mathbb{P}(\text{“all one specific category”}) = 1 - \prod_{i=1}^{m} \mathbb{E}_{\theta_t \sim \text{Beta}(1,\tau-1)} \left[1 - \theta_t^D\right] \tag{3.12}$$

$$= 1 - \prod_{i=1}^{m} 1 - \mathbb{E}_{\theta_t \sim \text{Beta}(1,\tau-1)} \left[\theta_t^D\right] \tag{3.13}$$

now, by using the equality $\mathbb{E}[x^D] = \prod_{r=0}^{D-1} \frac{\alpha+r}{\alpha+\beta+r}$ for the D-th raw moment of a Beta(α, β) distribution [54]

$$= 1 - \left(1 - \prod_{r=0}^{D-1} \frac{1+r}{\tau+r}\right)^m \tag{3.14}$$

$$= 1 - \left(1 - \frac{1}{\tau} \cdot \frac{2}{\tau+1} \cdot \ldots \cdot \frac{D}{\tau+D-1}\right)^m \tag{3.15}$$

$$= 1 - \left(1 - \frac{D!\,\tau!}{(\tau+D-1)!}\right)^m \tag{3.16}$$

We discussed in Section 4.3 that the `PestControl` benchmark obtains a good solution at $\boldsymbol{x} = \mathbf{5}$. One could assume that `BODi` performs well on this benchmark because its dictionary will likely contain this point. However, we observe that the

probability is effectively zero for $\tau = 5$, $m = 128$, and $D = 25$ (see Figure 3.19), which are the choices for the `PestControl` benchmark in Deshwal et al. [21]. This raises the question of (i) whether our hypothesis is wrong and (ii) what the reason for `BODi`'s performance degradation on the `PestControl` benchmark is.

We show that `BODi`'s reference implementation differs from the algorithmic description in an important detail, causing `BODi` to be considerably more likely to sample category five on `PestControl` (or the "last" category for arbitrary benchmarks) than any other category.

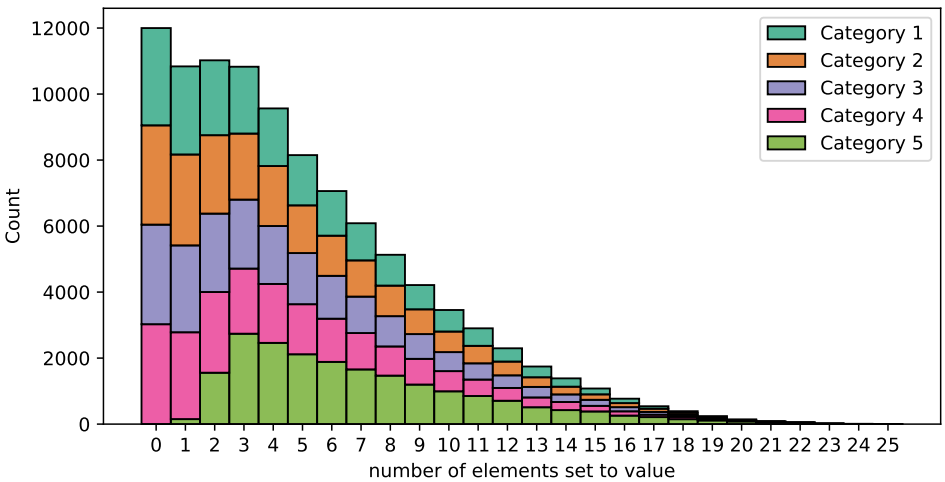


Figure 3.20: Histograms over the number of dictionary element entries set to each category for 20,000 repetitions of the sampling of dictionary elements for the `PestControl` benchmark. For each of the five categories and each value on the x-axis, the figure shows how often the number of entries in a dictionary element equals the value on the x-axis for the given category. For example, the count for $x = 0$ and category 5 is zero, indicating that all 20,000 dictionary points had at least one entry '5'. There is a considerably higher chance for a dictionary element entry to be set to category five than to any of the other categories.

Figure 3.20 shows five histograms over the number of dictionary elements set to each category. The values on the x-axis give the number of elements in a 25-dimensional categorical vector being set to a specific category. One would expect that the histograms have a similar shape regardless of the category. However, for category 5, we see that more elements are set to this category than for the other categories: The probability of k elements being set to category 5 is almost twice as high as the probability of being set to another category for $k \geq 3$. In contrast, the probability that no element in the vector belongs to category 5 is virtually zero. This behavior is beneficial for the `PestControl` benchmark, which obtained the best value found during our experiments for $\boldsymbol{x}^* = (5, 5, \ldots, 5, 1)$ (see Section 4). While

we see that the probability of each dictionary entry being set to category 5 is very low, we assume that we sample sufficiently many dictionary elements within a small Hamming distance to the optimizer such that BODi's GP can use this information to find the optimizer.

The reason for oversampling of the last category lies in a rounding issue in sampling dictionary elements. In particular, for a given dictionary element $\boldsymbol{a}_i$ and a corresponding vector $\boldsymbol{\theta}$ with $|\boldsymbol{\theta}| = \tau$, for each $i \in \{1, 2, \ldots, \tau - 1\}$, Deshwal et al. [21] set $\lfloor D\boldsymbol{\theta}_i \rfloor$ elements to category i. The remaining $D - \sum_{i=1}^{\tau-1} \lfloor D\boldsymbol{\theta}_i \rfloor$ elements are then set to category τ. This causes the last category to be overrepresented in the dictionary elements. For the choices of the PestControl benchmark, $D = 25$ and $\tau = 5$, the first four categories had a probability of ≈ 0.1805. In contrast, the last one had a probability of ≈ 0.278 for 10^8 simulations[12]. We assume that the higher probability of the last category is the reason for the performance difference between the modified and the unmodified version of the PestControl benchmark.

H.2 COMBO on Categorical Problems

On the categorical PestControl benchmark, COMBO [56] behaved similar to BODi [21]

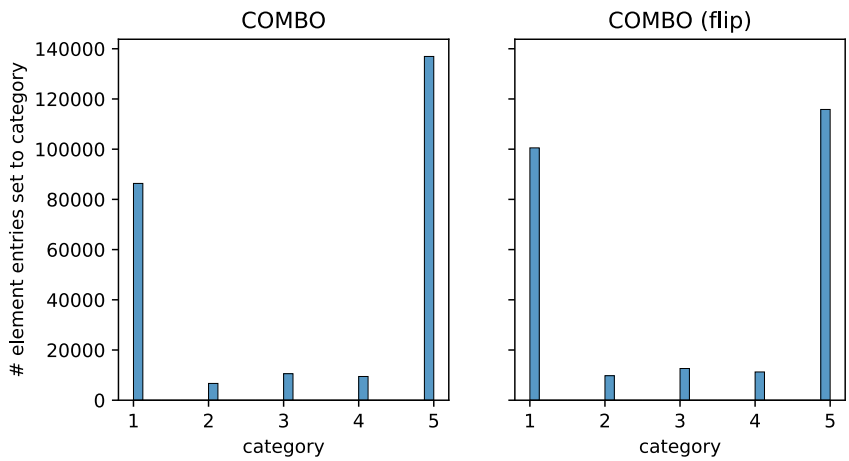


Figure 3.21: The histogram shows how many element entries of a 25-dimensional dictionary element are set to each of the five categories. There is a considerably higher chance for a dictionary element entry to be set to category 1 or 5 than to one of the other categories.

[12]The 95% confidence intervals for categories 1–5 are (0.1799, 0.1807), (0.1802, 0.1810), (0.1803, 0.1811), (0.1801, 0.1809), (0.2775, 0.2783). Pairwise Wilcoxon signed-rank tests between categories 1–4 and category 5 gives p values of 0 ($W \approx 4.7 \cdot 10^{10}$ each).

The histogram in Figure 3.21 shows how many element entries of a 25-dimensional dictionary element are set to each of the five categories. We see that the first and the last categories are over-represented on both benchmark versions. As discussed in Section 4, this benchmark attains the best observed value for $\boldsymbol{x}^* = (5, 5, \ldots, 5, 1)$. We observe that on the original and modified version of the benchmark, where the categories are shuffled, `COMBO` sets disproportionately many elements to categories one and five. Note that for the modified version of the benchmark, all categories are equally likely in the optimal solution.

We argue that this behavior is at least partially caused by implementation error in the construction of the adjacency matrix and the Laplacian for categorical problems[13]. This error causes categorical variables to be modeled like ordinal variables. According to Oh et al. [56], categorical variables are modeled as a complete graph (see Figure 3.22).

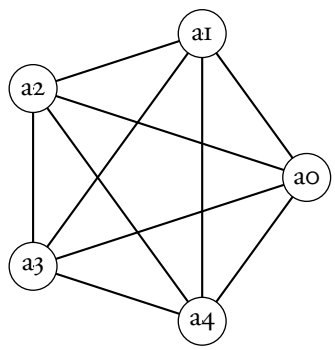


Figure 3.22: Following the description in the paper of Oh et al. [56], a categorical variable with five categories is modeled as a complete graph.

Looking into the source code of `COMBO`, we find the adjacency matrix for the first category of a categorical variable with five categories is constructed as

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

which is the adjacency matrix for a five-vertex path graph.

[13] `https://github.com/QUVA-Lab/COMBO/blob/9529eabb86365ce3a2ca44fff08291a09a853ca2/COMBO/experiments/test_functions/multiple_categorical.py#L137`, last access: 2023-04-26

References

[1] Webpage of the Fourth Max-SAT Evaluation. `http://www.maxsat.udl.cat/09/index.php?disp=submitted-benchmarks`. Last access: 2023-05-02.

[2] MaxSAT Evaluation 2018 : Solver and Benchmark Descriptions. 2018. URL `http://hdl.handle.net/10138/237139`. Publisher: Department of Computer Science, University of Helsinki.

[3] André Abramé and Djamal Habet. AHMAXSAT: Description and evaluation of a branch and bound Max-SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 9(1):89–128, 2014.

[4] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. MaxSAT Evaluation 2018: New developments and detailed results. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):99–131, 2019.

[5] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[6] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. *International Conference on Machine Learning*, pages 462–471. PMLR, 2018.

[7] Petr Baudiš and Petr Pošík. Online Black-Box Algorithm Portfolios for Continuous Optimization. *Parallel Problem Solving from Nature – PPSN XIII*, pages 40–49, Cham, 2014. Springer International Publishing.

[8] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(2), 2012.

[9] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, 2011.

[10] Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

[11] Mickaël Binois, David Ginsbourger, and Olivier Roustant. A warped kernel improving robustness in Bayesian optimization via random embeddings. *International Conference on Learning and Intelligent Optimization (LION)*, pages 281–286. Springer, 2015.

[12] Mickaël Binois, David Ginsbourger, and Olivier Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of global optimization*, 76(1):69–90, 2020.

[13] Jim Boelrijk, Bernd Ensing, Patrick Forré, and Bob WJ Pirok. Closed-loop automatic gradient design for liquid chromatography using Bayesian optimization. *Analytica Chimica Acta*, 1242, 2023.

[14] Mohamed Amine Bouhlel, Nathalie Bartoli, Rommel G. Regis, Abdelkader Otsmane, and Joseph Morlier. Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method. *Engineering Optimization*, 50(12):2038–2053, 2018.

[15] Benjamin Burger, Phillip M Maffettone, Vladimir V Gusev, Catherine M Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M Alston, Buyi Li, Rob Clowes, et al. A mobile robotic chemist. *Nature*, 583(7815):237–241, 2020.

[16] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian Gait Optimization for Bipedal Locomotion. *Learning and Intelligent Optimization*, pages 274–290. Springer International Publishing, 2014.

[17] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[18] Zachary Cosenza, Raul Astudillo, Peter Frazier, Keith Baar, and David E Block. Multi-Information Source Bayesian Optimization of Culture Media for Cellular Agriculture. *Biotechnology and Bioengineering*, 2022.

[19] Samuel Daulton, Xingchen Wan, David Eriksson, Maximilian Balandat, Michael A. Osborne, and Eytan Bakshy. Bayesian Optimization over Discrete and Mixed Spaces via Probabilistic Reparameterization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[20] Aryan Deshwal, Syrine Belakaria, Janardhan Rao Doppa, and Dae Hyun Kim. Bayesian optimization over permutation spaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6515–6523, 2022.

[21] Aryan Deshwal, Sebastian Ament, Maximilian Balandat, Eytan Bakshy, Janardhan Rao Doppa, and David Eriksson. Bayesian Optimization over High-Dimensional Combinatorial Spaces via Dictionary-based Embeddings. *International Conference on Artificial Intelligence and Statistics*, pages 7021–7039. PMLR, 2023.

[22] Adel Ejjeh, Leon Medvinsky, Aaron Councilman, Hemang Nehra, Suraj Sharma, Vikram Adve, Luigi Nardi, Eriko Nurvitadhi, and Rob A. Rutenbar. HPVM2FPGA: Enabling True Hardware-Agnostic FPGA Programming. *Proceedings of the 33rd IEEE International Conference on Application-specific Systems, Architectures, and Processors*, 2022.

[23] David Eriksson and Martin Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 493–503. PMLR, 27–30 Jul 2021.

[24] David Eriksson and Matthias Poloczek. Scalable Constrained Bayesian Optimization. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 730–738. PMLR, 13–15 Apr 2021.

[25] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5496–5507, 2019.

[26] CA Floudas, HK Fung, SR McAllister, M Mönnigmann, and R Rajgaria. Advances in protein structure prediction and de novo protein design: A review. *Chemical Engineering Science*, 61(3):966–988, 2006.

[27] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[28] Peter I. Frazier and Jialei Wang. *Bayesian Optimization for Materials Design*, pages 45–75. Springer International Publishing, Cham, 2016. ISBN 978-3-319-23871-5.

[29] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[30] Seyede Fatemeh Ghoreishi and Douglas Allaire. Multi-information source constrained Bayesian optimization. *Structural and Multidisciplinary Optimization*, 59:977–991, 2019.

[31] Seyede Fatemeh Ghoreishi and Douglas L Allaire. A fusion-based multi-information source optimization approach using knowledge gradient policies.

2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, page 1159, 2018.

[32] Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, 1990.

[33] Florian Hase, Loïc M Roch, Christoph Kreisbeck, and Alán Aspuru-Guzik. Phoenics: a Bayesian optimizer for chemistry. *ACS central science*, 4(9):1134–1145, 2018.

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

[35] Erik Hellsten, Carl Hvarfner, Leonard Papenmeier, and Luigi Nardi. High-dimensional Bayesian Optimization with Group Testing. *arXiv preprint arXiv:2310.03515*, 2023.

[36] Erik Hellsten, Artur Souza, Johannes Lenfers, Rubens Lacouture, Olivia Hsu, Adel Ejjeh, Fredrik Kjolstad, Michel Steuwer, Kunle Olukotun, and Luigi Nardi. BaCO: A Fast and Portable Bayesian Compiler Optimization Framework. *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2023.

[37] Henry C Herbol, Weici Hu, Peter Frazier, Paulette Clancy, and Matthias Poloczek. Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization. *npj Computational Materials*, 4(1):1–7, 2018.

[38] José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space. *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1470–1479. PMLR, 06–11 Aug 2017.

[39] Yingjie Hu, Jian-Qiang Hu, Yifan Xu, Fengchun Wang, and Rong Zeng Cao. Contamination control in food supply chain. *Proceedings of the 2010 Winter Simulation Conference*, pages 2678–2681. IEEE, 2010.

[40] Zak E Hughes, Michelle A Nguyen, Jialei Wang, Yang Liu, Mark T Swihart, Matthias Poloczek, Peter I Frazier, Marc R Knecht, and Tiffany R Walsh. Tuning Materials-Binding Peptide Sequences toward Gold-and Silver-Binding Selectivity with Bayesian Optimization. *ACS nano*, 15(11):18260–18269, 2021.

[41] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.

[42] Anant Singh Jain and Sheik Meeran. Deterministic job-shop scheduling: Past, present and future. *European journal of operational research*, 113(2):390–434, 1999.

[43] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455, 1998.

[44] Jungtaek Kim, Seungjin Choi, and Minsu Cho. Combinatorial Bayesian optimization with random mapping functions to convex polytopes. *Uncertainty in Artificial Intelligence*, pages 1001–1011. PMLR, 2022.

[45] Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in Bayesian optimization with applications in aerospace engineering. *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[46] Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-Examining Linear Embeddings for High-Dimensional Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1546–1558, 2020.

[47] Chong Liu and Yu-Xiang Wang. Global optimization with parametric function approximation. *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 22113–22136, 2023.

[48] Daniel J Lizotte, Tao Wang, Michael H Bowling, Dale Schuurmans, et al. Automatic Gait Optimization With Gaussian Process Regression. *IJCAI*, volume 7, pages 944–949, 2007.

[49] Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

[50] Matthias Mayr, Faseeh Ahmad, Konstantinos I. Chatzilygeroudis, Luigi Nardi, and Volker Krüger. Skill-based Multi-objective Reinforcement Learning of Industrial Robot Tasks with Planning and Knowledge Integration. *CoRR*, abs/2203.10033, 2022.

[51] Luigi Nardi, David Koeplinger, and Kunle Olukotun. Practical design space exploration. *2019 IEEE 27th International Symposium on Modeling, Analysis,*

and Simulation of Computer and Telecommunication Systems (MASCOTS), pages 347–358. IEEE, 2019.

[52] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian Optimization in Embedded Subspaces. *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research (PMLR)*, pages 4752–4761, 09–15 Jun 2019.

[53] Diana M. Negoescu, Peter I. Frazier, and Warren B. Powell. The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[54] Kai Wang Ng, Guo-Liang Tian, and Man-Lai Tang. Dirichlet and related distributions: Theory, methods and applications. 2011.

[55] Changhai Nie and Hareton Leung. A survey of combinatorial testing. *ACM Computing Surveys (CSUR)*, 43(2):1–29, 2011.

[56] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian Optimization using the Graph Cartesian Product. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[57] Steve O'Hagan, Warwick B Dunn, Marie Brown, Joshua D Knowles, and Douglas B Kell. Closed-loop, multiobjective optimization of analytical instrumentation: gas chromatography/time-of-flight mass spectrometry of the metabolomes of human serum and of yeast fermentations. *Analytical Chemistry*, 77(1):290–303, 2005.

[58] Tom Packebusch and Stephan Mertens. Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, 49(16):165001, 2016.

[59] Daniel Packwood. *Bayesian Optimization for Materials Science*. Springer, 2017.

[60] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[61] Matthias Poloczek and David P Williamson. An experimental evaluation of fast approximation algorithms for the maximum satisfiability problem. *Journal of Experimental Algorithmics (JEA)*, 22:1–18, 2017.

[62] Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. *Advances in neural information processing systems*, 30, 2017.

[63] Warren B Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019.

[64] Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778, 2018.

[65] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006.

[66] Rommel G Regis and Christine A Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

[67] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[68] Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. *International Conference on Machine Learning*, pages 8276–8285. PMLR, 2020.

[69] Artur M Schweidtmann, Adam D Clayton, Nicholas Holmes, Eric Bradford, Richard A Bourne, and Alexei A Lapkin. Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. *Chemical Engineering Journal*, 352:277–282, 2018.

[70] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. *First Conference on Automated Machine Learning (Main Track)*, 2022.

[71] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[72] Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590 (7844):89–96, 2021.

[73] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14:199–222, 2004.

[74] Francisco J. Solis and Roger J-B. Wets. Minimization by Random Search Techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.

[75] Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte Carlo Tree Search based Variable Selection for High Dimensional Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022.

[76] Artur Souza, Leonardo B Oliveira, Sabine Hollatz, Matt Feldman, Kunle Olukotun, James M Holton, Aina E Cohen, and Luigi Nardi. DeepFreak: Learning crystallography diffraction patterns with automated machine learning. *arXiv preprint arXiv:1904.11834*, 2019.

[77] Hampus Gummesson Svensson, Esben Jannik Bjerrum, Christian Tyrchan, Ola Engkvist, and Morteza Haghir Chehreghani. Autonomous drug design with multi-armed bandits. *2022 IEEE International Conference on Big Data (Big Data)*, pages 5584–5592. IEEE, 2022.

[78] Alexander Thebelt, Calvin Tsay, Robert Lee, Nathan Sudermann-Merx, David Walz, Behrang Shafei, and Ruth Misener. Tree ensemble kernels for Bayesian optimization with known constraints over mixed-feature spaces. *Advances in Neural Information Processing Systems*, 35:37401–37415, 2022.

[79] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials Discovery*, 4:18–21, 2016.

[80] Xingchen Wan, Vu Nguyen, Huong Ha, Binxin Ru, Cong Lu, and Michael A. Osborne. Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces. *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10663–10674. PMLR, 18–24 Jul 2021.

[81] Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian global optimization of expensive functions. *Operations Research*, 68(6):1850–1865, 2020.

[82] Ke Wang and Alexander W Dowling. Bayesian optimization for chemical products and functional materials. *Current Opinion in Chemical Engineering*, 36:100728, 2022.

[83] 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 (NeurIPS)*, 33:19511–19522, 2020.

[84] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent Advances in Bayesian Optimization. *ACM Comput. Surv.*, 2023.

[85] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *Journal of Artificial Intelligence Research (JAIR)*, 55:361–387, 2016.

[86] James T Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions. *NeurIPS Workshop on Bayesian Optimization*, 2017.

[87] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[88] Juliusz Krzysztof Ziomek and Haitham Bou Ammar. Are Random Decompositions all we need in High Dimensional Bayesian Optimisation? *International Conference on Machine Learning*, pages 43347–43368. PMLR, 2023.

Paper IV

Exploring Exploration in Bayesian Optimization

Submitted to the Forty-First Conference on Uncertainty in Artificial Intelligence

Leonard Papenmeier*
Lund University

Nuojin Cheng*
University of Colorado Boulder

Stephen Becker
University of Colorado Boulder

Luigi Nardi
Lund University

Abstract

A well-balanced exploration-exploitation trade-off is crucial for successful acquisition functions in Bayesian optimization. However, there is a lack of quantitative measures for exploration, making it difficult to analyze and compare different acquisition functions. This work introduces two novel approaches – observation traveling salesman distance and observation entropy – to quantify the exploration characteristics of acquisition functions based on their selected observations. Using these measures, we examine the explorative nature of several well-known acquisition functions across a diverse set of black-box problems, uncover links between exploration and empirical performance, and reveal new relationships among existing acquisition functions. Beyond enabling a deeper understanding of acquisition functions, these measures also provide a foundation for guiding their design in a more principled and systematic manner.

*Equal contribution.

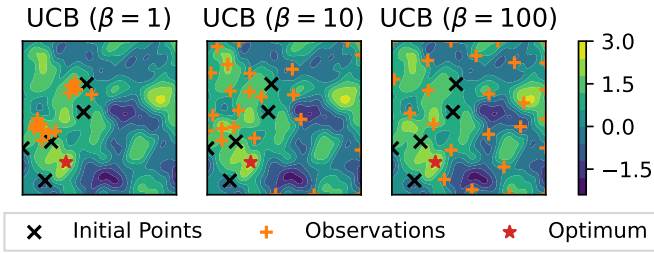


Figure 4.1: Observations of UCB with various β values (1, 10, and 100) on a two-dimensional GP-prior sample reveal the explorative behavior for different β. The black crosses are initial points, the orange plus signs are the observations of the BO phase, and the red star is the optimal location.

1 Introduction

Bayesian optimization (BO) is a widely used method to maximize black-box functions. Given a function $f : \mathcal{X} \to \mathbb{R}$, BO guides the optimization process by sequentially constructing probabilistic surrogates – typically a Gaussian process (GP) – and selecting new sampling points by maximizing an acquisition function (AF) $\alpha : \mathcal{X} \to \mathbb{R}$. The AF is a key component for any BO algorithm that chooses the point to evaluate next. As BO is typically used for problems where the underlying function is expected to be multi-modal, it is crucial that the AF exhibits *explorative* behavior, allowing it to discover different modes of the objective function, but also *exploitative* behavior, allowing it to focus on finding the optimum in a promising region of the search space $\mathcal{X}$. In short, a successful AF should exhibit a good exploration-exploitation trade-off (EETO) that balances these desiderata.

It is widely recognized in the BO community that different AFs exhibit varying degrees of explorative preference, and some AFs include parameters that allow explicit control over this behavior. For example, upper-confidence bound (UCB) [52] employs a parameter β to regulate the level of exploration. Fig. 4.1 illustrates an example of UCB with various β values applied to a two-dimensional GP prior with a length scale of 0.1. In this experiment, five initial points were fixed, and observations were collected over 25 iterations for each β value. A larger β produces a more dispersed layout of observations, indicating increased exploration. Similar behavior is observed in other acquisition functions, such as weighted expected improvement [51] and ε-greedy strategies [55]. However, when comparing AFs from different families – such as UCB versus weighted expected improvement or ε-greedy strategies – assessing their exploration preferences becomes challenging, as no universally accepted metric exists to quantify these characteristics. Understanding the exploration tendencies of different AFs is important, as this knowledge influences the selection of appropriate AFs for real-world problems, especially when a specific level of exploration is required.

In this work, we fill this gap by proposing two novel quantities to quantify the level of exploration. We make the following contributions:

- We propose two novel means for quantifying exploration named observation traveling salesman distance (OTSD) and observation entropy (OE). The first quantity is based on the total Euclidean distance of a traveling salesman tour connecting all the observation points in the search space, while the second adopts an information-theoretic approach and uses the empirical differential entropy of the observations.
- We introduce the first empirical taxonomy of acquisition function exploration based on these new methods, demonstrating their effectiveness in capturing exploration behavior.
- We provide an extensive evaluation across a diverse set of low- and high-dimensional synthetic and real-world benchmarks, demonstrating the exploration behavior of popular acquisition functions and their extensions. OTSD and OE strongly correlate in benchmark problems, cross-validating their reliability.

2 Background and Related Work

2.1 Gaussian Processes

A GP is a stochastic process that models an unknown function. It is characterized by the property that any finite set of function evaluations follows a multivariate Gaussian distribution. Assuming that the prior has a zero mean, a GP is uniquely determined by the current observations $\mathcal{D}_t := \{(\boldsymbol{x}_i, y_{\boldsymbol{x}_i})\}_{i=1}^t$ and the kernel function $\kappa(\boldsymbol{x}, \boldsymbol{x}')$. Given these, at stage t, the predicted mean of $y_{\boldsymbol{x}}$ at a new point $\boldsymbol{x}$ is $\mu_t(\boldsymbol{x}) = \boldsymbol{\kappa}_t(\boldsymbol{x})^T(\boldsymbol{K}_t)^{-1}\boldsymbol{y}_t$, and the predicted covariance between points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is $\mathrm{Cov}_t(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{\kappa}_t(\boldsymbol{x})^T(\boldsymbol{K}_t + \tilde{\sigma}\boldsymbol{I})^{-1}\boldsymbol{\kappa}_t(\boldsymbol{x}')$, where $[\boldsymbol{\kappa}_t(\boldsymbol{x})]_i = \kappa(\boldsymbol{x}_i, \boldsymbol{x})$, $[\boldsymbol{y}_t]_i = y_{\boldsymbol{x}_i}$, $\tilde{\sigma}$ is noise level, and $[\boldsymbol{K}_t]_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. At point $\boldsymbol{x}$, the GP posterior $y_{\boldsymbol{x}} \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t(\boldsymbol{x}))$, where $\sigma_t(\boldsymbol{x}) \coloneqq \mathrm{Cov}_t(\boldsymbol{x}, \boldsymbol{x})$; see Rasmussen et al. [48] for more details.

2.2 Acquisition Functions

One-step. One commonly used AF for BO is expected improvement (EI) [41], defined as $\alpha_{\mathrm{EI}}(\boldsymbol{x}) \coloneqq \mathbb{E}\big[\max\{y_{\boldsymbol{x}} - y_t^*, 0\}\big]$, where y_t^* denotes the current-best (i.e., incumbent) observation. A closely related function is probability of improvement

(PI) [40], which considers only the probability that a new observation $y_{\boldsymbol{x}}$ exceeds the incumbent y_t^*, without accounting for the magnitude of improvement: $\alpha_{\text{PI}}(\boldsymbol{x}) \coloneqq \mathbb{P}\left[y_{\boldsymbol{x}} > y_t^*\right]$. Both EI and PI have closed-form expressions and are therefore computationally efficient. Similarly, UCB is defined as $\alpha_{\text{UCB}}(\boldsymbol{x}) \coloneqq \mu_t(\boldsymbol{x}) + \sqrt{\beta_t}\,\sigma_t(\boldsymbol{x})$, where β_t is a parameter that balances exploration and exploitation. In contrast, information-theoretic AFs select the next sampling point to reduce uncertainty about a particular property of the optimum – whether its location as in entropy search (ES) or predictive entropy search (PES) [32, 33], its value as in max-value entropy search (MES) [59], or both as in joint entropy search (JES) [35, 56]. Thompson sampling (TS) implicitly balances the exploration-exploitation trade-off by maximizing posterior samples from a Gaussian process whose accuracy improves as more observations are incorporated [8]. However, TS has been criticized for being overly explorative. To address this, several approaches have been proposed to make it less explorative, including trust regions (TRs) that restrict the space over which the AF is maximized to a subregion of $\mathcal{X}$ [24] and random axis-aligned subspace perturbations (RAASP) sampling that only considers points close to the incumbent observation for the maximization of the AF [47].

Multi-step. One common property of all aforementioned AFs is that they assume that the next evaluation will be the last, i.e., they greedily maximize the simple or inference regret for the next iteration, assuming no more evaluations will be performed [59]. In contrast, multi-step AFs [27, 64, 39] consider the impact of the current choice for future evaluations: $\alpha_{\text{MS}}(\boldsymbol{x}) = v_1(\boldsymbol{x}|\mathcal{D}_t) + \mathbb{E}_y\left[\max_{\boldsymbol{x}_2}\left(v_1\left(\boldsymbol{x}_2|\mathcal{D}_t \cup \{(\boldsymbol{x}, y_{\boldsymbol{x}})\}\right) + \mathbb{E}_{y_2}\left[\ldots\right]\right)\right]$, where $v_1(\boldsymbol{x})$ is the one-step marginal value of $\boldsymbol{x}$, e.g., the expected improvement upon observing $\boldsymbol{x}$. See Jiang et al. [39] for details. Multi-step AFs, such as knowledge gradient (KG) [26], are computationally expensive since the expectations of α_{MS} must be approximated with Monte-Carlo methods and, therefore, are often limited to one lookahead step even though the theoretical framework can usually be extended to arbitrarily many lookahead steps [39]. At the same time, they are fundamentally different from previous AFs and may be characterized by a unique EETO [64].

Batch. Batching is a technique used where multiple function evaluations can be performed in parallel. Instead of re-conditioning the GP after a single new observation, one observes q points in parallel. Batching requires modifications of the AF to ensure that a batch contains a diverse set of candidates. One strategy for batch BO is using multi-point AFs that estimate the improvement of some utility upon observing q new points [58]. Other approaches include *local penalization* [28] that repels points from regions around points already included in the batch.

2.3 Expressions of Exploration

Quantifying Exploration. Although achieving a good balance between exploration and exploitation is widely recognized as a crucial component of an effective acquisition function, the BO community still lacks a simple and convenient metric for quantifying exploration. Several related metrics have been proposed – such as total center distance (TCD) [24, Fig. 6] and incumbent distance [36, Fig. 20] – but each has notable shortcomings. For instance, TCD focuses solely on the movement of the incumbent, and it fails when the first sample lands on the optimum (yielding a TCD of zero) or when the search oscillates between multiple optima. Similarly, incumbent distance measures the distance between the next query and the incumbent but does not reliably capture exploration, as repeatedly querying the opposite corner of the space can yield high values without representing meaningful explorative behavior.

Another approach employs Pareto analysis- to interpret the EETO [9, 25, 65, 17]. In this framework, exploration and exploitation are treated as distinct objectives, with the utility function depending on both the predicted mean μ and variance σ. Under this interpretation, many AFs, such as PI, EI, and UCB, can be accommodated. However, more sophisticated AFs, particularly those based on information-theoretic principles, do not conform to this framework.

Quantifying Exploration in Evolutionary Algorithms. The measurement of the exploration extends beyond Bayesian optimization. In evolutionary algorithms (EA) [21], this quantity is evaluated from two perspectives: genotypic and phenotypic [16]. Genotypic measures assess diversity in the input space using tree-based techniques [11, 15], Euclidean distance quantities [44], entropy-based methods [45], and individual-population similarity indices [37]. In contrast, phenotypic measures evaluate diversity in terms of fitness or performance, employing distance-based methods [13], entropy-based measures [1, 57], and local-attraction metrics [38]. Notably, the distance-based approaches mentioned above typically measure the distance from each individual to the population's center rather than the aggregate distance connecting all individuals, distinguishing them from our method. Additionally, entropy-based approaches have mainly focused on low-dimensional discrete domains – often using binning methods to approximate entropy – whereas our proposed techniques target high-dimensional continuous domains.

Quantifying Exploration in Reinforcement Learning. In reinforcement learning (RL), whether in simpler settings like multi-armed bandits (MABs), a special case of Markov decision processs (MDPs) where the state remains constant, or in more

complex MDP scenarios, an agent must balance exploration and exploitation to achieve long-term benefits.

In MAB problems, one measure of exploration is tracking the frequency with which each action (arm) is selected [43]. For general RL tasks, both states and actions must be considered. Some methods promote exploration by maximizing the entropy of the action distribution [61, 3] to encourage the agent to try diverse actions. Curiosity-driven exploration, often quantified using information gain [53, 34], provides another approach to exploration in the state-action space. Existing entropy-based exploration methods in RL often rely on parametric methods, like variational inference, which leverage prior knowledge of the underlying density functions (e.g., for actions) but may be inaccurate when the parametric assumptions fail. In BO, we do not have access to the true density of observations, so we adopt a non-parametric approach for density estimation. Consequently, our method is not directly comparable to previous entropy-based approaches in RL that depend on specific parametric families.

3 Quantifying Exploration

Motivated by the Cambridge [12] dictionary definition of *exploration* as "the activity of travelling to and around a place, especially one where you have never been [...] before, in order to find out more about it", we define exploration in the context of black-box optimization.

Definition 5 (Exploration)**.** The activity of sampling in a region of the search space, especially one that has never been sampled before, to learn more about a global optimum.

Quantifying the exploration preferences of AFs is crucial for understanding their behavior and for developing an effective AF portfolio to achieve better performance. In this section, we first summarize existing knowledge on the exploration tendencies of different acquisition functions and then propose two key methods to quantify exploration.

3.1 Analysis of Tribal Knowledge

The EI and PI AFs have been shown to explore relatively little [17], with PI being even less explorative than EI [6]. In contrast, the KG AF – which can be seen as a generalization of EI [63, p. 12] – has been reported to be more explorative than EI [26, p. 89]. Information-theoretic acquisition functions are generally considered

on the explorative side [33], although they can be surpassed by UCB with a high β value. At the extremes of the spectrum, random search (RS) samples points uniformly at random throughout the domain $\mathcal{X}$, while deterministic selection (DM) always selects the same fixed point; both completely disregard the probabilistic surrogate model. Empirical findings on the explorative behavior of AFs can be broadly summarized by the following informal ordering: RS $\succeq$ UCB (high β) $\succeq$ Information Theoretic ⓘ KG $\succeq$ EI $\succeq$ PI $\succeq$ UCB (low β) $\succ$ DM. Finally, although TS is known to be explorative [20], its exact placement in this ranking remains unclear. This ordering reflects a general quantitative understanding within the BO community; however, the relative explorative behavior of acquisition functions may vary depending on the specific problem setting. In particular, the relationship between information-theoretic acquisition functions and KG remains uncertain, as indicated by the question mark.

3.2 Exploration Methods

We introduce two quantities to evaluate the exploration behavior of different black-box optimization methods based on the locations of their observations in the search space.

Observation Traveling Salesman Distance (OTSD). OTSD quantifies the minimum Euclidean distance required to connect all observation points by formulating the problem as a traveling salesman problem (TSP). Given a set of t observation points $X_t \coloneqq \{\boldsymbol{x}_i\}_{i=1}^t$ from $\mathcal{D}_t$, OTSD is defined as the total length of the shortest possible route that visits each observation point exactly once and returns to the starting point. Mathematically, it is expressed as:

$$\mathrm{OTSD}(X_t) \coloneqq \min_{\tau \in S_t} \left(\sum_{i=1}^{t} \|\boldsymbol{x}_{\tau(i)} - \boldsymbol{x}_{\tau(i+1)}\| \right), \tag{4.1}$$

where τ is a permutation of $\{1, 2, \ldots, t\}$ representing a tour that visits all points with $\tau(t+1) \coloneqq \tau(1)$, $\|\cdot\|$ denotes the Euclidean distance, and S_t is the set of all possible permutations of t elements. OTSD increases monotonically with the number of observations.

Since the TSP is NP-hard, we approximate the solution using an insertion heuristic method [49], which has a time complexity of $\mathcal{O}(dT^2)$ for T observations in d dimensions, and the worst-case tour length is guaranteed to be at most twice the optimal distance. This approach conveniently allows us to track the OTSD for each $t \leq T$. The pseudocode for calculating OTSD is in Algorithm 4.1.

Algorithm 4.1 OTSD Insertion Heuristic

Input: Observation locations $X_t = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t\}$
Output: Estimated TSP distance, $\mathrm{OTSD}(X_t)$

Compute pairwise distances: $D(i,j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ for all $i, j \in \{1, \ldots, t\}, i \neq j$.
Initialize the permutation $\tau : \{1\} \to \{1\}$ that represents the (trivial) tour order on one point, $\boldsymbol{x}_1$.
For a tour on k points, define $\tau_i \coloneqq \tau(i \bmod k)$.
Initialize $\mathrm{OTSD} \leftarrow 0$
for $k = 2$ to t **do**
    For each consecutive pair in the tour τ on points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}\}$, compute the insertion cost for placing $\boldsymbol{x}_k$ between them: $\forall i = 1, \ldots, k-1$,
$\Delta C(i) \coloneqq D(\tau_i, k) + D(k, \tau_{i+1}) - D(\tau_i, \tau_{i+1})$
    Identify the insertion point $i^\star$ that minimizes $\Delta C(i)$.
    Update the permutation to τ on $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$ by inserting $\boldsymbol{x}_k$ at the optimal position $i^\star$.
    $\mathrm{OTSD} \leftarrow \mathrm{OTSD} + \Delta C(i^\star)$.
Return OTSD.

Normalized OTSD. To eliminate the influence of the problem dimensionality and ensure value consistency across different problems, we propose the *normalized OTSD*:

$$\mathrm{OTSD}_{\mathrm{norm}}(X_t) \coloneqq \frac{\mathrm{OTSD}(X_t)}{\Psi(d,t)}, \tag{4.2}$$

where d denotes the dimensionality of the problem and $\Psi(d,t) \coloneqq 2\sqrt{5d}\left(\frac{3t}{2}\right)^{1-1/d}$ is the upper bound derived in Proposition 4. Unlike OTSD, the normalized OTSD is non-monotonic; a constant value indicates that the AF does not change its behavior throughout the optimization, while higher and lower values suggest increased and decreased exploration, respectively. We use the normalized OTSD to aggregate results across different problems in Sec. 4.

Observation Entropy (OE). By treating observation points as samples from a random variable, we quantify the uniformity of the data distribution using empirical differential entropy. Higher entropy values indicate a more uniform spread of points, reflecting greater exploration. To estimate entropy without assuming an underlying distribution, we employ a non-parametric entropy estimator. Several methods exist, including histogram-based [29], kernel-density based [2], and nearest-neighbor based [42] approaches. Among them, the nearest-neighbor-based estimator, namely the Kozachenko-Leonenko (KL) estimator, stands out for its sample efficiency and

applicability in moderate-dimensional cases ($d \leq 50$), while other methods are very costly for $d \geq 10$.

We define OE using the KL estimator as follows:

$$\mathrm{OE}(X_t) \coloneqq \frac{d}{t} \sum_{i=1}^{t} \log\left(\varepsilon_i^k\right) + \psi(t) - \psi(1) + \log V_d, \tag{4.3}$$

where ε_i^k denotes the distance between $\boldsymbol{x}_i$ and its k-th nearest neighbor, $V_d \coloneqq \pi^{d/2}/\Gamma(1 + d/2)$ is the volume of the d-dimensional unit ball, $\Gamma(\cdot)$ denotes the Gamma function, and $\psi(t) \coloneqq \frac{\partial}{\partial t} \log \Gamma(t)$ is the digamma function. Following the recommendation of Berrett et al. [7], we set $k = \log(t)$ (using the natural logarithm), as increasing k with t improves the efficiency of the estimation.

OE is non-monotonic and may take on either positive or negative values. An increasing OE over BO iterations indicates that the observed samples are more uniformly distributed, suggesting more exploration. Because of its non-monotonic nature, a sharp change in OE signals that an AF is altering its level of exploration over iterations. The computational complexity of OE is $\mathcal{O}(dT^2)$ for T observations in d dimensions, or $\mathcal{O}(dTk)$ if distances are provided or $t \leq T$ gets updated sequentially. Algorithm 4.2 details the steps for computing OE.

Algorithm 4.2 Kozachenko-Leonenko Entropy Estimation

Input: Observation locations $X_t = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t\}$

Output: Estimated observation entropy $\mathrm{OE}(X_t)$

1. Compute pairwise distances: $D(i,j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ for all $i, j \in \{1, \ldots, t\}$ with $i \neq j$.
2. For each point $\boldsymbol{x}_i$, determine ε_i^k as the k-th smallest value in the set $\{D(i,j) : j \neq i\}$.
3. Compute the volume of the unit ball in $\mathbb{R}^d$: $V_d = \frac{\pi^{d/2}}{\Gamma\left(1+\frac{d}{2}\right)}$, where $\Gamma(\cdot)$ denotes the gamma function.
4. Evaluate the digamma functions $\psi(t)$ and $\psi(1)$, then compute the KL entropy estimator as given in Eq. (4.3).
5. **Return** $\mathrm{OE}(X_t)$.

We illustrate OTSD, normalized OTSD, and OE for various AFs on the 6-dimensional `Hartmann` function in Fig. 4.2. The left panel shows the OTSD, which increases monotonically as new observations are added. The center panel presents the normalized OTSD, providing a clearer distinction among the different AFs. The right panel displays the OE results. Since DM yields very low OE values (around -134), we exclude DM from the OE plot for better visualization. Overall, the

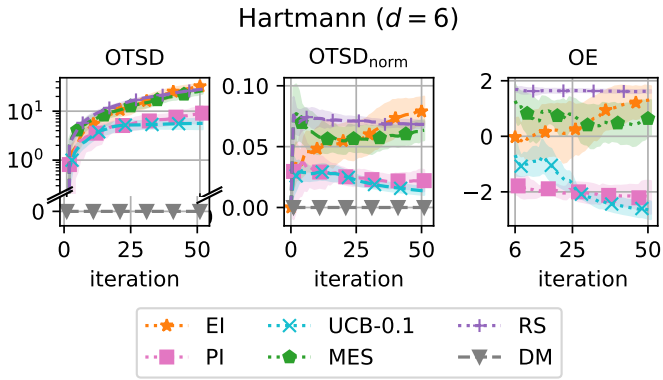


Figure 4.2: The exploration quantities OTSD and OE of RS, UCB ($\beta = 0.1$), EI, PI, MES, and deterministic selection (DM) on the 6-dimensional `Hartmann` function. From left to right, these plots show OTSD, normalized OTSD, and OE, respectively. DM values for OE (around -134) are hidden for better visualization. The shaded areas show the standard error of the mean.

figure demonstrates the explorative behavior of the various AFs and cross-validates the performance of OTSD and OE, as both yield consistent rankings.

The values of OTSD and OE are not directly comparable between different objective functions, as variations in input dimension, domain, and function landscape can significantly impact these quantities; neither quantity is invariant under a change of variables (for OE, this reflects that differential entropy is not invariant, unlike discrete entropy).

3.3 Exploration Bounds

In Fig. 4.2, we denote OTSD and OE as the statistical estimators based on the given input X_t. In this section, we establish bounds on their true values, denoted as $\text{OTSD}^*(t)$ and $\text{OE}^*(t)$ which depend only on t. The proofs for Propositions 4 and 5 are detailed in Appendices A.1 and A.2, respectively.

Proposition 4 (Upper and Lower Bounds for the True OTSD). *Let t denote the number of observations drawn from the unit cube $[0,1]^d$ in d-dimensional space ($d \geq 3$). Building on the bounds derived in [10, 5], the true observation traveling salesman distance $OTSD^*(t)$ satisfies*

$$0 \leq OTSD^*(t) \leq \Psi(d,t) \overset{def}{=} 2\sqrt{5d}\left(\frac{3}{2}t\right)^{1-1/d}. \tag{4.4}$$

Note that the minimum traveling salesman distance is zero, which occurs when all

points coincide.

Proposition 5 (Upper Bound for the True OE). *For any probability distribution supported in the unit cube of dimension d, the uniform distribution achieves the maximum differential entropy with a zero value. Conversely, the differential entropy can be made arbitrarily negative, implying that there is no finite lower bound.*

Typically, the sample set X_t does not consist of independently and identically distributed points, as they are chosen adaptively via the Bayesian optimization process. Consequently, it is not necessarily the case that $\mathrm{OE}(X_t) \leq 0$, even when the samples lie within the unit hypercube. Further discussion is provided in Appendix A.3.

Extension to Non-Euclidean Domains. In many practical problems, the input domain may be non-Euclidean. Examples include integers, ordinals, categoricals, protein sequences, strings, and graphs. Quantifying exploration on these non-Euclidean domains is particularly interesting. The OTSD can still be applied in these settings provided a suitable metric is available since the insertion heuristic in Algorithm 4.1 relies on the triangle inequality and thus only requires a metric space. However, the estimation of OE becomes more challenging. In particular, the KL estimator in Eq. (4.3) is designed for estimating differential entropy in Euclidean spaces. Defining an appropriate notion of entropy and developing an estimator for such irregular spaces is a case-by-case problem that requires further investigation and is beyond the scope of this paper.

4 Experiments

We evaluate OTSD (including normalized OTSD) and OE on a wide range of synthetic and real-world benchmarks to tackle the following three research questions:

RQ1. Are the proposed quantities consistent with the literature, i.e., do OTSD and OE show higher exploration levels for AFs that are known to be more explorative?

RQ2. How do AFs, whose exploration level has not yet been discussed, relate to others in terms of exploration?

RQ3. What is the relationship between the level of exploration and optimization performance?

4.1 Experimental Setup

Evaluation. For each run of an AF on a given benchmark, we record the locations $\boldsymbol{x}_i \in \mathcal{X}$ and their corresponding function values $y_{\boldsymbol{x}_i} \in \mathbb{R}$. From these observations, we compute the OTSD, the OE, and the performance, defined as the highest function value observed during the run. To aggregate OTSD results across different problems, we normalize OTSD using Eq. (4.2) and then average these normalized values across the selected benchmarks. We give runtimes for OTSD and OE in Appendix E; OTSD is fast to compute, requiring less than 200 ms while OE needs $\approx$ 5 s for 1,000 observations in a $20d$ space. We also empirically validate Proposition 4 in Appendix C.

Since there is no straightforward method to normalize OE and performance across problems with varying dimensions, we assess their relative rankings of competing methods. Specifically, we compute the mean OE or performance for each method on each problem over ten repetitions, rank these mean values per problem, and then average the rankings across problems to obtain the *mean relative ranking* (individual optimization performances for each benchmark are provided in Appendix D.6). Note that for OE, the ranking is reversed so that more explorative methods receive a larger rank, ensuring consistency with the normalized OTSD results. Because the KL estimator exhibits significant bias in high dimensions, we report OE only for low-dimensional experiments ($d \leq 20$); for high-dimensional real-world experiments, we exclusively use OTSD.

For better clarity, we only plot the mean values for normalized OTSD and the ranks. In Appendix D.2, we show the figures with the standard error of the mean.

Acquisition Functions. We study the following AFs: expected improvement (EI), probability of improvement (PI), max-value entropy search (MES), Thompson sampling (TS), and upper-confidence bound (UCB). We also include knowledge gradient (KG) in our comparison but only apply it to low-dimensional problems ($d \leq 10$) due to its high computational cost. We further compare with the popular `CMA-ES` [30], which is an evolutionary method for gradient-free continuous non-convex optimization using the implementation by Hansen et al. [31], version 3.3.0.

In addition to the simple BO setup, we experiment with two common techniques for high-dimensional Bayesian optimization (HDBO): TRs and RAASP, both introduced in the `TuRBO` algorithm [24]. In `TuRBO`, TRs, TS, and RAASP are interwoven and not studied independently. To allow assessing their individual effects, we therefore consider adaptations of these techniques.

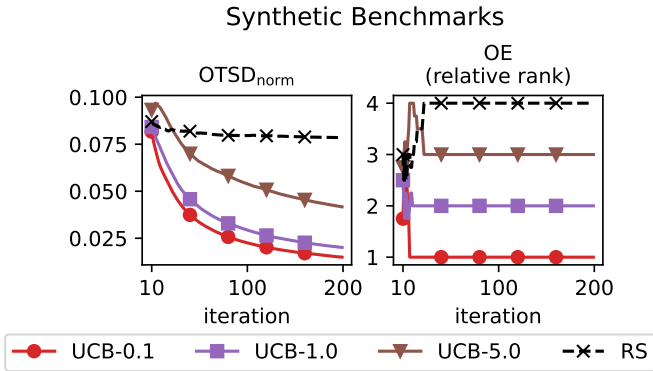


Figure 4.3: Normalized OTSD and OE rank averaged across all the synthetic benchmarks. A lower rank means lower exploration. We do not show the initial design of experiment phase.

Finally, we study the effect of batched evaluations. In particular, we use the batched variants of the aforementioned AFs where available with batch sizes of $q = 8$ and $q = 32$. While it is not uncommon to study the number of *batch evaluations* to assess the effect of batching, we always study the number of function evaluations as we are interested in how batching changes exploration, i.e., we count one batch of size q as q individual function evaluations.

Benchmarks. We evaluate several AFs and variations thereof on nine benchmark problems, ranging from low-dimensional synthetic problems to noisy, high-dimensional simulations. A summary of the benchmarks is given in Table 4.4 in Appendix B. The $2d$ Branin, $4d$ Levy, $6d$ Hartmann, and $8d$ Griewank problems are from Surjanovic and Bingham [54], the $8d$ Lasso-Diabetes and $180d$ Lasso-DNA problems from Šehić et al. [50], the $60d$ Rover and $14d$ Robot Pushing problems from Wang et al. [60], and the $124d$ Mopta08 problem from Eriksson and Jankowiak [23].

4.2 Empirical Validation of OTSD and OE

We begin by empirically validating that OTSD and OE effectively quantify exploration by applying them to methods that exhibit increasing levels of explorative behavior. Fig. 4.3 shows the normalized OTSD (left) and the mean OE ranks (right) averaged over the synthetic benchmarks. As expected, UCB with a low $\beta = 0.1$ (blue) achieves the lowest normalized OTSD and mean OE rank, followed by UCB with a moderate $\beta = 1$ (orange). UCB with a high $\beta = 5$ (green) achieves the highest OTSD and OE. The OTSD curves going down indicate that, as expected, the AFs become more exploitative over time. Appendix D.1 shows the same result

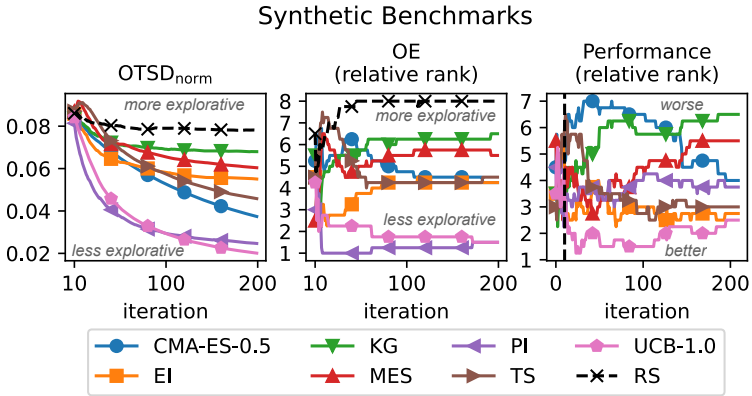


Figure 4.4: Normalized OTSD, average rank for OE and optimization performance on the synthetic benchmarks.

for real-world benchmarks; we also show the same analysis for `CMA-ES` with varying levels of σ_0.

4.3 Synthetic Benchmarks

Fig. 4.4 shows the performance of the various AF (without RAASP sampling and TRs), averaged over the four synthetic benchmarks. We observe that at the start of the optimization, right after the design of experiment (DoE) phase, PI is the least explorative AF as it has low normalized OTSD and the highest OE relative rank of all AFs. UCB-1.0 is similarly unexplorative, starting only slightly more explorative than PI and ending up overtaking PI as the least explorative AF. In contrast, TS starts as the most explorative AF according to both OTSD and OE. Eventually, MES, KG, and `CMA-ES` overtake TS as the most explorative AF. EI is on the same level of exploration as TS. TS and UCB-1 shows the best relative rank optimization performance.

Next, we study the effect of batching, TR, and RAASP sampling on the level of exploration for EI in Fig. 4.5; see Appendix D.5 for the same setting on other AFs. Batching increases exploration: EI with a batch size of 32 (red) is the most explorative variant as indicated by both OTSD and OE. The variant with the smaller batch size 8 (purple) is the next most explorative variant, followed by regular EI. RAASP and TRs, on the other hand, lower the level of exploration. Both RAASP and TRs get a similarly low OE rank, indicating that they are similarly unexplorative. However, while RAASP is the highest performing variant (orange), TRs reduce exploration on a similar level as RAASP, but it is one of the worst variants, as shown in the right panel of Fig. 4.5. Batching also degrades optimization performance, as is expected

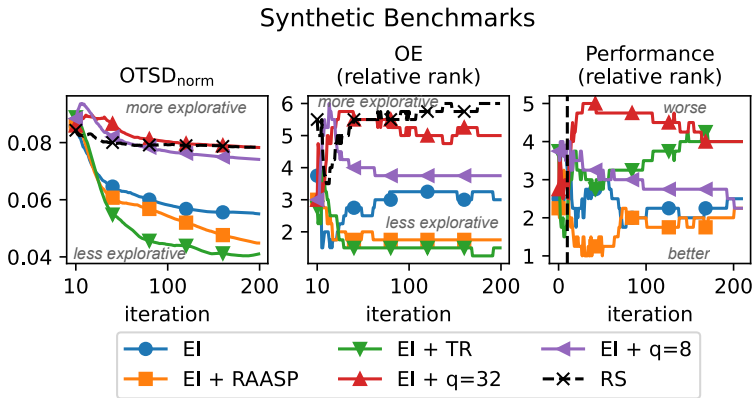


Figure 4.5: Normalized OTSD, average ranks for OE and optimization performance for EI and its variations on the synthetic benchmarks.

when plotting against the number of function evaluations.

4.4 Real-World Benchmarks

Fig. 4.7 shows the performance of the various AF (without RAASP sampling and TRs) on the real-world benchmarks. Since most real-world benchmarks are in high dimensions, we do not plot OE for the aggregated results. The exploration behavior in high dimensions is highly consistent, with minimal overlap between the OTSD curves. We hypothesize that EI and UCB-1.0 show the best optimization performance due to their balanced the EETO as shown by the OTSD. The overly-explorative TS shows the worst performance, followed by the less-explorative PI. Compared to the other methods, MES and TS show a unique behavior: MES initially becomes less explorative, indicated by a decreasing normalized OTSD, but then reverses this trend and becomes more explorative. TS shows the opposite behavior. Initially, it strives for pure exploration and is even more explorative than RS. We explain this behavior with TS choosing areas where the surrogate exhibits high posterior variance, choosing points distant to previous observations and hence yielding high OTSD. Later on, as the posterior variance of the surrogate decreases, TS becomes less explorative.

We also study the effect of TRs, RAASP, and batching on the real-world benchmarks. The results are similar to the synthetic case, so we save them for Appendix D.3.

Impact of the Dimensionality. In addition to distinguishing between synthetic and real-world scenarios, we examine OTSD and OE across low- and high-

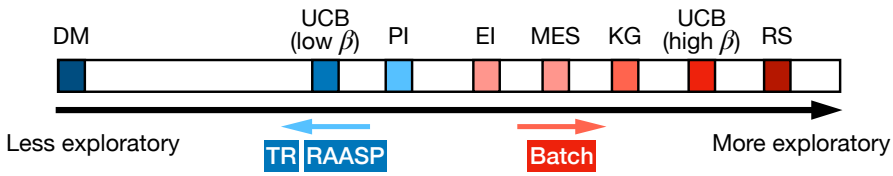


Figure 4.6: Empirical AF (and their variants) exploration taxonomy based on the quantitative exploration methods OTSD and OE. It represents a general understanding and may differ in specific problems.

dimensional regimes. Two key differences emerge: first, TS is considerably more explorative in high-dimensional benchmarks; second, while EI is more explorative than other methods in low dimensions, it becomes less so in high dimensions. One potential explanation is that in high-dimensional spaces, the GP may struggle to accurately learn the objective function, further encouraging explorative sampling, which significantly impacts TS. Moreover, lengthscales are underestimated in higher dimensions, causing EI to make more conservative predictions than in low-dimensional settings. Due to space limitations, we present their synthetic problem results in Appendix D.4.

4.5 Exploration Taxonomy

Our empirical results on synthetic and real-world benchmarks confirm the partial ordering KG $\succeq$ EI $\succeq$ PI, which was common knowledge within the community as discussed in Sec. 3.1. Furthermore, our findings reveal that KG is slightly more explorative than MES, addressing a missing link in previous studies. TS is challenging to classify because its behavior varies dramatically between low and high dimensions. In low dimensions, TS performs similarly to EI; while it becomes overly explorative in high dimensions – surpassing even RS in terms of normalized OTSD. Expanding the exploration taxonomy, we empirically demonstrate that techniques such as trust regions (TRs) and RAASP consistently reduce exploration, whereas batching tends to increase exploration. Finally, our findings indicate that the best-performing methods tend not to exhibit extreme exploration quantities; their OTSD and OE values are neither excessively high nor excessively low compared to other approaches. However, they are often on the less explorative side of the spectrum, as demonstrated by the decrease in OTSD during the optimization iterations. To summarize our results, we present a revised empirical AF exploration taxonomy in Fig. 4.6 based on our new OTSD and OE quantities.

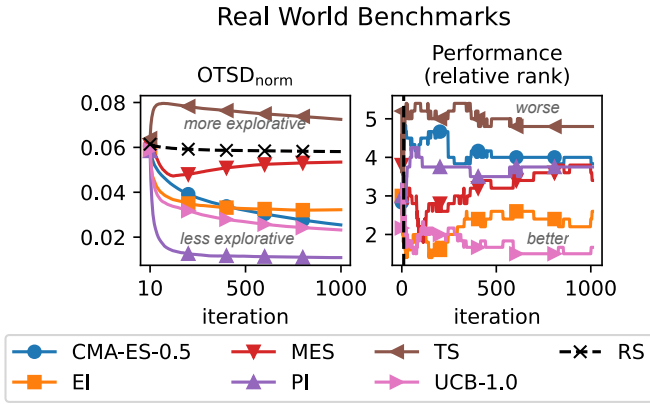


Figure 4.7: Normalized OTSD and average optimization performance rank on the real-world benchmarks.

5 Conclusion

In this work, we introduce two novel methods and their theoretical bounds to quantify the magnitude of exploration in various acquisition functions. We conduct extensive experiments on synthetic and real-world benchmarks, spanning low- and high-dimensional settings, to evaluate commonly used acquisition functions and their variants. Our empirical findings demonstrate that OTSD and OE effectively capture the level of exploration exhibited by these acquisition functions. Finally, we present the first empirical taxonomy of acquisition function exploration based on these quantification methods. These results offer valuable insights for designing new acquisition functions, constructing acquisition function portfolios, or controlling the optimization. For instance, an AF having higher OTSD than a random search (see TS in Fig. 4.7) can serve as a warning sign that this AF is too explorative for the problem at hand. In such situations, one could either switch to more local AF according to the taxonomy or combine the AF with RAASP sampling. Similarly, if an AF approaches the OTSD of a random search after being less explorative at the beginning of the optimization (see MES in Fig. 4.7), it could be an early-stopping indicator where the AF exhaustively visited all local minima and the optimization can come to an end.

Limitations and Future Work. While our results are obtained by extensive empirical experimentation, they do not consider the effect of GP hyperparameters and hyperpriors on the behavior of AFs and are limited to continuous domains. In the future, we will expand the taxonomy to include additional AFs, for instance, other information-theoretic AFs [32, 33, 35, 14], and study the effect of kernel and likelihood functions and their hyperparameters on the behavior of AFs in Bayesian

optimization. Furthermore, we will study if OTSD and OE can be extended to other domains, such as non-Euclidean spaces.

Acknowledgments

This project was partly supported by the Wallenberg AI, Autonomous Systems, and Software program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725

A Proofs

A.1 OTSD Upper Bound

Proof. Our proof follows from Balogh et al. [5, Theorem 1.3]. In that work, the authors show that the d-norm length of a Hamiltonian cycle with t nodes in a d-dimensional unit cube, denoted by

$$s_d^{\mathrm{HC}}(t) \coloneqq \left(\sum_{i=1}^{t} \sqrt{\mathrm{e}_i^d} \right)^{1/d}, \tag{4.5}$$

satisfies

$$s_d^{\mathrm{HC}}(t) \leq 3\sqrt{5} \left(\frac{2}{3} \right)^{1/d} \sqrt{d}, \tag{4.6}$$

where

$$\mathrm{e}_i \coloneqq \|\boldsymbol{x}_{\tau(i)} - \boldsymbol{x}_{\tau(i)+1}\|_2 \tag{4.7}$$

represents the Euclidean distance between successive nodes.

The true total distance with t nodes, denoted by $\mathrm{OTSD}^*(t)$, is given by

$$\mathrm{OTSD}^*(t) \coloneqq \sum_{i=1}^{t} \mathrm{e}_i. \tag{4.8}$$

Applying Hölder's inequality,

$$\|\boldsymbol{e}\|_1 \leq \|\boldsymbol{e}\|_d \cdot \|\mathbf{1}\|_{(1-1/d)^{-1}}, \tag{4.9}$$

yields

$$\mathrm{OTSD}^*(t) = \sum_{i=1}^{t} \mathrm{e}_i \leq s_d^{\mathrm{HC}}(t) \cdot \left(\sum_{i=1}^{t} 1 \right)^{1-1/d}. \tag{4.10}$$

Since $\sum_{i=1}^{t} 1 = t$, it follows that

$$\mathrm{OTSD}^* \leq 3\sqrt{5} \left(\frac{2}{3} \right)^{1/d} t^{1-1/d} \sqrt{d} = 2\sqrt{5d} \left(\frac{3}{2} t \right)^{1-1/d}. \tag{4.11}$$

□

A.2 OE Upper Bound

We show that the distribution with maximum differential entropy in the unit cube $[0,1]^d$ is the uniform distribution.

Proof. We wish to determine the probability density function $p(\boldsymbol{x})$, for $\boldsymbol{x} \in [0,1]^d$, that maximizes the differential entropy

$$H[p] = -\int_{[0,1]^d} p(\boldsymbol{x}) \ln p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{4.12}$$

subject to the normalization constraint

$$\int_{[0,1]^d} p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = 1. \tag{4.13}$$

To enforce this constraint, we introduce a Lagrange multiplier λ and consider the augmented functional

$$\mathcal{L}[p] = -\int_{[0,1]^d} p(\boldsymbol{x}) \ln p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \lambda \left(\int_{[0,1]^d} p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} - 1 \right). \tag{4.14}$$

We then compute the first variation $\delta\mathcal{L}$ with respect to an arbitrary variation $\delta p(\boldsymbol{x})$, yielding

$$\delta\mathcal{L} = -\int_{[0,1]^d} \delta p(\boldsymbol{x}) \Big[\ln p(\boldsymbol{x}) + 1 \Big] \, \mathrm{d}\boldsymbol{x} + \lambda \int_{[0,1]^d} \delta p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{4.15}$$

For $\delta\mathcal{L}$ to vanish for all admissible variations $\delta p(\boldsymbol{x})$, the integrand must be zero:

$$-\ln p(\boldsymbol{x}) - 1 + \lambda = 0 \quad \text{for all } \boldsymbol{x} \in [0,1]^d. \tag{4.16}$$

Solving for $\ln p(\boldsymbol{x})$, we obtain

$$\ln p(\boldsymbol{x}) = \lambda - 1, \tag{4.17}$$

which implies that

$$p(\boldsymbol{x}) = e^{\lambda - 1}. \tag{4.18}$$

Since $p(\boldsymbol{x})$ is constant over $[0,1]^d$, we can determine the constant by imposing the normalization condition:

$$\int_{[0,1]^d} p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = e^{\lambda-1} \cdot 1 = 1. \tag{4.19}$$

Thus,

$$e^{\lambda-1} = 1 \quad \Longrightarrow \quad \lambda - 1 = 0 \quad \Longrightarrow \quad \lambda = 1, \tag{4.20}$$

yields

$$p(\boldsymbol{x}) = 1 \quad \text{for all } \boldsymbol{x} \in [0,1]^d. \tag{4.21}$$

This completes the proof that the maximum entropy distribution on the d-dimensional unit cube is indeed the uniform distribution. Therefore,

$$\mathrm{OE}^*(t) \leq H[p_{\mathsf{unif}}] = 0. \tag{4.22}$$

□

A.3 KL Estimation Consistency and Bias

Many studies have examined the estimation bias of the Kozachenko-Leonenko estimator (see Eq. (4.3)). The original work by Kozachenko and Leonenko [42] established the consistency of the estimator under mild conditions when $k = 1$. Furthermore, Pál et al. [46] demonstrated both the consistency and the convergence rate of the nearest-neighbor-based estimator for Rényi entropies under the assumption that the entropy support is bounded. In addition, Delattre and Fournier [18] extended these results to non-compactly supported densities, providing an upper bound for the bias of order $\mathcal{O}(t^{-2/d})$. A more recent study [19] presents a consistency result for the KL estimator even when the density function is not smooth. However, in our study the sample points in X_t are generally not independent and identically distributed, as they are collected via a Bayesian optimization process. Consequently, we believe that the practical performance of OE does not align with the findings of these previous studies.

B Details on the Experimental Setup

We run each AF in a basic BO setup where we first initialize the GP with ten observations that we sample uniformly at random from $\mathcal{X}$ and then start the BO loop for 200 iterations for synthetic and 1000 iterations for real-world problems. Similarly, we run `CMA-ES` with a population size of 5 with different initial step sizes $\sigma_0 \in \{0.05, 0.1, 0.5\}$. UCB allows one to specify an exploration parameter β, which we set to $\beta \in \{0.1, 1, 5\}$.

To evaluate the AFs, we use the default `SingleTaskGP` GP model provided by `BoTorch` version `0.12.0` [4] as well as `BoTorch`'s provided methods to fit the GP and maximize the AF. In the case of TS, we sample from the GP posterior using pathwise conditioning [62] and maximize the posterior sample using `BoTorch`'s `optimize_posterior_samples` function with 1024 initial random samples and 20 restarts of the gradient descent (GD) optimizer.

For RAASP sampling, we rely on the `sample_around_best` parameter for `BoTorch`'s AF optimizer that, with a probability of $\min\left(1, \frac{20}{d}\right)$, substitutes a parameter's current best configuration with a value sampled from a truncated Gaussian, centered on the incumbent observation. For TRs, we follow `TuRBO`'s specification for finding the TR bounds. We then configure the AF maximizer to respect these bounds, similar to Eriksson [22] but dropping the sparse perturbations.

Table 4.4 summarizes the benchmarks used in this work.

Table 4.4: Benchmark summary.

Name	d	Noise	Synth.
Branin	2	✗	✓
Levy	4	✗	✓
Hartmann	6	✗	✓
Griewank	8	✗	✓
Lasso-Diabetes	8	✗	✗
Robot Pushing	14	✓	✗
Rover	60	✗	✗
Moptao8	124	✗	✗
Lasso-DNA	180	✗	✗

C Empirical Verification of the OTSD Bound

To verify the upper bound stated in Proposition 4, we take advantage of the normalized OTSD defined in Eq. (4.2) which makes the OTSD independent from

the dimensionality d. If the approximation error to compute OTSD is ignored, the theoretical bound implies that $\mathrm{OTSD}_{\mathrm{norm}}(t)$ should remain below 1 for all t (or with approximation error considered, $\mathrm{OTSD}_{\mathrm{norm}}(t)$ should remain below 2 for all t).

We show $\mathrm{OTSD}_{\mathrm{norm}}$ for Random Search (RS) across various dimensionalities of the problem in Fig. 4.8. As the figure illustrates, all values remain well below the theoretical threshold of 1. Given that RS represents a highly explorative scenario, this result empirically corroborates the bound in Proposition 4. Moreover, we observe that for higher-dimensional problems, the normalized OTSD converges to a specific constant, whereas for lower-dimensional problems, the convergence is less pronounced. This suggests that while the bound is reliable in high dimensions, it may not be as tight in low dimensions.

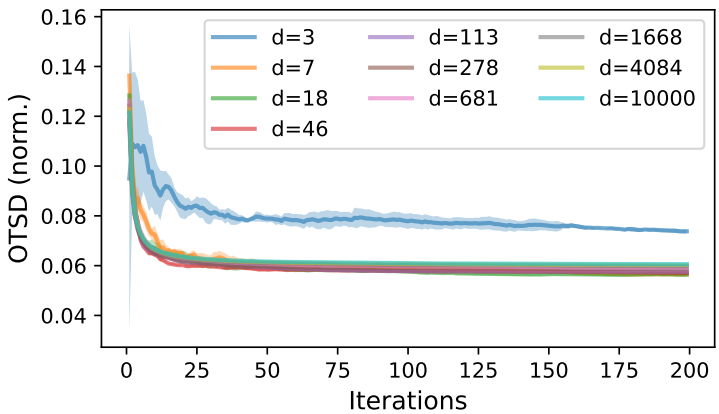
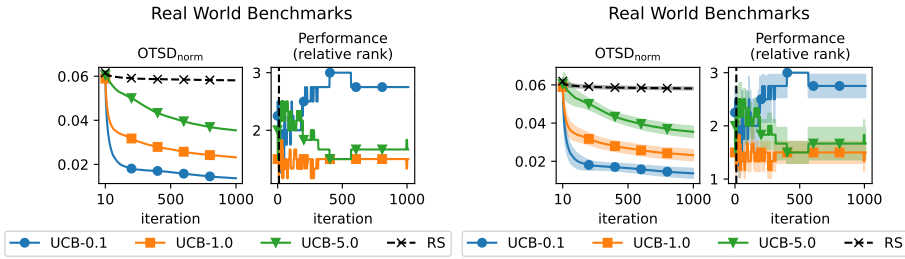


Figure 4.8: Normalized OTSD of Random Search for varying problem dimensions.

D Additional Experiments

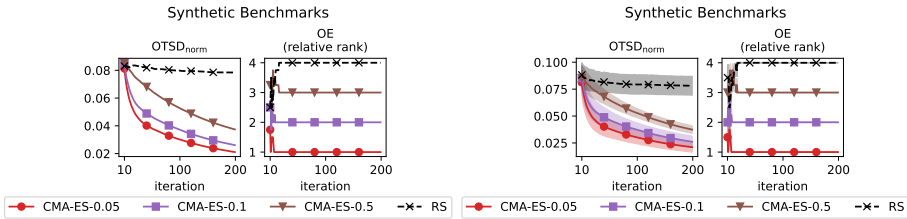
D.1 Empirical Validation of OTSD and OE

In this section, we study the OTSD for varying β-values for UCB in real-world settings and for `CMA-ES` with different step sizes, further supporting the adequacy of OTSD and OE for quantifying exploration.



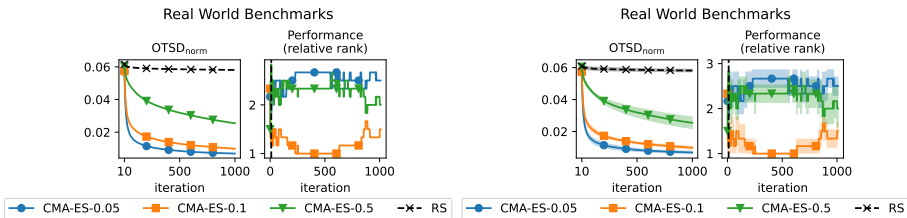
(a) Without error bars. (b) With error bars.

Figure 4.9: Normalized OTSD and mean ranks of the empirical performance for UCB with varying β-parameters on the real-world benchmarks.



(a) Without error bars. (b) With error bars.

Figure 4.10: Normalized OTSD and mean ranks of the empirical performance for `CMA-ES` with varying σ_0-parameters on the synthetic benchmarks.



(a) Without error bars. (b) With error bars.

Figure 4.11: Normalized OTSD and mean ranks of the empirical performance for`CMA-ES` with varying σ_0-parameters on the real-world benchmarks.

D.2 Error Bars for Main Text Figures

We report the figures from the main text with error bars indicating the standard error of the mean.

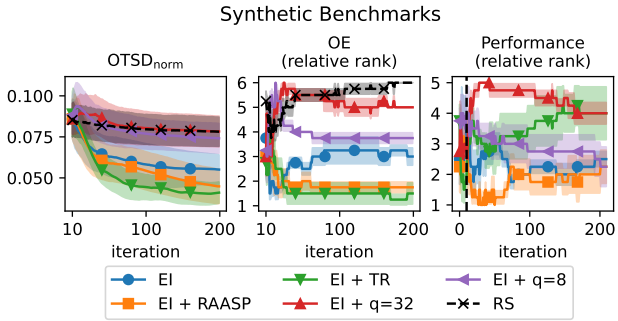


Figure 4.14: Normalized OTSD, average ranks for OE and optimization performance for EI and its variations on the synthetic benchmarks.

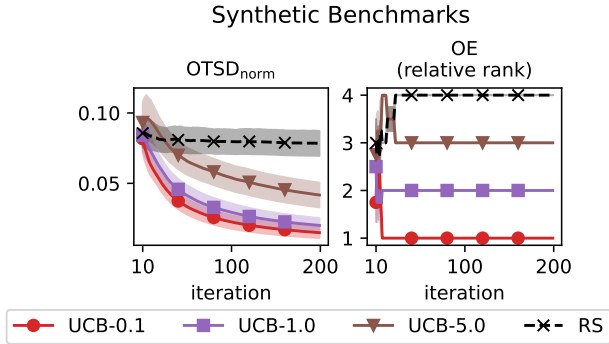


Figure 4.12: Normalized OTSD and OE rank averaged across all the synthetic benchmarks. A lower rank means lower exploration. We do not show the initial design of experiment phase.

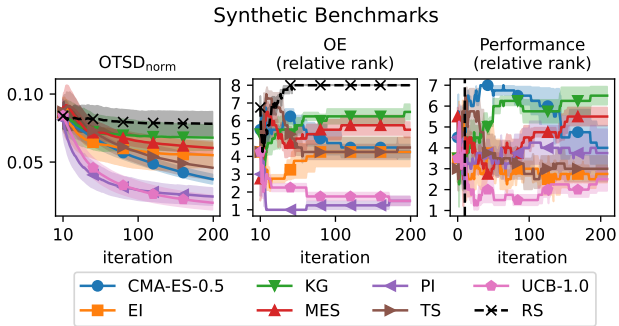


Figure 4.13: Normalized OTSD, average rank for OE and optimization performance on the synthetic benchmarks.

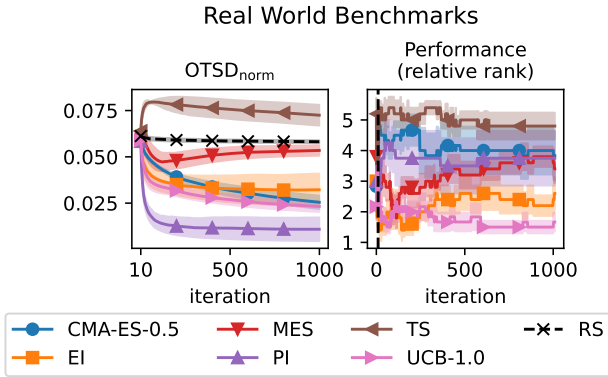
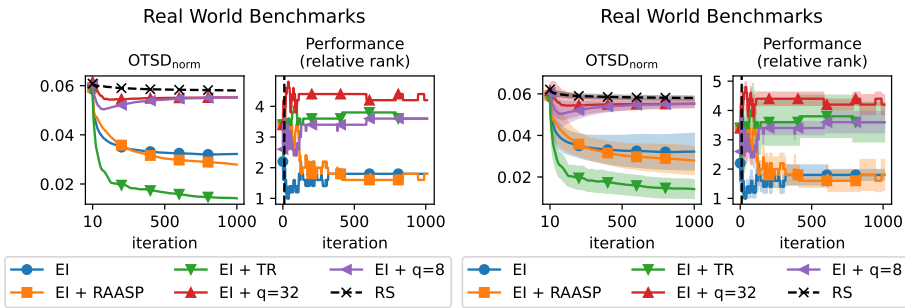


Figure 4.15: Normalized OTSD and average optimization performance rank on the real-world benchmarks.

D.3 TRs, RAASP, and batching on the Real-World Benchmarks



(a) Without error bars.

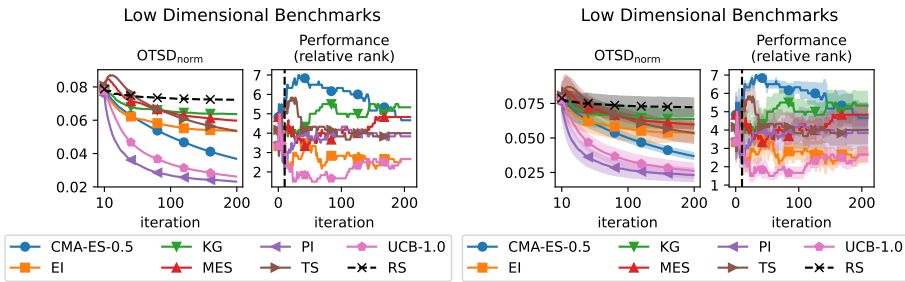
(b) With error bars.

Figure 4.16: Normalized OTSD and average optimization performance rank on the real-world benchmarks for EI and its variations. TRs and RAASP sampling promote exploitation.

Fig. 4.16 zooms in on the effect of batching, TR, and RAASP sampling. Here, the behavior is similar to the synthetic benchmarks: batching increases exploration and larger batch sizes lead to more exploration while RAASP sampling and TRs reduce exploration. In particular, the level of exploitation introduced by the TRs dominates all other methods. Compared to the optimization performance, both over-exploration and over-exploitation get punished: the most and least explorative methods ('EI + TR' and 'EI + q=32') show the worst empirical performance as indicated by the high average rank of the purple and blue curves in the right panel of Fig. 4.16.

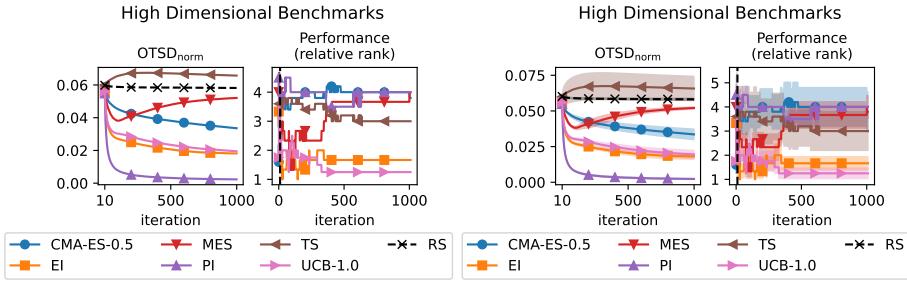
D.4 Impact of the Dimensionality

Next, we study how the dimensionality of problems affects exploration. To this end, we compare low-dimensional ($d \leq 20$, Fig. 4.17) and high-dimensional problems ($d > 20$, Fig. 4.18), unveiling significant differences between low- and high-dimensional regimes. While TS is eventually overtaken by MES in terms of exploration, it is by far the most explorative AF on high-dimensional problems. This is arguably due to the vast regions of high posterior uncertainty in high-dimensional spaces that allow diverse posterior samples. Similarly, MES is more explorative than other AFs (except for TS) in high-dimensional than low-dimensional spaces. Conversely, EI is considerably more explorative than UCB-1 in low-dimensional but not high-dimensional spaces. Arguably, the fast collapse of posterior uncertainty in low-dimensional spaces affects UCB-1 more than EI, making UCB almost as exploitative as PI. In both regimes, PI is most exploitative while showing mediocre to bad optimization performance. Brought together that the over-explorative TS also shows subpar optimization performance, we reinforce our conclusion that a balanced EETO is crucial for successful black-box optimizers.



(a) Without error bars. (b) With error bars.

Figure 4.17: Normalized OTSD and optimization performance ranks on the low-dimensional problems.



(a) Without error bars. (b) With error bars.

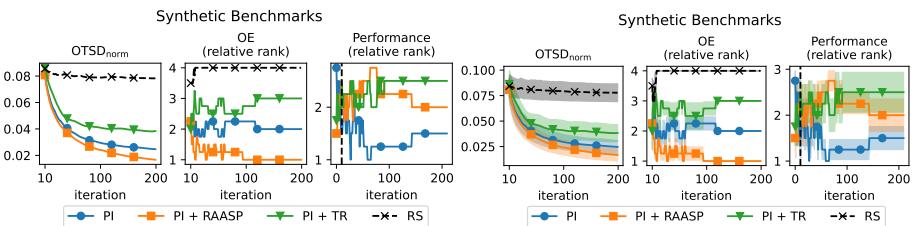
Figure 4.18: Normalized OTSD and optimization performance rank on the high-dimensional problems.

D.5 Optimizer Variations on Different Acquisition Functions

Here, we study the effect of RAASP sampling, TRs, and batching on all AFs used in Section 4.

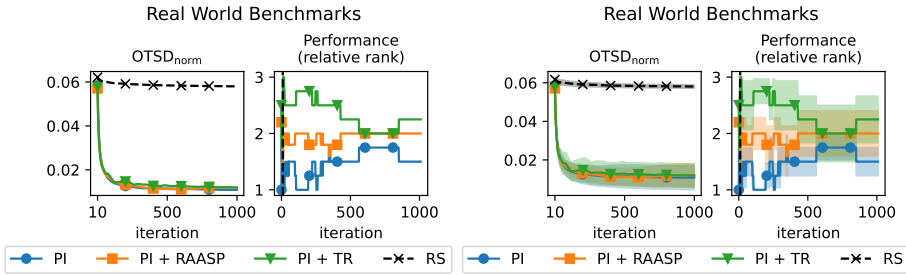
Probability of Improvement

For PI, we did not study batching as it is not implemented in `BoTorch`, so we only focus on TRs and RAASP sampling.



(a) Without error bars. (b) With error bars.

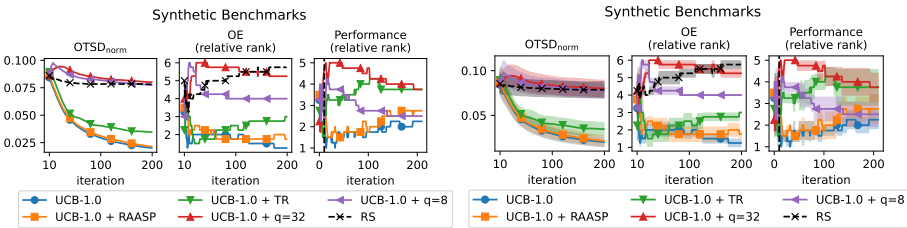
Figure 4.19: Effect of TRs and RAASP sampling on PI in the context of synthetic benchmarks: Both methods reduce the level of exploration.



(a) Without error bars. (b) With error bars.

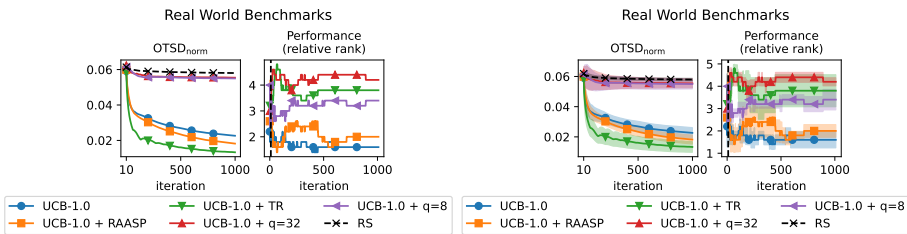
Figure 4.20: Effect of TRs and RAASP sampling on PI in the context of real-world benchmarks: Both methods reduce the level of exploration.

Upper Confidence Bounds



(a) Without error bars. (b) With error bars.

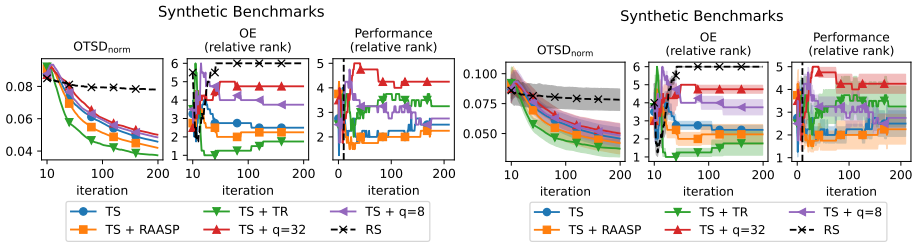
Figure 4.21: Effect of TRs, RAASP sampling, and batching on UCB-1 in the context of synthetic benchmarks:RAASP sampling and TRs reduce exploration, batching increases it.



(a) Without error bars. (b) With error bars.

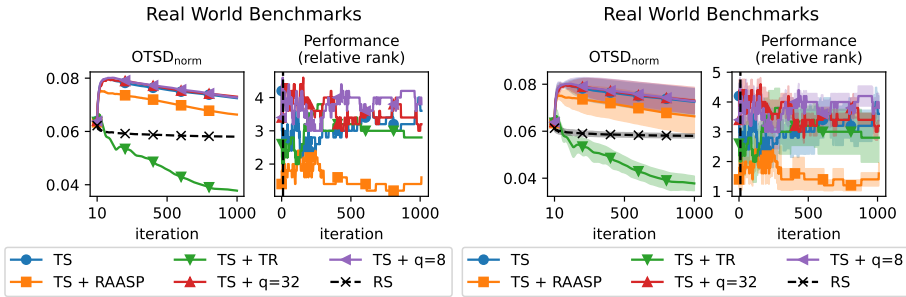
Figure 4.22: Effect of TRs, RAASP sampling, and batching on UCB-1 in the context of real-world benchmarks:RAASP sampling and TRs reduce exploration, batching increases it.

Thompson Sampling



(a) Without error bars. (b) With error bars.

Figure 4.23: Effect of TRs, RAASP sampling, and batching on TS in the context of synthetic benchmarks:RAASP sampling and TRs reduce exploration, batching increases it. RAASP sampling also improves optimization performance.

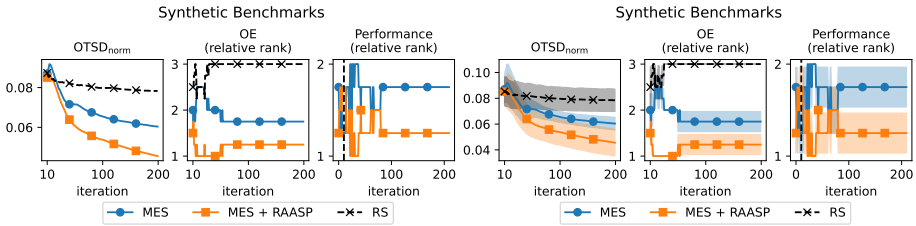


(a) Without error bars. (b) With error bars.

Figure 4.24: Effect of TRs, RAASP sampling, and batching on TS in the context of real-world benchmarks:RAASP sampling and TRs reduce exploration, batching increases it. RAASP sampling also improves optimization performance.

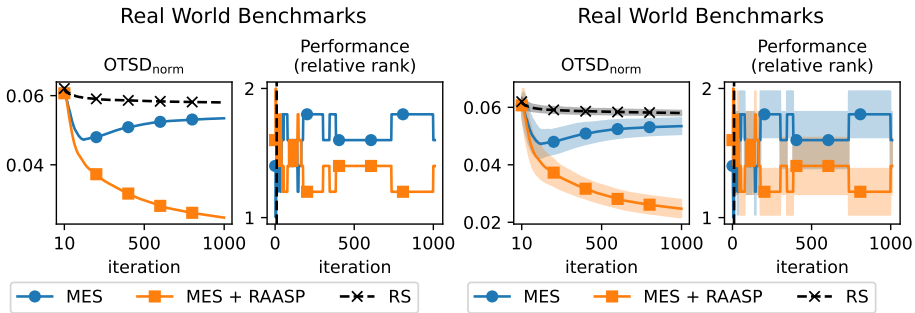
Max-Value Entropy Search

For MES, we only study the effect of RAASP sampling as we observed model-fitting errors for the TRs. Furthermore, batching is not implemented for MES in `BoTorch`.



(a) Without error bars. (b) With error bars.

Figure 4.25: Effect of RAASP sampling on MES in the context of synthetic benchmarks: RAASP sampling reduces the level of exploration.

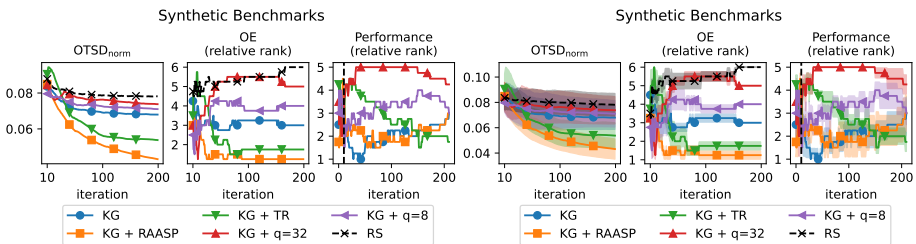


(a) Without error bars. (b) With error bars.

Figure 4.26: Effect of RAASP sampling on MES in the context of real-world benchmarks: RAASP sampling reduces the level of exploration.

Knowledge Gradient

We only ran KG for low-dimensional synthetic benchmarks due to its high computational cost in high dimensions.



(a) Without error bars. (b) With error bars.

Figure 4.27: Effect of TRs, RAASP sampling, and batching on KG in the context of synthetic benchmarks:RAASP sampling and TRs reduce exploration, batching increases it.

D.6 Optimization Performance

We present the optimization performance of various acquisition functions (AFs) and their variants for each individual benchmark. These results form the basis for the performance ranking plots in Section 4. For synthetic benchmarks with known optimal values, we show the simple regret, whereas we show the best-observed function value at each iteration for the real-world benchmarks with unknown optima.

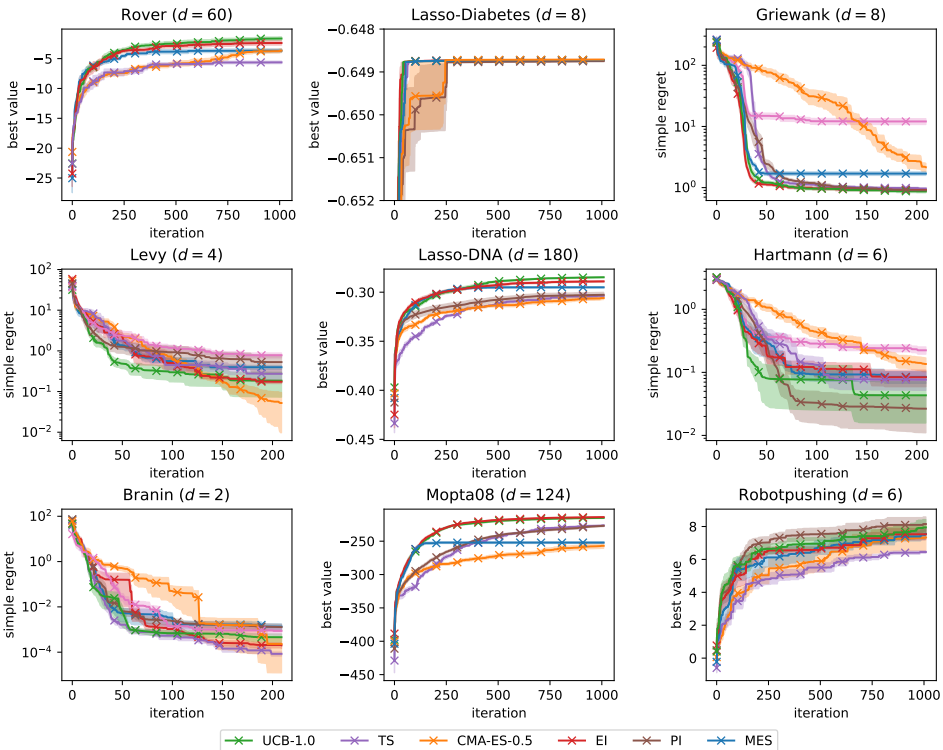


Figure 4.28: Optimization performance of the basic optimizer configuration on the different benchmarks.

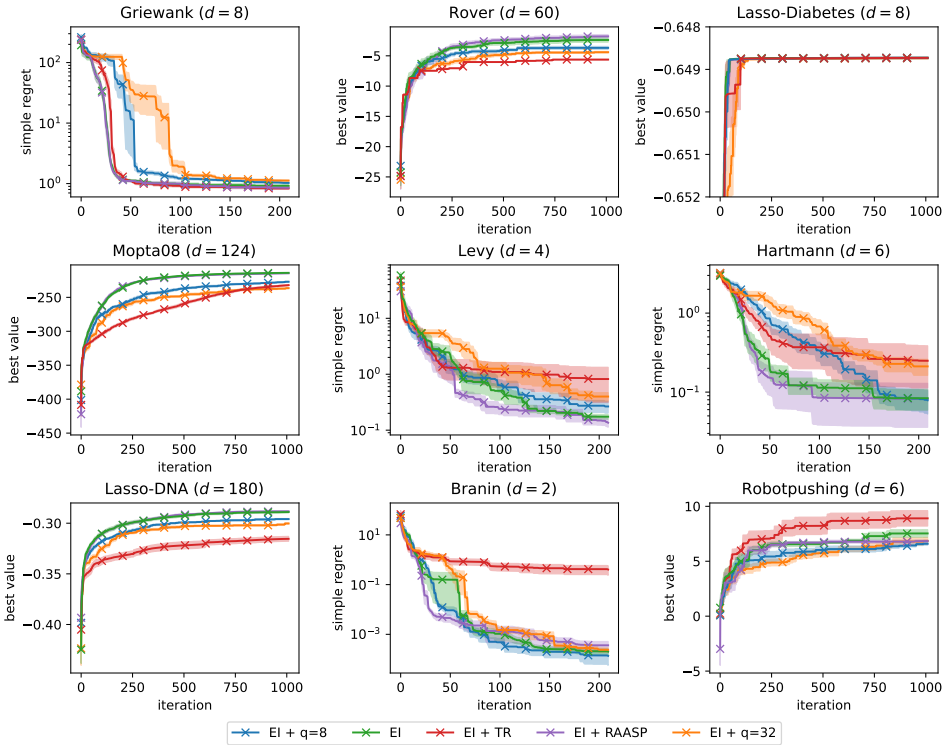


Figure 4.29: Optimization performance of the different EI variations on the benchmarks.

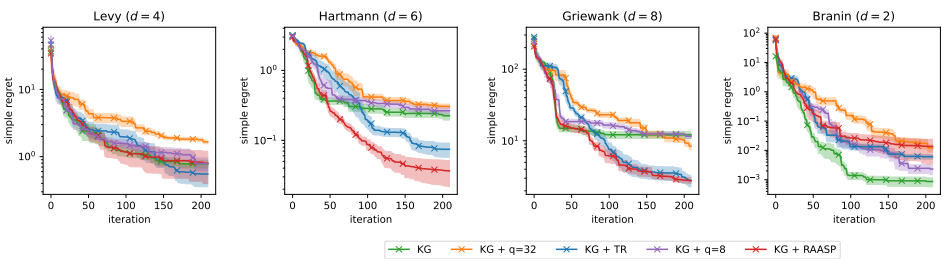


Figure 4.30: Optimization performance of the different KG variations on the benchmarks.

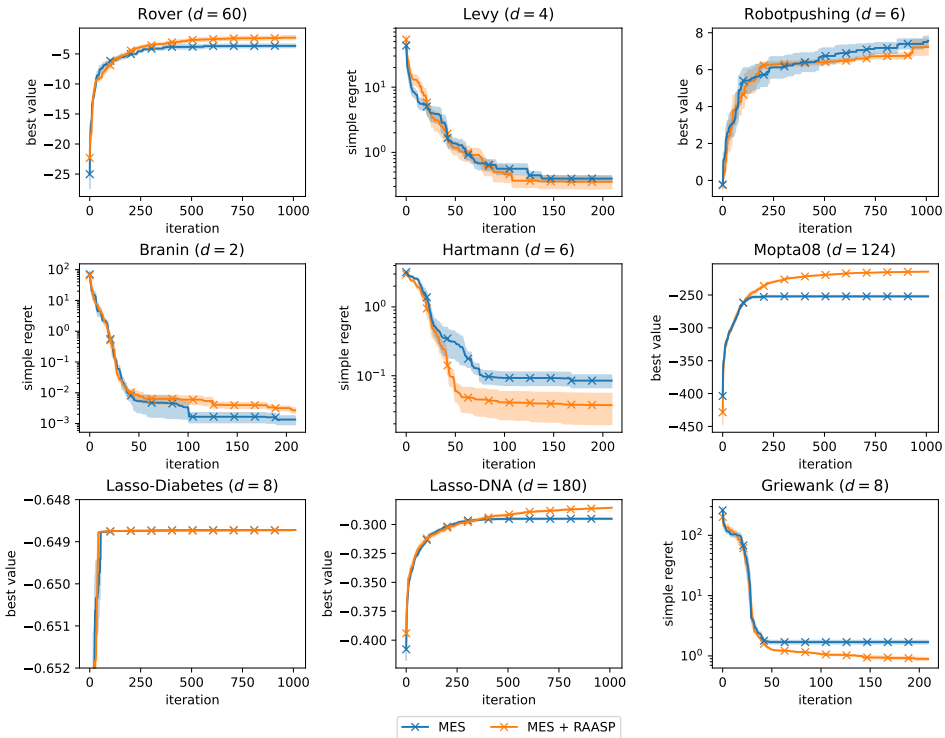


Figure 4.31: Optimization performance of the different MES variations on the benchmarks.

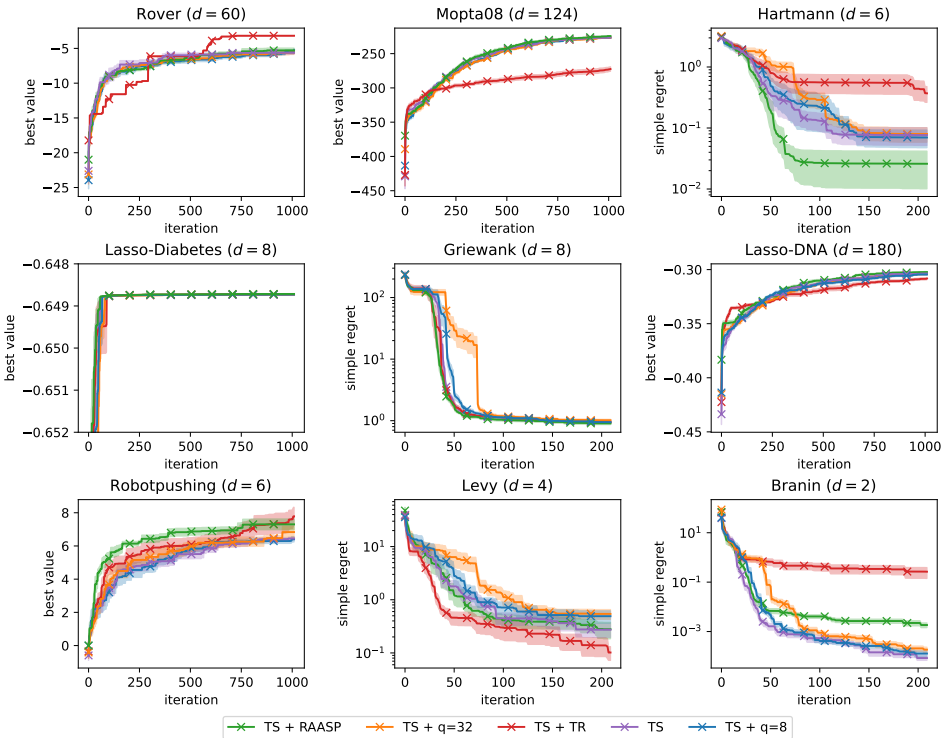


Figure 4.32: Optimization performance of the different TS variations on the benchmarks.

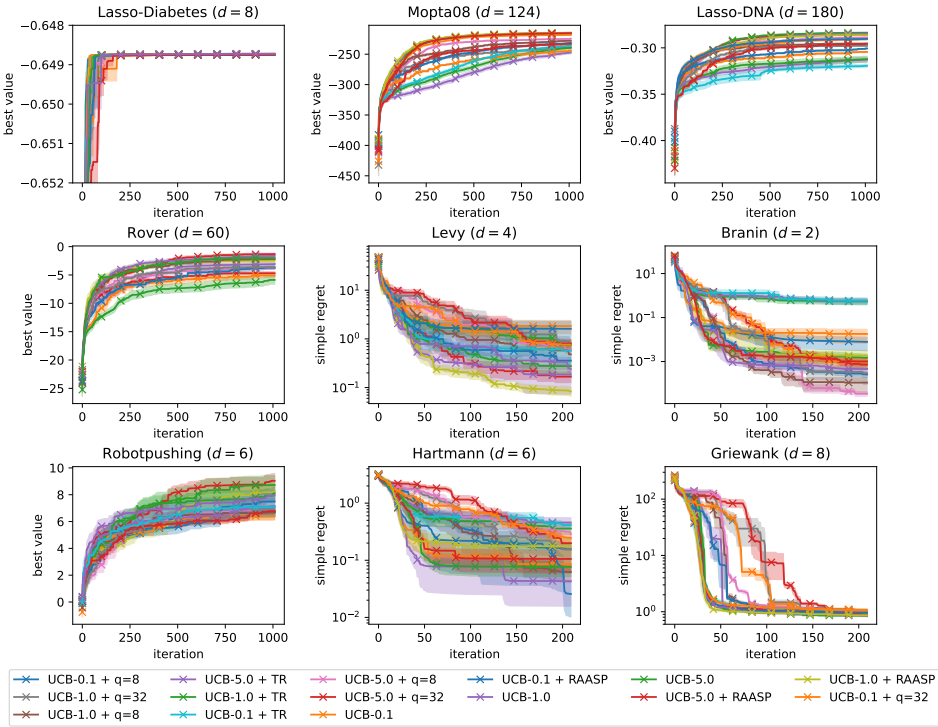


Figure 4.33: Optimization performance of the different UCB variations on the benchmarks.

E Runtime Analysis

We analyze the runtimes for OTSD and OE. We measure OTSD and OE on random sequences of length 1000 in different dimensionalities, ranging from 10 – 1000 for OTSD and from 2 – 20 for OE. We repeat each experiment 20 times and observe the runtimes for calculating OTSD and OE. Fig. 4.34 shows the runtimes for computing OTSD and OE. Computing OTSD is fast; for 1000 observation points in 10 dimensions it takes less than 200 ms and in 1000 dimensions less than 300 ms. OE is considerably more costly and restricted to low-dimensional spaces.

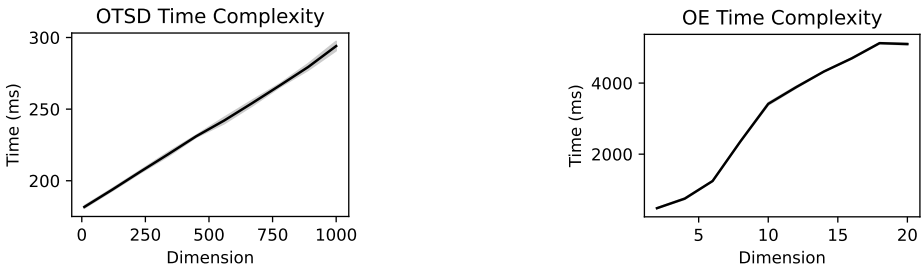


Figure 4.34: Runtimes for OTSD and OE in different dimensions. Empirically, OTSD scales linearly with the number of dimensions. OE is costlier.

References

[1] Salem F Adra and Peter J Fleming. Diversity management in evolutionary many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 15(2):183–195, 2010.

[2] Ibrahim Ahmad and Pi-Erh Lin. A nonparametric estimation of the entropy for absolutely continuous distributions (corresp.). *IEEE Transactions on Information Theory*, 22(3):372–375, 1976.

[3] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. *International conference on machine learning*, pages 151–160. PMLR, 2019.

[4] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[5] József Balogh, Felix Christian Clemen, and Adrian Dumitrescu. On a traveling salesman problem for points in the unit cube. *Algorithmica*, 86(9):3054–3078, 2024.

[6] Carolin Benjamins, Elena Raponi, Anja Jankovic, Koen van der Blom, Maria Laura Santoni, Marius Lindauer, and Carola Doerr. PI is back! Switching Acquisition Functions in Bayesian Optimization. *2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*, 2022.

[7] Thomas B Berrett, Richard J Samworth, and Ming Yuan. Efficient multivariate entropy estimation via k-nearest neighbour distances. *The Annals of Statistics*, 47(1):288–318, 2019.

[8] Hildo Bijl, Thomas B Schön, Jan-Willem van Wingerden, and Michel Verhaegen. A sequential Monte Carlo approach to Thompson sampling for Bayesian optimization. *arXiv preprint arXiv:1604.00169*, 2016.

[9] Bernd Bischl, Simon Wessing, Nadja Bauer, Klaus Friedrichs, and Claus Weihs. MOI-MBO: multiobjective infill for parallel model-based optimization. *Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers 8*, pages 173–186. Springer, 2014.

[10] Béla Bollobás and Amram Meir. A travelling salesman problem in the k-dimensional unit cube. *Operations research letters*, 11(1):19–21, 1992.

[11] Edmund Burke, Steven Gustafson, Graham Kendall, and Natalio Krasnogor. Advanced population diversity measures in genetic programming. *Parallel Problem Solving from Nature—PPSN VII: 7th International Conference Granada, Spain, September 7–11, 2002 Proceedings 7*, pages 341–350. Springer, 2002.

[12] Cambridge. Cambridge online dictionary. `https://dictionary.cambridge.org/us/dictionary/english/exploration`. Accessed: 2025-02-08.

[13] Nachol Chaiyaratana, Theera Piroonratana, and Nuntapon Sangkawelert. Effects of diversity control in single-objective and multi-objective genetic algorithms. *Journal of Heuristics*, 13:1–34, 2007.

[14] Nuojin Cheng, Leonard Papenmeier, Stephen Becker, and Luigi Nardi. A unified framework for entropy search and expected improvement in Bayesian optimization. *arXiv preprint arXiv:2501.18756*, 2025.

[15] Matej Črepinšek, Marjan Mernik, and Shih-Hsi Liu. Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees. *International Journal of Innovative Computing and Applications*, 3(1):11–19, 2011.

[16] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):1–33, 2013.

[17] George De Ath, Richard M Everson, Alma AM Rahat, and Jonathan E Fieldsend. Greed is Good: Exploration and Exploitation Trade-offs in Bayesian Optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1 (1):1–22, 2021.

[18] Sylvain Delattre and Nicolas Fournier. On the Kozachenko-Leonenko entropy estimator. *Journal of Statistical Planning and Inference*, 185:69–93, 2017.

[19] Luc Devroye and László Györfi. On the consistency of the Kozachenko-Leonenko entropy estimate. *IEEE Transactions on Information Theory*, 68(2): 1178–1185, 2021.

[20] Bach Do, Taiwo Adebiyi, and Ruda Zhang. Epsilon-greedy Thompson sampling to Bayesian optimization. *Journal of Computing and Information Science in Engineering*, 24(12), 2024.

[21] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

[22] David Eriksson. Trust Region Bayesian Optimization (TuRBO). BoTorch Tutorials, 2025. URL `https://botorch.org/docs/tutorials/turbo_1/`. Accessed: 2025-02-06.

[23] David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

[24] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5496–5507, 2019.

[25] Zhiwei Feng, Qingbin Zhang, Qingfu Zhang, Qiangang Tang, Tao Yang, and Yang Ma. A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization. *Journal of Global Optimization*, 61:677–694, 2015.

[26] Peter I Frazier. *Knowledge-Gradient Methods for Statistical Learning*. PhD thesis, Princeton University Princeton, 2009.

[27] David Ginsbourger and Rodolphe Le Riche. Towards Gaussian process-based optimization with finite time horizon. *mODa 9–Advances in Model-Oriented Design and Analysis: Proceedings of the 9th International Workshop in Model-Oriented Design and Analysis held in Bertinoro, Italy, June 14-18, 2010*, pages 89–96. Springer, 2010.

[28] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. *Artificial intelligence and statistics*, pages 648–657. PMLR, 2016.

[29] László Györfi and Edward C Van der Meulen. Density-free convergence properties of various estimators of entropy. *Computational Statistics & Data Analysis*, 5(4):425–436, 1987.

[30] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[31] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. Last access: 05/09/2022. License: BSD-3-Clause.

[32] Philipp Hennig and Christian J Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[33] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. *Advances in Neural Information Processing Systems*, 27, 2014.

[34] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

[35] Carl Hvarfner, Frank Hutter, and Luigi Nardi. Joint Entropy Search for Maximally-Informed Bayesian Optimization. *Advances in Neural Information Processing Systems*, 35:11494–11506, 2022.

[36] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian Optimization Performs Great in High Dimensions. Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20793–20817. PMLR, 21–27 Jul 2024.

[37] Kazuyuki Inoue, Taku Hasegawa, Naoki Mori, and Keinosuke Matsumoto. Analyzing exploration exploitation trade-off by means of PI similarity index and dictyostelium based genetic algorithm. *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2548–2555. IEEE, 2015.

[38] Jernej Jerebic, Marjan Mernik, Shih-Hsi Liu, Miha Ravber, Mihael Baketarić, Luka Mernik, and Matej Črepinšek. A novel direct measure of exploration and exploitation based on attraction basins. *Expert Systems with Applications*, 167: 114353, 2021.

[39] Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33:18039–18049, 2020.

[40] Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21:345–383, 2001.

[41] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455, 1998.

[42] Lyudmyla F Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problems of Information Transmission*, 23:9–16, 1987.

[43] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[44] Brian McGinley, John Maher, Colm O'Riordan, and Fearghal Morgan. Maintaining healthy population diversity using adaptive crossover, mutation, and selection. *IEEE Transactions on Evolutionary Computation*, 15(5):692–714, 2011.

[45] Alfonsas Misevičius. Generation of grey patterns using an improved genetic evolutionary algorithm: Some new results. *Information Technology and Control*, 40(4):330–343, 2011.

[46] Dávid Pál, Barnabás Póczos, and Csaba Szepesvári. Estimation of Rényi entropy and mutual information based on generalized nearest-neighbor graphs. *Advances in neural information processing systems*, 23, 2010.

[47] Bahador Rashidi, Kerrick Johnstonbaugh, and Chao Gao. Cylindrical Thompson sampling for high-dimensional Bayesian optimization. *International Conference on Artificial Intelligence and Statistics*, pages 3502–3510. PMLR, 2024.

[48] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006.

[49] Daniel J Rosenkrantz, Richard Edwin Stearns, and Philip M Lewis. Approximate algorithms for the traveling salesperson problem. *15th Annual Symposium on Switching and Automata Theory (swat 1974)*, pages 33–42. IEEE, 1974.

[50] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. *International Conference on Automated Machine Learning*, pages 2–1. PMLR, 2022.

[51] András Sóbester, Stephen J Leary, and Andy J Keane. On the Design of Optimization Strategies Based on Global Response Surface Approximation Models. *Journal of Global Optimization*, 33:31–59, 2005.

[52] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *Proceedings of the International Conference on Machine Learning*, 2010.

[53] Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal Bayesian exploration in dynamic environments. *Artificial General*

Intelligence: 4th International Conference, AGI 2011, Mountain View, CA, USA, August 3-6, 2011. Proceedings 4, pages 41–51. Springer, 2011.

[54] Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets, optimization test problems. `https://www.sfu.ca/~ssurjano/optimization.html`. Accessed: 2024-09-01.

[55] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[56] Ben Tu, Axel Gandy, Nikolas Kantas, and Behrang Shafei. Joint Entropy Search for Multi-objective Bayesian Optimization. *Advances in Neural Information Processing Systems*, 35:9922–9938, 2022.

[57] Mikdam Turkey and Riccardo Poli. A model for analysing the collective dynamic behaviour and characterising the exploitation of population-based algorithms. *Evolutionary Computation*, 22(1):159–188, 2014.

[58] Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian global optimization of expensive functions. *Operations Research*, 68(6):1850–1865, 2020.

[59] Zi Wang and Stefanie Jegelka. Max-value Entropy Search for Efficient Bayesian Optimization. *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

[60] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[61] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[62] James T Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Pathwise Conditioning of Gaussian Processes. *Journal of Machine Learning Research*, 22(105):1–47, 2021.

[63] Jian Wu. *Knowledge gradient methods for Bayesian optimization*. PhD thesis, Cornell University, 2017.

[64] Jian Wu and Peter Frazier. Practical two-step lookahead Bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[65] Antanas Žilinskas and James Calvin. Bi-objective decision making in global optimization based on statistical models. *Journal of Global Optimization*, 74: 599–609, 2019.

Paper v

A Unified Framework for Entropy Search and Expected Improvement in Bayesian Optimization

Accepted at the Forty-Second International Conference on Machine Learning

Nuojin Cheng*
University of Colorado Boulder

Leonard Papenmeier*
Lund University

Stephen Becker
University of Colorado Boulder

Luigi Nardi
Lund University

Abstract

Bayesian optimization (BO) is a widely used method for optimizing expensive black-box functions, with expected improvement (EI) being one of the most commonly used acquisition functions (AFs). In contrast, information-theoretic AFs aim to reduce uncertainty about the function's optimum and are often considered fundamentally distinct from EI. In this work, we challenge this prevailing perspective by introducing a unified theoretical framework, variational entropy search (VES), which reveals that EI and information-theoretic AFs are more closely related than

*Equal contribution.

previously recognized. We demonstrate that EI can be interpreted as a variational inference approximation of the popular information-theoretic AF, named max-value entropy search (MES). Building on this insight, we propose VES-Gamma, a novel AF that balances the strengths of EI and MES. Extensive empirical evaluations across both low- and high-dimensional synthetic and real-world benchmarks demonstrate that VES-Gamma is competitive with state-of-the-art AFs and, in many cases, outperforms EI and MES.

1 Introduction

Bayesian optimization (BO) is a widely used technique for maximizing black-box functions. Given a function $f : \mathcal{X} \to \mathbb{R}$, BO iteratively refines a probabilistic surrogate of f, typically a Gaussian process (GP), and selects the next evaluation point accordingly. At each iteration, the next sampling point is determined by maximizing an AF $\alpha : \mathcal{X} \to \mathbb{R}$. An effective AF must balance the exploration-exploitation trade-off (EETO), where exploitation prioritizes sampling points predicted by the surrogate to yield high objective values, while exploration targets regions with the potential to uncover even better values.

EI [23] is one of the most widely used AFs, valued for its simple formulation, computational efficiency, and strong empirical performance. The core idea behind EI is to maximize the expected improvement over the current best-observed value, which typically requires a noise-free assumption. More recently, [37, 11] have introduced the concepts of information-theoretic AFs, which represents a paradigm shift in BO. Unlike EI, which focuses on directly maximizing potential improvement, information-theoretic AFs aim to reduce uncertainty about the function f's optimal position and/or value, often through entropy-based measures. Due to their fundamentally different underlying philosophies and selection criteria, EI and information-theoretic AFs are widely regarded as distinct methodologies within the BO community [12].

Despite their apparent differences, we argue that EI and information-theoretic AFs share deeper theoretical connections than previously recognized. Understanding this relationship is crucial, as it provides novel insights into designing new AFs. By unifying the perspectives of both sides, we introduce VES-Gamma, a new AF that effectively balances their strengths, resulting in a robust AF that adapts well to diverse optimization problems. VES-Gamma inherits the performance of EI while incorporating information-theoretic considerations.

In summary, we make the following key contributions:

1. We introduce the VES framework which shows that EI can be interpreted as a special case of the popular information-theoretic AF MES. This unified theoretical perspective reveals that these two types of AFs are more closely related than previously recognized.

2. We propose VES-Gamma as an intermediary between EI and MES, incorporating information-theoretic principles while maintaining EI's strength in performance.

3. We provide an extensive evaluation across a diverse set of low- and high-dimensional synthetic, GP samples, and real-world benchmarks, demonstrating that VES-Gamma consistently performs competitively and, in many cases, outperforms both EI and MES.

2 Background and Related Work

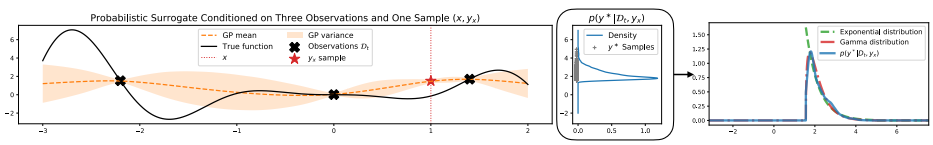


Figure 5.1: MES aims to optimize $\boldsymbol{x}$ such that the entropy (averaged over all $y_{\boldsymbol{x}}$) of the maximum values $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ is reduced. The left figure illustrates a noiseless GP conditioned on the observations $\mathcal{D}_t$ with three points (black crosses) and a sample $y_{\boldsymbol{x}}$ at $x = 1$ drawn from $p(y_{\boldsymbol{x}} \mid \mathcal{D}_t)$ (red star). The mid and right panels illustrate the density $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ (blue curves). When $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ is approximated using an exponential distribution (green dashed curve), this leads to the VES-Exp AF that is equivalent to EI. Furthermore, VES-Gamma, which approximates $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ using a Gamma distribution (red dash-dot curve), leads to a more accurate approximation and a generalized version of EI.

2.1 Gaussian Processes

A GP is a stochastic process that models an unknown function. It is characterized by the property that any finite set of function evaluations follows a multivariate Gaussian distribution. Assuming that f has a zero mean, a GP is uniquely determined by the current observations $\mathcal{D}_t \coloneqq \{(\boldsymbol{x}_i, y_{\boldsymbol{x}_i})\}_{i=1}^t$ and the kernel function $\kappa(\boldsymbol{x}, \boldsymbol{x}')$. Given these, at stage t, the predicted mean of $y_{\boldsymbol{x}}$ at a new point $\boldsymbol{x}$ is $\mu_t(\boldsymbol{x}) = \boldsymbol{\kappa}_t(\boldsymbol{x})^T (\boldsymbol{K}_t)^{-1} \boldsymbol{y}_t$, and the predicted covariance between points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is $\mathrm{Cov}_t(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{\kappa}_t(\boldsymbol{x})^T (\boldsymbol{K}_t)^{-1} \boldsymbol{\kappa}_t(\boldsymbol{x}')$, where $[\boldsymbol{\kappa}_t(\boldsymbol{x})]_i = \kappa(\boldsymbol{x}_i, \boldsymbol{x})$, $[\boldsymbol{y}_t]_i = y_{\boldsymbol{x}_i}$, and $[\boldsymbol{K}_t]_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$; see Rasmussen et al. [29] for more details.

2.2 Acquisition Functions

Various AFs have been proposed to balance exploration and exploitation in optimization tasks, each tailored to different problem characteristics and assumptions. These include probability of improvement (PI), EI [23], upper-confidence bound (UCB) [32], knowledge gradient (KG) [9], and information-theoretic AFs [37, 11, 13, 38, 16, 36]. Below, we discuss two types of AFs relevant to this study.

Expected Improvement. EI is one of the most commonly used AFs and is formulated as follows:

$$\alpha_{\text{EI}}(\boldsymbol{x}) = \mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}\big[\max\{y_{\boldsymbol{x}}, y_t^*\}\big] - y_t^*, \tag{5.1}$$

where y_t^* is the maximum observed value in $\mathcal{D}_t$, and $\mathbb{E}_{p(\cdot)}$ denotes the expectation with respect to the predictive density $p(\cdot)$. The $-y_t^*$ term at the end can be dropped since it is constant with respect to $\boldsymbol{x}$.

Information-Theoretic AFs. Information-theoretic AFs form a family of methods designed to select $\boldsymbol{x}$ such that its evaluation reduces uncertainty regarding the optimal points of the objective function. This uncertainty is quantified using differential entropy, defined as $\mathbb{H}[y] \coloneqq \mathbb{E}_{p(y)}[-\log p(y)]$. Similarly, the conditional entropy is expressed as $\mathbb{H}[y|x] \coloneqq \mathbb{H}[x, y] - \mathbb{H}[x]$.

The first information-theoretic AF for BO is entropy search (ES) [11], which is formulated as:

$$\alpha_{\text{ES}}(\boldsymbol{x}) = \mathbb{H}[\boldsymbol{x}^* \mid \mathcal{D}_t] - \mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}\left[\mathbb{H}[\boldsymbol{x}^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}]\right]. \tag{5.2}$$

Here, the random variable $\boldsymbol{x}^*$ represents the location of the maximum.

Predictive entropy search (PES) [13] offers a reformulation of ES that is computationally more efficient:

$$\alpha_{\text{PES}}(\boldsymbol{x}) = \mathbb{H}[y_{\boldsymbol{x}} \mid \mathcal{D}_t] - \mathbb{E}_{p(\boldsymbol{x}^*|\mathcal{D}_t)}\left[\mathbb{H}[y_{\boldsymbol{x}} \mid \mathcal{D}_t, \boldsymbol{x}^*]\right]. \tag{5.3}$$

Since directly estimating the entropy with $\boldsymbol{x}^*$ is expensive, following the PES format, MES[38] introduced an alternative approach that focuses on reducing the differential entropy of the 1D maximum value y^*:

$$\begin{aligned}\alpha_{\text{MES}}(\boldsymbol{x}) &= \mathbb{H}[y^* \mid \mathcal{D}_t] - \mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}\left[\mathbb{H}[y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}]\right] \\ &= \underbrace{\mathbb{H}[y_{\boldsymbol{x}} \mid \mathcal{D}_t]}_{\text{closed-form}} - \mathbb{E}_{p(y^*|\mathcal{D}_t)}\underbrace{\left[\mathbb{H}[y_{\boldsymbol{x}} \mid \mathcal{D}_t, y^*]\right]}_{\text{non-closed-form}}.\end{aligned} \tag{5.4}$$

Unlike MES and its subsequent extensions [16, 35] which approximate $p(y_{\boldsymbol{x}} \mid \mathcal{D}_t, y^*)$ using a truncated Gaussian, we focus on directly estimating $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ via variational inference (VI).

2.3 Related Work

Variational Inference and Evidence Lower Bound. VI is a widely used technique in Bayesian modeling to approximate intractable posterior distributions [24, 15, 21]. It relies on maximizing the evidence lower bound (ELBO) to approximate the log-likelihood $\log p(\tilde{\boldsymbol{x}})$ in the presence of latent variables $\boldsymbol{z}$. The log-likelihood can be decomposed as follows:

$$\log p(\tilde{\boldsymbol{x}}) \geq \mathbb{E}_{q(\boldsymbol{z})}\left[\log\left(\frac{p(\tilde{\boldsymbol{x}} \mid \boldsymbol{z})p(\boldsymbol{z})}{q(\boldsymbol{z})}\right)\right], \tag{5.5}$$

where $p(\boldsymbol{z})$ is a fixed prior distribution, and $q(\boldsymbol{z})$ is a variational approximation to the true posterior $p(\boldsymbol{z} \mid \tilde{\boldsymbol{x}})$. The ELBO is formally defined as:

$$\mathrm{ELBO}(p(\tilde{\boldsymbol{x}} \mid \boldsymbol{z}); q(\boldsymbol{z})) \coloneqq \mathbb{E}_{q(\boldsymbol{z})}\left[\log\left(\frac{p(\tilde{\boldsymbol{x}} \mid \boldsymbol{z})p(\boldsymbol{z})}{q(\boldsymbol{z})}\right)\right]. \tag{5.6}$$

By maximizing the ELBO, VI indirectly maximizes the log-likelihood $\log p(\tilde{\boldsymbol{x}})$, thereby improving the quality of the posterior approximation. In many applications, such as variational autoencoders (VAEs) [21] and variational diffusion [20], both the conditional likelihood $p(\tilde{\boldsymbol{x}} \mid \boldsymbol{z})$ and the variational distribution $q(\boldsymbol{z})$ are parameterized using neural networks (NNs). Since both the expectation reference probability and the term inside the ELBO are parameterized, one common strategy is to estimate the gradient using finite Monte Carlo samples and the *reparameterization trick* to optimize the parameters. We adopt this approach, which enables efficient gradient-based optimization and has been widely applied in the BO community [40].

Improving the Expected Improvement. It is widely recognized that EI can be prone to over-exploitation [27, 6, 7]. To mitigate this issue, Hoffman et al. [14] and Kandasamy et al. [19] propose to use a portfolio of AFs, which assigns probabilities to different AFs at each step. Another approach is Weighted EI (WEI), which adaptively adjusts the weights of the components within the EI AF [31, 5]. Similarly, Qin et al. [27] suggest "weakening" EI using sub-optimal points suggested by the AF to mitigate its over-exploitative behavior. However, these methods are primarily based on heuristics. Furthermore, information-theoretic AFs are often excluded from these design enhancements, as they are generally considered distinct from heuristic AFs such as PI, EI, UCB, or KG.

Entropy Approximation in Information-theoretic acquisition functions. Estimating entropy in information-theoretic AFs is computationally expensive and typically requires approximation techniques. Methods such as ES and PES employ sampling-based approaches, including Markov-Chain Monte-Carlo (MCMC) and expectation propagation. In contrast, MES derives an explicit approximation [38, Eq. 6], which was later interpreted as a VI formulation by Takeno et al. [34]. This variational perspective has since been extended to multi-objective optimization [28]. However, this approximation scheme lacks flexibility in tuning the variational distributions. Furthermore, to the best of our knowledge, most MES-based methods focus on approximating $p(y_{\boldsymbol{x}} \mid y^*, \mathcal{D}_t)$. An exception is Ma et al. [22], which approximates $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ using a Gaussian distribution. While this approach provides computational advantages, the inherent symmetry of the Gaussian distribution does not align with the properties of y^*.

3 Variational Entropy Search

3.1 Entropy Search Lower Bound

The idea behind our VES framework is to maximize a variational lower bound of MES with a predetermined family of densities to approximate $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$. Since we assume noiseless observations, the support is $[\max\{y_{\boldsymbol{x}}, y_t^*\}, +\infty)$. VES is illustrated in Fig. 5.1. The lower bound is formalized in Theorem 6 and proven in Appendix A.1.

Theorem 6. *The MES AF in Eq.* (5.4) *adheres to the Barber-Agakov (BA) bound [4, 25] and can be bounded from below as follows:*

$$\begin{aligned} \alpha_{MES}(\boldsymbol{x}) &= \mathbb{H}[y^* \mid \mathcal{D}_t] - \mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}\big[\mathbb{H}[y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}]\big] \\ &\geq \mathbb{H}[y^* \mid \mathcal{D}_t] + \mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}\big[\log q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})\big], \end{aligned} \tag{5.7}$$

where $q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ is any chosen density function that is absolutely continuous with respect to $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$.

Since the first term on the right-hand side of Eq. (5.7), $\mathbb{H}[y^* \mid \mathcal{D}_t]$, is independent of both q and $\boldsymbol{x}$, we can omit it. This leads us to define the remaining term as the entropy search lower bound (ESLBO):

$$\text{ESLBO}(\boldsymbol{x}; q) \coloneqq \mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}\big[\log q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})\big], \tag{5.8}$$

where $p(y^*, y_{\boldsymbol{x}} \mid \mathcal{D}_t)$ represents a joint density, which can be sampled using GP path sampling [13, 38].

To optimize $\alpha_{\text{MES}}(\boldsymbol{x})$, we adopt the VI approach [24], indirectly maximizing $\alpha_{\text{MES}}(\boldsymbol{x})$ by instead maximizing ESLBO. To ensure computational feasibility, the VI method constrains the density q to a predefined family $\mathcal{Q}$. When parameterizing q within $\mathcal{Q}$, the problem becomes tractable by solving for q and $\boldsymbol{x}$ iteratively, as detailed in Algorithm 5.1.

Notably, this procedure, known as expectation maximization (EM), is analogous to maximizing the ELBO in Eq. (5.5). We conclude our discussion by summarizing the correspondence between ESLBO and ELBO in Table 5.5.

Algorithm 5.1 VES Framework

Input: Observations $\mathcal{D}_t$, variational family $\mathcal{Q}$, number of inner iteration N
Output: Next sampling location $\boldsymbol{x}_{t+1}$

1: initialize $\boldsymbol{x}_{t+1}^{(0)}$
2: **for** $n = 1 : N$ **do**
3: $\quad q^{(n)}(y^*) \leftarrow \arg\max_{q \in \mathcal{Q}} \text{ESLBO}(\boldsymbol{x}_{t+1}^{(n-1)}; q)$
4: $\quad \boldsymbol{x}_{t+1}^{(n)} \leftarrow \arg\max_{\boldsymbol{x}_{t+1}} \text{ESLBO}(\boldsymbol{x}_{t+1}; q^{(n)})$
5: return $\boldsymbol{x}_{t+1}^{(N)}$

3.2 EI Through the Lens of the VES Framework

In this section, we aim to establish an explicit connection between the VES and EI AFs, allowing us to see EI through the lens of a VI approximation of the information-theoretical MES AF. We define $\mathcal{Q}$ as the set of all exponential density functions, $\mathcal{Q}_{\text{exp}}$, parameterized by the $\lambda > 0$ exponential density parameter and with support bounded from below by $\max\{y_{\boldsymbol{x}}, y_t^*\}$. The variational density function q is given by

$$q(y^*|\mathcal{D}_t, y_{\boldsymbol{x}}; \lambda) = \lambda e^{-\lambda(y^* - \max\{y_{\boldsymbol{x}}, y_t^*\})} \mathbf{1}_{y^* \geq \max\{y_{\boldsymbol{x}}, y_t^*\}}. \tag{5.9}$$

For noiseless observations, the indicator function $\mathbf{1}_{y^* \geq \max\{y_{\boldsymbol{x}}, y_t^*\}}$ always equals one and can be omitted. Plugging in q from Eq. (5.9) into the ESLBO (Eq. (5.8)) yields a new λ-parameterized AF. Since this AF stems from the exponential distribution, we name it VES-Exp. Theorem 7 shows that the next sampling point generated from VES-Exp within Algorithm 5.1 will be the same as for the EI AF; the theorem is proven in Appendix A.2.

Theorem 7. *When the family $\mathcal{Q}_{\text{exp}}$ is selected as in Eq.* (5.9) *and the function is noiseless,*

Table 5.5: Comparison of key aspects between the ELBO and ESLBO approaches.

Property	ELBO Approach	ESLBO Approach
Primary Variable	$p(\tilde{\boldsymbol{x}} \mid \boldsymbol{z})$	$\boldsymbol{x}$
Variational Variable	$q(\boldsymbol{z})$	$q(y^* \mid y_{\boldsymbol{x}}, \mathcal{D}_t)$
Lower Bound Formulation	$\text{ELBO}(q(\boldsymbol{z}); p(\tilde{\boldsymbol{x}} \mid \boldsymbol{z}))$	$\text{ESLBO}(q; \boldsymbol{x})$

ESLBO in Eq. (5.8) *turns into*

$$\begin{aligned} \mathit{ESLBO}(\boldsymbol{x}; \lambda) = \log \lambda - \lambda \underbrace{\mathbb{E}_{p(y^*|\mathcal{D}_t)}[y^*]}_{\text{constant}} \\ + \lambda \underbrace{\mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}\left[\max\{y_{\boldsymbol{x}}, y_t^*\}\right]}_{\text{EI}}. \end{aligned} \tag{5.10}$$

Maximizing ESLBO$(\boldsymbol{x}; \lambda)$ *in Eq.* (5.10) *with respect to* $\boldsymbol{x}$ *and* λ *yields the same* $\boldsymbol{x}$ *solution as the maximization of EI in Eq.* (5.1).

The key idea behind the proof is that, following Algorithm 5.1, the ESLBO in Eq. (5.10) always converges within two iterations. Regardless of the positive value of λ, the value of $\boldsymbol{x}$ that maximizes $\text{ESLBO}(\boldsymbol{x}; \lambda)$ remains the same. Consequently, starting from an arbitrary initial point $\boldsymbol{x}^{(0)}$, a positive $\lambda^{(1)}$ is derived, ensuring that ESLBO reaches its maximum value in the next iteration.

Theorem 7 reveals that EI can be viewed as a special case of MES, giving a new information-theoretic interpretation of the most popular AF in use today. However, the exponential distribution has a fairly rigid parametric form that does not capture the characteristics of $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$. Fig. 5.1 (right) shows an example of the structural limitations of the exponential density in green. We generate 1000 samples from an example distribution $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$, and observe that it significantly deviates from an exponential distribution. Specifically, the density of $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ is non-monotonic, exhibiting a peak before decreasing near $\max\{y_{\boldsymbol{x}}, y_t^*\}$ (approximately 1.55), while exponential distributions are necessarily monotonic.

This observation motivates the need to enrich the variational distributions $\mathcal{Q}$ to allow more flexibility. A natural extension is to use a Gamma distribution, which is a generalization of the exponential distribution. The Gamma density approximation in the previous example is shown in red in Fig. 5.1 (right). The next section introduces VES-Gamma, which is a more general AF that extends VES-Exp and its equivalent EI AF.

3.3 VES-Gamma: A Generalization of EI

VES-Gamma defines $\mathcal{Q}$ as the Gamma distribution parameterized by $k, \beta > 0$ with its support bounded from below by $\max\{y_{\boldsymbol{x}}, y_t^*\}$. The variational density is

$$\begin{aligned} q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}; k, \beta) = \frac{\beta^k}{\Gamma(k)} \left(y^* - \max\{y_{\boldsymbol{x}}, y_t^*\}\right)^{k-1} \\ \times e^{-\beta(y^* - \max\{y_{\boldsymbol{x}}, y_t^*\})} \mathbf{1}_{y^* \geq \max\{y_{\boldsymbol{x}}, y_t^*\}}, \end{aligned} \tag{5.11}$$

where $\Gamma(\cdot)$ denotes the Gamma function. The noise-free assumption allows us to omit the indicator function, and the ESLBO is reformulated as

$$\begin{aligned} \mathrm{ESLBO}(\boldsymbol{x}; k, \beta) = k \log \beta - \log \Gamma(k) \\ + (k-1)\mathbb{E}_{p(y^*, y_{\boldsymbol{x}} \mid \mathcal{D}_t)} \left[\log\left(y^* - \max\{y_{\boldsymbol{x}}, y_t^*\}\right)\right] \\ - \beta \mathbb{E}_{p(y^* \mid \mathcal{D}_t)}[y^*] + \beta \underbrace{\mathbb{E}_{p(y_{\boldsymbol{x}} \mid \mathcal{D}_t)}[\max\{y_{\boldsymbol{x}}, y_t^*\}]}_{\mathrm{EI}}. \end{aligned} \tag{5.12}$$

The ESLBO in Eq. (5.12) serves as the primary objective in the VES-Gamma algorithm. Eq. (5.12) consists of five terms, with the last term being the EI AF in Eq. (5.1) scaled by a multiplicative factor. The two hyperparameters, k and β, originally part of the Gamma distribution, dynamically balance different components of the objective. In particular, when $k = 1$, the Gamma distribution reduces to an exponential distribution, making the ESLBO in Eq. (5.12) equivalent to Eq. (5.10). In the following section, we discuss the approach for determining optimal values for k and β.

Auto-determination of Tradeoff Hyperparameters. For any fixed $\boldsymbol{x}$, the global maximum of the ESLBO in Eq. (5.12) with respect to k and β uniquely exists, as can be demonstrated through derivative analysis. Taking the partial derivatives of ESLBO in Eq. (5.12) and setting them to zero, we obtain:

$$\log \beta - \frac{\partial \log \Gamma(k)}{\partial k} + \mathbb{E}\left[\log z_{\boldsymbol{x}}^*\right] = 0, \quad \frac{k}{\beta} - \mathbb{E}\left[z_{\boldsymbol{x}}^*\right] = 0,$$

where the random variable $z_{\boldsymbol{x}}^* := y^* - \max\{y_{\boldsymbol{x}}, y_t^*\}$.

Substituting the second equation into the first yields:

$$\log k - \psi(k) = \log \mathbb{E}[z_{\boldsymbol{x}}^*] - \mathbb{E}[\log z_{\boldsymbol{x}}^*], \tag{5.13}$$

where $\psi(k) := \partial \log \Gamma(k) / \partial k$ is the digamma function [1], which can be efficiently approximated as a series. By Jensen's inequality, $\log \mathbb{E}[z_{\boldsymbol{x}}^*] - \mathbb{E}[\log z_{\boldsymbol{x}}^*] \geq 0$. Since

$\log k - \psi(k)$ is strictly decreasing and approaches zero asymptotically (see Fig. 5.2), the root of Eq. (5.13), $k^*_{\boldsymbol{x}}$, exists uniquely—except in the degenerate case where $z^*_{\boldsymbol{x}}$ is deterministic, in which case we apply a clamping function to prevent it from being zero.

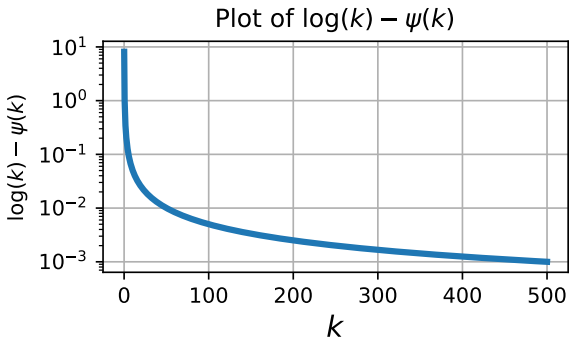


Figure 5.2: Plot of $\log k - \psi(k)$ for $k \in [0, 500]$. The function is strictly decreasing and asymptotically approaches zero.

With this analysis, the optimal value $k^*_{\boldsymbol{x}}$ is determined by solving the root of Eq. (5.13) using a one-dimensional root-finding method, where expectations of $z^*_{\boldsymbol{x}}$ are estimated via Monte Carlo sampling from $p(y^*, y_{\boldsymbol{x}} \mid \mathcal{D}_t)$. Once $k^*_{\boldsymbol{x}}$ is obtained, the optimal $\beta^*_{\boldsymbol{x}}$ follows as:

$$\beta^*_{\boldsymbol{x}} \leftarrow \frac{k^*_{\boldsymbol{x}}}{\mathbb{E}\left[z^*_{\boldsymbol{x}}\right]}. \tag{5.14}$$

Notably, the weighting parameters $k^*_{\boldsymbol{x}}$ and $\beta^*_{\boldsymbol{x}}$ are location-dependent, as $z^*_{\boldsymbol{x}}$ itself varies with $\boldsymbol{x}$. The VES-Gamma algorithm, which incorporates these principles, is detailed in Algorithm 5.2.

Although we provide both a theoretical justification and a practical implementation for the VES-Gamma AF, a deeper interpretation of the ESLBO in Eq. (5.12) remains an open research question. Due to the complex non-linear structure of Eq. (5.12), it is currently uncertain if there is a clear and straightforward interpretation of the various terms and the overall expression. As an example, we hypothesize that the third term acts as an "anti-EI" component, steering the VES-Gamma solution away from the EI recommendation to promote diversity, with the values of $\beta^*_{\boldsymbol{x}}$ and $k^*_{\boldsymbol{x}}$ dynamically balancing its influence. Investigating this hypothesis and further elucidating the role of each term within ESLBO will be the focus of future research.

Computational Cost of VES-Gamma. Implementing VES-Gamma in Algorithm 5.2 is computationally intensive. The number of inner iterations, N, must be sufficiently large for convergence, and each inner iteration

Algorithm 5.2 VES-Gamma

Input: Sample set $\mathcal{D}_t$, number of inner iterations N
Output: Next sampling location $\boldsymbol{x}_{t+1}$

initialize $\boldsymbol{x}_{t+1}^{(0)}$
for $n = 1 : N$ **do**
    Evaluate values of $\mathbb{E}\left[z_{\boldsymbol{x}}^*\right]$ and $\mathbb{E}\left[\log\left(z_{\boldsymbol{x}}^*\right)\right]$ by sampling $p(y^*, y_{\boldsymbol{x}} \mid \mathcal{D}_t)$ given $\boldsymbol{x} = \boldsymbol{x}_{t+1}^{(n-1)}$
    Solve $k^{(n)}$ from Eq. (5.13)
    Solve $\beta^{(n)}$ from Eq. (5.14)
    Update $\boldsymbol{x}_{t+1}^{(n)} \leftarrow \arg\max_{\boldsymbol{x}} \mathrm{ESLBO}(\boldsymbol{x}; k^{(n)}, \beta^{(n)})$ defined in Eq. (5.12)
return $\boldsymbol{x}_{t+1}^{(N)}$

requires estimating $\mathbb{E}[z_{\boldsymbol{x}}^*]$ by sampling a large number of y^*. Consequently, the overall BO loop takes significantly more time than EI and MES, as shown in Table 5.6 for $N = 50$. However, since black-box function evaluations are often expensive, the additional computational cost of VES-Gamma is not a major bottleneck in many real-world applications.

4 Results

4.1 Experimental Setup

We employ a consistent GP hyperparameter and prior setting across all benchmarks and AFs, evaluating BO performance using the simple regret $r(t) := f^* - \max_{(\boldsymbol{x}_i, y_{\boldsymbol{x}_i}) \in \mathcal{D}_t} y_{\boldsymbol{x}_i}$, where $f^* := \max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$. When f^* is unknown, we instead report the negative best function value, $-\max_{(\boldsymbol{x}_i, y_{\boldsymbol{x}_i}) \in \mathcal{D}_t} y_{\boldsymbol{x}_i}$.

To warm-start the optimization process, we initialize with 20 random samples drawn uniformly from $\mathcal{X}$ and model the GP using a 5/2-Matérn kernel with automatic relevance determination (ARD) and a dimensionality-scaled lengthscale prior [17]. Following the theoretical assumption in the VES framework, we only focus on experiments with noise-free observations. Although all benchmarks are noiseless, we allow the GP to accommodate potential non-stationarity or discontinuities in the underlying function.

Each experiment is repeated 10 times to estimate average performance, with results reported as mean $\pm$ one standard deviation. For problems with dimension less than 50, we run 100 iterations, otherwise 1000 iterations are

Table 5.6: Average duration of a BO loop for each AF. We measure the runtime on the `Branin`, `Levy`, and `Hartmann` benchmarks and average over benchmarks, BO iterations, and 10 outer repetitions. VES has a considerably higher runtime than the other AFs.

AF	average time per BO iteration
EI	$1.627s$ ($\pm 0.916s$)
MES	$1.120s$ ($\pm 0.472s$)
VES	$53.170s$ ($\pm 15.433s$)

computed. For numerical stability in VES-Gamma, we apply clamping: $z^*_{\boldsymbol{x}} = \max\{10^{-10}, y^* - \max\{y_{\boldsymbol{x}}, y^*_t\}\}$. The expectation in Eq. (5.12) is estimated via pathwise conditioning [41] using 128 posterior samples. Additionally, the number of inner iterations N in Algorithm 5.2 is set to 50, with early stopping applied if $\|\boldsymbol{x}^{(n-1)} - \boldsymbol{x}^{(n)}\| < d \cdot 10^{-5}$, where d denotes the problem dimension. We implement VES-Gamma and our other experiments using `BoTorch` [3]. We always compare against `LogEI` [2] and use EI and `LogEI` interchangeably. Upon acceptance, we will publish our implementation under a permissive open-source license.

Benchmarks. To evaluate VES, we consider three distinct categories of benchmark problems: synthetic benchmarks, GP samples, and real-world optimization tasks.

For synthetic benchmarks, we examine commonly used functions that are diverse in dimensionality and landscape complexity. Specifically, we evaluate the 2-dimensional `Branin`, the 4-dimensional `Levy`, the 6-dimensional `Hartmann`, and the 8-dimensional `Griewank` functions. These benchmarks are widely utilized in optimization studies and provide controlled testbeds for algorithmic comparisons [33].

For GP sample benchmarks, we draw from a GP prior with a $\nu = 5/2$ Matérn kernel. These experiments examine the impact of varying length scales ($\ell = \{0.5, 1, 2\}$) and dimensionalities ($d = \{2, 50, 100\}$) on algorithmic performance.

For real-world scenarios, we utilize a set of benchmarks reflecting practical high-dimensional problems. These include the 60-dimensional `Rover` problem [39], the 124-dimensional soft-constrained `Mopta08` [18] benchmark introduced in Eriksson and Jankowiak [8], the 180-dimensional `Lasso-DNA` problem from `LassoBench` [30], and the 388-dimensional `SVM` benchmark, also introduced in Eriksson and Jankowiak [8]. These tasks represent optimization challenges in engineering design, machine learning, and computational biology.

Due to space constraints, additional experiments are provided in Appendix B.

Table 5.7: KS two-sample test passing rate between VES-Exp and EI for various benchmarks.

	Passing Rate (%)
Branin ($d = 2$)	94.00
Hartmann ($d = 6$)	99.80
Rover ($d = 60$)	92.60
Moptao8 ($d = 124$)	93.20
Prior ($d = 2$)	93.60
Prior ($d = 50$)	94.60

4.2 Comparing VES-Exp and EI

Kolmogorov-Smirnov Test. After establishing the theoretical equivalence of VES-Exp and EI in Section 3.2, we aim to validate this equivalence in our practical implementation. To this end, we employ the Kolmogorov-Smirnov (KS) two-sample test with a significance level of $\alpha = 5\%$ to assess statistical similarity. The two samples consist of function values evaluated by each AF across 10 repeated trials, i.e., $Y_{\mathrm{EI}}(t) \coloneqq \{\boldsymbol{y}_t^i\}_{i=1}^{10}$, where $\boldsymbol{y}_t^i$ denotes the function evaluation at step t in the i-th trial. The null hypothesis states that the function evaluations from VES-Exp and EI originate from the same distribution.

We collect function values for all 500 iterations and consider a test successful (pass) for each iteration t if the null hypothesis is not rejected. We include six different benchmarks spanning low-dimensional synthetic problems to high-dimensional real-world scenarios. Additional implementation details on KS test are presented in Appendix C.

Empirical Equivalence Results Figure 5.3 illustrates the function values obtained by VES-Exp and EI, while Table 5.7 reports the passing rates of the KS test across six benchmarks. The results show that all passing rates exceed 90%, with the Hartmann benchmark achieving the highest proportion of accepted tests.

Several factors explain the remaining discrepancies between VES-Exp and EI. First, since both acquisition functions are non-convex, their optimization may yield different next sampling points $\boldsymbol{x}_{t+1}$ due to variations in initialization. Second, VES methods employ a clamping mechanism to ensure that $z_{\boldsymbol{x}}^*$ remains numerically positive, which introduces a dependency between y^* and $\boldsymbol{x}$. In practice, this violates the assumptions used in the proof in Appendix A.2. Finally, while EI has a closed-form expression, VES-Exp relies on Monte Carlo estimation, introducing numerical inexactness and potential discrepancies.

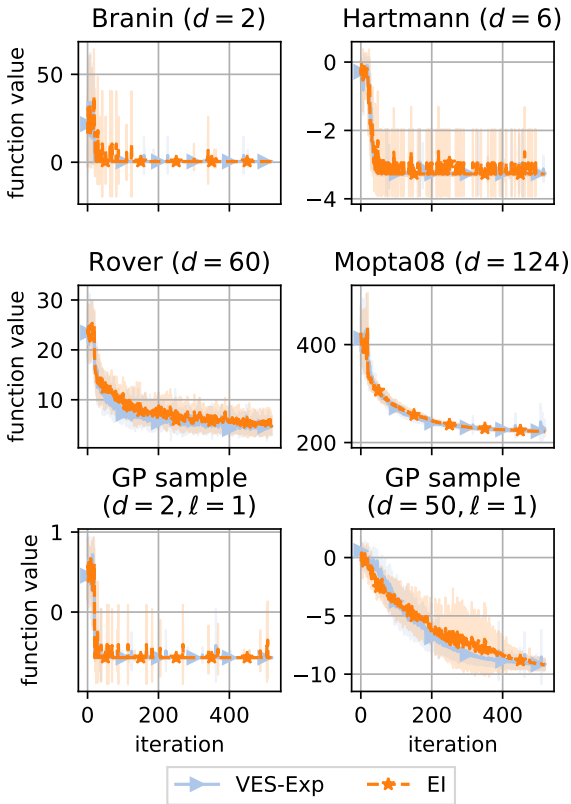


Figure 5.3: Function values observed at each BO iteration for the EI and VES-Exp AFs.

4.3 Performance of VES-Gamma

Synthetic Test Functions. Fig. 5.4 illustrates the performance of various methods, including MES, EI, and VES-Gamma, across four synthetic benchmark functions: `Branin` ($d = 2$), `Levy` ($d = 4$), `Hartmann` ($d = 6$), and `Griewank` ($d = 8$). The metric shown is the logarithm of the best value (or simple regret), averaged over 10 independent runs.

On `Branin`, VES-Gamma achieves the best performance, with MES and EI lagging behind. For `Levy`, VES-Gamma and EI are effectively tied for the best results, with MES showing slightly worse performance. On the `Hartmann` function, VES-Gamma outperforms all other methods. Finally, for the `Griewank` function, VES-Gamma and EI once again demonstrate similar performance, significantly outperforming MES.

Overall, these results highlight the robustness of VES-Gamma across diverse

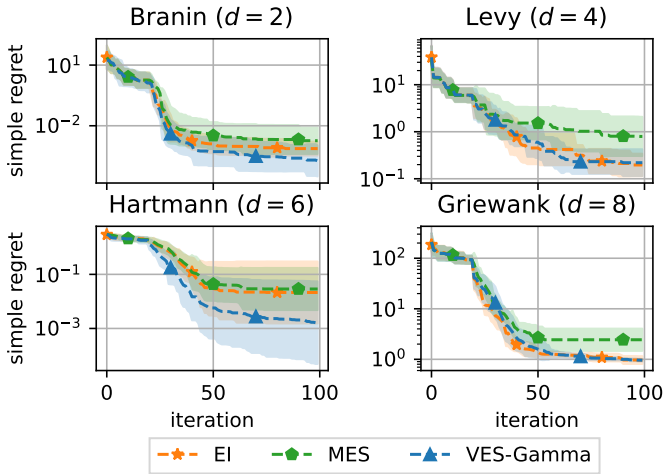


Figure 5.4: VES-Gamma, EI, and MES on the synthetic `Branin` ($d = 2$), `Levy` ($d = 4$), `Hartmann` ($d = 6$), and `Griewank` ($d = 8$) benchmark functions. Average log simple regret: VES-Gamma performs best on `Branin` and `Hartmann`, and it is competitive on `Levy` and `Griewank`.

synthetic benchmarks, consistently ranking among the top-performing methods.

GP Samples. Here, we study problem instances where the GP can be fitted without model mismatch. To this end, we sample realizations from an isotropic 100-dimensional GP prior with varying length scale $\ell = 0.05, 0.1, 0.25, 0.5$, using the same $5/2$-Matérn covariance function for the GP prior and the GP we fit to the observations.

Fig. 5.5 shows the optimization performance on the 100-dimensional GP prior samples. For $\ell = 0.05, 0.1, 0.25$, VES-Gamma outperforms EI and MES by a wide margin. EI and MES converge to a suboptimal solution. Only for $\ell = 0.5$ does EI reach the same quality as VES-Gamma, outperforming MES.

Real-World Benchmarks. Fig. 5.6 presents the performance of VES-Gamma, EI, and MES across four real-world optimization problems: the 60-dimensional `Rover` trajectory optimization, the 124-dimensional `Mopta08` vehicle optimization, the 180-dimensional weighted `Lasso-DNA` regression, and the 388-dimensional `SVM` hyperparameter tuning benchmarks.

Consistent with previous observations, VES-Gamma delivers strong performance, significantly outperforming all other acquisition functions on the `SVM` benchmark. It also ranks among the top-performing methods, alongside EI, on the `Mopta08`

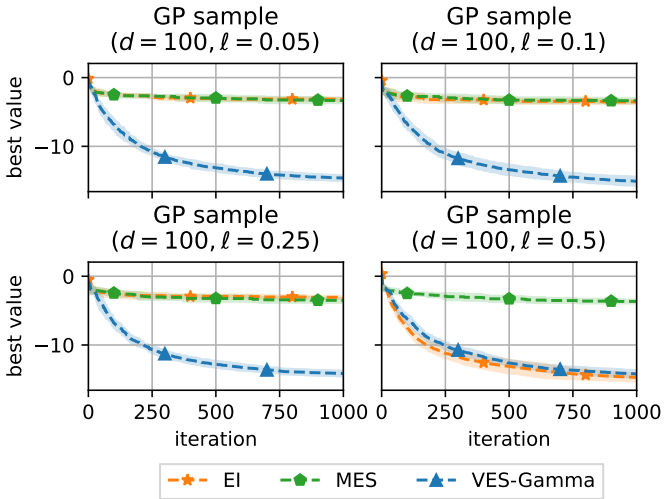


Figure 5.5: Performance curves (best values up to each iteration). VES-Gamma shows superior performance on all but one problem where it performs as good as acsEI.

and `Lasso-DNA` benchmarks. On the `Rover` problem, VES-Gamma performs comparably to EI, while MES achieves the best results in this scenario. MES exhibits mixed performance across the benchmarks, achieving the best results on `Rover` but falling behind on the `Mopta08` and `SVM` problems.

Overall, VES-Gamma demonstrates robust and consistent performance across all benchmarks, establishing itself as a versatile and reliable AF for high-dimensional real-world optimization problems.

5 Conclusion

In this work, we introduce VES, a unified framework that bridges EI and information-theoretic AFs through a variational inference approach. We demonstrate that EI can be interpreted as a special case of MES, revealing a deeper theoretical connection between these two widely used methodologies in BO. Building on this insight, we propose VES-Gamma, a novel acquisition function that dynamically balances the strengths of EI and MES. Comprehensive benchmark evaluations across a diverse set of low- and high-dimensional optimization problems highlight the robust and consistently high performance of VES-Gamma. These results underscore the potential of the VES framework as a promising foundation for developing more adaptive and efficient AF in BO.

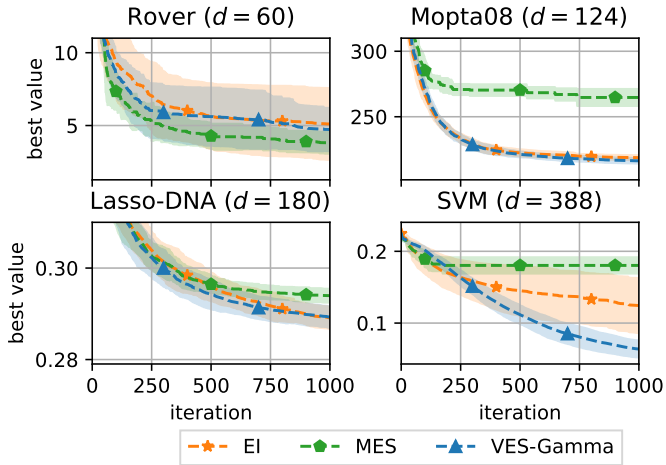


Figure 5.6: Performance curves (best function value up to each iteration). VES-Gamma outperforms all other AFs on `SVM` and performs well on the other problems.

Limitations and future work. While the Gamma distribution offers flexibility, future work will explore alternative variational distributions to enhance the adaptability of VES-Gamma. Another key direction is improving computational efficiency. Additionally, extending the theoretical framework to noisy settings remains an open challenge, requiring adaptations in variational inference to account for stochastic density supports.

Impact. This paper presents work that aims to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgements

This project was partly supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725

A Proofs

A.1 ESLB Proof

The MES AF in Eq. (5.4) can be lower bounded as follows,

Proof.

$$\begin{aligned}
\alpha_{\text{MES}}(\boldsymbol{x}) &= \mathbb{H}[y^* \mid \mathcal{D}_t] - \mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}\mathbb{H}[y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}] \\
&= \mathbb{H}[y^* \mid \mathcal{D}_t] + \mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}[\log(p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}))] \\
&= \mathbb{H}[y^* \mid \mathcal{D}_t] + \mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}\left[\log\left(\frac{p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})}{q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})}\right)\right] \\
&= \mathbb{H}[y^* \mid \mathcal{D}_t] + \mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}[\log(q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}))] \\
&+ \mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}[D_{\text{KL}}\big(p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}) \| q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})\big)] \\
&\geq \mathbb{H}[y^* \mid \mathcal{D}_t] + \mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}[\log(q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}))],
\end{aligned}$$

where the Kullback-Leibler divergence $D_{\text{KL}}(p(x)\|q(x)) \coloneqq \mathbb{E}_{p(x)}\left[\log(p(x)/q(x))\right]$. The inequality is tight if and only if $\mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}[D_{\text{KL}}\big(p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})\|q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})\big)] = 0$, which implies $p(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}}) = q(y^* \mid \mathcal{D}_t, y_{\boldsymbol{x}})$ for all $y_{\boldsymbol{x}} \mid \mathcal{D}_t$. □

A.2 VES-Exp and EI Algorithmic Equivalence

Theorem 7 is proved as follow:

Proof. By restricting the variational distributions to exponential distributions, we slightly abuse the input notations of ESLBO in (5.8) and define:

$$\begin{aligned}
\text{ESLBO}(\lambda, \boldsymbol{x}) &= \mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}\left[\log\left(\lambda \exp\left(-\lambda(y^* - \max\{y_{\boldsymbol{x}}, y_t^*\})\right)\right)\right] \\
&= \log\lambda - \lambda\mathbb{E}_{p(y^*, y_{\boldsymbol{x}}|\mathcal{D}_t)}\left[(y^* - \max\{y_{\boldsymbol{x}}, y_t^*\})\right] \\
&= \log\lambda - \lambda\underbrace{\mathbb{E}_{p(y^*|\mathcal{D}_t)}[y^*]}_{\text{constant}} + \lambda\underbrace{\mathbb{E}_{p(y_{\boldsymbol{x}}|\mathcal{D}_t)}\left[\max\{y_{\boldsymbol{x}}, y_t^*\}\right]}_{\text{EI AF}}.
\end{aligned} \tag{5.15}$$

Beginning with an arbitrary initial value $\boldsymbol{x}^{(0)}$, we determine the corresponding parameter

$$\lambda^{(1)} = \frac{1}{\mathbb{E}_{p(y^*, y_{\boldsymbol{x}^{(0)}}|\mathcal{D}_t)}\left[(y^* - \max\{y_{\boldsymbol{x}^{(0)}}, y_t^*\})\right]}, \tag{5.16}$$

which is derived by taking the derivative of (5.15) and letting it equal zero. With λ fixed, ESLBO$(\lambda^{(1)}, \boldsymbol{x})$ produces the same result as the EI AF in (5.1). We then compute $\lambda^{(2)}$ based on $\boldsymbol{x}^{(1)}$ following (5.16). Regardless of the specific value of $\lambda^{(2)}$, the ESLBO function consistently yields the same result, $\boldsymbol{x}^{(1)}$. This consistency ensures that the VES iteration process converges in a single step. The final outcome, represented as $(\boldsymbol{x}^{(1)}, \lambda^{(2)})$, indicates that the corresponding $q(y^* \mid y_{\boldsymbol{x}}, \mathcal{D}_t)$ is the closest approximation to $p(y^* \mid y_{\boldsymbol{x}}, \mathcal{D}_t)$ within $\mathcal{Q}_{\text{exp}}$ (in the sense that minimizes their Kullback-Leibler divergence). □

B Additional Experimental Results

In this section, we evaluate VES-Gamma (Algorithm 5.2) on additional benchmarks.

B.1 Synthetic Test Functions

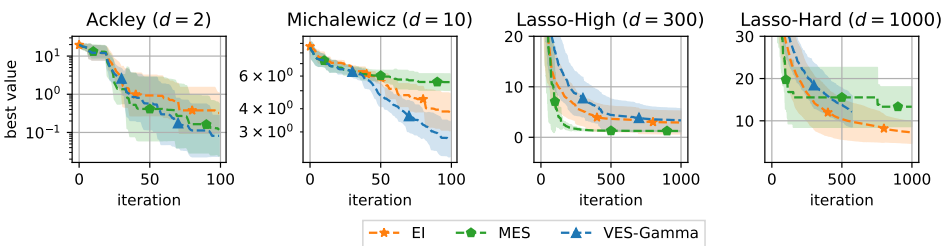


Figure 5.7: Performance plots for EI, MES, and VES-Gamma on additional synthetic benchmark functions. VES-Gamma shows robust performance throughout the bank.

Fig. 5.7 shows the performance of the different AFs, EI, MES, VES-Gamma-Sequential, and VES-Gamma, on additional synthetic benchmark functions: the `Ackley` and `Michalewicz` test functions[14] and the `Lasso-High` and `Lasso-Hard` benchmarks [30]. On the 1000-dimensional `Lasso-Hard` problem, VES-Gamma ran into a timeout after 48 hours. Therefore, we plot the mean up to the minimum number of iterations performed across all repetitions. VES-Gamma demonstrates robust performance across the benchmarks, outperforming all other AFs on `Ackley`, MES, and VES-Gamma-Sequential on `Michalewicz`, and performing similarly to the other AFs on the `Lasso` benchmarks. VES-Gamma and MES perform considerably worse than VES-Gamma, especially on the more high-dimensional problems.

[14] `https://www.sfu.ca/~ssurjano/optimization.html`

C Kolmogorov-Smirnov Test Statistic

The KS two-sample test is a non-parametric statistical method used to determine whether two samples are drawn from the same continuous distribution. It compares their empirical cumulative distribution functions (ECDFs) and calculates a test statistic that quantifies their maximum difference. Given two independent samples as function evaluations from VES-Exp $\{X_1, X_2, \ldots, X_{n_1}\}$ and from EI $\{Y_1, Y_2, \ldots, Y_{n_2}\}$, their ECDFs are defined as:

$$F_X(x) = \frac{1}{n_1} \sum_{i=1}^{n_1} \mathbb{I}(X_i \leq x), \quad F_Y(x) = \frac{1}{n_2} \sum_{j=1}^{n_2} \mathbb{I}(Y_j \leq x),$$

where $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if the condition is true and 0 otherwise. The KS test statistic is given by:

$$D = \sup_x |F_X(x) - F_Y(x)|,$$

where $\sup_x$ denotes the supremum over all possible values of x. This statistic measures the maximum absolute difference between the ECDFs of the two samples.

Statistical Hypotheses. The hypotheses for the KS test are defined as:

- Null hypothesis (H_0): $F_X(x) = F_Y(x)$ for all x (the two samples come from the same distribution).
- Alternative hypothesis (H_a): $F_X(x) \neq F_Y(x)$ for at least one x (the two samples come from different distributions).

To test these hypotheses, the test p-value is solved using the KS survival function:

$$p_{\text{test}} = Q_{\text{KS}} \left(\sqrt{\frac{n_1 n_2}{n_1 + n_2}} D \right),$$

where $Q_{\text{KS}}(\cdot)$ represents the survival function of the Kolmogorov distribution:

$$Q_{\text{KS}}(z) = 2 \sum_{k=1}^{\infty} (-1)^{k-1} e^{-2k^2 z^2}.$$

Alternatively, the significance level $\alpha = 0.05$ can be tested using the critical value:

$$D_{0.05} \approx \sqrt{-\frac{1}{2}\ln\left(0.025\right)} \cdot \sqrt{\frac{n_1+n_2}{n_1 n_2}}.$$

If $D > D_{0.05}$, we reject the null hypothesis and consider it as failure (not pass).

D VES-Gamma Computational Acceleration

Table 5.6 highlights the higher computational cost of VES methods compared to EI and MES. However, we observe that a technique known as variable projection (VarPro) [10, 26] can be leveraged to accelerate the computation of VES under certain conditions, which VES-Gamma satisfies.

The key idea behind VarPro is that when the function ESLBO has a specific structure,

$$\max_{\boldsymbol{x};k,\beta} \text{ESLBO}(\boldsymbol{x};k,\beta) = \max_{\boldsymbol{x}} \left(\underbrace{\max_{k,\beta} \text{ESLBO}(\boldsymbol{x};k,\beta)}_{\varphi(\boldsymbol{x})} \right), \tag{5.17}$$

and the solution to $\max_{k,\beta} \text{ESLBO}(\boldsymbol{x};k,\beta)$ is unique, then $\varphi(\boldsymbol{x})$ is differentiable, with

$$\frac{d}{d\boldsymbol{x}}\varphi(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\text{ESLBO}(\boldsymbol{x}, k^*_{\boldsymbol{x}}, \beta^*_{\boldsymbol{x}}), \tag{5.18}$$

where k^* and β^* are the unique values that maximize ESLBO.

Following the proof in Eq. (5.13), we establish that the solutions $k^*_{\boldsymbol{x}}$ and $\beta^*_{\boldsymbol{x}}$ are unique. This confirms that it is feasible to implement the VarPro strategy to accelerate the computation of VES-Gamma, eliminating the need for the iterative scheme in Algorithm 5.1. This ongoing work aims to reduce the computational cost of VES-Gamma to a level comparable to EI and MES.

References

[1] Milton Abramowitz, Irene A Stegun, and Robert H Romer. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 1988.

[2] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected Improvements to Expected Improvement for Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36: 20577–20612, 2023.

[3] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[4] David Barber and Felix Agakov. The IM Algorithm: A variational approach to Information Maximization. *Advances in Neural Information Processing Systems*, 16(320):201, 2004.

[5] Carolin Benjamins, Elena Raponi, Anja Jankovic, Carola Doerr, and Marius Lindauer. Self-Adjusting Weighted Expected Improvement for Bayesian Optimization. *International Conference on Automated Machine Learning*, pages 6–1. PMLR, 2023.

[6] Julian Berk, Vu Nguyen, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Exploration Enhanced Expected Improvement for Bayesian Optimization. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II 18*, pages 621–637. Springer, 2019.

[7] George De Ath, Richard M Everson, Alma AM Rahat, and Jonathan E Fieldsend. Greed is Good: Exploration and Exploitation Trade-offs in Bayesian Optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1 (1):1–22, 2021.

[8] David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

[9] Peter I Frazier, Warren B Powell, and Savas Dayanik. A Knowledge-Gradient Policy for Sequential Information Collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

[10] Gene H Golub and Victor Pereyra. The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate. *SIAM Journal on Numerical Analysis*, 10(2):413–432, 1973.

[11] Philipp Hennig and Christian J Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[12] Philipp Hennig, Michael A Osborne, and Hans P Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

[13] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. *Advances in Neural Information Processing Systems*, 27, 2014.

[14] Matthew Hoffman, Eric Brochu, and Nando de Freitas. Portfolio Allocation for Bayesian Optimization. *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 327–336, 2011.

[15] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 2013.

[16] Carl Hvarfner, Frank Hutter, and Luigi Nardi. Joint Entropy Search for Maximally-Informed Bayesian Optimization. *Advances in Neural Information Processing Systems*, 35:11494–11506, 2022.

[17] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian Optimization Performs Great in High Dimensions. Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20793–20817. PMLR, 21–27 Jul 2024.

[18] Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. *MOPTA 2008 Conference (20 August 2008)*, 2008.

[19] Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly. *Journal of Machine Learning Research*, 21(81):1–27, 2020.

[20] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in Neural Information Processing Systems*, 34:21696–21707, 2021.

[21] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[22] Haitong Ma, Tianpeng Zhang, Yixuan Wu, Flavio P Calmon, and Na Li. Gaussian Max-Value Entropy Search for Multi-Agent Bayesian Optimization. *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10028–10035. IEEE, 2023.

[23] Jonas Mockus. The application of Bayesian methods for seeking the extremum. *Towards global optimization*, 2:117, 1998.

[24] John Paisley, David Blei, and Michael Jordan. Variational Bayesian Inference with Stochastic Search. *Proceedings of the International Conference on Machine Learning*, 2012.

[25] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On Variational Bounds of Mutual Information. *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[26] Clarice Poon and Gabriel Peyré. Smooth over-parameterized solvers for non-smooth structured optimization. *Mathematical Programming*, 201(1):897–952, 2023.

[27] Chao Qin, Diego Klabjan, and Daniel Russo. Improving the Expected Improvement Algorithm. *Advances in Neural Information Processing Systems*, 30, 2017.

[28] Jixiang Qing, Henry B Moss, Tom Dhaene, and Ivo Couckuyt. $\{\mathrm{PF}\}^2$ES: Parallel Feasible Pareto Frontier Entropy Search for Multi-Objective Bayesian Optimization. *26th International Conference on Artificial Intelligence and Statistcs (AISTATS) 2023*, volume 206, pages 2565–2588, 2023.

[29] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006.

[30] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. *International Conference on Automated Machine Learning*, pages 2–1. PMLR, 2022.

[31] András Sóbester, Stephen J Leary, and Andy J Keane. On the Design of Optimization Strategies Based on Global Response Surface Approximation Models. *Journal of Global Optimization*, 33:31–59, 2005.

[32] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *Proceedings of the International Conference on Machine Learning*, 2010.

[33] Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets, optimization test problems. `https://www.sfu.ca/~ssurjano/optimization.html`. Accessed: 2024-09-01.

[34] Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian Optimization with Max-value Entropy Search and its Parallelization. *International Conference on Machine Learning*, pages 9334–9345. PMLR, 2020.

[35] Sho Takeno, Takuya Tamura, Kaito Shitara, and Masayuki Karasuyama. Sequential and Parallel Constrained Max-value Entropy Search via Information Lower Bound. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 20960–20986. PMLR, June 2022.

[36] Ben Tu, Axel Gandy, Nikolas Kantas, and Behrang Shafei. Joint Entropy Search for Multi-objective Bayesian Optimization. *Advances in Neural Information Processing Systems*, 35:9922–9938, 2022.

[37] Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44:509–534, 2009.

[38] Zi Wang and Stefanie Jegelka. Max-value Entropy Search for Efficient Bayesian Optimization. *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

[39] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[40] James T Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions. *NeurIPS Workshop on Bayesian Optimization*, 2017.

[41] James T Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Pathwise Conditioning of Gaussian Processes. *Journal of Machine Learning Research*, 22(105):1–47, 2021.

Paper VI

Understanding High-Dimensional Bayesian Optimization

Accepted at the Forty-Second International Conference on Machine Learning

Leonard Papenmeier
Lund University

Matthias Poloczek
Amazon

Luigi Nardi
Lund University

Abstract

Recent work reported that simple Bayesian optimization methods perform well for high-dimensional real-world tasks, seemingly contradicting prior work and tribal knowledge. This paper investigates the 'why'. We identify fundamental challenges that arise in high-dimensional Bayesian optimization and explain why recent methods succeed. Our analysis shows that vanishing gradients caused by Gaussian process initialization schemes play a major role in the failures of high-dimensional Bayesian optimization and that methods that promote local search behaviors are better suited for the task. We find that maximum-likelihood estimation of Gaussian process length scales suffices for state-of-the-art performance. Based on this, we propose a simple variant of maximum-likelihood estimation called MSR that leverages these findings to achieve state-of-the-art performance on a comprehensive

set of real-world applications. We also present targeted experiments to illustrate and confirm our findings.

1 Introduction

Bayesian optimization (BO) has found wide-spread adoption for optimizing expensive-to-evaluate black-box functions that appear in aerospace engineering [27, 24], drug discovery [34], robotics [26, 6, 40, 29] or finance [3].

While BO has proven reliable in low-dimensional settings, high-dimensional spaces are challenging due to the curse of dimensionality (COD) that demands exponentially more data points to maintain the same precision with increasing problem dimensionality. Several approaches have extended BO to high-dimensional spaces under additional assumptions on the objective function that lower the data demand, such as additivity [9, 21, 14, 54] or the existence of a low-dimensional active subspace [51, 33, 25, 36]. Without such assumptions, it was widely believed that BO with a Gaussian process (GP) surrogate is limited to approximately 20 dimensions for common evaluation budgets [12, 31]. Recently, Hvarfner et al. [17] and Xu and Zhe [53] reported that simple BO methods perform well on high-dimensional real-world benchmarks, often surpassing the performance of more sophisticated algorithms.

Due to its many impactful applications, high-dimensional Bayesian optimization (HDBO) has seen active research in recent years [4, 37]. While the boundaries have been pushed significantly, the causes of performance gains have not always been thoroughly identified. For example, in the case of the `BODi` [8] and `COMBO` [35] algorithms, later work found that the methods benefited from specific benchmark structures prevalent in their evaluation [37]. Similarly, an evaluation of Nayebi et al. [33] showed that the performance of some prior methods is sensitive to the location of the optimum in the search space.

Thus, answering the recent call for more scrutiny and exploratory research [16], we first identify fundamental challenges arising in high dimensions and then examine state-of-the-art HDBO methods to understand how they mitigate these obstacles. Equipped with these insights, we propose a simpler approach that uses maximum-likelihood estimation (MLE) of the GP length scales called MLE Scaled with random axis-aligned subspace perturbations (RAASP) (MSR). We demonstrate that MSR is sufficient for state-of-the-art HDBO performance without the need for specifying a prior belief on length scales as in maximum a-posteriori estimation (MAP). We note that practitioners usually do not possess such priors and instead rely on empirical performances on benchmarks.In particular, we change the initialization of length

scales to avoid vanishing gradients of the GP likelihood function that easily occur in high-dimensional spaces but, so far, have been overlooked for BO. Furthermore, we provide empirical evidence suggesting that good BO performance on extremely high-dimensional problems (on the order of 1000 dimensions) is due to local search behavior and not to a well-fit surrogate model. In summary, we make the following contributions.

1. We identify fundamental challenges that arise in high-dimensional Bayesian optimization and explain why recent methods succeed. We show that vanishing gradients and local search behaviors are important in HDBO.

2. We find that MLE of GP length scales suffices for state-of-the-art performance. We propose a simple variant of MLE called `MSR`.

3. We evaluate `MSR` on a comprehensive set of real-world applications and a series of targeted experiments that illustrate and confirm our findings.

2 Problem Statement and Related Work

We aim to find $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$, where $f : \mathcal{X} \to \mathbb{R}$ is an unknown, only point-wise observable, and expensive-to-evaluate black-box function and $\mathcal{X} = [0, 1]^d$ is the d-dimensional search space, sometimes called "input space". BO is a popular approach to optimize problems with the above characteristics. We give a summary of BO and GPs in Appendix A and restrict the discussion to high-dimensional BO.

Extending the scope of BO to high-dimensional problems was, for a long time, considered "as one of the holy grails" [51] or "one of the most important goals" [33] of the field. Several contributions extended the scope of BO to specific high-dimensional problems. However, for the longest time, no fully scalable method has been found to extend in arbitrarily high-dimensional spaces without making additional assumptions about the problem structure. The root problem of extending Bayesian optimization (BO) to high dimensions is the curse of dimensionality (COD) [4] that not only requires exponentially many more data points to model f with the same precision but also complicates the fitting of the GP hyperparameters and the maximization of the acquisition function (AF). The growing demand for training samples stems from increasing point distances in high dimensions, where the average distance in a d-dimensional hypercube is $\sqrt{d}$ [23].

This paper focuses on HDBO operating directly in the search space. Other methods for HDBO include linear [33, 51, 25, 36, 37] or non-linear [48, 31, 28, 5, 7] embeddings

to map from a low-dimensional subspace to the input space.

High-dimensional BO in the input space. In the literature, HDBO operating directly in the high-dimensional search space $\mathcal{X}$ is often considered infeasible due to the COD. Numerous approaches have been proposed, often leveraging assumptions made on the objective function f such as additivity [21, 13, 50, 32, 54] or axis-alignment [10, 15, 46], which simplify the problem and improve sample efficiency if they are met. Other methods identify regions in the search space relevant for the optimization, for example, using trust regions (TRs) [43, 39, 11] or by partitioning the space [49].

Recently, multiple works re-evaluated basic BO setups for high-dimensional problems, presenting state-of-the-art performance on various high-dimensional benchmarks with only small changes to basic BO strategies. Hvarfner et al. [17] use a dimensionality-scaled log-normal length scale hyperprior that shifts the mode and mean of the log-normal distribution by a factor of $\sqrt{d}$, designed to counteract the increased distance between randomly sampled points. To optimize the AF, they change `BoTorch`'s [2] default strategy of performing Boltzmann sampling on a set of quasi-randomly generated points by sampling over both a set of quasi-randomly generated points and a set of points that are generated by perturbing the 5% best-performing points. By perturbing 20 dimensions on average, this strategy creates candidates closer to the incumbent observations and enforces a more exploitative behavior [44, 43, 11]. The effect of the sampling strategy was recently revisited by Rashidi et al. [41]. They argue that `TuRBO`'s RAASPs are crucial to performance on high-dimensional benchmarks and, motivated by this observation, derive the cylindrical Thompson sampling (TS) strategy that maintains locality but drops the requirement of axis alignment. Independently of Hvarfner et al. [17], Xu and Zhe [53] reported that "standard GPs can be excellent for HDBO" which they show empirically on several high-dimensional benchmarks. They use a uniform $\mathcal{U}(10^{-3}, 30)$ length scale hyperprior, which, in their experiments, performs superior to `BoTorch`'s Gamma $\Gamma(3, 6)$ length scale hyperprior.

3 Facets of the Curse of Dimensionality

This section discusses how the curse of dimensionality impacts high-dimensional Bayesian optimization and techniques to mitigate these challenges.

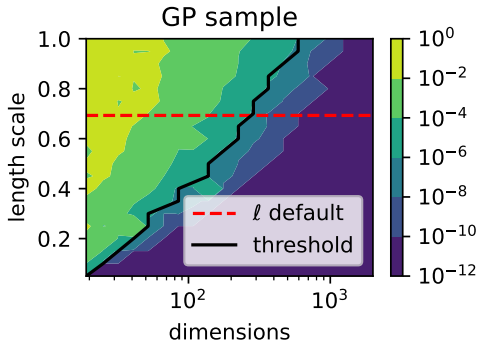


Figure 6.1: Maximum MLE gradient magnitude for the 50 first gradient steps initialized with different initial length scales (y-axis) and problem dimensionalities (x-axis). With short initial length scales, the gradients vanish even for low dimensions.

3.1 Vanishing Gradients at Model Fitting

Bayesian optimization uses a probabilistic surrogate model of f to guide the optimization. GPs are the most popular surrogates due to their analytical tractability. They allow for different likelihoods, mean, and covariance functions, each often exposing several hyperparameters, including the function variance σ_f^2, the noise variance σ_n^2, and the d model length scales $\boldsymbol{\ell}$ that need to be fitted to the task at hand. In the absence of prior information about the objective function f, maximum-likelihood estimation (MLE) is commonly used to fit the model hyperparameters by maximizing the GP marginal log-likelihood (MLL):

$$\boldsymbol{\theta}^*_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|X, \boldsymbol{\theta}), \tag{6.1}$$

Here, X are points in the search space $\mathcal{X}$, $\boldsymbol{y}$ are the associated function values, and $\boldsymbol{\theta}$ is the vector of GP hyperparameters. See Appendix A for more information.

The MLL is usually maximized using a multi-start gradient descent (GD) approach. A crucial component of fitting a GP is choosing starting points for the MLE hyperparameters. In this section, we show that an ill-suited length scale initialization scheme can cause the gradient of the MLL function with respect to the GP length scales to vanish for high-dimensional problems. Thus, the length scales remain at the numerical values that they have been initialized to and will not be fitted to the objective function.

Fig. 6.1 shows the severity of the vanishing-gradients phenomenon. We plot the maximum magnitude (element-wise) of the length scale gradient across 50 gradient updates of an isotropic GP with a $5/2$-Matérn kernel as a function of the input dimensionality of the objective function (x-axis) and the initial value for the

length scale hyperparameter with which the gradient-based optimizer starts when maximizing the MLE (y-axis). The objective function is sampled from a GP with a $5/2$-Matérn kernel and $\ell = 0.5$, i.e., a GP prior sample. The dashed line shows the default initial length scale of $\ln 2$ used in `GPyTorch`. We consider gradients smaller than the machine precision for single floating point numbers 'vanished'. The reason is that even after 500 gradient updates, the length scale would change at most by $\approx 6 \times 10^{-5}$ from the value with which the gradient-based optimizer was initialized.

Methods to Mitigate Vanishing Gradients. One strategy to counteract the vanishing gradients is to replace MLE with maximum a-posteriori estimation (MAP) by choosing a hyperprior on the length scales that prefers long length scales:

$$\boldsymbol{\theta}^*_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \underbrace{\log p(\boldsymbol{y}|X, \boldsymbol{\theta})}_{\text{evidence}} + \underbrace{\log p(\boldsymbol{\theta})}_{\text{prior}}. \tag{6.2}$$

MAP maximizes the unnormalized log posterior, which is the sum of the MLL and the log prior. We sometimes use the terms 'MLL' and 'unnormalized log-posterior' interchangeably if what is meant is clear from the context. The gradient of the recently popularized dimensionality-scaled log-normal hyperprior of [17] directs the optimizer toward the mode of the hyperprior $\log p(\boldsymbol{\theta})$ that corresponds to long length scales.

If the gradient-based optimizer of the MLE reaches sufficiently long length scales, the MLL $\log p(\boldsymbol{y}|X, \boldsymbol{\theta})$ no longer vanishes. Fig. 6.2 shows the length scales of a GP conditioned on random observations of a realization drawn from an isotropic 1000-dimensional GP prior with length scale $\ell = 0.5$ and a $5/2$-Matérn kernel when fitting with MLE and MAP. We initialize the length scales with $\ln 2 \approx 0.69$, draw 1 and 10 observations uniformly at random, and maximize the MLL for 500 iterations with MLE and MAP, using a log-normal hyperprior. We average over 10 random restarts. When conditioning on only 1 observation, the MAP optimization starts at the initialization $\ln 2$ and converges to the hyperprior mode. For 10 observations, the length scales first move toward the mode but then converge to a different point, which trades off the attraction of the prior mode and the ground truth length scale. With MLE, the length scale does not change because of the vanishing gradients issue.

To "ensure meaningful correlation" in increasingly high-dimensional space, Hvarfner et al. [17] scale a log-normal hyperprior for the GP length scales with the problem's dimensionality. Xu and Zhe [53] pursue a different approach of initializing the gradient-based optimizer of the length scales with large values. Specifically, they posit a uniform $\mathcal{U}(10^{-3}, 30)$ hyperprior to the length scales and initialize the gradient-based optimization of the automatic relevance determination (ARD) kernel with d samples from the hyperprior. Although their method does not scale the hyperprior

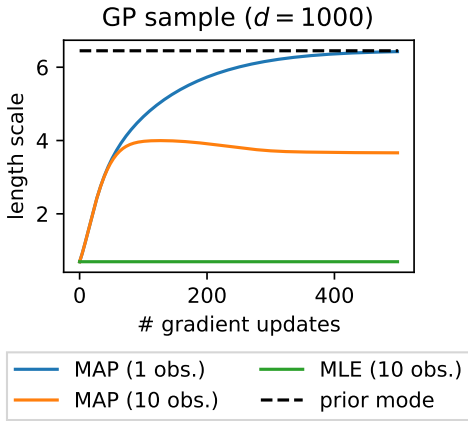


Figure 6.2: Gradient-based optimization of a GP length scale with MAP and MLE. When conditioning the GP on only one random observation, MAP converges to the prior mode. We plot $\pm$ one standard error, which is too small to be visible.

mode with the problem's dimensionality, the expected length scale of ≈ 15 is sufficiently long to avoid vanishing gradients for the problems studied by the authors.

While these methods mitigate the issue of vanishing gradients, increasing length scales by any constant factor does not always solve it, as Fig. 6.1 indicates. Whether scaling the hyperprior of the length scale with the problem's dimensionality achieves a good fit of the surrogate model depends on the properties of f. The success of the two methods described above is related to the effect of the increased length scales on mitigating the problem of vanishing gradients. If the underlying function varies quickly, the GP needs to use a short length scale to model f. In such cases, the GP cannot model the function globally, as shown in Appendix B.4.

3.2 Vanishing Gradients of the Acquisition Function

Several popular acquisition functions for BO, such as upper-confidence bound (UCB) [47], expected improvement (EI) [20], and probability of improvement (PI) [18], rely solely on the GP posterior, exhibiting only small variation when the posterior itself changes only moderately. In high-dimensional spaces, the expected distance between two points sampled uniformly at random increases with $\sqrt{d}$ [23]. Thus, there are typically large 'unexplored regions' in the search space where the algorithm has not sampled. Suppose a commonly used GP with constant prior mean function and a stationary kernel. The GP posterior corresponds to the prior in those regions unless the kernel has sufficiently long length scales. Therefore, the GP posterior and the acquisition surface are flat in those vast unexplored regions. AFs

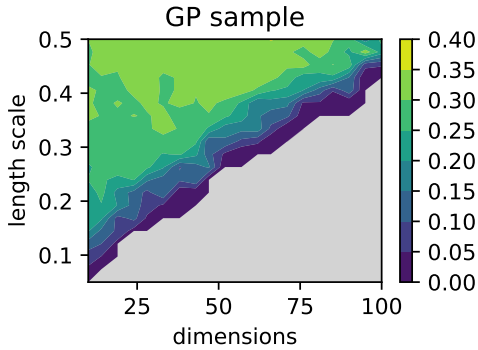


Figure 6.3: Average distances between the initial and the final candidates of `LogEI` for various model length scales and dimensionalities without RAASP sampling. Values in the gray region are numerically zero. In high dimensions, the gradient of the AF vanishes, causing no movement of the gradient-based optimizer.

are usually optimized with gradient-based approaches. Thus, these 'flat' areas of the AF also lead to vanishing gradients, but this time of the AF. This affects the selection of the next sample in the BO procedure.

The `LogEI` [1] AF provides a numerically more stable implementation of EI, mitigating the problem of vanishing gradients but not solving it, as we demonstrate next. Fig. 6.3 shows the average distances the gradient-based optimizer travels for `LogEI`. The surrogate is a GP with 20 observations drawn uniformly at random. The unknown objective function is a realization of the same GP. We run `BoTorch`'s multi-start GD acquisition function optimizer, initialized with 512 random samples and 5 random restarts, and measure the distances between the starting and end points of the gradient-based optimization of the AF. Average distances increase with $\sqrt{d}$; thus the plot shows average distances between the starting and endpoints of the gradient-based optimization that are normalized by $d^{-\frac{1}{2}}$. Gray regions correspond to a numerically zero average distance, indicating vanishing gradients. Vanishing gradients of the AF remain a problem, even when using `LogEI` and operating in moderate dimensions.

Methods to Mitigate Vanishing Gradients of the AF. One technique for handling vanishing gradients in the AF optimization is *locality*. `TuRBO` [11], for example, uses TRs to constrain the optimization to a subregion of the search space. Even if the GP surrogate is uninformative, `TuRBO` performs local search and can optimize high-dimensional problems even if data is scarce. `TuRBO` also uses RAASP sampling [41, 44], i.e., with a probability of $\min\left(1, \frac{20}{d}\right)$, it replaces the value of each dimension of the incumbent solution with a value drawn uniformly at random within the TR bounds. This process is repeated multiple times to create several candidates evaluated

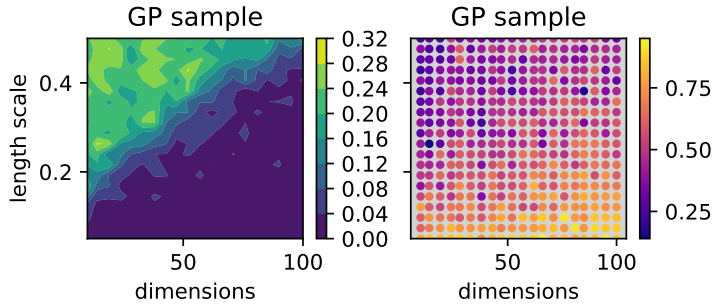


Figure 6.4: Left: Average distances between the initial and the final candidates of `LogEI` with RAASP sampling. The vanishing gradient issue decreases. Right: Fraction of multi-start GD candidates originating from the RAASP samples when evaluating `LogEI` on random samples. In high dimensions, RAASP samples are increasingly more likely to get picked, even for longer length scales.

on a realization of the GP posterior, a process known as Thompson sampling. The point of maximum value is then chosen for evaluation in the next iteration of the BO loop. This design choice further enforces locality as new candidates only differ on average from the incumbent in 20 dimensions for $d \geq 20$.

Differing slightly from `TuRBO`'s approach, RAASP sampling has been implemented in `BoTorch`'s AF maximization and can optionally be enabled with the parameter `sample_around_best`. `BoTorch` augments a set of globally sampled candidates with the RAASP samples, resulting in twice as many initial candidates. It perturbs the best-performing points by replacing the value of each dimension with a probability of $\min\left(1, \frac{20}{d}\right)$ by a value drawn from a Gaussian distribution, truncated to stay within the bounds of the search space. `BoTorch` then chooses the points of maximum initial acquisition value to start the GD optimization of the acquisition function.

With increasing dimensionality or descending length scale, the starting points for the multi-start GD routine chosen by the AF maximizer are increasingly more likely to originate from the RAASP samples. Fig. 6.4 (right panel) illustrates this. Here, we draw realizations from GPs, initialized with different dimensionalities (x-axis) and length scales (y-axis). For each realization, we maximize the acquisition function with RAASP sampling and plot the percentage of candidates of maximum acquisition value originating from the RAASP samples across all candidates. A higher percentage indicates a more 'local' sampling. We further average across five random restarts. The percentage of RAASP candidates with maximum acquisition value increases with the input dimensionality and decreases with the length scale. At the same time, those candidates stay close to the initial candidates. This is shown in the bottom right of Fig. 6.4 (left panel), which shows the average distance traveled by candidate points of the AF maximizer when using RAASP sampling. With RAASP sampling, candidates travel a positive distance, visualized by the lack of gray color. This indicates

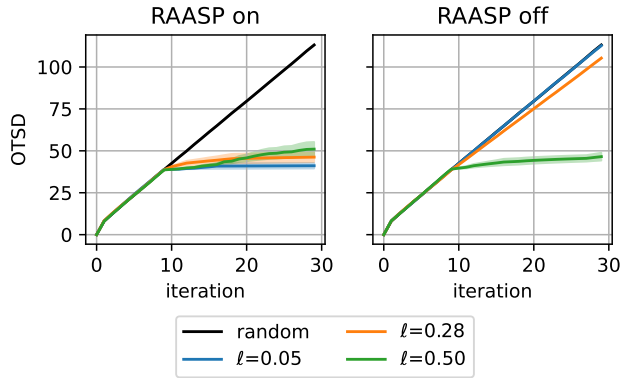


Figure 6.5: OTSD for `BoTorch`'s AF maximizer operating on a 100-dimensional space and GPs of various length scales with and without RAASP sampling. The behavior of short-length-scale GPs reverts to local search ($\ell = 0.05$ in the left panel) with RAASP sampling and to local search without RAASP sampling ($\ell = 0.05$ and $\ell = 0.28$ in the right panel). Shaded areas show the standard error of the mean obtained by 10 random repetitions. In the right panel, the blue line masks the black line.

a reduction of the vanishing-gradient issues. We attribute this to candidates being created close to the incumbent observations where the AF is less likely to be flat.

In general, when the length scales of the GP are short and the dimensionality is high, BO shows a local-search-like behavior with RAASP sampling and a random search behavior without it. We demonstrate this using the observation traveling salesman distance (OTSD) [38], which quantifies an algorithm's exploration by finding the shortest path connecting all observations made up to a certain iteration. The OTSD curve of an algorithm A consistently lying above the curve of algorithm B indicates that algorithm A is more explorative as its observations are more spread more evenly across $\mathcal{X}$. The OTSD is always monotonic increasing. Fig. 6.5 shows the OTSDs for 100-dimensional GPs with different length scales, each initialized with 10 random samples in the design of experiment (DoE) phase and subsequently optimized with `LogEI` and RAASP sampling for 20 iterations. Unless the model length scale is sufficiently long for the AF gradient not to vanish (as for $\ell = 0.5$ in the right panel of Fig. 6.5), the AF maximizer picks one of the initial random candidates without further optimizing it. This is supported by the trajectories for the BO phase (iteration ≥ 9) following the trajectory of the DoE phase (iteration < 9) for $\ell = 0.05$ and $\ell = 0.28$ in the right panel of Fig. 6.5. We generally recommend the RAASP sampling method as it improves BO by automatically reverting to local search when encountering flat AFs.

3.3 Bias-Variance Trade-off for fitting the GP

GP models are commonly fitted by maximizing the MLL, either using unbiased MLE estimation or using MAP, which places a hyperprior on one or several GP hyperparameters. MLE exhibits a higher variance and sensitivity to noise, particularly when fitting a model in high-dimensional spaces with scarce data. MAP, on the other hand, has a lower variance in the length-scale estimates but comes at the cost of bias unless accurate prior information is available. The MLL is given by

$$\begin{aligned}\log p(\boldsymbol{y}|X,\boldsymbol{\theta}) = &\underbrace{-\frac{1}{2}\boldsymbol{y}^\intercal \left(K(X,X)+\sigma_n^2 I\right)^{-1}\boldsymbol{y}}_{\text{data fit}} \\ &\underbrace{-\frac{1}{2}\log|K(X,X)+\sigma_n^2 I|}_{\text{complexity penalty}} - \frac{n}{2}\log 2\pi\end{aligned} \tag{6.3}$$

The first and second terms are often called data fit and complexity penalty [42]. For more details, see Appendix A. This section explores the bias-variance trade-off between these two popular approaches for GP model fitting.

MLE. Fig. 6.6 shows the length scale obtained when using (blue) or MAP (orange) to fit a GP surrogate model with an $^5/_2$-ARD-Matérn to a realization drawn from an isotropic GP prior with length scale $\ell = 1$ and noise term 10^{-8}. We examine a 10 and a 50-dimensional GP and repeat each experiment 50-times. As before, the gradient-based optimizer starts with an initial length scale of $\frac{\sqrt{d}}{10}$. We observe that the length scales estimated by MLE vary significantly less for the 10-dimensional function than for the 50-dimensional one. As we increase the number of observations on which the GP surrogate is conditioned, the variance of the estimated length scales decreases.

The dotted curves in the bottom row of Fig. 6.7 show the likelihood surface for MLE as specified in Eq. (6.3). The penalty term $\frac{1}{2}\log|K|$ and the data fit term $-\frac{1}{2}\boldsymbol{y}^\intercal \left(K(X,X)+\sigma_n^2 I\right)^{-1}\boldsymbol{y}$ are shown in the first two rows, respectively; the constant term $-\frac{n}{2}\log 2\pi$ is omitted from the figure. We account for the different number of samples between the left and right figures by scaling the penalty, data fit, MLE, and MAP terms with s^{-1}. We show the surface for $s = 5$ (left) and $s = 50$ (right) data points. As the length scales increase, the entries of the kernel matrix increase. The determinant $|K|$ decays more quickly, and the penalty term decreases, adding a more distinct global trend to the likelihood surface. In the limit, $\ell \to \infty$, the kernel matrix becomes a matrix of ones, and the determinant becomes zero. In low dimensions, the data fit term decreases for long length scales, but the

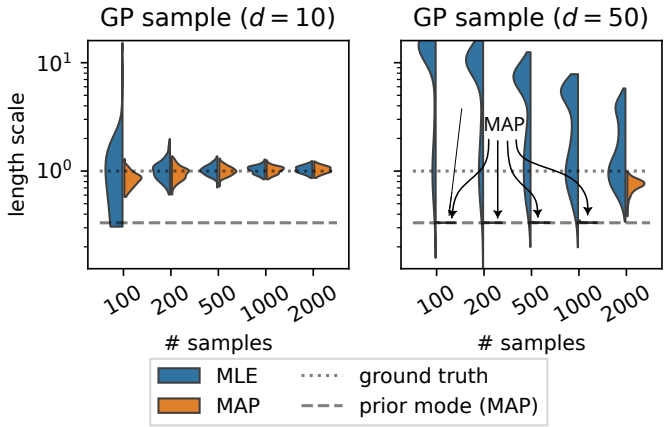


Figure 6.6: Average length scales (y-axis) obtained by MLE (blue) and MAP (orange) for different numbers of randomly sampled observations (x-axis) for a 10- and for a 50-dimensional GP prior sample. The obtained length scales differ substantially for the higher dimensional function if few points have been observed.

decreasing penalty compensates for this, resulting in a relatively flat MLL surface. The fast decay of the data fit term increases the "signal-to-noise" ratio, making it easier for the optimizer to converge in 10 than 100 dimensions. This can be seen by comparing the green and brown MLL curves in Fig. 6.7 for $s = 50$ samples. For more observations, MLL becomes smoother in all dimensions, as indicated by the left vs. right panel in Fig. 6.7.

MAP. MAP allows for incorporating prior beliefs about reasonable values for hyperparameters. However, practitioners often do not possess such prior information and hence resort to hyperpriors that reportedly perform well in benchmarks. Karvonen and Oates [22] criticized this as an 'arbitrary' determination of hyperparameters.

The orange distributions in Fig. 6.6 show the average length scales obtained by MAP with a Gamma$(3, 6)$ prior, which has been the default in `BoTorch` before version `12.0`. We use this prior as it has a substantial mass around its mode $1/3$, simplifying our analysis compared to wider priors, which reduce the difference between MLE and MAP. Compared to MLE, the MAP estimates vary less but exhibit significant bias. This is pronounced for the 50-dimensional GP sample, where the MAP estimates for the length scales revert to the prior mode for 100, 200, 500, and 1000 initial samples. The solid lines in the lower row of Fig. 6.7 show the surface for the MAP estimation, using the same GP sample as for MLE. The log prior term adds additional curvature, resulting in length scale estimates of lower variance. This is particularly noticeable for little data (left column of Fig. 6.7) and consistent with Fig. 6.6. With more

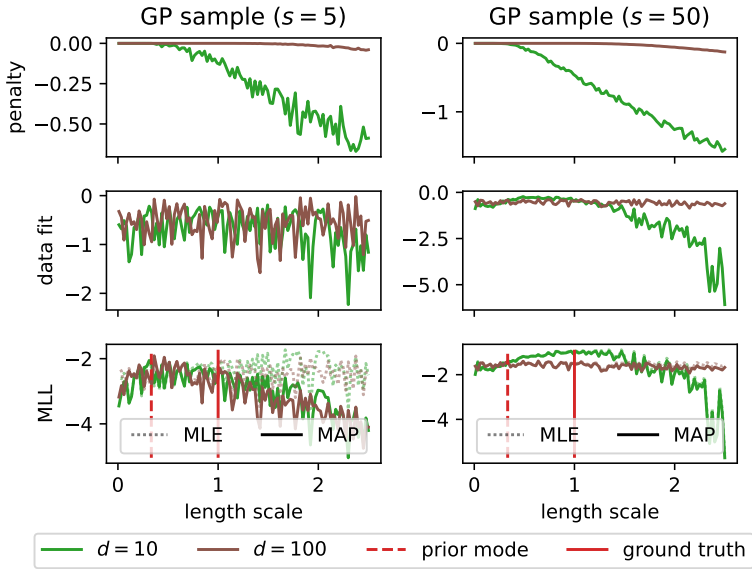


Figure 6.7: The MLL surface (bottom), the penalty (top row), and data fit terms (center) for various length scales and numbers of observations. Fewer observations lead to more erratic changes in the data fit term, leading to higher variance in the length scale estimates unless a prior gives additional shape to the surface.

data, MLE and MAP become increasingly more similar, with MAP's log posterior decreasing faster for longer length scales due to the Gamma prior.

4 Discussion

Experimental Setup and Benchmarks. We propose a simple initialization for the gradient-based optimizer used for fitting the length scales of the GP surrogate *via* MLE and evaluate its performance for BO tasks. In what follows, we suppose a BO algorithm with a $5/2$-ARD-Matérn kernel and `LogEI` [1]. To address the issue of vanishing gradients at the start of the MLE optimization, we choose the initial length scale as 0.1 and scale with $\sqrt{d}$ to account for the increasing distances of the randomly sampled DoE points. Thus, the initial length scale used in the optimization is $\frac{\sqrt{d}}{10}$, and we refer to this new Bayesian optimization method as 'MLE scaled'. The second method in the evaluation is the same BO method, but now using the default value $\ell = \ln 2$ of `GPyTorch` as initial length scale in MLE. This value is not scaled with d. Based on our analysis of the impact of RAASP sampling when optimizing the AF, we combine 'MLE scaled' with RAASP sampling and call the method 'MLE Scaled with RAASP' (`MSR`). The fifth method `DSP` is the new default in `BoTorch` [2], which uses a MAP estimate of length scales and initializes the optimization with the mode

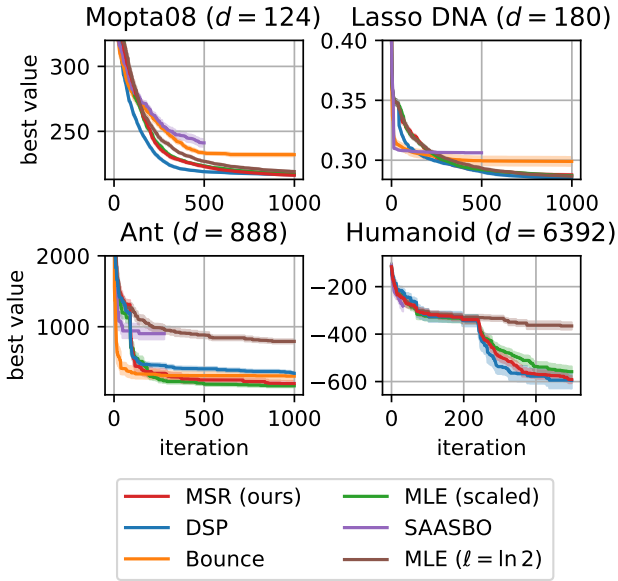


Figure 6.8: BO with the 'scaled' initialization of MLE performs comparably to the state-of-the-art in HDBO.

of the dimensionality-scaled prior (DSP) [17]. We also compare against `SAASBO` [10] and `Bounce` [37]. `SAASBO` has a large computational runtime. Hence, we run it only for 500 iterations and terminate runs that exceed 72 hours. That is why `SAASBO` has fewer evaluations for `Ant` and `Humanoid`.

Our benchmarks are the 124-dimensional soft-constrained version of the `Mopta08` benchmark [19] introduced by Eriksson and Jankowiak [10], the 180-dimensional `Lasso-DNA` benchmark [45], as well as two 888- and 6392-dimensional `Mujoco` benchmarks used by Hvarfner et al. [17]. The first two benchmarks are noise-free, while the others exhibit observational noise.

MLE Works Well for HDBO. Fig. 6.8 shows the performance of the BO methods on the four real-world applications. Each plot gives the average objective value of the best solution found so far in terms of the number of iterations. We show the ranking of the methods according to the final performance in Table 6.8 in Appendix B.1. The confidence bands indicate the standard error of the mean. MSR achieves competitive performance across all benchmarks, matching the SOTA `DSP`. Notably, MSR outperforms DSP on $124d$ `Mopta08` and $888d$ `Ant`, and performs slightly worse than `DSP` other benchmarks. `Bounce` [37] is consistently outperformed by MSR, MLE, and DSP but surpasses `SAASBO` [10], especially on `Ant`. Although the constant length scale initialization ($\ell = \ln 2$) without RAASP achieves satisfactory results on lower-

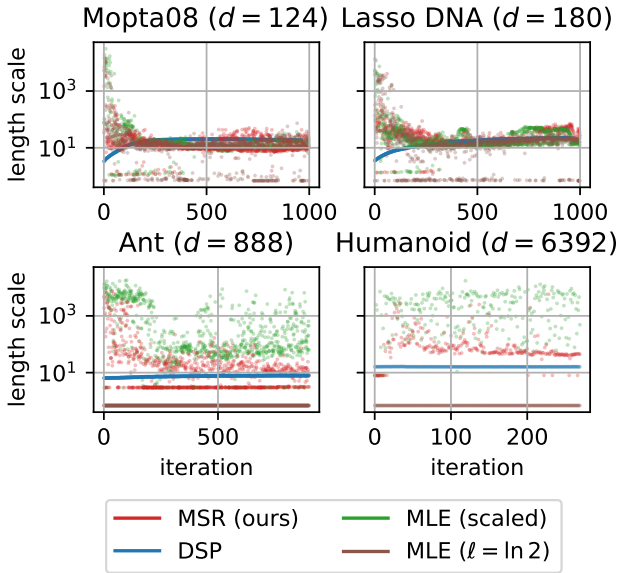


Figure 6.9: Average length scales of `MSR` and the other methods. RAASP sampling gives more consistent length scale estimates.

dimensional benchmarks such as $124d$ `Mopta08` and $180d$ `Lasso-DNA`, it fails on higher-dimensional benchmarks like $888d$ `Ant` and $6392d$ `Humanoid`. We attribute this breakdown to vanishing gradients as shown in Fig. 6.12 in Appendix B.2.

Fig. 6.9 compares the mean length scales per BO iteration, averaged over dimensions and 15 repetitions. For the 124-dimensional `Mopta08` and 180-dimensional `Lasso-DNA` applications, 'MLE ($\ell = \ln 2$)' learns length scales similar to the other methods. However, this constant initialization strategy fails to make progress from the initial value for the more high-dimensional problems in the bottom row of Fig. 6.9. We attribute this to vanishing gradients of the MLL as discussed in Sec. 3 and highlighted in Fig. 6.12 in Appendix B.2.

RAASP Reduces Variance. Fig. 6.9 shows a surprising behavior for `DSP`. As one would anticipate, the estimated length scales are typically close to the mode of the hyperprior at the start of the optimization. However, they then converge to an even higher value on all benchmarks but the 6392-dimensional `Humanoid` benchmark. Furthermore, the deviation from the prior mode is more pronounced for the lower-dimensional benchmarks, being in line with our analysis of the bias-variance trade-off in Sec. 3.3. At the beginning of its execution, the Bayesian optimization algorithm that uses MLE with scaled initial length scales ('MLE (scaled)') uses longer length scales than all other methods. The resulting estimates vary significantly for the high-

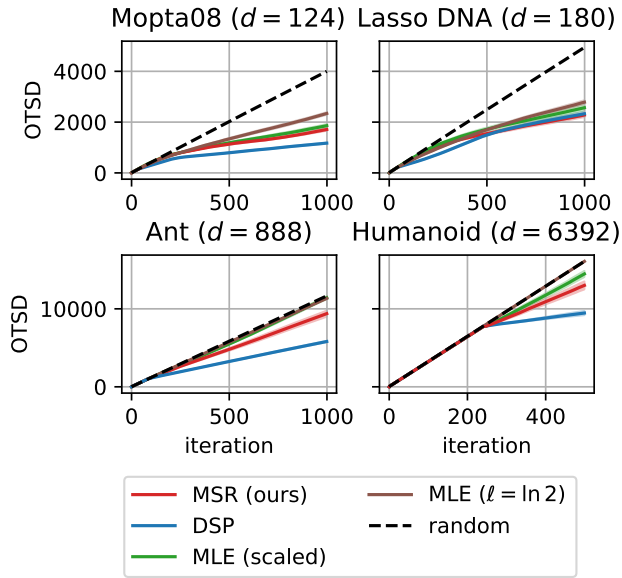


Figure 6.10: `DSP` exhibits the least exploration. MLE with fixed initial length scales performs like random search on `Ant` and `Humanoid`.

dimensional `Ant` and `Humanoid` problems, supporting our analysis in Sec. 3.3 where we study the comparatively high variance of MLE compared to MAP.

The length scales obtained by `MSR` lie between the values of `DSP` and of 'MLE without RAASP sampling' (green dots, 'MLE (scaled)') for most benchmarks. An exception is `Ant` where `MSR` sometimes results in shorter length scales than `DSP`. Overall, the RAASP sampling, which is the only difference between `MSR` and 'MLE (scaled)', obtains more consistent length scale estimates.

RAASP Promotes Locality. Fig. 6.10 compares the amount of exploration that the algorithms perform through the lens of the OTSD metric; see Section 3.2 for details on OTSD. We observe that `DSP` (blue curves) is the most exploitative method on all benchmarks, being in line with the fact that, after MLE with constant length scale initialization ('MLE ($\ell = \ln 2$)'), `DSP` has the shortest length scales on most benchmarks. The fact that 'MLE ($\ell = \ln 2$)' is the most *explorative* method, coinciding with the 'random search' line in Fig. 6.10, despite having the *shortest* length scales is explained by our analysis in Sec. 3.2: the random initial points for the AF optimization are not further optimized if the gradient of the AF vanishes, because the method does not employ RAASP sampling. We observe that 'MLE ($\ell = \ln 2$)' does not learn longer length scales during the optimization for `Ant` and `Humanoid`, as indicated by the horizontal brown line in in Fig. 6.9. Thus, it does not recover

later. For Mopta08 and Lasso-DNA, which have a lower input dimension, the effect is less pronounced because 'MLE ($\ell = \ln 2$)' sometimes learns longer length scales that avoid vanishing gradients of the AF. MLE with scaled initial length scale values (green curves in Fig. 6.10) is the second-most explorative method. However, this is not due to vanishing gradients but caused by overly long length scales, shown as green dots in Fig. 6.9. MSR (red curves) is more explorative than DSP and more exploitative than 'MLE (scaled)', which does not use RAASP sampling. This is consistent with the shorter length scales of MSR (red dots in Fig. 6.9), confirming that MSR not only yields more consistent length scale estimates but also acts more local than its RAASP-sampling-free equivalent.

Notes on Popular HDBO Benchmarks. In Fig. 6.9, all methods converge to similarly long length scales on the Mopta08 and Lasso-DNA benchmarks. This is likely attributable to a specific property of these popular benchmarks, as we discuss in Appendix C. In short, these benchmarks have a simple structure that benefits models with long length scales, posing the risk of algorithms being 'overfitted' to these benchmarks.

5 Conclusions and Future Work

Our analysis reveals fundamental challenges in HDBO while offering practical insights for improving HDBO methods. We demonstrate that common approaches for fitting Gaussian processes cause vanishing gradients in high dimensions. We propose an initialization for MLE that achieves state-of-the-art performance on established HDBO benchmarks without requiring the assumptions of maximum a-posteriori estimation. Finally, we provide empirical evidence that combining MLE with RAASP sampling reduces the variance of the length scale estimates and yields values closer to the ones learned by DSP, providing a fresh argument for the inclusion of RAASP sampling in HDBO.

In future work, we will continue to carefully vet popular benchmarks and propose novel, challenging benchmarks that preserve the traits of real-world applications. Furthermore, this work emphasizes the importance of numerical stability for the performance of Bayesian optimization in high dimensions. Thus, we propose approaching the development of models and acquisition functions from this perspective.

Impact. This paper presents work that aims to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which

we feel must be specifically highlighted here.

Acknowledgements

This project was partly supported by the Wallenberg AI, Autonomous Systems, and Software program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725

A A Review of Gaussian Processes and Bayesian Optimization

A.1 Gaussian Processes

Gaussian processes (GPs) model a distribution over functions, i.e., assume that f is drawn from a GP: $f \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}') + \sigma_n^2 \mathbb{1}_{[\boldsymbol{x}=\boldsymbol{x}']})$ where m and k are the mean and covariance function of the GP, respectively [42]. Common kernel functions include the radial basis function (RBF) kernel:

$$k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{r}{2}\right), \tag{6.4}$$

or the $5/2$automatic relevance determination (ARD)-Matérn kernel:

$$k_{\text{Mat}5/2}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \left(1 + \sqrt{5r} + \frac{5r}{3}\right) \exp\left(-\sqrt{5r}\right) \tag{6.5}$$

with $r = \sum_{i=1}^{d} \frac{(x_i - x_i')^2}{\ell_i^2}$. Here, $\boldsymbol{\ell}$ is a d-dimensional vector of component-wise length scales. Thus, the kernel's number of hyperparameters (HPs) in Eq. (6.5) is $d + 1$.

Given some training data $\mathcal{D} \coloneqq \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, $X \coloneqq (\boldsymbol{x}_1^\intercal, \ldots, \boldsymbol{x}_N^\intercal)^\intercal$, $\boldsymbol{y} = (y_1 \ldots, y_N)^\intercal$, the function values $\boldsymbol{y}_*$ of a set of query points X_* is normally distributed as

$$\boldsymbol{y}_* | X, \boldsymbol{y}, X_* \sim \mathcal{N}(K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1} \boldsymbol{y}, \tag{6.6}$$

$$K(X_*, X_*) - K(X_*, X)(K(X, X)^{-1} + \sigma_n^2 I) K(X, X_*)) \tag{6.7}$$

Let $\boldsymbol{\theta} = \{\sigma_n^2, \sigma_f^2, \boldsymbol{\ell}\}$ be the set of GP hyperparameters. The GP surrogate is then typically fitted by maximizing the marginal log-likelihood w.r.t. $\boldsymbol{\theta}$, also known as maximum-likelihood estimation (MLE), i.e.

$$\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|X, \boldsymbol{\theta}) \tag{6.8}$$

$$\begin{aligned}\log p(\boldsymbol{y}|X, \boldsymbol{\theta}) = &-\frac{1}{2}\boldsymbol{y}^\intercal \left(K(X,X) + \sigma_n^2 I\right)^{-1} \boldsymbol{y} \\ &- \frac{1}{2}\log|K(X,X) + \sigma_n^2 I| - \frac{n}{2}\log 2\pi\end{aligned} \tag{6.9}$$

With a gradient-based approach, this is done by maximizing Eq. (6.9), which is usually multi-modal and difficult to optimize. The gradient of the marginal log-likelihood w.r.t. θ_i is given by

$$\begin{aligned}\frac{\partial}{\partial \theta_i}\log p(\boldsymbol{y}|X, \boldsymbol{\theta}) = &\frac{1}{2}\boldsymbol{y}^\intercal \left(K + \sigma_n^2 I\right)^{-1} \frac{\partial K}{\partial \theta_i} \left(K + \sigma_n^2 I\right)^{-1} \boldsymbol{y} \\ &- \frac{1}{2}\mathrm{tr}\left(\left(K + \sigma_n^2 I\right)^{-1} \frac{\partial K}{\partial \theta_i}\right),\end{aligned} \tag{6.10}$$

where $\frac{\partial K}{\partial \theta_i}$ is the symmetric Jacobian matrix of partial derivatives w.r.t. θ_i.

One often endows the GP hyperparameters with hyperpriors and seeks the mode of the posterior distribution, known as maximum a-posteriori estimation (MAP):

$$\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|X, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}). \tag{6.11}$$

A.2 Bayesian Optimization

Bayesian optimization (BO) is an iterative approach, alternating between fitting the model and choosing query points. Query points are found by maximizing an acquisition function (AF), e.g., expected improvement (EI) [30]. The EI AF measures how much observing a point $\boldsymbol{x}$ is expected to improve over the best function value observed thus far. It is defined as

$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}_{f(\boldsymbol{x})}\left[[f(\boldsymbol{x}) - y^\star]_+\right] = (\mu_N(\boldsymbol{x}) - y^\star)\,\Phi(Z) + \sigma_N(\boldsymbol{x})\phi(Z) \tag{6.12}$$

with $Z = \frac{\mu(\boldsymbol{x}) - y^\star}{\sigma(\boldsymbol{x})}$, Φ and ϕ being the standard normal cumulative density function (CDF) and probability density function (PDF), and μ_N and σ_N^2 being the posterior mean and posterior variance at $\boldsymbol{x}$, i.e.,

$$\mu_N(\boldsymbol{x}) = K(\boldsymbol{x}, X)(K(X,X) + \sigma_n^2 I)^{-1}\boldsymbol{y} \tag{6.13}$$

$$\sigma_N^2(\boldsymbol{x}) = (k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2)(K(X,X) + \sigma_n^2 I)^{-1}K(X, \boldsymbol{x}) \tag{6.14}$$

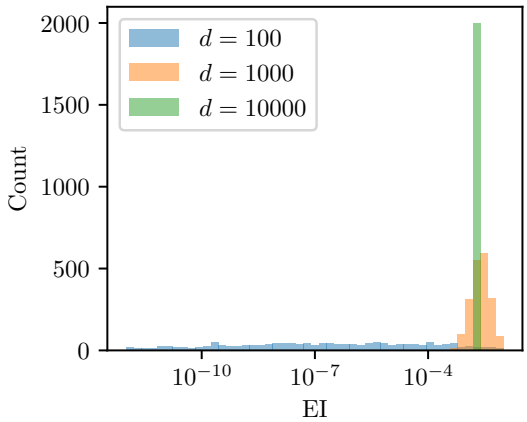


Figure 6.11: Distribution of EI values for GPs in various dimensionalities. When conditioning on the same amount of data points and maintaining the length scale as the dimensionality grows, the distribution of EI values becomes more peaked.

As discussed by Ament et al. [1], EI often suffers from vanishing gradients, which only worsens in high-dimensional spaces due to the plethora of flat regions. This is shown in Fig. 6.11. Here, we condition a GP on 100 random points in $[0, 1]^d$ for which we obtain function values by drawing from a GP prior with an isotropic RBF kernel with $\ell = 10$. We evaluate the AF on 2000 points drawn from a scrambled Sobol sequence and plot the histograms for various dimensionalities. As the dimensionality grows, there are more equal EI values, indicating flat regions in the AF and possible problems with vanishing gradients of the EI function. Ament et al. [1] propose `LogEI`, a numerically more stable version of EI that solves many of the numerical issues with EI.

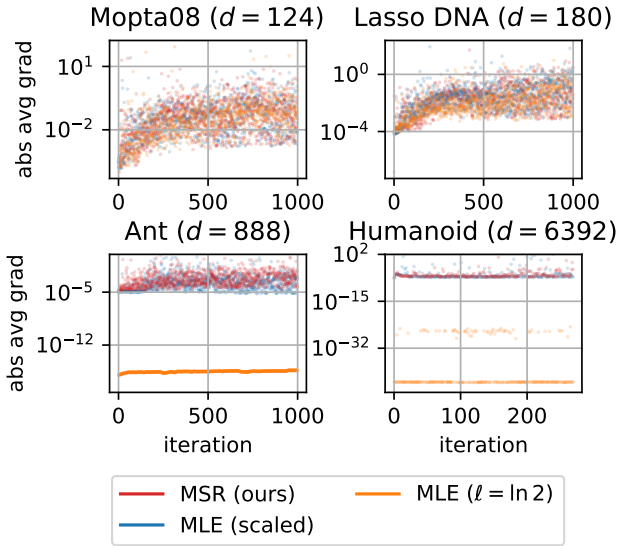


Figure 6.12: Mean absolute value of the gradients for the different MLE methods, including the proposed `MSR`. The constant length scale initialization exhibits vanishing gradients for the high-dimensional `Ant` and `Humanoid` problems.

B Additional Experiments

B.1 Ranking of Optimization Algorithms

Table 6.8: Ranking for the different optimizers on the benchmark problems according to their final performance. `SAASBO` is excluded from the comparison as it was not run for the entirety of the optimization; `Bounce` ran into memory issues on `Humanoid` and, therefore, does not have a rank on this benchmark.

	MSR	DSP	Bounce	MLE (scaled)	MLE ($\ell = \ln 2$)
`Mopta08` ($d = 124$)	1	2	5	3	4
`Lasso-DNA` ($d = 180$)	4	1	5	3	2
`Ant` ($d = 888$)	2	4	3	1	5
`Humanoid` ($d = 6392$)	2	1	-	3	4

B.2 MLE Gradients for Real-World Experiments

Complementing our analysis in Sec. 4, Fig. 6.12 shows the average absolute gradients of the different MLE methods, including our proposed `MSR` method. The constant length scale initialization ('MLE ($\ell = \ln 2$)') is the only method consistently exhibiting vanishing gradients on the 888-dimensional `Ant` and the

6392-dimensional Humanoid problems as depicted by the solid orange lines for those benchmarks. On Mopta08 and Lasso DNA, all methods have non-vanishing gradients. Furthermore, both MLE methods scaling the initial length scale do not suffer from vanishing gradients on Ant and Humanoid.

B.3 High-dimensional Benchmark Functions

By the no-free-lunch theorem [52], the relative performance of an optimization algorithm depends on the properties of the problems it operates on. Here, we show that for several benchmark problems, no state-of-the-art algorithm strictly dominates the other methods.

Table 6.9 shows the relative performances of CMA-ES, DSP, and TuRBO after 1000 optimization steps on the 100-dimensional versions of the Levy, Schwefel, and Griewank benchmarks. We evaluate Levy in the bounds $[-10, 10]$, Schwefel in the bounds $[-500, 500]$, and Griewank in the bounds $[-600, 600]$ by scaling from the unit hypercube in which the GP operates to the respective bounds before evaluating the function.

To better understand the reason for the performance differences, we study the observation traveling salesman distance (OTSD) for the different functions, shown in Fig. 6.13. Plots of the 2-dimensional versions of all three benchmarks are shown in Fig. 6.16 and the performance and OTSD plot of 100-dimensional Levy figure can be found in Fig. 6.14. CMA-ES shows the lowest level of exploration and has the lowest OTSD on the Schwefel function, where it outperforms the two other algorithms. On Griewank (see Fig. 6.15), DSP has the highest average OTSD performs best while the less explorative CMA-ES shows the worst performance. For Levy, both TuRBO and DSP are relatively explorative and outperform CMA-ES by a considerable margin. We conclude that more explorative algorithms are advantageous on the benchmarks with a clear global trend like Griewank, which resembles a paraboloid, and Levy, which has a parabolic shape along the x_1 dimension (see Fig. 6.16). In contrast, the Schwefel benchmark is more "stationary" in that a point's function value depends less on that point's absolute position in the space. Noteworthy,

Table 6.9: Relative performances of CMA-ES, DSP, and TuRBO after optimizing for 1000 iterations averaged over 10 repetitions. No algorithm performs best for all benchmarks.

Benchmark	Rank 1	Rank 2	Rank 3
Levy100	TuRBO	DSP	CMA-ES
Schwefel100	CMA-ES	DSP	TuRBO
Griewank100	DSP	TuRBO	CMA-ES

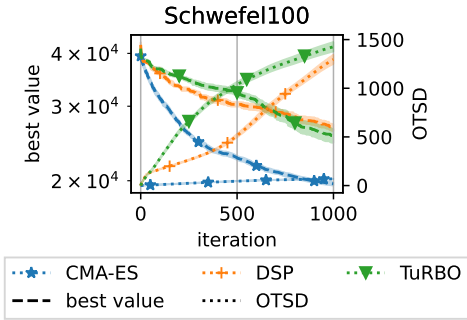


Figure 6.13: OTSD (solid lines) and performance curves (dashed lines) of the 100-dimensional `Schwefel` function

stationarity as assumed by GP models with a stationary covariance function, which, by far, are the most common covariance functions for high-dimensional Bayesian optimization (HDBO). A more local approach such as `CMA-ES` is beneficial on this highly multi-modal benchmark.

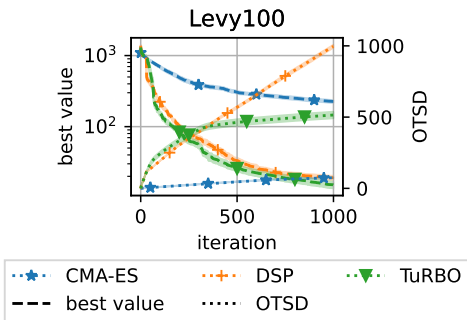


Figure 6.14: OTSD (solid lines) and performance curves (dashed lines) of the 100-dimensional `Levy` function

Figs. 6.14 and 6.15 show the OTSD and performance plots for the 100-dimensional `Levy` and `Griewank` functions. Fig. 6.16 shows the two-dimensional versions of the `Levy`, `Griewank`, and `Schwefel` benchmark functions.

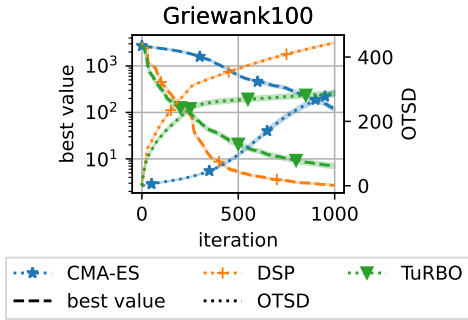


Figure 6.15: OTSD (solid lines) and performance curves (dashed lines) of the 100-dimensional `Griewank` function

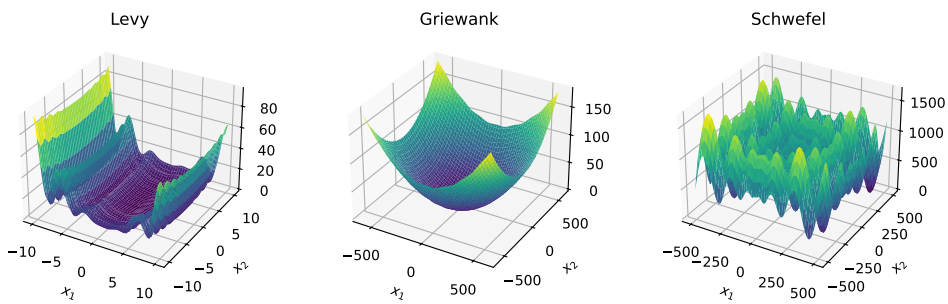


Figure 6.16: The two-dimensional versions of the `Levy`, `Griewank`, and `Schwefel` benchmark functions used above.

B.4 Hard Optimization Problems

We reiterate that the curse of dimensionality (COD) remains a reality and exists even for low-dimensional problems. Fig. 6.17 shows 100 evaluations made by EI on a 2-dimensional GP prior sample as the benchmark function. There is no model mismatch; the length scales of the surrogate model are set to the correct value ($\ell = 0.025$). However, EI operates locally and fails to find the global optimum (marked by a red cross).

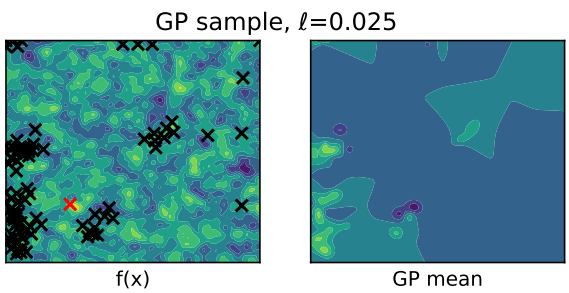


Figure 6.17: `LogEI` run on a two-dimensional GP prior sample for 100 evaluations. The right panel shows the posterior mean at the end of the optimization. For highly multimodal benchmarks, EI reverts to a local search behavior and does not obtain a global optimum (red cross).

C Popular Benchmarks Seem Simpler Than Expected

This section examines two popular high-dimensional benchmarks, the $180d$ `Lasso-DNA` and $124d$ `Mopta08`. In what follows, we will demonstrate that many input variables seem to have little influence on the objective value. These empirical findings suggest that these benchmarks are not truly as high-dimensional as their nominal number of input variables might suggest, potentially misleading the perceived difficulty of these benchmarks and confounding the assessment of what state-of-the-art (SOTA) algorithms will be able to accomplish in practice.

We begin by collecting the top 10% best points identified by the SOTA algorithm using dimensionality-scaled prior (DSP) [17] for each benchmark. For each dimension x_i separately, we then count how often these points lie on the boundary of the search space. If over half of the points place x_i on the boundary, we label x_i as *secondary*, and as *dominant* otherwise. To ensure consistency across runs, we perform 15 repetitions and consider those dimensions that have been identified as dominant in eight or more of the repetitions.

Table 6.10 reports the number of dominant and secondary dimensions. In both benchmarks, a large fraction of the dimensions are classified as *secondary*, meaning that the best solutions obtained by DSP method set the corresponding input variables to value near the boundary of the search space. Note that our finding is directionally aligned with the 43 active dimensions that Šehić et al. [45] reported for `Lasso-DNA`, using a different method to estimate the number of relevant dimensions.

We further investigate the impact of each set of dimensions by replacing, at each iteration, either the dominant dimensions (f_{dominant}) or the secondary dimensions ($f_{\text{secondary}}$) with randomly chosen binary values. The rows $\bar{f}_{\text{dominant}}$ and $\bar{f}_{\text{secondary}}$ in Table 6.10 indicate the corresponding average objective values (with standard errors in parentheses). Replacing secondary dimensions impairs the result only slightly, whereas randomizing the dominant dimensions yields a markedly larger performance drop. A two-sided Wilcoxon test shows that all differences are statistically significant at $p < 0.001$.

Next, we examine how Gaussian process (GP) models account for this structure during HDBO. As illustrated in Fig. 6.18, the estimated length scales of secondary dimensions are typically large, which is consistent with the observation that they influence the (predicted) objective value less. Intuitively, if a dimension's optimal value often lies at the boundary, the GP can assign it a very large length scale, learning the trend toward the boundary with few evaluations and leaving more of the budget to explore the truly dominant dimensions. In effect, the BO loop focuses on those dimensions that genuinely drive performance.

Table 6.10: The number of dimensions identified as dominant and secondary and the average function values obtained when replacing the dominant ($\bar{f}_1$) or secondary ($\bar{f}_2$) parameters with uniformly random values. The average values and standard deviations for uniformly random points are shown as $\bar{f}_{\text{rand}}$. Replacing secondary parameters harms performance considerably less than replacing dominant parameters.

	`Lasso-DNA` (d=180)	`Mopta08` (d=124)
# dominant	69.33	30.80
# secondary	110.67	93.20
$\bar{f}_{\text{dominant}}$	0.315 ($\pm 4 \times 10^{-4}$)	328.824 (± 1.920)
$\bar{f}_{\text{secondary}}$	0.445 ($\pm 4 \times 10^{-3}$)	430.474 (± 8.164)
$\bar{f}_{\text{rand}}$	0.410 ($\pm 3.3 \times 10^{-2}$)	403.428 (± 48.976)

Below, we demonstrate that for these problems, BO will set most of the input variables to exactly zero or one – that is, to the boundary of the search space – regardless of whether the GP is fit by MLE or MAP. Thus, we conclude that the task effectively has a lower dimensionality than its nominal number of input variables. While this property was already recognized for `Lasso-DNA`, our results demonstrate for the first time that it also applies to `Mopta08`. Furthermore, it is interesting to note that a simple GP surrogate fitted via MLE or MAP can leverage this structure, a behavior that had not been previously observed.

Above, we showed that 1) dimensions of the best points observed by MLE and MAP that predominantly lie on the border have significantly longer length scales than the "dominant" dimensions, and 2) most of the dimensions have marginal impact. MLE shows a similar behavior and is omitted for brevity. We further omit the 388-dimensional `SVM` benchmark as it is known to have a low-dimensional effective subspace [10].

To complement our analysis, we show that MLE and MAP assign dominant dimensions mainly values at the boundary. This indicates that the GP model actually makes use of the specific characteristics of these benchmarks.

Figs. 6.19 and 6.20 further show that BO consistently evaluates a large share of the parameters at the border. Fig. 6.19 shows this for DSP by Hvarfner et al. [17] whereas Fig. 6.20 uses MLE as proposed in Section 4. The general trend is that, during the course of the optimization, increasingly many parameters are evaluated at the border, which is consistent with Fig. 6.18.

We thus argue that two HDBO benchmarks do not fully capture the complexity of HDBO because of this simple structure. While this is to be expected for `Lasso-DNA` and `SVM`, this property has not been discussed for the soft-constrained `Mopta08` benchmark introduced in Eriksson et al. [11] to the best of our knowledge.

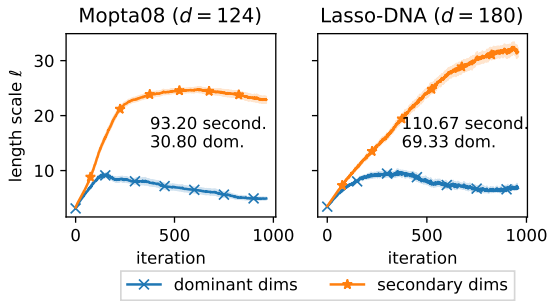


Figure 6.18: The mean average length scales of "dominant" and "secondary" dimensions for the `Mopta08` (left) and `Lasso-DNA` (right) benchmarks for DSP.

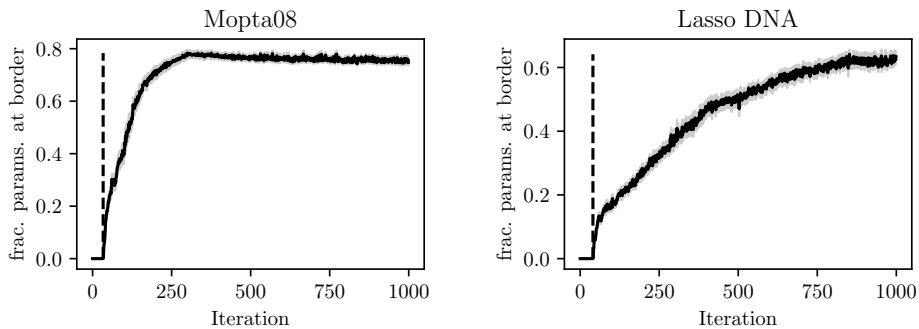


Figure 6.19: Fraction of dimensions set to a value at the border (0 or 1) by DSP. The shaded area shows the standard error of the mean across 15 repetitions.

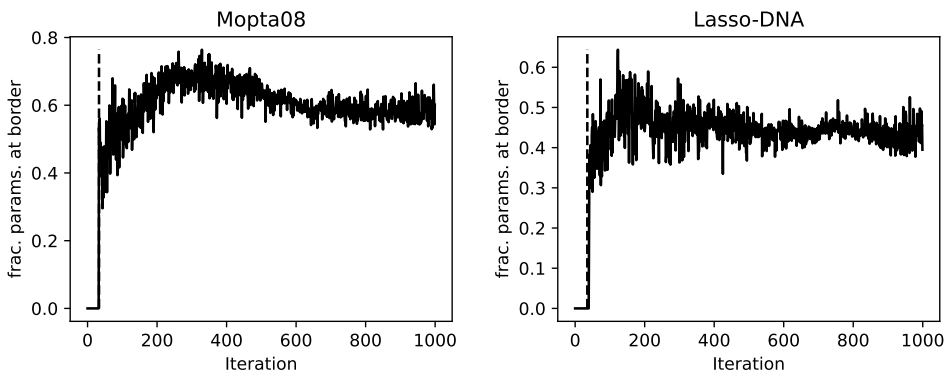


Figure 6.20: Fraction of dimensions set to a value at the border (0 or 1) by our MLE method. The shaded area shows the standard error of the mean across 15 repetitions.

References

[1] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected Improvements to Expected Improvement for Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[3] Petr Baudiš and Petr Pošík. Online Black-Box Algorithm Portfolios for Continuous Optimization. *Parallel Problem Solving from Nature – PPSN XIII*, pages 40–49, Cham, 2014. Springer International Publishing.

[4] Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

[5] Mohamed Amine Bouhlel, Nathalie Bartoli, Rommel G. Regis, Abdelkader Otsmane, and Joseph Morlier. Efficient global optimization for high-dimensional constrained problems by using the Kriging models combined with the partial least squares method. *Engineering Optimization*, 50(12):2038–2053, 2018.

[6] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[7] Jingfan Chen, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. Semi-supervised Embedding Learning for High-dimensional Bayesian Optimization. *arXiv preprint arXiv:2005.14601*, 2020.

[8] Aryan Deshwal, Sebastian Ament, Maximilian Balandat, Eytan Bakshy, Janardhan Rao Doppa, and David Eriksson. Bayesian Optimization over High-Dimensional Combinatorial Spaces via Dictionary-based Embeddings. *International Conference on Artificial Intelligence and Statistics*, pages 7021–7039. PMLR, 2023.

[9] David K Duvenaud, Hannes Nickisch, and Carl Rasmussen. Additive Gaussian Processes. *Advances in neural information processing systems*, 24, 2011.

[10] David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

[11] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5496–5507, 2019.

[12] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[13] Jacob Gardner, Chuan Guo, Kilian Weinberger, Roman Garnett, and Roger Grosse. Discovering and exploiting additive structure for Bayesian optimization. *International Conference on Artificial Intelligence and Statistics*, pages 1311–1319, 2017.

[14] Eric Han, Ishank Arora, and Jonathan Scarlett. High-dimensional Bayesian optimization via tree-structured additive models. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7630–7638, 2021.

[15] Erik Hellsten, Carl Hvarfner, Leonard Papenmeier, and Luigi Nardi. High-dimensional Bayesian Optimization with Group Testing. *arXiv preprint arXiv:2310.03515*, 2023.

[16] Moritz Herrmann, F Julian D Lange, Katharina Eggensperger, Giuseppe Casalicchio, Marcel Wever, Matthias Feurer, David Rügamer, Eyke Hüllermeier, Anne-Laure Boulesteix, and Bernd Bischl. Position: Why We Must Rethink Empirical Research in Machine Learning. *Forty-first International Conference on Machine Learning*, 2024.

[17] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian Optimization Performs Great in High Dimensions. Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20793–20817. PMLR, 21–27 Jul 2024.

[18] Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21:345–383, 2001.

[19] Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. *MOPTA 2008 Conference (20 August 2008)*, 2008.

[20] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455, 1998.

[21] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. *International conference on machine learning (ICML)*, pages 295–304, 2015.

[22] Toni Karvonen and Chris J Oates. Maximum likelihood estimation in Gaussian process regression is ill-posed. *Journal of Machine Learning Research*, 24(120):1–47, 2023.

[23] Mario Köppen. The curse of dimensionality. *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8, 2000.

[24] Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in Bayesian optimization with applications in aerospace engineering. *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[25] Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional Bayesian optimization. *Advances in Neural Information Processing Systems 33*, NeurIPS, 2020.

[26] Daniel J Lizotte, Tao Wang, Michael H Bowling, Dale Schuurmans, et al. Automatic Gait Optimization With Gaussian Process Regression. *IJCAI*, volume 7, pages 944–949, 2007.

[27] Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

[28] Natalie Maus, Haydn Thomas Jones, Juston Moore, Matt Kusner, John Bradshaw, and Jacob R. Gardner. Local latent space bayesian optimization over structured inputs. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[29] Matthias Mayr, Faseeh Ahmad, Konstantinos I. Chatzilygeroudis, Luigi Nardi, and Volker Krüger. Skill-based Multi-objective Reinforcement Learning of Industrial Robot Tasks with Planning and Knowledge Integration. *CoRR*, abs/2203.10033, 2022.

[30] Jonas Mockus. The Bayesian approach to global optimization. *System Modeling and Optimization: Proceedings of the 10th IFIP Conference New York City, USA, August 31–September 4, 1981*, pages 473–481. Springer, 2005.

[31] Riccardo Moriconi, Marc Peter Deisenroth, and KS Sesh Kumar. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109:1925–1943, 2020.

[32] Mojmir Mutny and Andreas Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[33] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian Optimization in Embedded Subspaces. *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research (PMLR)*, pages 4752–4761, 09–15 Jun 2019.

[34] Diana M. Negoescu, Peter I. Frazier, and Warren B. Powell. The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[35] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian Optimization using the Graph Cartesian Product. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[36] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the Scope as You Learn: Adaptive Bayesian Optimization in Nested Subspaces. *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

[37] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Bounce: a Reliable Bayesian Optimization Algorithm for Combinatorial and Mixed Spaces. *arXiv preprint arXiv:2307.00618*, 2023.

[38] Leonard Papenmeier, Nuojing Cheng, Stephen Becker, and Luigi Nardi. Exploring Exploration in Bayesian Optimization. 2025. URL `leonard.papenmeier.io/otsd-paper`.

[39] Giulia Pedrielli and Szu Hui Ng. G-STAR: A new kriging-based trust region method for global optimization. *2016 Winter Simulation Conference (WSC)*, pages 803–814. IEEE, 2016.

[40] Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778, 2018.

[41] Bahador Rashidi, Kerrick Johnstonbaugh, and Chao Gao. Cylindrical Thompson sampling for high-dimensional Bayesian optimization. *International Conference on Artificial Intelligence and Statistics*, pages 3502–3510. PMLR, 2024.

[42] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006.

[43] Rommel G Regis. Trust regions in Kriging-based optimization with expected improvement. *Engineering optimization*, 48(6):1037–1059, 2016.

[44] Rommel G Regis and Christine A Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

[45] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. LassoBench: A High-Dimensional Hyperparameter Optimization Benchmark Suite for Lasso. *International Conference on Automated Machine Learning*, pages 2–1. PMLR, 2022.

[46] Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte Carlo Tree Search based Variable Selection for High Dimensional Bayesian Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022.

[47] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *Proceedings of the International Conference on Machine Learning*, 2010.

[48] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 11259–11272, 2020.

[49] 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 (NeurIPS)*, 33:19511–19522, 2020.

[50] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[51] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *Journal of Artificial Intelligence Research (JAIR)*, 55:361–387, 2016.

[52] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[53] Zhitong Xu and Shandian Zhe. Standard Gaussian Process is All You Need for High-Dimensional Bayesian Optimization. *arXiv preprint arXiv:2402.02746*, 2024.

[54] Juliusz Krzysztof Ziomek and Haitham Bou Ammar. Are Random Decompositions all we need in High Dimensional Bayesian Optimisation? *International Conference on Machine Learning*, pages 43347–43368. PMLR, 2023.