IMT School for Advanced Studies, Lucca
Lucca, Italy

Global and preference-based optimization using surrogate-based methods

PhD Program in Systems Science

Track in CSSE

XXXV Cycle

By

Mengjia Zhu

2024

The dissertation of Mengjia Zhu is approved.

PhD Program Coordinator: Prof. Alberto Bemporad, IMT School for Advanced Studies Lucca

Advisor: Prof. Alberto Bemporad, IMT School for Advanced Studies Lucca

The dissertation of Mengjia Zhu has been reviewed by:

Prof. Fabio Schoen, University of Florence

Prof. Benoît Chachuat, Imperial College London

IMT School for Advanced Studies Lucca
2024

“Sometimes I wish I could have been a bit more relaxed, but then I wouldn’t have been the same player.“

— Steffi Graf

Contents

List of Figures

List of Tables

Acknowledgements

I would like to begin by expressing my sincere appreciation to my supervisor, Prof. Alberto Bemporad, for his continuous guidance and support throughout this journey. His profound expertise in the research field has consistently provided me with valuable feedback and insightful suggestions to guide me in the right direction. He has always been approachable for discussions, available via email and in-person meetings. Furthermore, he has played a crucial role in helping me develop good research habits and be more organized regarding the research process. Additionally, he has consistently challenged me to think critically and independently. I have always held in high regard his work ethic, exceptional efficiency, and amazing coding skills. Moreover, he has aided me in honing my writing and presentation skills. Under his supervision, my research journey has been a constant source of inspiration and fulfillment.

I would like to extend my sincere gratitude to Prof. Dario Piga for his invaluable support and collaboration throughout our joint research projects. I am immensely grateful for his constant availability and willingness to provide guidance and assistance. The fruitful discussions and constructive feedback have significantly contributed to the quality of our work.

I also want to thank Dr. Hasan Esen, Dr. Maximilian Kneissl, and Dr. Adam Molin from DENSO Automotive Deutschland GmbH, and Dr. Dejan Ničković, and Dr. Edgar A. Aguilar from Austrian Institute of Technology GmbH for our collaboration on the joint industrial project. This expe-

rience has been truly invaluable and has significantly contributed to my personal and professional growth.

I am grateful to Dr. Ehecatl Antonio del Río Chanona for providing me the opportunity to undertake the Erasmus+ traineeship at the Imperial College London. I also appreciate the supervision and collaboration of Dr. Ehecatl Antonio del Río Chanona, Dr. Ye Seol Lee, and Prof. Kim Jelfs as well as the insightful discussions and collaborations with Dr. Austin Mroz and Dr. Lingfeng Gui for our project on experimental design. Additionally, I thank all my colleagues for their camaraderie and enjoyable interactions during my visit.

I am also grateful to Dr. Loris Cannelli and Dr. Francesco Farina for our collaboration on the distributed version of GLIS.

I want to thank Prof. Fabio Schoen and Prof. Benoît Chachuat for taking the time to review my thesis and offering valuable feedback to enhance its quality.

The journey of these past years at IMT has truly impacted me in many dimensions. I want to express my deep appreciation to my dear friends for their support, companionship, and the joyous/touching moments we have shared. The"Tea + party" group with Asli, Kristina, Luisana, and Serenella has been an enormous support since day one of my Ph.D. studies. Over these years, we have created cherished memories and participated in each other's important life events. I also want to thank my friends Sampath, Stefano, Daniele, Nick, Liang, Hamid, and Shokh, as well as Mario from our *DYSCO* research unit for the walks around the wall, the ping-pong/foosball games, all the discussions, and the emotional support. I would also like to thank Di for the support and for making me feel at home. I thank people from the IMT offices/receptions/canteen, especially Barbara, Daniela and Silvia, for helping with all the documentations and emotional

support.

I want to thank Fang, Xinyi, Jiachun, Xuan, Qixuan, and Anne for the company through all these years. I am truly fortunate to have them by my side. I also express my deep appreciation to Na and Jiaqi for dragging me out of emotional holes, for the ups and downs, and for everything we have experienced together.

I would like to thank Prof. Sandro Macchietto, Prof. Lilo Pozzo, and Prof. Nirmala Savage for their continuous support academically and personally.

Finally, I would like to thank my parents and my grandma for their generous and continuous support along my personal and educational journey over the years. Nothing would be possible without them.

Vita

August 29, 1995	Born, Zhejiang, China
2017	B.S. in Chemical Engineering, minor in Math Final mark: 3.72/4.00 University of Washington, Seattle, US
2019	MSc in Advanced Chemical Engineering with PSE Final mark: Distinction Imperial College London, London, UK
2019 - 2024	PhD in System Science IMT School for Advanced Studies Lucca, Lucca, Italy
Aug 2023 - Jan 2024	Visiting PhD student Imperial College London, London, UK

Publications

1. M. Zhu, A. Bemporad, M. Kneissl, and H. Esen, "Learning critical scenarios in feedback control systems for automated driving," *IEEE 26th International Conference on Intelligent Transportation Systems*, Bilbao, Spain, 2023, pp. 321-328.
2. L.Cannelli, M. Zhu, F. Farina, A. Bemporad, and D. Piga, "Multi-agent active learning for distributed black-box optimization," *IEEE Control Systems Letters*, vol. 7, pp. 1488-1493, 2023.
3. A. Molin, E. Aguilar, D. Nickovic, M. Zhu, A. Bemporad, and H. Esen, "Specification-guided critical scenario identification for automated driving," *25th International Symposium on Formal Methods*, 2023.
4. M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," in *IEEE Trans. on Control Systems Technology*, vol. 30, no. 5, pp. 2176–2187, 2022.
5. M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," in *European Control Conference*, Rotterdam, Netherlands, 2021, pp. 638–645.

Submitted

1. M. Zhu, A. Mroz, L. Gui, K. Jelfs, A. Bemporad, EA. del Río Chanona, and Y. Lee, "Discrete and mixed-variable experimental design with surrogate-based approach," *submitted for publication*, 2024.
2. M. Zhu and A. Bemporad, "Global and preference-based optimization with mixed variables using piecewise affine surrogates," *submitted for publication*, 2023.

Presentations

1. M. Zhu, “Preference-based Optimization“, IPSE Seminar at *University of Surrey*, Surrey, UK, 2024.
2. M. Zhu, “Learning critical scenarios in feedback control systems for automated driving,“ at *26th IEEE International Conference on Intelligent Transportation Systems*, Bilbao, Bizkaia, Spain, 2023.
3. M. Zhu, “Preference-based MPC calibration,” at *European Control Conference*, virtual, 2021.
4. M. Zhu, “Preference-based MPC calibration,” at *AUTOMATICA.IT Workshop*, virtual, 2020.

Abstract

This thesis explores methodologies in black-box and preference-based optimization, addressing three key research questions. Firstly, it introduces a semi-automated calibration approach that eliminates the need for an explicit performance index by relying on human calibrator preferences. Secondly, the thesis delves into preference-based global optimization algorithms that address optimization problems where the analytic expression of the objective function is unknown and the optimization is subject to unknown constraints. The proposed algorithm, C-GLISp, extends the active preference learning framework to handle these unknown constraints. Lastly, the thesis tackles the challenge of optimization problems involving mixed variables and linear constraints. To address this challenge, we present a novel surrogate-based global optimization algorithm, named PWAS. The algorithm constructs a piecewise affine surrogate of the objective function over feasible samples and utilizes exploration functions to efficiently navigate the feasible domain using mixed-integer linear programming solvers. Additionally, a preference-based version of the algorithm, PWASp, is introduced to handle situations where only pairwise comparisons between samples are available instead of direct objective function evaluations. The efficiency and effectiveness of the proposed approaches are demonstrated via benchmark studies. Additionally, the practical applicability of PWAS is discussed via experimental design case stuides.

List of Notations

General

- $\mathbb{R}$ is the set of all real numbers
- $\mathbb{Z}$ is the set of all integers
- $\mathbb{N}$ is the set of all natural numbers
- $\mathbb{R}^n$ denotes the set of real vectors of dimension n with $n \in \mathbb{N}$
- $\mathbb{R}^{m \times n}$ denotes the set of matrices of dimension $m \times n$ (m rows and n columns) with $n, m \in \mathbb{N}$
- $\mathbb{Z}^n$ denotes the set of integer vector of dimension n with $n \in \mathbb{N}$
- $x \in \mathbb{R}^n$ denotes the optimization vector
- $\mathcal{X}$ is often used to denote the set of x within linear/nonlinear equality/inequality constraints
- $x^\intercal$ denotes the transpose of vector x
- x^* denotes the optimum x identified after optimization
- $\|x\|_p$ denotes the general p-norm of vector x
- Given $\ell \in \mathbb{R}^n$, ℓ^j denotes the jth element of vector ℓ
- Given $A \in \mathbb{R}^{m \times n}$, A^{ij} denotes the element at row i and column j of matrix A
- $|\mathcal{A}|$ denotes the cardinality of the set $\mathcal{A}$
- $\lfloor a \rfloor$ denotes the greatest integer smaller than or equal to a, for $a \in \mathbb{R}$
- $\lceil a \rceil$ denotes the smallest integer greater than or equal to a, for $a \in \mathbb{R}$
- $\hat{f}$ is often used to denote the surrogate approximation of the (unknown) latent function f

Model Predictive Control

- $\|\nu\|_Q^2$ is the weighted squared norm, *i.e.*, $\nu^\top Q\nu$
- N_p is the prediction horizon and N_u is the control horizon
- $u_{t+k|t}$, $k = 1, \ldots, N_p$ denotes the control action at time step $t + k$ up to the end of the prediction horizon given the information at time step t
- Q_u denotes the positive-semidefinite weight matrix of vector u

List of Abbreviations

BO	Bayesian Optimization
BoTorch	BO in PyTorch
C-GLISp	GLISp under unknown constraints
CAMD	Computer-aided Molecular Design
CMA-ES	Covariance Matrix Adaptation - Evolution Strategy
CoCaBO	Continuous and Categorical BO
CoCaBO-0.5	CoCaBO with trade-off parameter $\lambda = 0.5$
CoCaBO-auto	CoCaBO with λ optimized as a hyperparameter
CSTR	Continuous Stirring Tank Reactor
DEAP	Distributed Evolutionary Algorithm in Python
DoE	Design of experiment
EDA	Estimation of Distribution
EDBO	Experimental Design via BO
EXP3BO	Exponential-weight algorithm for Exploration and Exploitation with BO
GLIS	Global minimum using Inverse distance weighting and Surrogate radial basis functions
GLISp	Preference-based GLIS
GP	Gaussian Process
GPT	Generative Pre-trained Transformer
HTE	High-throughput experimentation
IDW	Inverse Distance Weighting
LHS	Latin Hypercube Sampling
LK	Lane-Keeping

LLM	Large Language Model
LP	Linear Programming
LPV	Linear parameter-varying model
LTV	Linear time-varying model
MIDACO	Mixed Integer Distributed Ant Colony Optimization
MILP	Mixed-integer Linear Programming
MISO	Mixed-Integer Surrogate Optimization
MPC	Model Predictive Control
NOMAD	Nonlinear Optimization with the Mesh Adaptive Direct search
OA	Obstacle-Avoidance
PARC	Piecewise Affine Regression and Classification
PBO	Preference-based Bayesian Optimization
PWA	PieceWise Affine function
PWAS	Global optimization using PWA surrogates
PWASp	Preference-based PWAS
QP	Quadratic Programming
RBF	Radial Basis Function
PSO	Particle Swarm Optimization
SMAC	sequential model-based algorithm configuration
SVM	Support Vector Machine
TPE	Tree-structured Parzen Estimator

Chapter 1

Overview

1.1 Introduction

In today's dynamic and complex world, pursuing optimal solutions across various domains has become essential. Whether in engineering, finance, or data analysis, the ability to efficiently navigate the vast solution space to find the best outcomes is a compelling challenge. Researchers and practitioners have developed diverse methodologies and algorithms to tackle these intricate optimization problems. Global optimization methods have emerged as indispensable tools for finding optimal solutions within a given problem space. On the other hand, the applicability of many traditional optimization techniques is often constrained by assumptions of known objective functions and explicit problem structures. Black-box optimization, also known as derivative-free optimization, is a powerful technique used to find the optimal solution for a problem when the underlying function is unknown or cannot be directly accessed. However, given an input, the objective function can be measured. Optimization methods that tackle black-box optimization problems have been widely used in various fields, including calibration in control engineering, hyperparameter tuning in machine learning, and portfolio design in finance.

1.2 Thesis outline and contributions

In this thesis, we focus on the following research questions. The content of the research questions is partially or fully reprinted from [1–3]:

M. Zhu and A. Bemporad, "Global and preference-based optimization with mixed variables using piecewise affine surrogates," *submitted for publication*, 2023.

1. Automating the calibration of the parameters of a control policy through global optimization requires quantifying a closed-loop performance function. However, this can be impractical in many situations [1]. What are possible ways to solve such black-box global optimization problems without measuring the objective function?

2. Preference-based global optimization algorithms minimize an unknown objective function only based on whether the function is better, worse, or similar for given pairs of candidate optimization vectors. Such optimization problems arise in many real-life examples, such as finding the optimal calibration of the parameters of a control law. The calibrator can judge whether a particular combination of parameters leads to a better, worse, or similar closed-loop performance. The search for the optimal parameters is often subject to unknown constraints. For example, the vector of calibration

parameters must not lead to closed-loop instability [2]. What can we do to encourage feasible sampling when there exist unknown constraints?

3. Optimization problems involving mixed variables, *i.e.*, variables of numerical and categorical nature, can be challenging to solve, especially in the presence of complex constraints. Moreover, when the objective function is the result of a complicated simulation or experiment, it may be expensive to evaluate [3]. What are possible ways to solve such black-box global optimization problems while maintaining the integrality of the integer variables and respecting mixed-integer constraints? Can we extend such methods to problems where the objective function can not be assessed, but we can assess the performance based on pairwise comparisons?

As we will demonstrate in this thesis, the aforementioned research questions can be tackled using surrogate-based optimization methods for global and preference-based optimization problems. In this thesis, we show the formulation of the optimization problems and their solution methods, which are organized as follows:

- In Chapter 2, we present a general literature review and some background information for the subjects of global, black-box, preference-based, and mixed-variable black-box optimization, and model predictive control.

- In Chapter 3, we address Question 1, where we suggest a semi-automated calibration approach that requires instead a human calibrator to express a *preference* on whether a certain control policy is "better" than another one, therefore eliminating the need of an explicit performance index. In particular, we focus our attention on semi-automated calibration of *Model Predictive Controllers* (MPCs), for which we attempt computing the set of best calibration parameters by employing the recently-developed active preference-based optimization algorithm GLISp. Based on the preferences expressed by the human operator, GLISp learns a surrogate of the underlying closed-loop performance index that the calibrator (unconsciously) uses and proposes, iteratively, a new set of calibration parameters to him or her for testing and for comparison against previous experimental results. The resulting semi-automated calibration procedure is tested on two case studies, showing the capabilities of the approach in achieving near-optimal performance within a limited number of experiments.

The content of this Chapter and this abstract are from [1]:

> ©2021 EUCA. Reprinted, with permission, from M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," in *European Control Conference*, Rotterdam, Netherlands, 2021, pp. 638–645.

- In Chapter 4, we address Question 2, where we extend an active preference learning algorithm introduced recently to handle unknown constraints. The proposed method, called C-GLISp, looks for an optimizer of the problem only based on *preferences* expressed on pairs of candidate vectors, and on whether a given vector is reported *feasible* and/or *satisfactory*. C-GLISp learns a surrogate of the underlying objective function based on the expressed preferences, and a surrogate of the probability that a sample is feasible and/or satisfactory based on whether each of the tested vectors was judged as such. The surrogate functions are used to propose a new candidate vector for testing and assessment iteratively. Numerical benchmarks and a semi-automated control calibration task demonstrate the effectiveness of C-GLISp, showing that it can reach near-optimal solutions within a small number of iterations.

  The content of this Chapter and this abstract are from [2]

  > ©2021 IEEE. Reprinted, with permission, from M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," in *IEEE Trans. on*

> *Control Systems Technology*, vol. 30, no. 5, pp. 2176–2187, 2022.

- In Chapter 5, we address Question 3, where we propose a novel surrogate-based global optimization algorithm to solve linearly constrained mixed-variable problems up to medium-size (around 100 variables after encoding and 20 constraints) based on constructing a piecewise affine surrogate of the objective function over feasible samples. We introduce two types of exploration functions to efficiently search the feasible domain via mixed-integer linear programming solvers. We also provide a preference-based version of the algorithm, which can be used when only pairwise comparisons between samples can be acquired while the underlying objective function to minimize remains unquantified. The two algorithms are tested on mixed-variable benchmark problems with and without constraints. The results show that, within a small number of acquisitions, the proposed algorithms can often achieve better or comparable results than other existing methods.

  The content of this Chapter and this abstract are reprinted from [3]:

  > M. Zhu and A. Bemporad, "Global and preference-based optimization with mixed variables using piecewise affine surrogates," *submitted for publication*, 2023.

- In Chapter 6, we apply PWAS, the algorithm developed in Chapter 5, to experimental design problems. Specifically, we focus on three case studies, each with a different size of design space and numerical complexity: i) optimization of reaction conditions for Suzuki–Miyaura cross-coupling (fully categorical), ii) optimization of crossed-barrel design to augment mechanical toughness (mixed-integer), and iii) solvent design for enhanced Menschutkin reaction kinetics (mixed-integer and categorical with linear constraints). By comparing with conventional optimization algorithms, we offer insights into the practical applicability of *PWAS*.

  The content of this Chapter and this abstract are reprinted from [4]:

M. Zhu, A. Mroz, L. Gui, K. Jelfs, A. Bemporad, EA. del Río Chanona, and Y. Lee, “Discrete and mixed-variable experimental design with surrogate-based approach“, *submitted for publication*, 2024.

- In Chapter 7, we provide some concluding remarks on the presented materials in this thesis and some open problems for future research.

Chapter 2

Preliminaries

In this chapter, we will present a general literature review and some background information that will serve as the foundation for the following chapters. Specifically, we will delve into the following subjects: global, black-box, preference-based, and mixed-variable derivative-free optimization, along with model predictive control.

2.1 Global optimization

Global optimization methods aim to find the global optimum of a given objective function within a known feasible region. It is widely used in different real-world applications such as portfolio optimization in finance[5], experimental design in engineering[6], and scheduling in operations management [7–9].

The general formulation of global optimization problems is stated as follows [1]:

$$\begin{aligned} \text{find } x^* \in \quad & \arg\min_x f(x) \\ \text{s.t.} \quad & \ell \le x \le u \\ & x \in \mathcal{X} \end{aligned} \tag{2.1}$$

[1]In this thesis, we define optimization problems as minimization problems. However, one can always translate these problems into maximization by defining the objective function as $-f(x)$.

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ models some quantity that we want to minimize. The optimization vector $x \in \mathbb{R}^n$ is the vector of optimization variables x_i that we aim to identify optimally. We assume x is always bounded, and the upper (u) and lower (ℓ) bounds and other arbitrary constraints ($\mathcal{X}$) define the feasible region of x, which we denote as D in the sequel. The global and local optimizers are defined as follows:

- A vector $x^* \in \mathbb{R}^n$ is a global optimizer of (2.1), if $x^* \in D$ and $f(x) \geq f(x^*), \forall x \in D$
- A vector $x^* \in \mathbb{R}^n$ is a local optimizer of (2.1), if $x^* \in D$ and there exists a neighborhood $\mathcal{N}$ of x^* such that $f(x) \geq f(x^*), \forall x \in D \cap \mathcal{N}$

Different approaches can be applied to solve (2.1). In the following, we provide a summary for some common methods.

- **Exhaustive search methods**, such as grid search [10] and random search [11], involve enumerating all possible solutions within a given domain. Therefore it can guarantee that the global minimum in (2.1) is achieved if the search space is discrete and finite. However, this approach can be computationally inefficient, especially for problems with high dimensions and/or with continuous search space.
- **Gradient-based approaches**, such as Newton's method and steepest descent method, often rely on smoothness and convexity assumptions of (2.1), limiting their applicability to find global optimum for non-convex problems [12, 13].
- **Probabilistic and population-based search methods**, such as simulated annealing [14], genetic algorithms [15], and particle swarm optimization [16], search the feasible region by trading off between exploration and exploitation, making them suitable for global optimization problems with non-convex and multimodal objective functions. However, tuning is often required for the hyperparameters involved in the algorithm and the algorithms can be slow to converge for complex problems [17].

- **Surrogate-based methods** (also often referred to as response surface methods) use surrogate models, such as kriging [18] and radial basis functions [19], to approximate the objective function. These methods can be helpful when the objective function is expensive to evaluate. In this case, using surrogates to guide the search process can reduce the number of function evaluations. On the other hand, surrogate-based methods are often not suitable for problems with high dimensions, and the design of suitable surrogate models and acquisition functions can be challenging.

In this thesis, our primary focus is on global optimization methods suitable for addressing complex problems involving simulation or physical experiments. These problems typically involve finding the global optimum without a known analytic expression for the objective function (*i.e.*, the expression of $f(x)$ in (2.1)). However, we can often measure the objective function given an input decision vector. These types of optimization problems are commonly referred to as black-box optimization problems. Evaluating the black-box input/output function can be computationally expensive. Hence, it is crucial to minimize the number of function evaluations required to find a near-optimal solution. Additionally, due to the lack of expression of the objective function, methods that require derivative information can not be applied directly.

2.2 Black-box optimization

In this section, we briefly discuss optimization methods suitable to solve black-box optimization problems.

- **Direct search methods** [20], such as pattern search [21], coordinate search [22], Mesh Adaptive Direct-Search [23, 24], Nelder-Mead [25], and Hooke and Jeeves [26] algorithms, iteratively probe and evaluate the objective function without explicitly using its derivatives or assuming of any specific structure. These methods are generally robust and are especially useful when the objective function is non-differentiable (or when its gradient information is unavailable), discontinuous, or noisy. However, they can be computationally expensive since they rely on a large number of function evaluations.

- **Surrogate-based methods** are often coupled with active learning strategies when dealing with black-box optimization problems. For instance, Bayesian Optimization (BO) [27] is the widely used nowadays.

- **Probabilistic and population-based search methods**: see the discussion in Section 2.1.

- **Metaheuristic algorithms** [28], such as Ant colony optimization [29], Harmony search [30], and Firefly algorithm [31], are optimization techniques inspired by natural phenomena or problem-solving heuristics. These methods often require no gradient information, tend to explore the search space more globally, and can be parallelized. On the other hand, they often require a large number of function evaluations (high computational cost), lack of interpretability of the solution, and require parameter tunning.

This thesis concentrates on black-box optimization problems with expensive-to-evaluate objective functions. As a result, we focus on employing surrogate-based methods. In the following, we summarize their general procedures (see, *e.g.*, [32]), which are also schematically summarized in Figure 2.1 [3].

1. Define the *objective function* to optimize (*e.g.*, the performance of a simulation or a real-world problem). It needs to provide the output based on an input decision vector, but it does not necessarily require an explicit analytic expression.

2. Select a *surrogate model* to approximate the behavior of the objective function (*e.g.*, Gaussian processes [33], radial basis functions [19, 32], *etc.*.). One may select the model based on the prior knowledge available and the characteristics of the problems (deterministic/stochastic).

3. Select an *exploration model* (*e.g.*, space-filling methods, *i.e.*, disperse points to promote a broad coverage across the domain [32], probability of improvement [33], *etc.*.). An appropriate exploration model encourages exploration in unvisited feasible regions, aiming to decrease uncertainties in the surrogate model and avoid getting stuck in local optima.

4. Generate an initial set of samples using some *initial sampling strategies* (*e.g.*, Latin hypercube sampling). For sample efficiency, it is important to ensure diversity and coverage across a wide range of input space.

5. Evaluate the initial samples.

6. Build the surrogate model by training it with the initial samples and their corresponding function evaluations.

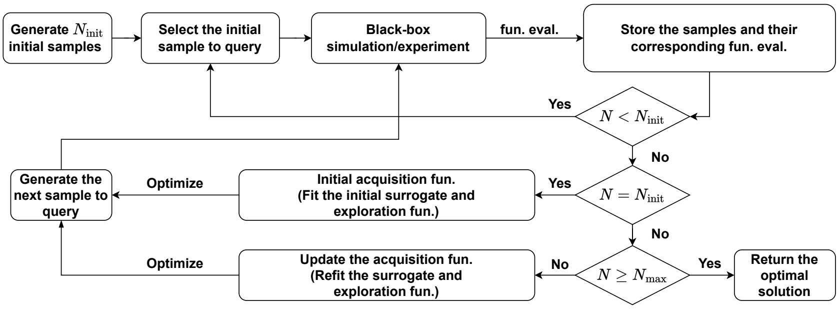
7. Select the next sample to test by optimizing the *acquisition function*, which trades off between the exploitation of the objective function predictions based on the surrogate model and the exploration of the input space (prediction uncertainty/improvement/diversity) based on the exploration model.

8. Evaluate the objective function with the newly queried sample from step 7.

9. Update the surrogate model by incorporating the new sample and its function evaluation. The predictability of the surrogate model is improved by iteratively updating the surrogate model throughout the optimization procedure,

10. Repeat steps 7 to 9 until a stopping criterion is met (*e.g.*, a maximum number of function evaluations, a converge threshold, *etc.*).

In general, measurements of the objective function at sampled points are required for surrogate-based optimization methods to construct the surrogate model. However, certain application domains pose challenges



Generate N_{init} initial samples
Select the initial sample to query
Black-box simulation/experiment
fun. eval.
Store the samples and their corresponding fun. eval.
Yes
$N < N_{\text{init}}$
No
Generate the next sample to query
Optimize
Initial acquisition fun. (Fit the initial surrogate and exploration fun.)
Yes
$N = N_{\text{init}}$
No
Optimize
Update the acquisition fun. (Refit the surrogate and exploration fun.)
No
$N \geq N_{\text{max}}$
Yes
Return the optimal solution

Figure 2.1: General procedures for surrogate-based optimization methods 2.1 [3].

where the objective function may not be quantifiable (*e.g.*, qualitative descriptions) or involve multiple objectives without pre-determined relative importance [34]. Conversely, an experienced human decision-maker can easily evaluate the performance of the optimization outcome and expresses his/her preferences through pairwise comparisons. When optimization methods rely on preference information instead of function evaluations, they are referred to as preference-based optimization methods in this thesis.

2.3 Preference-based optimization

In this section, we start with an overview of preference-based optimization methods, followed by defining the specific problem we address in this thesis. Finally, we summarize GLISp, the optimization method that this thesis is built upon.

In the context of preference-based optimization, various methods have been developed to handle optimization problems where the objective function is not explicitly defined, not quantifiable, or difficult to evaluate [33, 35–38]. These methods aim to leverage the knowledge of preferences provided by the user to guide the optimization process. These approaches are often interactive and involve an iterative process of incorporating user preferences into the optimization process. They typically rely on preference elicitation techniques [39], such as direct assessment (see *e.g.*, [40]), pairwise comparisons (see *e.g.*, [1, 38]), or interactive visualizations (see *e.g.*, [33, 41]), to guide the search for preferred solutions.

Different approaches are used to model the preferences, which can be generally classified into three categories [37]: learning utility functions [42–44], learning preference relations [45], and function approximation [46].

In this thesis, our main focus lies on preference-based optimization methods that learn a surrogate model respecting the preferences expressed by the decision-maker based on *pairwise comparisons*. The surrogate, once learned, is used to guide the search toward the decision vector that leads to the most preferable outcome [38]. This fall into the category of *utility function learning*, where the preferences are considered *constraints* for the space of utility functions [37, 42–44]. In the following, we provide the general formulation for preference-based optimization problems.

Different from the problem formulation in (2.1), for preference-based optimization, the latent objective function $f(x)$ is unknown/immeasurable. However, we assume that we can always express a preference between two choices. Formally, given two decision vectors x_1 and x_2, we define the *preference function* $\pi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \{-1, 0, 1\}$ as

$$\pi(x_1, x_2) = \begin{cases} -1 & \text{if } x_1 \text{ ``better'' than } x_2 \\ 0 & \text{if } x_1 \text{ ``as good as'' } x_2 \\ 1 & \text{if } x_2 \text{ ``better'' than } x_1. \end{cases} \tag{2.2}$$

And the following properties holds for all $x_1, x_2 \in \mathbb{R}^n$ [38, 47]:

$$\pi(x_1, x_1) = 0, \quad \pi(x_1, x_2) = -\pi(x_2, x_1)$$
$$\pi(x_1, x_2) = \pi(x_2, x_3) = -1 \Rightarrow \pi(x_1, x_3) = -1$$

The modeling assumption [38] is that the decision-maker assigns preferences based on an underlying latent function f (cf. formulation (2.1)) in his/her mind that he or she wants to minimize:

$$\pi(x_1, x_2) = \begin{cases} -1 & \text{if } f(x_1) < f(x_2) \\ 0 & \text{if } f(x_1) = f(x_2) \\ 1 & \text{if } f(x_1) > f(x_2). \end{cases} \tag{2.3}$$

Here, the latent function f is unknown to the optimization algorithm and the decision-maker. We can only obtain the preference expressed by the decision-maker, *i.e.*, the evaluation of the preference function π.

The goal is to find the optimal decision vector $x^* \in D$ such that x^* is "better" (or "not worse") than any other feasible x according to the preference function π, which translates to the following mathematical formulation

$$\text{Find } x^\star \text{ such that } \pi(x^\star, x) \leq 0, \ \forall x \in D. \tag{2.4}$$

Considering the relationship between the expressed preference and the underlying latent function as noted in (2.3), optimization problem (2.4) implies $f(x^*) \leq f(x)$, $\forall x \in D$, which indicates that x^* is a global minimizer of f on D.

2.3.1 GLISp

This thesis builds upon the preference-based optimization method GLISp [38]. In the following, we note the general steps of GLISp in solving preference-based optimization problem (2.4), and refer readers to [38] for detailed descriptions [2].

2.3.1.1 Training a surrogate function from preferences

Assume that we have generated $N \geq 2$ samples $\{x_1 \ \dots \ x_N\}$ of the optimization vector, with $x_i, x_j \in \mathbb{R}^{n_x}$ such that $x_i \neq x_j$, $\forall i \neq j$, $i, j = 1, \dots, N$. For each of these vectors, an experiment/simulation is performed and the decision-maker has provided a *preference vector* $B = [b_1 \ \dots \ b_M]^T \in \{-1, 0, 1\}^M$ with

$$b_h = \pi(x_{i(h)}, x_{j(h)}), \tag{2.5}$$

where M is the number of expressed preferences, $1 \leq M \leq \binom{N}{2}$, $h \in \{1, \dots, M\}$, $i(h), j(h) \in \{1, \dots, N\}$, $i(h) \neq j(h)$. Note that the element

[2]The content of this subsection is is based on [38] and is partially reprinted from ©2021 EUCA. Reprinted, with permission, from M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," in *European Control Conference*, Rotterdam, Netherlands, 2021, pp. 638–645.

b_h of vector B represents the preference expressed by the decision-maker between the experimental/simulation performance achieved with vector $x_{i(h)}$ and $x_{j(h)}$.

The observed preferences are then used to learn a surrogate function $\hat{f} : \mathbb{R}^{n_x} \to \mathbb{R}$ of the underlying latent function f. The surrogate $\hat{f}$ is constructed by imposing the constraints

$$\hat{\pi}(x_{i(h)}, x_{j(h)}) = \pi(x_{i(h)}, x_{j(h)}),\ \forall h = 1, \dots, M, \tag{2.6}$$

on $\hat{f}$, where $\hat{\pi}$ is defined from $\hat{f}$ as in (2.3).

The function $\hat{f}$ is parametrized as the following linear combination of Radial Basis Functions (RBFs) [48, 49]:

$$\hat{f}(x) = \sum_{k=1}^{N} \beta_k \phi(\epsilon d(x, x_i)), \tag{2.7}$$

where $d : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}$ is the squared Euclidean distance

$$d(x_1, x_2) = \|x_1 - x_2\|_2^2, \tag{2.8}$$

$\epsilon > 0$ is a scalar parameter, $\phi : \mathbb{R} \to \mathbb{R}$ is an RBF, and $\beta = [\beta_1\ \dots\ \beta_N]^T$ are the unknown coefficients to be computed based on the preferences imposed in (2.6). Examples of RBFs are $\phi(\epsilon d) = \frac{1}{1+(\epsilon d)^2}$ (*inverse quadratic*), $\phi(\epsilon d) = e^{-(\epsilon d)^2}$ (*Gaussian*), $\phi(\epsilon d) = (\epsilon d)^2 \log(\epsilon d)$ (*thin plate spline*), see more examples in [32, 48].

According to (2.6) and the preference relation (2.3), the following constraints are imposed on $\hat{f}$:

$$\begin{array}{ll} \hat{f}(x_{i(h)}) \le \hat{f}(x_{j(h)}) - \sigma + \varepsilon_h & \text{if } \pi(x_{i(h)}, x_{j(h)}) = -1 \\ \hat{f}(x_{i(h)}) \ge \hat{f}(x_{j(h)}) + \sigma - \varepsilon_h & \text{if } \pi(x_{i(h)}, x_{j(h)}) = 1 \\ |\hat{f}(x_{i(h)}) - \hat{f}(x_{j(h)})| \le \sigma + \varepsilon_h & \text{if } \pi(x_{i(h)}, x_{j(h)}) = 0 \end{array} \tag{2.9}$$

for all $h = 1, \dots, M$, where $\sigma > 0$ is a given tolerance and ε_h are positive slack variables.

Accordingly, and similarly to Support Vector Machines (SVMs) [50], the coefficient vector β describing the surrogate $\hat{f}$ is obtained by solving the convex Quadratic Programming (QP) problem

$$
\begin{array}{rl}
\min_{\beta,\varepsilon} & \displaystyle\sum_{h=1}^{M} c_h\varepsilon_h + \frac{\lambda}{2}\sum_{k=1}^{N}\beta_k^2 \\
\text{s.t.} & \displaystyle\sum_{k=1}^{N}(\phi(\epsilon d(x_{i(h)}, x_k) - \phi(\epsilon d(x_{j(h)}, x_k))\beta_k \\
& \quad \leq -\sigma + \varepsilon_h, \qquad\qquad \forall h:\ b_h = -1 \\
& \displaystyle\sum_{k=1}^{N}(\phi(\epsilon d(x_{i(h)}, x_k) - \phi(\epsilon d(x_{j(h)}, x_k))\beta_k \\
& \quad \geq \sigma - \varepsilon_h, \qquad\qquad \forall h:\ b_h = 1 \\
& \displaystyle\left|\sum_{k=1}^{N}(\phi(\epsilon d(x_{i(h)}, x_k) - \phi(\epsilon d(x_{j(h)}, x_k))\beta_k\right| \\
& \quad \leq \sigma + \varepsilon_h, \qquad\qquad \forall h:\ b_h = 0 \\
& h = 1, \dots, M
\end{array}
\tag{2.10}
$$

where c_h are positive weights, for example $c_h = 1$, $\forall h = 1, \dots, M$. The slack variables ε_h in (2.10) are used to relax the constraints imposed by the preference vector B in (2.6). Constraint infeasibility might be due to an inappropriate selection of the RBF (namely, poor flexibility in the parametric description of the surrogate $\hat{f}$) and/or to outliers in the acquired preferences, for instance, due to inconsistent assessments done by the decision-maker. The scalar λ in the cost function (2.10) is a regularization parameter. When $\lambda > 0$, problem (2.10) is a QP problem that admits a unique solution because $c_h > 0$ for all $h = 1, \dots, M$. If $\lambda = 0$, problem (2.10) becomes a Linear Program (LP), whose solution may not be unique.

We remark that computing the surrogate function $\hat{f}$ requires one to choose the hyper-parameter ϵ defining the shape of the RBFs ϕ in (2.7). This parameter can be chosen through K-fold cross-validation [51], by testing the capabilities of $\hat{f}$ in reconstructing the preferences in slices of the dataset not used to estimate $\hat{f}$.

2.3.1.2 Acquisition function

Once a surrogate $\hat{f}$ is estimated, this function can be in principle minimized in order to find the optimal vector x. More specifically, the following steps can be followed: (i) generate a new sample by pure minimization of the estimated surrogate function $\hat{f}$ defined in (2.7), *i.e.*,

$$x_{N+1} = \arg\min \hat{f}(x) \text{ s.t. } x \in \mathcal{X};$$

(ii) ask the decision-maker to evaluate the preference $\pi(x_{N+1}, x_N^\star)$, where $x_N^\star \in \mathbb{R}^{n_x}$ is the best vector of optimization variables found so far, corresponding to the smallest index $i^\star$ such that

$$\pi(x_{i^\star}, x_i) \leq 0, \; \forall i = 1, \ldots, N; \tag{2.11}$$

(iii) update the estimate of $\hat{f}$ through (2.10); and (iv) iterate over N.

Such a procedure, which only *exploits* the current available observations in finding the optimal vector x, may easily miss the global minimum of (2.4). Therefore, looking only at the surrogate function $\hat{f}$ is not enough to search for a new sample x_{N+1}. A term promoting the *exploration* of the parameter space should thus be considered.

In GLISp, an acquisition function is employed to balance exploitation *vs.* exploration when generating the new sample x_{N+1}. As proposed in [32], the exploration function is constructed by using the inverse distance weighting (IDW) function $z : \mathbb{R}^{n_x} \to \mathbb{R}$ defined by

$$z(x) = \begin{cases} 0 & \text{if } x \in \{x_1, \ldots, x_N\} \\ \tan^{-1}\left(\frac{1}{\sum_{i=1}^N w_i(x)}\right) & \text{otherwise} \end{cases} \tag{2.12}$$

where $w_i(x) = \frac{1}{d^2(x, x_i)}$. Clearly $z(x) = 0$ for all parameters already tested, and $z(x) > 0$ in $\mathbb{R}^{n_x} \setminus \{x_1, \ldots, x_N\}$. The arc tangent function in (2.12) avoids that $z(x)$ gets excessively large far away from all sampled points.

Then, given an exploration parameter $\delta \geq 0$, the *acquisition function* $a : \mathbb{R}^{n_x} \to \mathbb{R}$ is constructed as

$$a(x) = \frac{\hat{f}(x)}{\Delta \hat{f}} - \delta z(x), \tag{2.13}$$

where

$$\Delta \hat{f} = \max_i \{\hat{f}(x_i)\} - \min_i \{\hat{f}(x_i)\}$$

is the range of the surrogate function on the samples in $\{x_1, \ldots, x_N\}$ and is used in (2.13) as a normalization factor to simplify the choice of the exploration parameter δ. Clearly $\Delta\hat{f} \geq \sigma$ if at least one comparison $b_h = \pi(x_{i(h)}, x_{j(h)}) \neq 0$.

As discussed below, given a set $\{x_1, \ldots, x_N\}$ of samples and a vector B of preferences defined by (2.5), the next x_{N+1} to test is computed as the solution of the (non-convex) optimization problem

$$x_{N+1} = \arg\min_{x \in \mathcal{X}} a(x). \tag{2.14}$$

Different optimization algorithms can be used to solve problem (2.14). For instance, the methods discussed in Sections 2.1 and 2.2. Note that the construction of the acquisition function a is rather heuristic, therefore finding highly accurate solutions of (2.14) is not required.

In the acquisition function (2.13), the exploration parameter δ promotes sampling the space in $\mathcal{X}$ in areas that have not been explored yet. While, as observed earlier, $\delta = 0$ can make the GLISp algorithm rely only on the surrogate function $\hat{f}$ and miss the global optimum, setting $\delta \gg 1$ makes the GLISp algorithm exploring the entire feasible region regardless of the results of the comparisons. For a sensitivity analysis example of GLISp with respect to δ, the reader is referred to [38, Section 7.5]. Regarding the other main hyper-parameter ϵ of GLISp defining the RBF in (2.7), during the active learning phase K-fold cross-validation is executed repeatedly to automatically choose and possibly adapt ϵ.

As per discussed, most surrogate-based optimization methods that tackle black-box and preference-based optimization problems focus on the input space that consists solely of continuous variables. On the other hand, many optimization problems in real-world applications involve mixed variables, *i.e.*, continuous, integer, and categorical variables. Moreover, mixed-variable constraints are often present in such applications. Therefore, it is important to suggest feasible samples to test, leading to mixed-variable derivative-free optimization problems.

2.4 Mixed-variable black-box optimization

In this section, we discuss some common algorithms in the literature that tackles black-box optimization problems with mixed variables. We refer the reader to [52] for a comprehensive review and comparison of algorithms and software for mixed-integer derivative-free optimization. In the following, we list out some main algorithms categorized based on their characteristics.

- **Surrogate-based methods** (see Section 2.1 for a general discussion): Mixed-Integer Surrogate Optimization (MISO) [53], Sequential Model-based Algorithm Configuration (SMAC) [54], Tree-structured Parzen Estimator (TPE) [55]
- **Direct search methods** (see Section 2.2 for a general discussion): Nonlinear Optimization with the Mesh Adaptive Direct search (NOMAD) [56], Brute Force Optimizer (BFO) [57], Design Analysis Kit for Optimization and Terascale Applications (DAKOTA) [58]
- **Metaheuristic algorithms** (see Section 2.2 for a general discussion): Mixed Integer Distributed Ant Colony Optimization (MIDACO) [59]

2.4.1 PARC

The mixed-variable black-box optimization method developed in Chapter 5 of this thesis uses PARC (Piecewise Affine Regression and Classification) [60] to fit the surrogate function. In the following, we provide an overview of PARC and refer readers to [60] for detailed analysis [3].

The general procedures of PARC are illustrated in Fig. 2.2. PARC is a block descent algorithm, where it first groups samples in K_{init} clusters. Clusters containing fewer samples than a predefined minimum

[3]The content of this subsection is based on [60] and is partially reprinted from M. Zhu, A. Mroz, L. Gui, K. Jelfs, A. Bemporad, EA. del Río Chanona, and Y. Lee, "Discrete and mixed-variable experimental design with surrogate-based approach", *submitted for publication*, 2024 and M. Zhu and A. Bemporad, "Global and preference-based optimization with mixed variables using piecewise affine surrogates," *submitted for publication*, 2023.

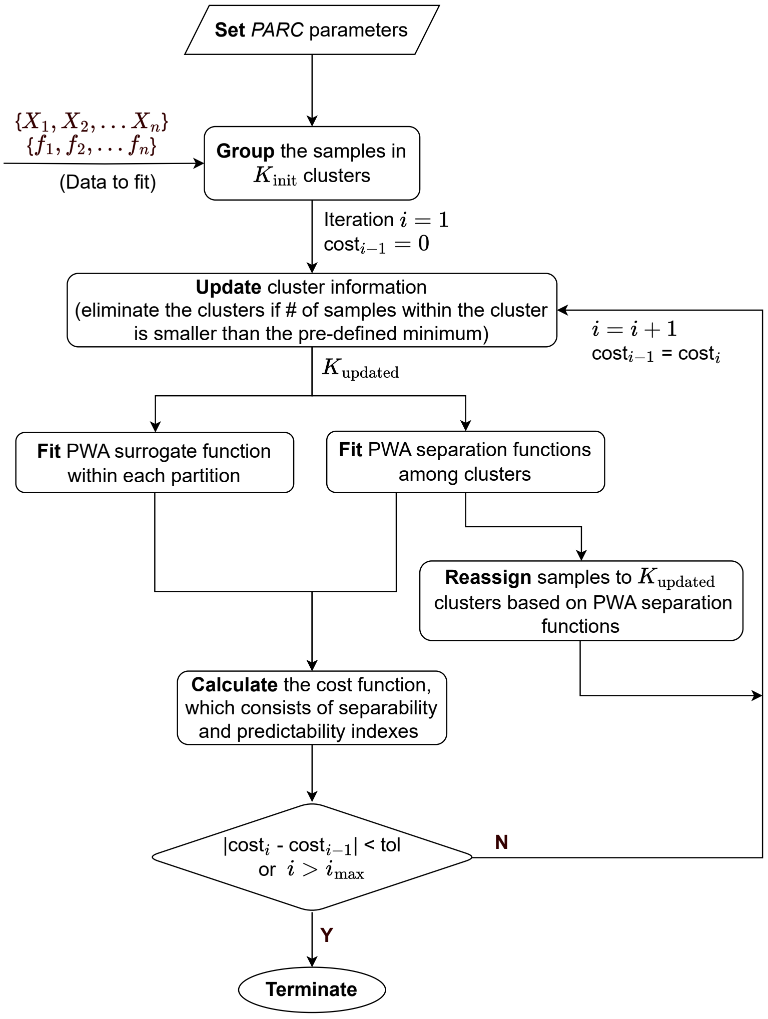

Set *PARC* parameters
$\{X_1, X_2, \dots X_n\}$
$\{f_1, f_2, \cdots f_n\}$
(Data to fit)
Group the samples in K_{init} clusters
Iteration $i = 1$
cost$_{i-1} = 0$
Update cluster information (eliminate the clusters if # of samples within the cluster is smaller than the pre-defined minimum)
$i = i + 1$
cost$_{i-1}$ = cost$_i$
K_{updated}
Fit PWA surrogate function within each partition
Fit PWA separation functions among clusters
Reassign samples to K_{updated} clusters based on PWA separation functions
Calculate the cost function, which consists of separability and predictability indexes
\|cost$_i$ - cost$_{i-1}$\| < tol or $i > i_{\text{max}}$
N
Y
Terminate

Figure 2.2: The flowchart of PARC.

threshold are discarded, resulting in K_{updated} remaining clusters. We then fit PWA (Piecewise affine) separation functions among these clusters to form K_{updated} partitions of the design space. Within each partition, a PWA surrogate function is fitted to make predictions. A cost function comprising separability and predictability indexes is then calculated, which balances between the enhancement of separability among different partitions and the improvement of predictability within each partition. PARC terminates when either the difference in the value of the cost function between two consecutive evaluations falls below a predefined tolerance, or the number of iterations surpasses a predetermined maximum limit. Otherwise, PARC reassigns samples to K_{updated} clusters based on the newly fitted PWA separation functions and then iterates the procedure until termination criteria are met.

The number K_{init} of partitions is a hyper-parameter of PARC that must be selected by trading off between having a more flexible surrogate function (large K_{init}) and limit computations (small K_{init}). As noted in Fig. 2.2, PARC can adaptively update K_{init} during surrogate fitting, which makes it more flexible and robust when fitting the surrogate, and can help reduce computational complexities without significantly compromising prediction accuracy. While we refer the reader to [60] for a detailed analysis of the PARC algorithm, for mere illustration here we show two surrogate-fitting examples: one in the continuous domain, and another in the mixed continuous and categorical domain. We note that problems in 2D are selected for demonstration purposes, and PARC can handle problems with high dimensions.

For the continuous function, we consider the Branin function [61]:

$$\begin{aligned} f(x_1, x_2) &= a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s \\ &\quad a = 1,\ b = \frac{5.1}{4\pi^2},\ c = \frac{5}{\pi},\ r = 6,\ s = 10,\ t = \frac{1}{8\pi} \\ &\quad -5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15. \end{aligned} \tag{2.15}$$

We use 800 randomly generated training samples (x_{1k}, x_{2k}) to fit a PWA surrogate of f in (2.15) by using the PARC algorithm with the initial partition $K_{\text{init}} = 10$. Figure 2.3 shows the final polyhedral partition and Figure 2.4 the PWA surrogate of the Branin function fit by PARC. We observe that the surrogate fitted by PARC captures the general shape of the nonlinear Branin function.

For the mixed continuous and categorical domain, we consider the

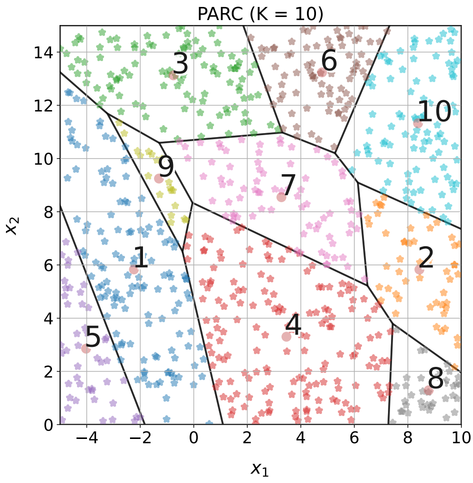
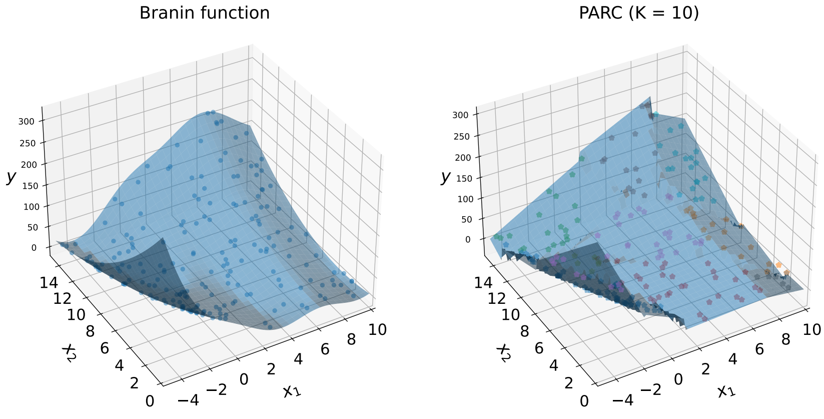


Figure 2.3: The polyhedral partition with $K_{\text{init}} = 10$ initial partitions. Note that the final partition is also 10 (no partition is discarded). The dots in the figure are the training data for the PARC algorithm.



(a) Analytical **(b)** PARC with $K_{\text{init}} = 10$

Figure 2.4: Branin function fitted analytically and by PARC. The dots in the figure are the test data (200 samples) for the PARC algorithm.

following synthetic function:

$$f(x_1, x_2) = \begin{cases} x_1^2 + 2x_1 + 1 & x_2 = 0 \\ x_1 + 100 & x_2 = 1 \\ (1 - x_1)^3 & x_2 = 2 \end{cases} \tag{2.16}$$
$$-5 \leq x_1 \leq 5,\ x_2 \in \{0, 1, 2\},$$

where with different categorical values of x_2, function evaluations ($f(x_1, x_2)$) can vary significantly. We use 960 randomly generated training samples with 10 initial partitions (K_{init}) to fit the PWA surrogates. Figure 2.5 shows the final polyhedral partition. In this case, we observe that 2 initial partitions were discarded during surrogate fitting, resulting in 8 remaining partitions. In Figure 2.6, we show the function (2.16) fitted analytically, and fitted by PARC with $K_{\text{updated}} = 8$ partitions, where we observe that PARC can make good predictions at different values of the categorical variable (x_2), despite their distinct characteristics.

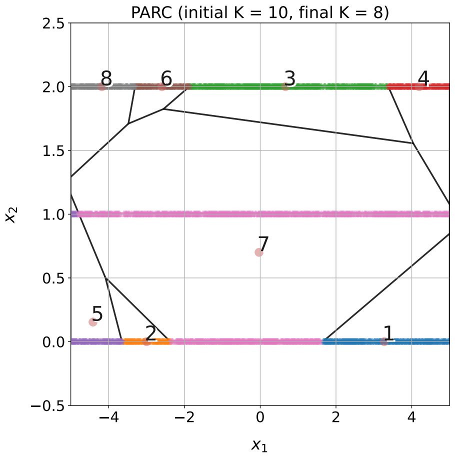
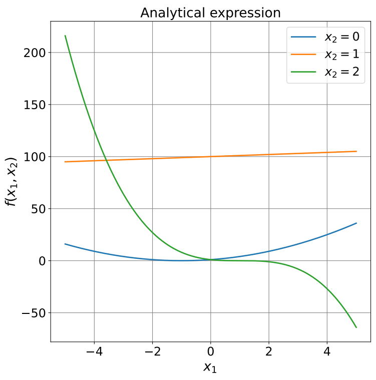
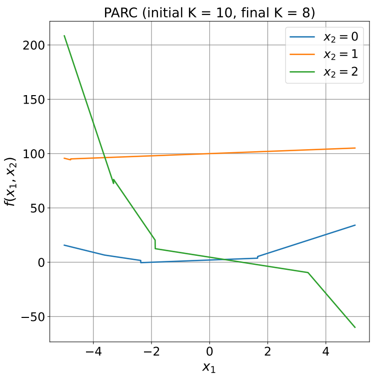


Figure 2.5: The polyhedral partition with $K_{\text{init}} = 10$ initial partitions. Note that the final partition is 8 (2 partitions are discarded). The dots in the figure are the training data for the PARC algorithm.



(a) Analytical



(b) PARC with $K_{\text{init}} = 8$ final partitions

Figure 2.6: Function (2.16) fitted analytically and by PARC.

2.5 Model predictive control

In later chapters of this thesis, we will apply optimization methods to solve problems in control applications, specifically in model predictive control (MPC). In this section, we provide an overview of MPC. MPC is an advanced model-based control strategy used to optimize the performance of dynamic systems in engineering and industrial applications. It chooses a control action based on predictions of the system's future behavior made using a simplified dynamical model of the system [62, 63]. In the following, we summarize the main elements and key steps involved in MPC [62, 63]:

- MPC requires a *dynamical model* of the system to be controlled. It describes the system dynamics, constraints, and input-output relationships, which will be used to predict how the state variables of the system evolve over time. The model can be obtained based on prior knowledge of the system (first principle models) or through system identification techniques [64–66]. In the following, we discuss briefly about *Linear parameter-varying (LPV)* [67–69] and *Linear*

Time-varying (LTV) [68, 70] models.

  - The LPV model characterizes the system's evolution over time by accounting for parameter values at each time step. By incorporating parameter variations, LPV models can effectively capture the time-varying dynamics of the system, leading to more precise representations and control designs. This modeling approach is particularly beneficial when systems operate under changing conditions or when uncertainties exist regarding the parameters involved.
  - The LTV model characterizes dynamic changes in the system's behavior over the prediction horizon. These changes may arise from various factors, including time-varying inputs, varying operating conditions, or evolving system parameters. Moreover, it is possible to embed measured disturbances in the model. The flexibility of the LTV model allows for a more accurate depiction of the system's temporal dynamics, enabling a better understanding and analysis of its behavior under changing circumstances.

- MPC operates by adopting a receding horizon approach, where it forecasts the future behavior of the system within a specified finite time horizon, namely *prediction horizon*, by recursively utilizing the system model. The prediction horizon is divided into several discrete time steps.

- MPC also considers a *control horizon*, which determines the duration during which control actions are computed and implemented on the system. Generally, the control horizon is shorter than the prediction horizon.

- MPC utilizes a *cost function* that quantifies the performance of the system under control and represents the control objectives. The cost function is formulated based on the system's state variables and control inputs. It can comprise various terms such as setpoint tracking, control effort (aggressiveness), and constraints. The design of the cost function often requires a careful balance among conflicting objectives whose relative importance are often accounted for using different weight factors.

- MPC can handle both hard (*e.g.*, physical limitations, and safety specifications) and soft (*e.g.*, user comfort) *constraints*, which is one of its main advantages.

- The control problem in MPC is formulated as an *optimization problem*, where the goal is to find the optimal control sequence over the prediction horizon that minimizes the cost function while satisfying the constraints. Depending on the characteristics of the cost function, different optimization algorithms can be employed.

- Once the optimization problem is solved, only the *first control action* of the optimal control sequence is implemented in the system. The system evolves. Then the process is repeated at the next time step, where the optimization problem is solved again with updated measurements and predictions from previous time steps.

- MPC inherently incorporates *feedback* control by continuously updating the system model and recalculating control actions based on current measurements. This adaptive nature enables MPC to effectively handle uncertainties, disturbances, and variations in system dynamics that may arise during operation. By leveraging real-time information, MPC can dynamically adjust its control strategy to maintain system performance and stability.

Implementing an MPC controller entails certain challenges. One such challenge is the tuning of MPC parameters, such as the control and prediction horizons, to achieve optimal performance. These parameters need to be carefully selected based on the system's characteristics and desired control objectives. Additionally, defining the cost function for the MPC can be a nontrivial task. Determining the appropriate weighting of various factors in the cost function, such as setpoint tracking, control effort, and constraints, may require prior knowledge or iterative refinement to strike the right balance for effective control. In Chapter 3, we will show how alternative cost function can be defined for the MPC and in Chapter 4, we show an MPC calibration example where *unknown* (in terms of explicit analytic expression) but assessable (by the decision-maker) constraints are present.

Chapter 3

Preference-based MPC calibration

3.1 Introduction

The design of *Model Predictive Controllers* (MPC) typically requires the tuning of several parameters such as the prediction and control horizons, the weight matrices defining the cost function, various numerical solver tolerances, *etc*. A calibrator typically adjusts these knobs based on experience and trial-and-error, until the closed-loop system behaves as he or she desires. Such a tuning process thus requires skilled calibrators, domain knowledge, and it can be costly and time consuming.

To automate the tuning process, usually a figure of merit is defined to quantitatively assess closed-loop performance, and experiment-driven optimization algorithms are usually adopted to find near-optimal MPC parameters. In particular, black-box global optimization methods based on surrogate functions, like *Bayesian Optimization* (BO) [33] and the re-

The content of this Chapter is from ©2021 EUCA. Reprinted, with permission, from M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," in *European Control Conference*, Rotterdam, Netherlands, 2021, pp. 638–645. and ©2021 IEEE. Reprinted, with permission, from M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," in *IEEE Trans. on Control Systems Technology*, vol. 30, no. 5, pp. 2176–2187, 2022.

cently introduced GLIS (GLobal optimization based on Inverse distance weighting and radial basis function Surrogates) algorithm [32], have been recently applied for MPC parameter tuning [71, 72], choice of the MPC predictive model [73, 74] and also in other control engineering problems and applications such as PID and state-feedback control tuning [75, 76], position and force control in robot manipulators [77, 78], and control of mobile robots and quadrotors [79, 80], just to cite a few.

The aforementioned optimization approaches iteratively suggest new parameters to be tested based on a surrogate function, which is estimated from closed-loop performance figures gathered from previous experiments. An exploration term takes care of sufficiently covering the set of parameters to search, thus avoiding the solver to be trapped in a local minimum.

However, in order to use such methods for MPC calibration, it is essential to have a well-defined performance index that captures the desired closed-loop behavior of the system. Unfortunately, in many practical control applications a performance is usually formulated based on multiple criteria, and thus it is difficult for a calibrator to formally define and quantify objectively the scoring function. On the other hand, it is usually easier for a calibrator to express a preference (such as "controller A is better than controller B") between the outcome of two experiments.

3.1.1 Contribution

Motivated by the above consideration, in this chapter, we propose a novel calibration approach to tune the MPC parameters based on pairwise preferences between experiment outcomes. The approach is semi-automated in that the control parameters to be tested are selected automatically by an algorithm while closed-loop performance is assessed manually by the calibrator, that therefore is no longer required to formulate a performance index upfront.

The proposed approach for preference-based MPC calibration relies on the derivative-free global optimization algorithm recently developed in [38] (see also in Section 2.3.1), which iteratively proposes a new comparison to the calibrator to make, based on actively learning a surrogate of the latent objective function from past sampled decision vectors and pairwise preferences. Preference-based optimization method is becoming popular in the field of reinforcement learning (RL) [81] and a comprehensive review is presented in the survey paper [37]. Inverse (reinforcement) learning and semi-supervised machine learning approaches are also closely related to preference-based learning, which has been applied to parameter tuning, such as for automated driving [82–84]. The main concern of the inverse-learning approach is that it can not improve upon the demonstrations by the expert. The algorithm implemented in this chapter [38], called GLISp (preference-based GLIS) proposed a different acquisition function form that balances the trade-off between exploitation and exploration. GLISp has been shown to be very efficient in terms of number of experiments and comparisons required to compute the global optimum, which is a major drawback of many preference-based RL methods [37].

We show the efficiency of the proposed preference-based MPC calibration in two case studies. The first one considers the control of a *Continuous Stirring Tank Reactor* (CSTR), while the second one is related to automated driving of a vehicle with obstacle avoidance. In both cases, overall satisfactory performance is achieved within a relatively small number of closed-loop experiments, without the hard and time-consuming need to specify a quantitative scoring function driving a fully-automated MPC calibration.

The rest of this chapter is organized as follows. Section 3.2 describes the MPC calibration problem. Some notes on the preference-based tuning approach based on GLISp is presented in Section 3.3. The application of the proposed approach for two case studies are discussed in Section 3.4. Finally, conclusions are drawn in Section 3.5.

3.2 Problem description

Let us consider the problem of controlling a nonlinear multi-input multi-output system described by the continuous-time state-space representation:

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= g(x,u), \end{aligned} \tag{3.1}$$

where $x \in \mathbb{R}^{n_x}$ and $\dot{x} \in \mathbb{R}^{n_x}$ are the state vector and its time derivative, respectively; $u \in \mathbb{R}^{n_u}$ is the control input; $y \in \mathbb{R}^{n_y}$ is the vector of controlled outputs; and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ are the state and output mappings, respectively.

A popular strategy to achieve a reference-tracking objective for a system described by (3.1) under input and output constraints, is linear-time varying (LTV) MPC, a.k.a. real-time iteration scheme [85, 86]. The MPC is designed based on the following predictive model obtained via linearization of (3.1) around a nominal trajectory $\bar{x}_k$, $\bar{u}_k$, $\bar{y}_k$ and discretization with sampling time T_s, resulting in the prediction model

$$\begin{aligned} \tilde{x}_{k+1} &= A_k \tilde{x}_k + B_k \tilde{u}_k \\ \tilde{y}_k &= C_k \tilde{x}_k + D_k \tilde{u}_k, \end{aligned} \tag{3.2}$$

where subscript k denotes the value at time step k, $\tilde{x}_k = x_k - \bar{x}_k$, $\tilde{u}_k = u_k - \bar{u}_k$ and $\tilde{y}_k = y_k - \bar{y}_k$.

At each sampling time t, the MPC action $u_{t|t}$ to apply to the system is computed by solving the Quadratic Programming (QP) problem:

$$\begin{aligned} \min_{\{u_{t+k|t}\}_{k=0}^{N_u-1},\varepsilon} \; & \sum_{k=0}^{N_p-1} \left\| y_{t+k|t} - y_{t+k}^{\text{ref}} \right\|_{Q_y}^2 \\ & + \sum_{k=0}^{N_p-1} \left\| u_{t+k|t} - u_{t+k}^{\text{ref}} \right\|_{Q_u}^2 \\ & + \sum_{k=0}^{N_p-1} \left\| \Delta u_{t+k|t} \right\|_{Q_{\Delta u}}^2 + Q_\varepsilon \left\| \varepsilon \right\|^2 \end{aligned} \tag{3.3}$$

s.t. model equation (3.2) and the following constraints

$$\begin{aligned}
& y_{\min} - \varepsilon \leq y_{t+k|t} \leq y_{\max} + \varepsilon,\ k = 1, \ldots, N_p \\
& u_{\min} \leq u_{t+k|t} \leq u_{\max},\ k = 1, \ldots, N_p \\
& \Delta u_{\min} \leq \Delta u_{t+k|t},\ \ k = 1, \ldots, N_p \\
& \Delta u_{t+k|t} \leq \Delta u_{\max},\ \ k = 1, \ldots, N_p \\
& u_{t+N_u+j|t} = u_{t+N_u|t},\ \ j = 1, \ldots, N_p - N_u,
\end{aligned} \tag{3.4}$$

where $\|\nu\|_Q^2$ is the weighted squared norm, *i.e.*, $\nu^\top Q \nu$; u_{ref} and y_{ref} are the input and output references, respectively; and $\Delta u_{t+k|t} = u_{t+k|t} - u_{t+k-1|t}$.

Several tuning parameters appear in the MPC problem (3.3) - (3.4) and must be tuned, such as:

- prediction horizon N_p and control horizon N_u;
- positive-semidefinite weight matrices Q_y, Q_u, $Q_{\Delta u}$;
- positive constant Q_ε used to soften the constraints and thus to guarantee feasibility of the optimization problem (3.3);
- tolerances used in the QP algorithm solving (3.3)–(3.4).

In order to compact the notation, all the MPC knobs are collected in a parameter vector $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, where $\mathcal{X}$ is a given bounded set in which the optimal tuning for x is sought.

In this chapter, we present an experiment-driven approach to tune the MPC knobs x in order to optimize the overall closed-loop performance based on pairwise preferences expressed by a controller calibrator. More precisely, we assume that we do not have a quantitative scoring function used by the calibrator to measure the closed-loop performance index and thus to be usable for auto-tuning x by global optimization. The only assumption we make is that for a given pair of different MPC parameters $x_1, x_2 \in \mathcal{X}$, only a preference expressed by the calibrator is available in terms of "performance achieved with parameters x_1 was better (or worse, or similar) than the one achieved with x_2".

The rationale of our problem formulation is that, in multi-criteria decision making such as in control system design, it is difficult (and sometimes impossible) to quantify an overall scoring function, but anyway it is easier for a calibrator to assess a *preference* between the outcomes of two experiments.

In order to compute the optimal MPC parameters x, the active preference learning algorithm GLISp [38] is used, which proposes the experiments to be performed for tuning x, through pairwise comparisons, that we reviewed in Section 2.3.1.

3.3 Preference-based tuning

The active learning algorithm GLISp iteratively suggest a sequence of MPC parameters $x_1, \ldots, x_N$ to be tested and compared such that x_N approaches the "optimal" combination of parameters as N grows. Specifically, we aim to find $x^\star \in \mathcal{X}$ such that $x^\star$ is "better" (or "no worse") than any other parameter x according to the preference function π, namely

$$x^\star \text{ such that } \pi(x^\star, x) \leq 0,\ \forall x \in \mathcal{X}. \tag{3.5}$$

If such a vector $x^\star$ is found, clearly we have also found a global minimizer $x^\star$ of J on $\mathcal{X}$, where J is an underlying closed-loop performance index that the calibrator wants to minimize, as (2.3) and (3.5) implies that $J(x^\star) \leq J(x), \forall x \in \mathcal{X}$.

Since the analytic expression and evaluations of J are unknown, we first estimate the surrogate function $\hat{J}$ following the steps noted in Section 2.3.1.1. Then the combined acquisition function (2.14) is optimized use the Particle Swarm Optimization (PSO) algorithm of [87].

We finally remark that, in executing the GLISp, some values of the MPC knobs may lead to an unstable closed-loop behaviour. In this case, the experiment should be interrupted (e.g., for safety reasons) and the calibrator may still express a preference. It is also possible that in the initialization phase of GLISp a comparison should be performed over two experiments both exhibiting closed-loop instability. In this case, one experiment can be preferred to another one, for instance, based on the time it could run before being interrupted.

3.4 Case studies

We apply the proposed preference-based MPC calibration method on two case studies: control of a *Continuous Stirring Tank Reactor* (CSTR) and automated driving with obstacle avoidance. The role of the calibrator is played by myself, which compares the observed closed-loop performance manually according to a qualitative scoring function constructed in her mind based on engineering insights.

In both case studies we set the maximum number of function evaluations $N_{\max} = 50$, the exploration parameter $\delta = 0.3$, and the tolerance parameter $\sigma = 10^{-6}$. The GLISp algorithm is initialized with $N_{\text{init}} = 10$ random samples generated by Latin hypercube sampling [88] using the *lhsdesign* function of the *Statistics and Machine Learning Toolbox* of MATLAB [89]. The hyper-parameter ϵ defining the RBF functions (2.7) is initialized as 1 and updated at iterations $10, 20, 30, 40$ via K-fold cross-validation with $K = 3$. The tests are run on an x64 with Intel i7-8550U 1.8 GHz CPU and 8GB of RAM. The *Model Predictive Control Toolbox* of MATLAB [90] is used for MPC design and simulation.

3.4.1 CSTR optimal steady-state switching policy

System description

The first case study considers the control of a CSTR, extensively described in [91]. The CSTR system consists of a jacketed non-adiabatic tank where an exothermic reaction occurs. The tank is assumed to be perfectly mixed with constant inlet and outlet rate. The chemical laws describing the CSTR reaction can be derived based on the energy and material balance, and are given by:

$$\begin{aligned}\frac{dT(t)}{dt} &= \frac{F}{V}(T_f(t)-T(t))-\frac{H}{C_p\rho}r(t)-\frac{US}{C_p\rho V}(T(t)-T_c(t))\\ \frac{dC_A(t)}{dt} &= \frac{F}{V}(C_{Af}(t)-C_A(t))-r(t),\end{aligned} \tag{3.6}$$

where V [m^3] is the volume of the reactor, and F [m^3/hr] is the rate of reactant A feeding the tank, which is equal to the rate of the product stream that exists from the reactor. Both V and F are assumed to be constant. $C_A(t)$ and $C_{A_f}(t)$ [kgmol/m^3] represent the concentration at time t of reactant A in the tank and in the inlet feed stream, respectively. $T(t)$, $T_f(t)$ and $T_c(t)$ [K] are the temperature of the reactor, of the inlet feed stream and of the coolant stream, respectively. Constant heat of reaction H [kcal/kgmol], fluid heat capacity C_p [kcal/(kg K)] and density ρ [kg/m^3] are assumed. U [kcal/(m K hr)] and S [m^2] are the overall heat transfer coefficient and heat exchange area, respectively. $r(t)$ [kgmol/(m^{3}hr)] is the reaction rate per unit volume, which can be calculated through Arrhenius rate law:

$$r(t) = k_0 \exp\left(\frac{-E}{RT(t)}\right) C_A(t) \tag{3.7}$$

where k_0 [hr^{-1}] is the pre-exponential factor; E [kcal/kgmol] is the activation energy for the reaction; and R [Kcal/(kgmol K)] is the gas constant. The CSTR process parameters are taken from [91] and listed in Table 3.1.

Control objectives

Initially, the plant is operating at steady state with a reactant concentration $C_A = 8.56$ kgmol/m^3 and low conversion. The objective is to design an MPC controller to achieve a steady-state concentration $C_A = 2$ kgmol/m^3, thus increasing the conversion rate. The feed stream concentration $C_{A_f}(t)$ and temperature $T_f(t)$ are treated as measured disturbances, and the coolant temperature $T_c(t)$ is the control input. In this test, the condition of the feed stream is kept constant, with T_f = 298.15 K and C_{A_f} = 10 kgmol/m^3.

Table 3.1: CSTR parameters [91]

Parameter	Value	unit
F/V	1.0	hr^{-1}
k_0	$3.49e7$	hr^{-1}
H	5960	kcal/kgmol
E	11843	kcal/kgmol
ρC_p	500	kcal/(m^3K)
US/V	150	kcal/(m^3K hr)
T_{c0}	298	K
R	1.987	Kcal/(kgmol K)

At each time step, the nonlinear model of the CSTR system in (3.6) is linearized around its operating point, and a linear MPC problem is solved. Although the output variables of interest are both the reactor temperature $T(t)$ and the concentration $C_A(t)$ in the product stream, only the latter is tracked and compensated by MPC.

Different competitive objectives should be taken into account in the MPC calibration, such as fast steady-state transition, reasonable final target achievement, and low energy consumption. The following guidelines are used to assist the calibration. First, the steady-state switching is expected to be completed within two days. Second, the final achieved steady state should be within ±3% of the desired value $C_A = 2$ kgmol/m^3. Third, in order to take into account energy consumption due to the cooling process, the temperature of the coolant stream $T_c(t)$ is restricted to be in the range of $[284 \ \ 310]$ K, with a maximum change at each time step ($T_{c_{max}}$) set to 10 K. The first two requirements just reflect the desired performance, and thus are not treated as *hard* constraints during calibration.

The following three design parameters are tuned: the sampling time T_s used to close the loop and to discretize the continuous-time model (3.6), the prediction horizon N_p, and the weight $Q_{\Delta u}$ penalizing the input change. These tuning parameters T_s, N_p and $\log(Q_{\Delta u})$ are restricted to the ranges $[0.25 \ 1.5]$ hr, $[4 \ 40]$ and $[-5 \ 3]$, respectively. The control horizon N_u is set equal to $N_p/3$ and rounded to the closet integer. Other MPC knobs are fixed, with Q_y and Q_u equal to $\left[\begin{smallmatrix}0 & 0\\ 0 & 1\end{smallmatrix}\right]$ and 0, respectively.

Calibration process

At each iteration of GLISp, a set of new MPC parameters is suggested and the closed-loop experiment is performed in simulation. The experiment is interrupted either when the steady-state condition is reached or after 48 hrs (namely, two simulated days). The experiment is also interrupted if an unsafe behavior is observed. The calibrator is then asked to compare the performance of the experiment with the best performance achieved until that time and choose the one she prefers based on the control objectives and the aforementioned guidelines.

As an example, Fig. 3.1 shows one iteration of the semi-automated

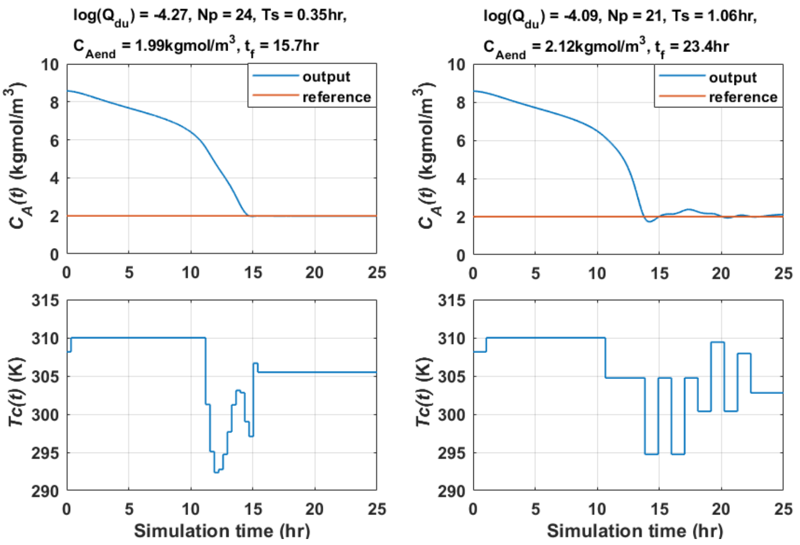


Figure 3.1: CSTR query window for one iteration of GLISp. The top subplots show the change of concentration C_A over the simulation time. The bottom subplots show the variations of coolant temperature T_c over simulation time. The results of the left experiment are preferred to the right one because of a faster transient and settling time, and smaller variations of T_c.

calibration process. At the top of the figure, the MPC design parameters, the achieved values of $C_{A_{end}}$ and duration of the switching process t_f are displayed. Both the MPC designs achieved the desired $C_{A_{end}}$ within 48 hrs. However, the transient and settling times of the left-hand-side experiment are shorter than the right-hand-side one. Moreover, the variations of the input signal T_c are smaller for the experiment on the left. Therefore, in this iteration, the left-hand-side MPC design is preferred.

Results

The GLISp algorithm terminates after $N_{\max} = 50$ iterations (namely, closed-loop experiments) and hence 49 comparisons. The best MPC design parameters T_s, N_p and $\log(Q_{\Delta u})$ are found to be 0.31 hr, 26 and -1.79, respectively. Fig. 3.2 (left panels) shows the corresponding trajectory of reactant concentration $C_A(t)$ and of the manipulated variable $T_c(t)$.

Fully-automated calibration

For the sake of comparison, a fully-automated calibration is performed. This requires to define a multi-objective quantitative scoring function describing the expected closed-loop performance. This step can be very hard and time-consuming, requiring proper scaling and balancing of the competitive control objectives.

The following closed-loop performance index function to be minimized is constructed after several trial-and-error iterations based on the preferences of the calibrator:

$$\frac{t_f}{t_{f_{max}}} + \frac{\sum_{k=1}^{N_T}(T_{c_k} - T_{c_{k-1}})^2}{T_{c_{max}}\, N_T} + \frac{|C_{A_{end}} - C_{A_{ref}}|}{AR\%\, C_{A_{ref}}} \tag{3.8}$$

where $t_{f_{max}} = 48$ hrs is the maximum duration of the switching process; $T_{c_{max}} = 10$ K is the maximum allowed temperature change of the coolant fluid between each time step; T_{c_k} and $T_{c_{k-1}}$ are the temperature of the coolant fluid at time step k and $k-1$, respectively; N_T is the total number of time steps in the steady-state transition; $AR\% = 3\%$ is the acceptable range of the final steady state concentration; $C_{A_{end}}$ and $C_{A_{ref}}$ [kgmol/m^3] are the achieved and desired final steady-state concentration of reactant A.

The GLIS algorithm (without preference) of [32] is used for experiment-driven optimization of the cost function in (3.8) with respect to the MPC design parameters. For a fair comparison with the proposed preference-based algorithm, 50 function evaluations are performed. This value actually does not account for the number of trials (namely, experiments) needed to construct the scoring function in (3.8).

The obtained optimal MPC parameters T_s, N_p and $\log(Q_{\Delta u})$ are 0.25 hr, 26 and -0.91. Fig. 3.2 (right panels) shows the corresponding closed-loop trajectory of $C_A(t)$ and of the manipulated variable $T_c(t)$. It can be observed that the preference- and the non-preference- based approaches achieve similar closed-loop performance.

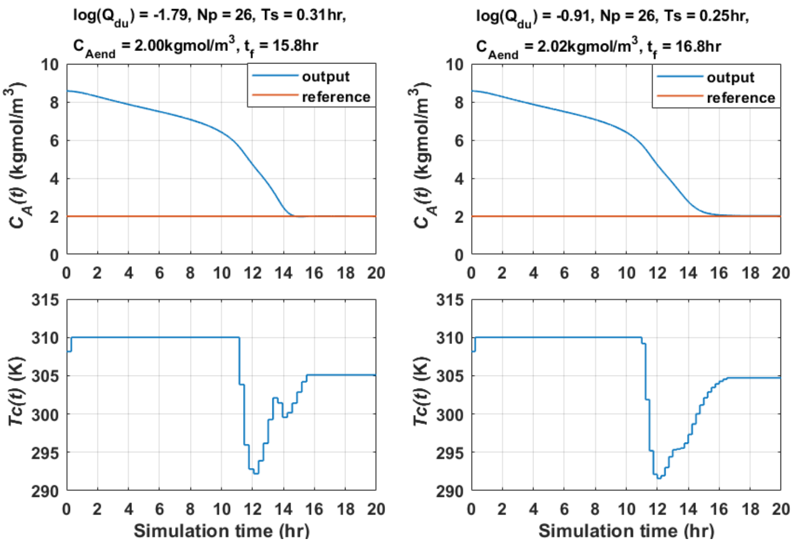


Figure 3.2: CSTR closed-loop performance obtained by calibrating MPC parameters through the proposed semi-automated preference-based approach (left panels) and through a fully-automated approach minimizing the scoring function (3.8) (right panels).

Overall, this case study demonstrates the capability of the semi-automated approach for solving calibration tasks with multiple competitive objectives. For such calibration tasks, when using a fully-automated approach, significant efforts need to be devoted to construct a proper performance scoring function as in (3.8). Therefore, the semi-automated approach can greatly reduce calibration time and efforts by eliminating this step.

3.4.2 Automated driving vehicle

System description

As a second case study, we consider the problem of lane-keeping (LK) and obstacle-avoidance (OA) in automated driving. MPC is employed to command vehicle velocity and steering angle to provide a smooth and safe drive. A simplified two degree-of-freedom bicycle model is used to describe the vehicle dynamics and simulate the experiment, with the front wheel as the reference point. The state variables $s = [x_f\ w_f\ \theta]'$ in the model are the longitudinal x_f and lateral w_f [m] positions of the front wheel, and the yaw angle θ [rad]. The manipulated variables $u = [v\ \psi]'$ are the commanded vehicle velocity v [m/s] and steering angle ψ [rad]. The standard continuous-time kinematic equations

$$\begin{aligned}\dot{x}_f =& v\cos(\theta+\psi)\\ \dot{w}_f =& v\sin(\theta+\psi)\\ \dot{\theta} =& \frac{v\sin(\psi)}{L}\end{aligned} \tag{3.9}$$

are used to model the evolution of the vehicle, where L [m] is the vehicle length. Here, full state observation is assumed, *i.e.*, the control output $y = s$.

The resulting discrete-time state-space model of (3.9) following the

MPC strategy noted in Section 3.2:

$$
\begin{aligned}
\tilde{s}_{k+1} = & \begin{bmatrix} 1 & 0 & -\bar{v}_k \sin(\bar{\theta}_k + \bar{\psi}_k) T_s \\ 0 & 1 & \bar{v}_k \cos(\bar{\theta}_k + \bar{\psi}_k) T_s \\ 0 & 0 & 1 \end{bmatrix} \tilde{s}_k \\
& + \begin{bmatrix} \cos(\bar{\theta}_k + \bar{\psi}_k) T_s & -\bar{v}_k \sin(\bar{\theta}_k + \bar{\psi}_k) T_s \\ \sin(\bar{\theta}_k + \bar{\psi}_k) T_s & \bar{v}_k \cos(\bar{\theta}_k + \bar{\psi}_k) T_s \\ \frac{\sin(\bar{\psi}_k)}{L} T_s & \frac{\bar{v}_k \cos(\bar{\psi}_k)}{L} T_s \end{bmatrix} \tilde{u}_k \\
\tilde{y}_k = & \tilde{s}_k,
\end{aligned}
\tag{3.10}
$$

is then used at each sampling time to compute the MPC action.

Control objectives

In tuning the MPC parameters, the objective is to keep the vehicle at the same horizontal lane with constant speed and to overtake other moving vehicles in an optimal way if they are within safety distance. However, similar to the CSTR case, it is difficult to define a proper quantitative scoring function for this multi-objective calibration task (for example, due to the ambiguity of transferring "optimal obstacle avoidance" into a mathematical formula). Therefore, to achieve good MPC performance using a fully-automated approach not based on preferences, either the calibrator needs to find a proper scoring function via trial-and-error (like in the previous CSTR case) or an advanced path planner model is required to provide a well-defined reference path to compare with. Both procedures can be time-consuming and computationally heavy. On the other hand, when using the semi-automated approach discussed in this chapter, none of the pre-mentioned steps is required.

We describe the test scenario as follows. Note, for ease of assessment, the unit of v and ψ described in the following text as well as in the figures are represented in km/hr and degree (°), respectively. The controlled vehicle is initially at position (x_f, w_f) = (0, 0) m with $\theta = 0°$. Another vehicle (obstacle) is at position (30, 0) m and moving horizontally at a constant speed of 40 km/hr. The shape of both vehicles is assumed to be rectangular, with a length of 4.5 m and a width of 1.8 m. During nominal LK conditions, the vehicle being controlled moves horizontally at 50 km/hr, with w_f = 0 m and $\dot{w}_f$ = 0 m/s . Once the obstacle is within a safety distance, the vehicle being controlled should pass it while keeping a safe lateral distance between them. In this case, the horizontal and lateral safety distances are 10 m and 3 m, respectively. The vehicle controlled by the MPC can vary its velocity in the range of [40, 70] km/hr during the LK period and [50, 70] km/hr during the OA period, and its reference velocities are set to 50 and 60 km/hr, respectively. For both LK and OA periods, θ can take values in the range of [-45, 45]°, with its rate of change between each time step limited to [-5, 5]°/s.

Five MPC design parameters are tuned. The sampling time T_s is allowed to vary in the range $[0.085\ 0.5]$ s. The prediction horizon N_p is restricted to $[10\ 30]$ and the control horizon N_u is taken as a fraction ϵ_c of N_p rounded to the closest integer. Here, ϵ_c can take values in the range $[0.1\ 1]$. The weight matrix of manipulated variables ($Q_{\Delta u}$) is set to be diagonal $Q_{\Delta u} = \begin{bmatrix} q_{u11} & 0 \\ 0 & q_{u22} \end{bmatrix}$ and the values of $\log(q_{u11})$ and $\log(q_{u22})$ are restricted in the interval [-5, 3]. The other MPC design parameters are fixed, with Q_y and Q_u set to $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, respectively.

Calibration process

The calibrator selects the preferred controller based on the following observations: (i) during both LK and OA periods, the worst-case computational time (t_{comp}) required for solving the QP problem (3.3) at each time step needs to be smaller than T_s, so that the MPC can be implemented in real time; (ii) during the LK phase, the vehicle should move at constant speed with w_f and θ close to 0 m and 0°; (iii) during the OA phase, the vehicle should keep reasonable safety distance away from the obstacle and should guarantee passengers' comfort (*i.e.*, aggressive lateral movements during overtaking should be avoided); (iv) the velocity in both LK and OA period should be close to the reference value with its variations kept to minimum; (v) variations of steering angles should not be aggressive; (vi) when there is a conflict combination among aforementioned criteria, criterion (i) has the highest priority and if the conflict is among criteria (ii)–(v), preference is given to the one leading to safer driving practice based on calibrator's experience.

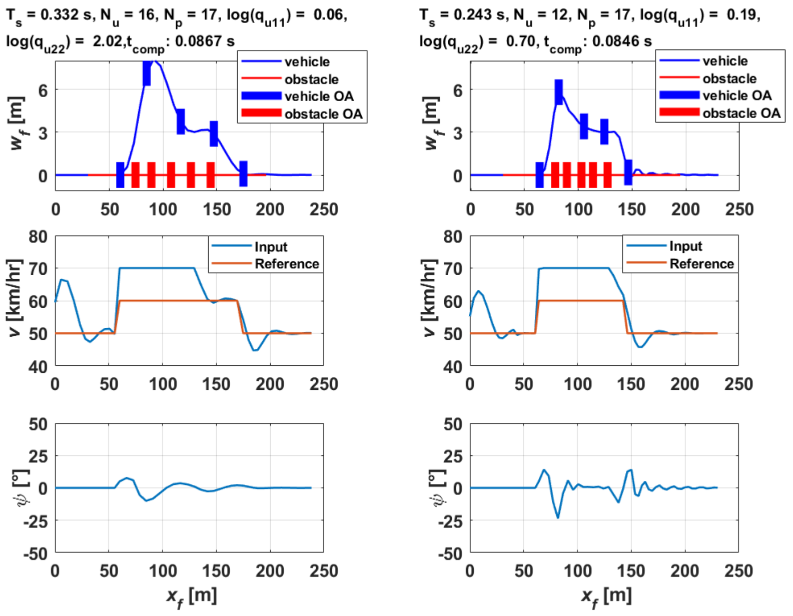


Figure 3.3: Vehicle control query window. The top subplots show the location trajectories of the vehicle and the obstacle, in which the "vehicle OA" and "obstacle OA" bars show five relative positions of the vehicle and obstacle during the obstacle-avoidance phase. The middle subplots show the actual and reference velocity v at different longitudinal positions. The steering angle ψ over the longitudinal position is depicted in the bottom subplots. For ease of assessment, the unit of v and ψ in the figure is converted to km/hr and degree (°), respectively. The results on the right panels are preferred.

The closed-loop test is simulated for 15 seconds. Fig. 3.3 shows the query window for one iteration of the calibration process. The MPC design parameters and t_{comp} are displayed at the top of the figure. In both of the cases illustrated in the figure, the QP problem (3.3) is solved within the chosen sampling time T_s. The performance of the right-hand-side experiment is preferred since in the left-hand-side experiment large lateral movements are present. Indeed, these movements can be much more dangerous (as the car may cross to the other lane) comparing to slightly more aggressive ψ variations.

Results

The GLISp algorithm terminates after $N_{\max} = 50$ function evaluations and 49 comparisons. The best MPC design parameters T_s, ϵ_c, N_p, $\log(q_{u11})$ and $\log(q_{u22})$ are found to be equal to 0.085 s, 0.310, 16, 0.261 and 0.918, respectively with a worst-case computational time t_{comp} needed to solve the QP problem (3.3) equal to 0.0808 s.

The final closed-loop results achieved by the designed MPC are depicted in Fig. 3.4, which demonstrates that solely based on calibrator's preferences, after only 50 experiments, the proposed algorithm is able to tune the MPC parameters with satisfactory performance. It is also important to remark that, for the same problem, the authors were not able to find, via several trial-and-error tests, a proper scoring function of the closed-loop performance to be used for a fully-automated calibration.

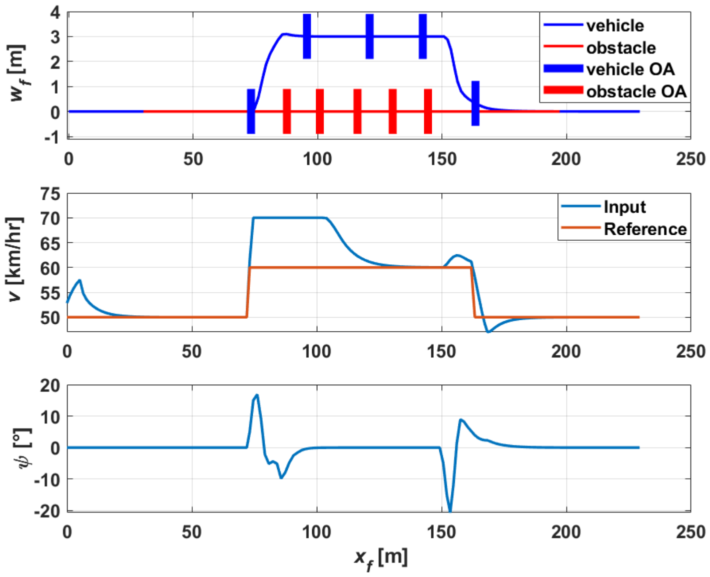


Figure 3.4: Vehicle control final performance obtained by the designed MPC controller.

3.5 Conclusion

In this chapter, a novel semi-automated MPC calibration approach was presented which turns out to be efficient in terms of number of experiments required for calibration. The key feature of the proposed methodology is that it allows calibration only based on pairwise preferences between the outcomes of the experiments, and thus it is very useful for calibration tasks with qualitative, subjective, or hard-to-quantify performance index functions, and with calibrators having limited MPC design knowledge. The same preference-based approach can be also used for calibration of other type of controllers, such as PIDs.

Chapter 4

Handling unknown constraints in preference-based optimization with applications to controller calibration

4.1 Introduction

Active learning algorithms for black-box global optimization problems have been studied since the sixties under different names [92–95]. These algorithms solve the problem by optimizing a surrogate of the objective function, which is estimated by exploring the space of the optimization variables. General procedures of surrogate-based active learning meth-

The content of this Chapter is from ©2021 IEEE. Reprinted, with permission, from M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," in *IEEE Trans. on Control Systems Technology*, vol. 30, no. 5, pp. 2176–2187, 2022.

ods were discussed in Section 2.2. In particular, nowadays *Bayesian Optimization* (BO) [27] is widely used to solve problems in which the cost function can only be quantified after running an experiment, such as in experimental controller calibration [96] and in automated machine learning [97].

Successful applications of global optimization algorithms based on active learning for the calibration of Model Predictive Control (MPC), PID, and state-feedback control laws were presented in [71, 72, 75, 76], in which the tuning parameters of the controller are the optimization variables and a quantitative characterization of the resulting closed-loop performance after running a simulation or experiment is the objective to optimize. These algorithms were also used for model selection [73, 74], for controller tuning in robotic manipulation and trajectory tracking [77, 78, 98], in optimizing gait parameters in robotic bipedal locomotion [79], and for "safe" optimization of position controller parameters of quadrotors [80].

A limitation of many black-box optimizers is that they require quantifying an objective function after running an experiment. However, many real-world controller calibration problems involve multiple objectives to optimize, such as settling time, overshoots, actuation effort, computational burden, and other performance-related metrics. The relative weights of such objectives can be hard to assign, and sometimes even impossible to quantify, as they are the result of a qualitative judgment. On the other hand, a skilled calibrator can often assess closed-loop performance of certain tuning combinations in terms of "this test was better than the other one," *i.e.*, by pairwise comparisons. Thus, when quantifying an objective function is difficult or impossible, we can instead use these expressed *preferences* to learn an underlying surrogate function to be optimized, which leads to the area of preference-based learning algorithms.

Preference-based Bayesian optimization (PBO) has been proposed in [99–102]. Preference-based *reinforcement learning* (RL) has also drawn much attention in recent years [81]. The reader is referred to the survey paper [37] for a comprehensive review. Note that sample efficiency, which is related to the credit assignment task in RL, is a major challenge in many preference-based RL methods [37]. A more sample-efficient active preference learning method, called GLISp (an extension of GLIS), was proposed in [38]. GLISp learns a surrogate of the underlying preference relations by solving a Quadratic Programming (QP) problem, whose constraints reflect the expressed *preference* on whether a specific candidate is better than the other. Then, it iteratively proposes a new candidate for testing to the decision-maker for comparison. This experiment-driven preference-based approach was tested in [1] (see also in Chapter 3 of this thesis) for semi-automated MPC calibration (automatic selection of control parameters, manual assessment of performance by comparisons), demonstrating its effectiveness in terms of the number of experiments needed to reach near-optimal closed-loop performance.

Some real-life control design problems often also involve constraints that are unknown beforehand or for which it is not possible to find an explicit analytic expression. This is a challenge since safe exploration can be essential in many control applications, and infeasible experiments can be dangerous and costly. Several methods have been proposed in the literature to handle unknown constraints and encourage safe exploration. In [103], Sui *et al.* presented a stagewise safe BO with Gaussian processes, which they later extended to allow multiple safety constraints independent of the objective function [104]. A general formulation for constrained BO and a modified version of the expected improvement acquisition function was illustrated in [105], which handles noisy constraint observations and considers cases in which the objective and constraint functions are decoupled. In [106], a *sequential model-based optimization method* was proposed. The unknown feasible region boundaries are first reconstructed from data through *support vector machines* (SVM). Then, a global optimization step is performed via BO. Different from aforementioned methods, GLISp accounts for unknown constraints *implicitly* in the preferences expressed by the decision maker by making the samples that are infeasible lose the pairwise comparisons against the feasible ones.

This chapter extends GLISp to handle unknown constraints in the active learning phase *explicitly*, therefore encouraging safe exploration. Besides expressing preferences, the decision-maker is asked to label an experiment as feasible and if its outcome is overall satisfactory (yes/no). Based on such labels, a surrogate of the probability of constraint feasibility and experiment's satisfaction is learned via an *Inverse Distance Weighting* (IDW) interpolant function [32]. The surrogates are properly integrated within the acquisition function to find the next point to test.

We show the efficiency and effectiveness of the proposed method, called *C-GLISp*, in three numerical benchmarks, and on an extension of the case study originally proposed in [1] on semi-automated MPC calibration for automated driving.

MATLAB and Python implementations of C-GLISp are also provided and available at `http://cse.lab.imtlucca.it/~bemporad/glis`.

The rest of the chapter is organized as follows. The problem of preference-based optimization with unknown constraints is formulated in Section 4.2. The proposed active-learning algorithm and details for its practical implementation are discussed in Section 4.3. Numerical benchmarks showing the properties and the effectiveness of the proposed method are reported in Section 4.4, while the case study on semi-automated MPC calibration for automated driving is presented in Section 4.5. Conclusions are drawn in Section 4.6.

4.2 Problem formulation

Let $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ be the space of decision vectors x. We are interested in minimizing an (unknown) objective function $f : \mathcal{D} \to \mathbb{R}$ subject to the constraint that x belongs to, which is an (unknown) feasibility set $\Omega_G \subseteq \mathcal{D}$.

We assume that we cannot represent the set Ω_G, but rather that, given a vector $x \in \mathcal{D}$, a decision-maker can assess the value of the *feasibility* *function* $G : \mathcal{D} \to \{0, 1\}$ defined as

$$G(x) = \begin{cases} 0 & \text{if } x \notin \Omega_G \\ 1 & \text{if } x \in \Omega_G. \end{cases} \tag{4.1}$$

The idea is that all the values x outside Ω_G are considered as "unacceptable" by the decision-maker. For example, an unacceptable x can be a set of controller parameters leading to an unstable closed-loop behavior or to a control law that is too expensive to compute in real-time.

Furthermore, we assume that the objective function f cannot be directly quantified, but rather it can be indirectly observed in two ways:

1. For a given sample $x \in \mathcal{D}$, the decision-maker is requested to say whether or not x leads to certain "satisfactory performance". Formally, we can define a *satisfaction set* $\Omega_S \subseteq \mathcal{D}$ and a *satisfaction function* $S : \mathcal{D} \to \{0, 1\}$ as

$$S(x) = \begin{cases} 0 & \text{if } x \notin \Omega_S \\ 1 & \text{if } x \in \Omega_S, \end{cases} \tag{4.2}$$

   where the set Ω_S contains all the vectors x leading to a performance that the decision-maker judges satisfactory. An analytic expression of Ω_S is therefore not available, only the value $S(x)$ is provided by the decision-maker for any given $x \in \mathcal{D}$. Note that Ω_S may not be a subset of Ω_G, for example when the preference-based optimization process is carried out in simulation: a sample may lead to satisfactory performance but would not be implementable due to hardware limitations. On the other hand, in cases of assessments based on physical experiments, Ω_S is necessarily a subset of Ω_G, as no performance would be available for evaluation when the parameters are infeasible.

2. For any pair $x_1, x_2 \in \mathcal{D}$, the decision-maker is requested to provide the output of the *preference function*: $\pi : \mathcal{D} \times \mathcal{D} \to \{-1, 0, 1\}$, which is implicitly defined according to the underlying hidden function f to be minimized, namely

$$\pi(x_1, x_2) = \begin{cases} -1 & \text{if } f(x_1) < f(x_2) \\ 0 & \text{if } f(x_1) = f(x_2) \\ 1 & \text{if } f(x_1) > f(x_2). \end{cases} \tag{4.3}$$

The rationale behind the above problem formulation is that often one encounters practical decision problems in which a function f is impossible to quantify, but anyway it is possible for a human operator to express a qualitative evaluation (*e.g.*, "good" or "bad") and a preference between the outcome of two experiments.

Formally, we want to find the optimal solution $x^\star \in \Omega_S \cap \Omega_G$ such that $x^\star$ is "better" (or "no worse") than any other x according to the preference function π, namely solve the following problem:

$$\text{find } x^\star \text{ such that } \pi(x^\star, x) \leq 0,\ \forall x \in \Omega_S \cap \Omega_G. \tag{4.4}$$

We propose to solve problem (4.4) iteratively as follows: (i) suggest a sequence of decision vectors $x_1, \dots, x_N \in \mathcal{D}$ to test, (ii) ask to evaluate the feasibility function $G(x_i)$ and the satisfaction function $S(x_i)$ for $i = 1, \dots, N$, and (iii) ask to evaluate the preference function $\pi(x_i, x_j)$ for M given pairs (i, j), $i, j = 1, \dots, N$, $i \neq j$, where M is the number of expressed preferences, $1 \leq M \leq \binom{N}{2}$. The goal is to propose candidate vectors x_N approaching the optimal solution $x^\star$ as N grows.

4.3 Proposed method

The preference-based optimization method described in this chapter to solve problem (4.4) is based on an extension of GLISp originally proposed in [38] (see also in Section 2.3.1). We refer to the new algorithm as "C-GLISp," whose aim is to handle unknown constraints expressed in terms of an approximation of the feasibility function G in (4.1) and of the satisfaction function S in (4.2).

Similarly to GLISp, C-GLISp involves two main phases: an initial random sampling and an active learning phase. In both phases, C-GLISp trains and updates three surrogate functions approximating the feasibility function G, the satisfaction function S, and the preference function π. During the active learning phase, the next point for evaluation is selected by optimizing an acquisition function which trades off *exploitation* (optimization only based on the surrogates describing the observed preferences and constraints) and *exploration* (searching unexplored areas of the domain $\mathcal{D}$). The goal of C-GLISp is to approach an optimal solution $x^\star$ as in (4.4) within a small number N of experiments.

4.3.1 Learning unknown constraint functions

We discuss how to train surrogates of the functions G and S that approximate, respectively, the feasibility constraint $x \in \Omega_G$ and the satisfaction constraint $x \in \Omega_S$. The idea is to ask the decision-maker to assess, once an experiment is performed, whether the constraints $x \in \Omega_G$ and $x \in \Omega_S$ are satisfied or not, and train surrogate functions of G and S based on the outcome of $N \geq 2$ of such queries. These queries are performed on a set of samples $\{x_1, \ldots, x_N\}$ iteratively proposed by C-GLISp. Compared to unconstrained preference-based optimization like GLISp, in which an infeasible/unsatisfactory sample only indirectly reveals itself as such by losing pairwise comparisons against feasible/satisfactory ones, C-GLISp exploits the information on whether $x \in \Omega_G$ and/or $x \in \Omega_S$ to facilitate the optimization process, in particular, to avoid exploration in the infeasible and/or unsatisfactory region and therefore reduce the number of samples $x_i \notin \Omega_G$ and/or $x_i \notin \Omega_S$.

The surrogate functions for G and S are constructed as follows. The decision-maker observes the outcome of the performed experiments, and he/she provides a *feasibility vector* $G_F = [G_1\ \ldots\ G_N]' \in \{0,1\}^N$ with

$$G_i = G(x_i), \tag{4.5}$$

and a *satisfaction vector* $S_F = [S_1\ \ldots\ S_N]' \in \{0,1\}^N$ with

$$S_i = S(x_i), \tag{4.6}$$

by assessing whether each experiment is feasible and satisfactory. Then, surrogates $\hat{G}$ of G and $\hat{S}$ of S are constructed from the observations G_F and S_F, respectively, as detailed below.

A surrogate function $\hat{G} : \mathcal{D} \to \mathbb{R}$ predicting the probability of satisfying the feasibility constraint $x \in \Omega_G$ is defined as

$$\hat{G}(x) = \sum_{i=1}^{N} \nu_i(x) G_i, \tag{4.7}$$

where $\nu_i(x) : \mathcal{D} \to \mathbb{R}$ for $i = 1 \ldots, N$ is defined as

$$\nu_i(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{if } x = x_j, j \neq i \\ \frac{w_i(x)}{\sum_{i=1}^{N} w_i(x)} & \text{otherwise.} \end{cases} \tag{4.8}$$

Here $w_i : \mathcal{D} \setminus \{x_i\} \to \mathbb{R}$ is the following IDW function [107]

$$w_i(x) = \frac{e^{-d^2(x,x_i)}}{d^2(x, x_i)}, \tag{4.9}$$

where $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ denotes the squared Euclidean distance

$$d(x, x_i) = \|x - x_i\|_2^2. \tag{4.10}$$

The surrogate function $\hat{S} : \mathcal{D} \to \mathbb{R}$ is defined similarly. The approach presented in this chapter aims at solving problems where experiments are expensive to run, so that data efficiency is essential. IDW interpolation functions are selected because of their high accuracy in this case. Other binary classification methods (*e.g.* logistic regression or random forests) would be less suitable in this context since their accuracy with a small number of training data is limited. Support vector machines (SVMs) [50] can be a potential substitute since it works well for small and medium-size training samples. However, our preliminary numerical tests have shown that IDW interpolation functions outperform SVM in our context. Also the predicted probabilities $\hat{G}$ and $\hat{S}$ from IDW interpolation functions are always between 0 and 1 by construction (see [32, Lemma 1-P2]). It is also worth noting that since the feasibility/satisfactory constraints are unknown, it may be impossible to distinguish between them. Hence, rather than model each feasibility/satisfactory constraint with a separate surrogate function, we model the probability of being feasible/satisfactory, making it easier to handle multiple feasibility/satisfactory constraints.

4.3.2 Learning the preference function

Radial basis functions (RBFs) [48, 49] are flexible and have been adopted to solve global optimization problems in [32, 48, 49, 108, 109] with success. Therefore, as in [38], we parameterize the surrogate function $\hat{f} : \mathcal{D} \to \mathbb{R}$ as a linear combination of RBFs [48, 49] as reviewed in Section 2.3.1.1

$$\hat{f}(x) = \sum_{k=1}^{N} \beta_k \phi(\epsilon d(x, x_i)). \tag{4.11}$$

Besides the *feasibility vector* G_F and the satisfaction vector S_F, the *preference vector* $B = [b_1 \; \dots \; b_M]^T \in \{-1, 0, 1\}^M$ is also assumed to be provided by the decision-maker, with

$$b_h = \pi(x_{i(h)}, x_{j(h)}), \tag{4.12}$$

for $x_i, \; x_j \in \mathcal{D}$ such that $x_i \neq x_j, \forall i \neq j, i, j = 1, \dots, N$.

4.3.3 Acquisition function

Minimizing $\hat{f}$ greedily to generate the next sample x_{N+1} may lead the solver converge to a point that is not the global optimum of (4.4). Hence, when selecting the next point x_{N+1}, besides *exploiting* the surrogate $\hat{f}$, some *exploration* should be considered to search regions with limited/no samples to reduce uncertainty of $\hat{f}$. Also, the feasibility and satisfactory regions are unknown and are only *implicitly* included in the surrogate function $\hat{f}$. Therefore, we also include terms to *explicitly* avoid the *exploration* in the regions with low probabilities of being feasible and satisfactory by penalizing the (estimated) infeasibility $x \notin \Omega_G$ and unsatisfactory performance $x \notin \Omega_S$.

The exploration term used in GLISp is the following IDW function $z : \mathcal{D} \to \mathbb{R}$

$$z(x) = \begin{cases} 0 & \text{if } x \in \{x_1, \dots, x_N\} \\ \tan^{-1}\left(\frac{1}{\sum_{i=1}^{N} r_i(x)}\right) & \text{otherwise,} \end{cases} \tag{4.13}$$

where $r_i(x) = \frac{1}{d^2(x, x_i)}$. Unlike GLISp, here we modify (4.13) into

$$\begin{aligned} z_N(x) = &\left(1 - \frac{N}{N_{max}}\right) \tan^{-1}\left(\frac{\sum_{i=1}^{N} r_i(x_N^*)}{\sum_{i=1}^{N} r_i(x)}\right) \\ &+ \frac{N}{N_{max}} \tan^{-1}\left(\frac{1}{\sum_{i=1}^{N} r_i(x)}\right) \end{aligned} \tag{4.14}$$

for $x \notin \{x_1, \dots, x_N\}$ and $z_N(x) = 0$ otherwise. In (4.14), x_N^* is the best decision variable found up to iteration N. The formulation in (4.14) is empirically observed to better escape from local minima. In (4.14), N_{max} is the maximum allowed number of experiments. The rationale behind (4.14) is that we encourage the exploration of regions of $\mathcal{D}$ further away from the current best solution in the early iterations and reduce its effects as the number N of experiments increases.

Then, we introduce the *acquisition function* $a : \mathcal{D} \to \mathbb{R}$ defined as

$$\begin{aligned} a(x) = & \frac{\hat{f}(x)}{\Delta \hat{F}} - \delta_E z_N(x) \\ & + \delta_G(1 - \hat{G}(x)) + \delta_S(1 - \hat{S}(x)), \end{aligned} \tag{4.15}$$

where $\delta_E \geq 0$ is the exploration parameter, and $\delta_G, \delta_S \geq 0$ weight the probability of a sample x to be infeasible and/or unsatisfactory, respectively. The term $\Delta \hat{F} = \max_i\{\hat{f}(x_i)\} - \min_i\{\hat{f}(x_i)\}$ is the range of the surrogate function $\hat{f}$ on the samples in $\{x_1, \dots, x_N\}$. It is used as a scaling factor in (4.15) to make each term in (4.15) comparable, which eases the selection of the hyper-parameters δ_E, δ_G, and δ_S.

The exploration parameter δ_E encourages sampling unexplored regions of the domain $\mathcal{D}$. Setting $\delta_E = 0$ makes C-GLISp rely heavily on the accuracy of the surrogate functions $\hat{f}$, $\hat{G}$, and $\hat{S}$, which may easily lead to missing the global optimum. On the other hand, setting $\delta_E \gg 1$ leads C-GLISp to explore the entire domain $\mathcal{D}$ regardless of the decision-maker's preferences and feasibility/satisfaction assessments.

Functions $\hat{G}$ and $\hat{S}$ in (4.15) aim at discouraging exploration in regions where the experiment is predicted to be infeasible (*i.e.*, $x \notin \Omega_G$) and/or unsatisfactory (*i.e.*, $x \notin \Omega_S$). Therefore, a poor selection of the hyperparameters δ_G and δ_S and/or a poor predictive capability of $\hat{G}$ and $\hat{S}$ (*e.g.*, due to a limited number of samples) can prevent finding new vectors that are actually feasible and/or satisfactory. To alleviate this issue, we suggest to adaptively tune δ_G and δ_S based on the sampled standard deviation obtained from leave-one-out cross-validation [51] of $\hat{G}$ and $\hat{S}$, respectively. More specifically, each available sample in the set $\{x_1, \dots, x_N\}$ is used once as a testing point and the remaining ones are used to train $\hat{G}$ and $\hat{S}$. The prediction $\hat{G}(x_i)$ and $\hat{S}(x_i)$ on the test sample x_i is compared with the corresponding labels $G(x_i)$ and $S(x_i)$ assigned by the decision-maker to compute the following sampled standard deviations of the error:

$$\hat{\sigma}_G = \min\left\{1,\ \sqrt{\frac{\sum_{i=1}^{N}(\hat{G}(x_i)-G(x_i))^2}{N-1}}\right\},$$
$$\hat{\sigma}_S = \min\left\{1,\ \sqrt{\frac{\sum_{i=1}^{N}(\hat{S}(x_i)-S(x_i))^2}{N-1}}\right\}. \tag{4.16}$$

The sampled standard deviations are then used to update, after each iteration, the weights δ_G and δ_S as follows:

$$\delta_G = (1-\hat{\sigma}_G)\delta_{G,\text{default}}, \tag{4.17a}$$
$$\delta_S = (1-\hat{\sigma}_S)\delta_{S,\text{default}}, \tag{4.17b}$$

where $\delta_{G,\text{default}}$ and $\delta_{S,\text{default}}$ are default values set by the user. Clearly, one should select $\delta_{G,\text{default}} > \delta_{S,\text{default}}$, so that the possible infeasibility is penalized more than a potential unsatisfactory behavior. The updated values of δ_G and δ_S are then used to construct the acquisition function $a(x)$ in (4.15).

The next sample x_{N+1} to test is obtained by minimizing $a(x)$, *i.e.*,

$$x_{N+1} = \arg\min_{x\in\mathcal{D}} a(x). \tag{4.18}$$

Different optimization methods can be used to solve problem (4.18) efficiently either via derivative-free [110], or derivative based algorithms.

C-GLISp updates the surrogates $\hat{f}$, $\hat{G}$, and $\hat{S}$, and the exploration function $z_N(x)$, by iteratively suggesting a new point x_{N+1} to test, and by receiving feedback from the decision-maker in terms of feasibility, overall satisfaction, and preferences between pairs of experiments. Algorithm 4.1 summarizes the flow of the proposed method.

Algorithm 4.1 C-GLISp: Preference learning algorithm with unknown constraint handling ©2021 IEEE

Input: Lower and upper bounds (ℓ, u), known constraint set if available; number $N_{\text{init}} \geq 2$ of initial samples, number $N_{\max} \geq N_{\text{init}}$ of maximum function evaluations; $\delta_E \geq 0$, $\delta_{G,\text{default}} \geq 0$ and $\delta_{S,\text{default}} \geq 0$; $\sigma > 0$; and $\epsilon > 0$; self-calibration index set $\mathcal{I}_{\text{sc}} \subseteq \{1, \ldots, N_{\max} - 1\}$.

```
1. Generate $N_{\text{init}}$ random samples $X = \{x_1, \ldots, x_{N_{\text{init}}}\}$ using Latin hypercube sampling method [88];
2. $N \leftarrow 1, i^\star \leftarrow 1$;
3. **While** $N < N_{\max}$ **do**
   3.1. **if** $N = 1$ **then**
        3.1.1. Observe feasibility $G_N$ and satisfaction $S_N$;
   3.2. **if** $N \geq N_{\text{init}}$ **then**
        3.2.1. **if** $N \in \mathcal{I}_{\text{sc}}$ **then** recalibrate $\epsilon$ through $K$-fold cross-validation;
        3.2.2. Solve (2.10) to obtain $\beta$ to define the surrogate function $\hat{f}$ (4.11);
        3.2.3. Update $\delta_G$ and $\delta_S$ as in (4.17);
        3.2.4. Define acquisition function $a$ as in (4.15);
        3.2.5. Solve optimization problem (4.18) and get $x_{N+1}$;
   3.3. $i(N) \leftarrow i^\star, j(N) \leftarrow N + 1$;
   3.4. Observe feasibility $G_{j(N)}$. satisfaction $S_{j(N)}$ and preference $b_N = \pi(x_{i(N)}, x_{j(N)})$ ;
   3.5. **if** $b_N = 1$ **then set** $i^\star \leftarrow j(N)$;
   3.6. $N \leftarrow N + 1$;
4. **End**.
```

Output: Computed best input $x^\star = x_{i^\star}$.

4.4 Optimization benchmarks

This section reports tests of C-GLISp on three constrained global optimization benchmarks to illustrate its effectiveness in solving optimization problems with unknown constraints. Computations are run on an Intel i7-8550U 1.8-GHz CPU laptop with 8GB of RAM. The Latin hypercube sampling method [88] (*lhsdesign* function of the *Statistics and Machine Learning Toolbox* of MATLAB [89]) is used in the initial sampling phase of C-GLISp. Particle Swarm Optimization (PSO) [87] is used to minimize the acquisition function as in (4.18).

C-GLISp is compared to the original GLISp and to PBO (with *expected improvement* as acquisition function) [100, Section 2.3]. For numerical benchmarks, C-GLISp, GLISp, and PBO assign the preferences on pairwise comparisons based on the combined assessments of the objective function value, feasibility, and performance satisfaction. For each test function, depending on the problem formulation, a maximum of three types of queries are obtained when we use C-GLISp, which are the preference relation (B), the feasibility label (G_F), and the satisfaction label (S_F). In contrast, GLISp and PBO only rely on the preference relation B. The goal of the comparison between C-GLISp and GLISp is to check if accounting the feasibility and/or satisfactory information explicitly in the acquisition function can encourage safe exploration from the comparisons. It is worth noting that the exact evaluation of the objective function and the constraints (feasibility and satisfactory outcomes) for these numerical benchmarks are unknown to the algorithms and are only used to construct a synthetic decision-maker.

Table 4.1 lists the specifications of the benchmarks. The original feasibility set of the function *Mishra's Bird function-constrained* (MBC) [111, 112] is modified so that the unconstrained global optimum in the search domain is no longer in the feasible area. The *camelsixhumps-hard constrained* (CHC) benchmark [32, 113] considers two feasibility constraints, and the unconstrained global optimum also differs from the constrained one. Lastly, the benchmark function *camelsixhumps-hard and soft constrained* (CHSC) [32, 113] has both feasibility and satisfaction constraints.

Table 4.1: Numerical benchmarks - Problem Specification ©2021 IEEE

Test function	Objective function	Unknown constraints	Search domain $\mathcal{D}$
Mishra's Bird function-constrained (modified) [111, 112] (MBC)	$f(x,y) = \sin(y)e^{(1-\cos(x))^2} + \cos(x)e^{(1-\sin(y))^2} + (x-y)^2$	Feasibility constraints: $(x+9)^2 + (y+3)^2 < 9$	$[-10.0, -2]$; $[-6.5, 0.0]$
camelsixhumps-hard constrained [32, 113] (CHC)	$f(x,y) = (4 - 2.1x^2 + x^{4/3})x^2 + xy + (4y^2-4)y^2$	Feasibility constraints: $g_1 \cap g_2$ $g_1 : \begin{bmatrix} 1.6295 & 1 \\ -1 & 4.4553 \\ -4.3023 & -1 \\ -5.6905 & -12.1374 \\ 17.6198 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} < \begin{bmatrix} 3.0786 \\ 2.7417 \\ -1.4909 \\ 1 \\ 32.5198 \end{bmatrix}$ $g_2 : x^2 + (y+0.1)^2 < 0.5$	$[-2, 2]$; $[-1, 1]$
camelsixhumps-hard and soft constrained [32, 113] (CHSC)	$f(x,y) = (4 - 2.1x^2 + x^{4/3})x^2 + xy + (4y^2-4)y^2$	Feasibility constraints: g_2 Satisfaction constraints: g_1 $g_1 : \begin{bmatrix} 1.6295 & 1 \\ 0.5 & 3.875 \\ -4.3023 & -4 \\ -2 & 1 \\ 0.5 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} < \begin{bmatrix} 3.0786 \\ 3.324 \\ -1.4909 \\ 0.5 \\ 0.5 \end{bmatrix}$ $g_2 : x^2 + (y+0.04)^2 < 0.8$	$[-2, 2]$; $[-1, 1]$

Table 4.2: Numerical benchmarks - Solver Specification ©2021 IEEE

Test function	Max number of fun. eval. N_{max}	Number of initial sampling N_{init}	Hyper-parameter values			RBF specifications (4.11)			Tolerance	Weights	Regularization
			δ_E	$\delta_{G,\text{default}}$	$\delta_{S,\text{default}}$	function	initial ϵ	recalibration steps	σ in (2.10)	c_h in (2.10)	λ in (2.10)
MBC	50	13	1.0	1.0	–	Inverse quadratic	1.0	$\{13, 22, 32, 41\}$	0.02	1.0	1e-6
CHC	100	25	2.0	2.0	–	Inverse quadratic	1.0	$\{25, 44, 63, 81\}$	0.01	1.0	1e-6
CHSC	50	13	1.0	1.0	0.5	Inverse quadratic	1.0	$\{13, 22, 32, 41\}$	0.02	1.0	1e-6

Same parameters (if relevant) are used in C-GLISp, GLISp, and PBO.

Table 4.3: Numerical benchmarks - Results ©2021 IEEE

Test function	Constrained optimum[a]				Feasibility[b]		
	Optimum	C-GLISp	GLISp	PBO	C-GLISp	GLISp	PBO
MBC	-48.4	-47.95	-48.33	-40.24	100	100	91
CHC	-0.5844	-0.3582	-0.5224	0.2571	96	66	33
CHSC	-0.9050	-0.8526	-0.8861	-0.6315	96 (95)	82 (84)	74 (72)

a- The median of computed constrained optima that are feasible out of 100 runs (the distribution over 100 runs is reported in Table 4.4).

b- Number of runs whose computed optimizers are feasible out of 100 runs. Values in parentheses indicate the number of runs the optimizer is satisfactory.

The two unconstrained optima for this test function are both feasible but not satisfactory.

The values of the hyper-parameters and benchmark testing specifications for C-GLISp, GLISp, and PBO are provided in Table 4.2. The number of initial samples (N_{init}) is selected as one fourth of the maximum number of function evaluations ($N_{max}/4$) rounded to the nearest integer. Three-fold cross-validation is used to update the hyper-parameter ϵ (4.11) at iterations within the self-calibration index set $\mathcal{I}_{\text{sc}}$, which are N_{init}, $N_{init} + (N_{max} - N_{init})/4$, $N_{init} + (N_{max} - N_{init})/2$, and $N_{init} + 3(N_{max} - N_{init})/4$, rounded to the closest integers. The tolerance σ in (2.10) is set to $1/N_{max}$. The default value $\delta_{G,\text{default}}$ in (4.17) is the same as δ_E, so that the feasibility term in (4.15) is comparable to the pure exploration term, while the default value $\delta_{S,\text{default}}$ is selected as $\delta_{G,\text{default}}/2$ to reduce its effects with respect to hard feasibility constraints. The parameters δ_G and δ_S are kept at their default values until N_{init} experiments have been performed, then they are updated each time a new point is added using equation (4.17). The remaining parameters of the solvers are set according to the defaults used or suggested in [38].

Table 4.3 reports the results obtained by running a Monte-Carlo simulation with 100 runs of C-GLISp, GLISp, and PBO to obtain statistically significant results. One of such runs of C-GLISp on all three numerical benchmarks is depicted in Fig. 4.1. Table 4.4 displays the distribution over 100 runs of the percentage difference between the achieved feasible solutions and the true constrained optimum. Overall, the results from Table 4.3 and 4.4 show that C-GLISp can find a feasible near-optimal solution more frequently than GLISp and PBO.

From the results on the benchmark function MBC, where the feasibility set Ω_G covers roughly one-third of the domain $\mathcal{D}$ (cf. Fig. 4.1), the performance of GLISp and C-GLISp are comparable. They always terminate the search with a feasible optimum (100 out of 100 runs), with 67% and 69% of them, respectively, located within 5% difference from the global solution (Table 4.4). On the other hand, PBO computes a feasible optimum in 91 runs, but with only around 39% within 5% difference (Table 4.4). When the constraint is more complex such as the one in CHC,

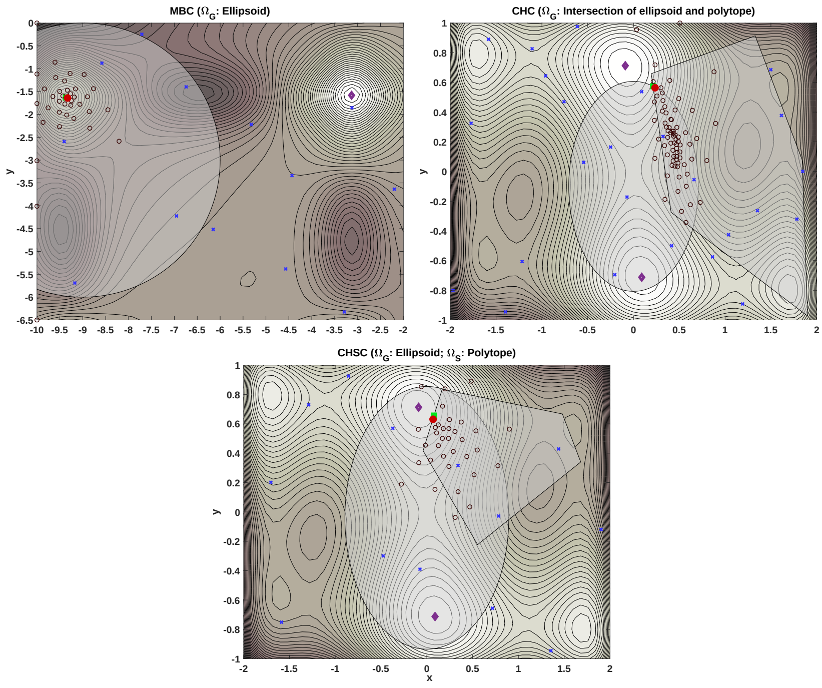


Figure 4.1: Algorithm C-GLISp. Level sets of the functions used in the three benchmarks, along with feasibility and satisfaction sets. Blue $\times$: points generated from initial sampling phase; black $\circ$: points generated from active learning phase; purple $\Diamond$: global unconstrained optimizer; red $\bullet$: constrained optimizer found after N_{max} iterations; green $\Box$: global constrained optimizer. As N increases, the points generated by C-GLISp approach the constrained optimizer, and most of the points generated during the active learning phase lay in the feasibility and satisfaction regions. ©2021 IEEE

the majority of the optima computed by C-GLISp (96 out of 100 runs) are feasible. In comparison, only 66 and 33 runs by GLISp and PBO, respectively, terminate with a feasible solution (Table 4.3 and Fig. 4.2). From Fig. 4.1, it is also observed that, for the test function CHC, after the initial sampling phase, most points generated in the active learning phase by C-GLISp are within the feasible region.

For the test function CHSC, C-GLISp can often find a near-optimal

Table 4.4: Distribution over 100 runs of the percentage difference between achieved and global optimum ©2021 IEEE

Benchmark	Algorithm	Number of runs within each interval			
		Intervals of % Difference from Global Optimum			
		(0,5]	(5,10]	(10,15]	(15,100]
MBC	PBO	39	4	2	18
	GLISp	67	1	2	3
	C-GLISp	69	6	1	5
		(0,5]	(5,20]	(20,50]	(50,100]
CHC	PBO	0	0	4	7
	GLISp	28	7	5	1
	C-GLISp	0	20	40	22
		(0,5]	(5,10]	(10,15]	(15,100]
CHSC	PBO	13	10	4	27
	GLISp	56	8	5	9
	C-GLISp	43	22	13	16

Note: only the runs with feasible solutions within 100% difference from the global optimum are counted.

solution that is both feasible and satisfactory. The performance of GLISp is slightly worse than C-GLISp in terms of the number of times a feasible and satisfactory solution is obtained. PBO can identify a feasible and satisfactory solution with a relatively high chance but still lower than both C-GLISp and GLISp. Also, its final outcome is worse, see Table 4.3.

Table 4.3 also shows that, within the same number of iterations, the median of the computed feasible constrained optima from GLISp is always closer to the global constrained optimum than the one computed from C-GLISp. This is because of the trade-off between trying to get a more accurate solution (which is often achieved by sampling multiple points close to the current best solution up to iteration N) and exploring a larger area to reduce uncertainty (in problems with unknown constraints, C-GLISp also tries to identify possible feasible regions).

For our problem setting, we have set a limit on the computational budget. Modification of the exploration term from (4.13) to (4.14) helps to better escape from local minima in the early iterations by encouraging the exploration of regions of $\mathcal{D}$ further away from the current best

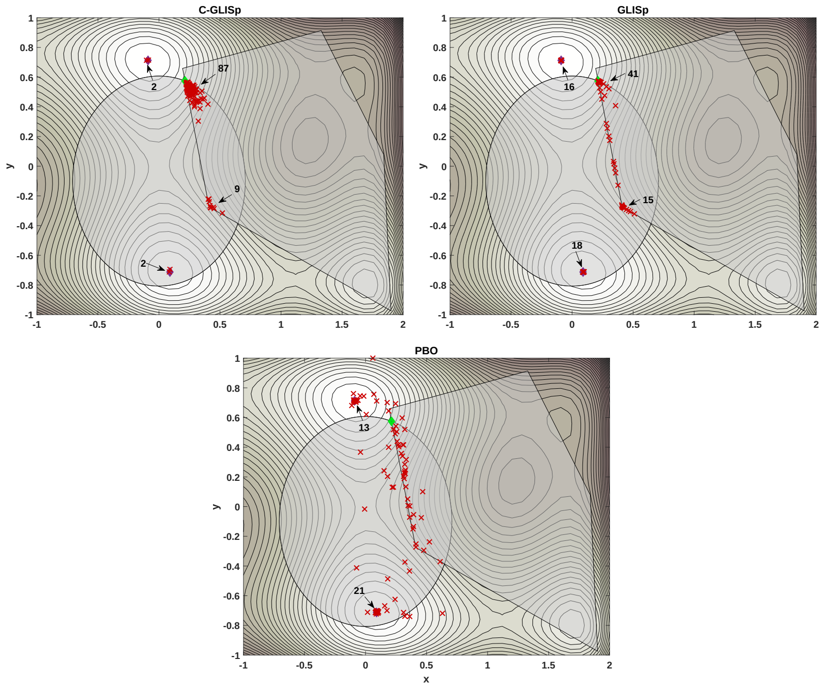


Figure 4.2: Benchmark CHC. Optimizers computed by C-GLISp, GLISp and PBO in 100 runs. Red ×: optimizer computed at the end of each run; purple ◊: unconstrained optimizer; green ◊: global constrained optimizer. Numbers in black with arrows indicate the number of overlapping points. ©2021 IEEE

solution. This modification is significant for problems with small feasible regions (relative to the search domain) and/or complex unknown constraints (*e.g.*, numerical benchmark CHC). This is because that the additional exploration introduced by the modification can help the solver identify the feasible region more quickly and start recommending feasible guesses faster, reducing the chance of the solver trapping into an infeasible local optimum. From the number of feasible solutions computed shown in Table 4.3 and the computed optimizers displayed in Fig. 4.2, we observe that the situation of trapping into an infeasible local optimum occurs to GLISp more frequently than to C-GLISp. However, GLISp can achieve a solution closer to the constrained optimum than C-GLISp (Table 4.4) when GLISp successfully identifies the feasible region. This is because more computational budget is then used to get closer to the optimum than to explore other potentially feasible regions as in C-GLISp. For problems where testing an infeasible solution is expensive and/or dangerous, it is better to be conservative and have a less optimal but feasible solution. As a result, C-GLISp is preferred over GLISp under these problem settings.

Overall, the results on the numerical benchmarks show that both C-GLISp and GLISp can approach near-optimal solutions within a small number of function evaluations. In all of the three benchmarks, both C-GLISp and GLISp outperform PBO (Table 4.3 and 4.4). An explanation for the superior performance of C-GLISp in identifying feasible/satisfactory solutions is that it explicitly leverages feasibility/satisfaction information in the acquisition function, while GLISp and PBO handle unknown constraints only through preference queries.

4.5 MPC calibration

To illustrate the application of C-GLISp to controller calibration, we discuss the design of an MPC controller for lane-keeping (LK) and obstacle-avoidance (OA) in automated driving. MPC is employed to command vehicle velocity and steering angle to provide a smooth and safe drive. The same problem was considered in Section 3.4.2 and is extended in this chapter to handle feasibility and satisfaction constraints. Specifically, we use C-GLISp to implement an iterative semi-automated calibration procedure. Different from the case study in Section 3.4.2, where we account for the information of feasibility and satisfaction conditions implicitly in the preference query (GLISp) as if they were, implicitly, underlying penalty functions, C-GLISp explicitly takes into account this information via direct queries as in (4.1) and (4.2).

As discussed in Section 3.4.2, the two main objectives involved in this control task are: (1) maintain the vehicle at the same horizontal lane with constant speed if no obstacles (other vehicles) are present; and (2) pass other moving vehicles if they are within a safety distance. We test C-GLISp with the same test scenario as in Section 3.4.2 to tune the same set of MPC design parameters, that is, the sampling time (T_s), prediction and control horizons (N_p, N_u), and weight matrix $Q_{\Delta u}$.

The role of the calibrator is played by myself. The maximum number of function evaluations N_{max} is set to 50, with $N_{init} = 10$. The default hyperparameters δ_E, δ_G and δ_S in (4.15) are set to 1, 1, and 0.5, respectively. The parameters σ, c_h and λ in (2.10) are set to 0.02, 1, and 1e-6, respectively. The hyperparameter ϵ characterizing the RBF function (4.11) is initialized to 1.0, and recalibrated at iterations 10, 20, 30, and 40 via 3-fold cross-validation.

The closed-loop experiment of the vehicle control is simulated for 15 seconds. Fig. 4.3 shows the query window for one iteration of the calibration process. Besides expressing a preference between the new experiment and the current best one, as in the case study discussed in Section 3.4.2, at each iteration, the calibrator is also asked to decide whether the newly proposed experiment is feasible and/or satisfactory. More specifically, the calibrator labels the control policy that leads to "unstable/unsafe" and "unimplementable" behavior as infeasible (*i.e.*, $G(x) = 0$). Examples of "unstable/unsafe" behaviors include but are not limited to: vehicle hits the obstacle; vehicle oscillates on the road, *etc.* "Unimplementable" cases are the ones whose computational time (t_{comp}) required for solving the QP problem of MPC (2.10) exceeds the sampling time T_s. As for being labeled as satisfactory, the following two criteria are used: (i) the lateral position of the vehicle does not exceed 5 m during the OA period (black dashed line on Fig. 4.3); and (ii) the vehicle does not oscillate during the LK period (in other words, the vehicle moves at a constant speed with w_f and θ close to 0 m and $0°$). These feasibility and satisfactory criteria are assumed to be unknown to C-GLISp and are learned based on the expressed feasibility and satisfactory labels.

For the pairwise comparisons, the calibrator expresses her preferences according to the following guidelines: (i) whether it is feasible; (ii) whether it is satisfactory; (iii) whether the vehicle guarantees passengers' comfort during the OA period, for example, by not changing velocities or moving the lateral position too aggressively; (iv) whether the deviations of the vehicle velocity from the reference values is minor in both LK and OA periods; (v) whether aggressive variations of steering angles are avoided. If a conflict combination among criteria mentioned above appears, criterion (i) has the highest priority, and if the conflict is among criteria (ii)–(v), the control policy that leads to qualitatively safer driving practice based on the calibrator's experience is preferred. Note that conditions (iii)-(v) and the method of judging safe driving practice are mainly qualitative/subjective, and it is difficult to express them in terms of quantitative metrics.

For the example query window illustrated in Fig. 4.3, conflicting combinations of the assessing criteria are observed. Compared to the experiment shown in the left panels of the figure, the experiment shown in the right panels has more aggressive lateral movements during the OA period. Furthermore, the changes in velocity and steering angle are greater in both frequency and magnitude in both LK and OA periods. The experiment shown in the left panels is feasible since it is implementable ($t_{comp} < T_s$) and stable, while the experiment shown in the right panels of the figure is infeasible since t_{comp} exceeds T_s. Additionally, both experiments fail to satisfy the satisfaction conditions. Above all, the performance of the experiment shown in the left panels is preferred according to criterion (vi).

4.5.1 Results

C-GLISp terminates after 50 simulated closed-loop experiments and 49 pairwise comparisons. The best MPC design parameters T_s, ϵ_c, N_p, $\log(q_{u11})$ and $\log(q_{u22})$ are determined to be 0.085 s, 0.100, 23, -0.323 and -3.71, respectively, with a worst-case computational time $t_{comp} =$ 0.0789 s. The closed-loop performance obtained via these MPC design parameters is depicted in Fig. 4.4. As shown in the figure, after only 50 simulated experiments, the proposed algorithm can tune the MPC parameters to achieve feasible and satisfactory performance, accomplishing the driving tasks with smooth and safe maneuvers.

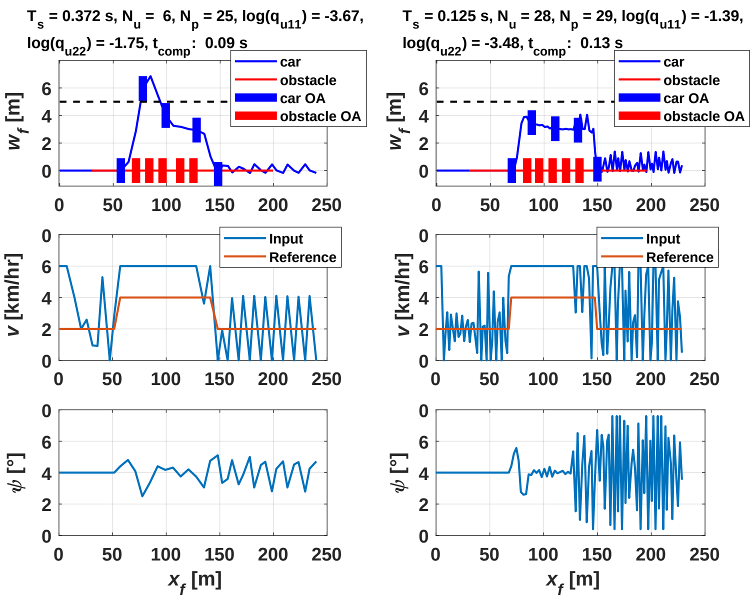


Figure 4.3: Vehicle control query window. The top subplots show the vehicle and obstacle positions. The "vehicle OA" and "obstacle OA" bars show five relative positions of the vehicle and obstacle during the OA period. The dashed lines indicate the lateral distance that the car should avoid exceeding (5 m in this case). The middle subplots show the actual and reference velocity v at different longitudinal positions. The steering angle ψ over the longitudinal position is depicted in the bottom subplots. The results on the left panels are preferred and feasible, while the results on the right panels are infeasible. The results on both sets of panels fail to satisfy the satisfaction conditions. ©2021 IEEE

4.6 Conclusions

The algorithm C-GLISp introduced in this chapter can handle preference-based global optimization with unknown objective functions and unknown constraints better than other existing black-box surrogate methods (PBO and GLISp), as illustrated through benchmark problems. The automated driving case study demonstrated the application of C-GLISp in semi-automated MPC calibration. Although convergence to global op-

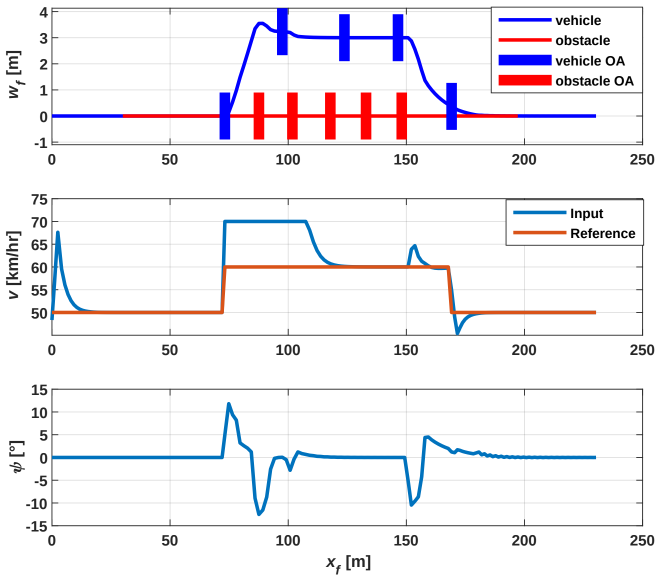


Figure 4.4: Final vehicle control performance obtained by the designed MPC controller. The top subplot shows the vehicle and obstacle positions. The "vehicle OA" and "obstacle OA" bars show five relative positions of the vehicle and obstacle during the OA period. The middle subplot shows the actual and reference velocity v at different longitudinal positions. The bottom subplot shows the steering angle ψ over the longitudinal position. ©2021 IEEE

timizers cannot be guaranteed, we observed that the C-GLISp can find satisfactory results within a small number of iterations and that it has a higher probability of proposing feasible samples during the exploration thanks to the introduction of additional information by the decision-maker that is used to synthesize corresponding surrogate functions. We finally note that, while we have used C-GLISp for controller calibration, the algorithm can be used in many other applications in which a few tuning parameters must be decided based on preferences under constraints that cannot be easily quantified.

Chapter 5

Global and Preference-based Optimization with Mixed Variables using Piecewise Affine Surrogates

5.1 Introduction

A large variety of decision problems in several application domains can have decision variables defined over a mixed-variable domain, *i.e.*, they can be of different types, such as continuous, integer, and categorical, which introduces a further challenge from an optimization perspective. Also, these problems frequently include constraints of mixed-integer nature, for example, constraints determined by logical conditions involving continuous and binary variables, and evaluating infeasible instances of the optimization variables may not be possible, for example, when the

The content of this Chapter is from M. Zhu and A. Bemporad, "Global and preference-based optimization with mixed variables using piecewise affine surrogates," *submitted for publication*, 2023.

corresponding function evaluation requires running a simulation or an experiment that is impossible or dangerous to execute. As a result, it is preferable to efficiently exploit the known admissible set of the problem to encourage feasible sampling.

Surrogate-based optimization techniques have been studied extensively to target black-box optimization problems with expensive-to-evaluate objective functions as discussed in the previous chapters. Although most of the literature has focused only on real-valued optimization variables, a few approaches have been adopted to handle integer and categorical variables [114]. Here, we distinguish integer variables as the ones representing ordinal relationships and categorical variables as those representing non-ordinal relationships. Integer variables are most commonly considered as continuous variables during the solution process and rounded to the nearest integer during post-analysis (*e.g.*, MISO [53]), while categorical variables are often first one-hot encoded and then treated as continuous variables in $[0, 1]$ when fitting the surrogate model, and then rounded and decoded after the optimization step (*e.g.*, MINOAN [115]). See also [116–118] for algorithms that have applied similar approaches to handle integer and categorical variables. Ploskas and Sahinidis [52] comprehensively analyzed and compared different algorithms, and related software packages, targeting bound-constrained mixed-integer derivative-free optimization problems. In their review, the authors observed that MISO [53] demonstrates superior performance when dealing with large (51 - 500 variables) and binary problems. On the other hand, NOMAD [56, 119] emerged as the top performer for mixed-integer, discrete (non-binary), small, and medium-sized (up to 50 variables) problems.

Most of the surrogate-based methods assume all the inputs as continuous and ordinal [52, 53, 115]. On the other hand, different classes for the categorical variables often represent different choices rather than ordinal relations. Therefore, if one attempts to fit the latent function using a unified surrogate in which categorical variables are one-hot encoded and treated as continuous vectors with entries in $[0, 1]$, sharp transitions might be observed in the constructed surrogate, leading to poor fitting qualities. Alternatively, one can fit different surrogate models to each categorical class [120–122]. However, as the number of categories and classes within each category increase, the size of the problem quickly blows up. To alleviate this issue, Ru et al. [123] propose an approach that makes efficient use of the information in the acquired data by combining the strengths of multi-armed bandits and BO based on Gaussian processes. This method has been shown to effectively solve bound-constrained problems with multiple categorical variables and multiple possible choices.

In addition to mixed variables, real-life optimization problems frequently contain constraints. In this case, if the integer and the one-hot encoded category variables are relaxed as continuous variables while optimizing, *i.e.*, the integrality of the variables is neglected, the constraints may not be satisfied after post-analysis, especially when equality constraints are present [115]. In [115], to maintain the integrality of the variables, the authors use one-hot encoding to convert integer and one-hot encoded categorical variables to auxiliary variables. However, infeasibility with respect to constraints is still allowed during the solution process in [115]. In [124], piecewise-linear neural networks are employed as surrogate models to address constrained discrete black-box optimization problems, and mixed-integer linear programming (MILP) is used to optimize the acquisition function. However, the no-good constraints used in [124] to tackle discrete-variable-only problems cannot be trivially transferred to the mixed-variable domain; hence, this approach cannot be directly applied to domains with mixed variables.

5.1.1 Contribution

In this chapter, we aim to solve medium-sized mixed-variable nonlinear optimization problems (say up to 100 variables after encoding) subject to mixed-integer linear equality and/or inequality constraints (say up to 20 constraints), where the optimization variables can be continuous, integer, and categorical. Specifically, we propose an algorithm that uses a piecewise affine (PWA) function as the surrogate, and we incorporate two types of exploration functions (distance-based and frequency-based) in the acquisition function to efficiently explore the feasible domain, therefore reducing the number of queries required to obtain satisfactory results. Moreover, as the initial samples play an essential role in fitting the surrogate, especially when the function-evaluation budget is limited, we propose incorporating the exploration function as part of the initial sampling strategy to obtain scattered initial samples when a large number of linear equality and inequality constraints are present.

We name the proposed algorithm as **PWAS**, short for **P**iece**w**ise **A**ffine **S**urrogate-based optimization. We show the efficiency and effectiveness of PWAS by comparing its performance with other existing solvers on a set of benchmark problems. We also present an extension of PWAS to solve problems in which function evaluations are unavailable, such as problems with multiple objectives whose relative weight is unclear or when only qualitative assessments are available, assuming that a decision-maker can express *preferences* between two candidate solution vectors. Such preference information is used to shape the PWA surrogate by the proposed algorithm that we name **PWASp**, short for **PWAS** based on **p**references. Python implementations of PWAS and PWASp are available on the GitHub repository (`https://GitHub.com/mjzhu-p/PWAS`).

The rest of the chapter is organized as follows. The description of the target problem is formulated in Section 5.2. The proposed surrogate-based optimization algorithms are discussed in Sections 5.3 and 5.4. Section 5.5 reports the numerical benchmarks demonstrating the effectiveness of the proposed method. Lastly, conclusions are discussed in Section 5.6.

5.2 Problem formulation

We consider a decision problem with n_c real variables grouped in vector $x \in \mathbb{R}^{n_c}$, n_{int} integer variables grouped in vector $y \in \mathbb{Z}^{n_{\text{int}}}$, and n_d categorical variables grouped in list $Z = [Z^1, \dots, Z^{n_d}]$, where each categorical variable Z^i can take values within its corresponding n_i classes,

$i = 1, \ldots, n_d$. Let us assume that each categorical variable Z^i is one-hot binary encoded into the subvector $[z_{1+d^{i-1}} \ \ldots \ z_{d^i}]^\mathsf{T} \in \{0,1\}^{n_i}$ for each $i = 1, \ldots, n_d$, where $d^0 = 0$, $d^i = \sum_{j=1}^{i} n_j$, and $z \in \{0,1\}^{d^{n_d}}$ is the complete vector of binary variables after the encoding, with $z \in \Omega_z = \{z \in \{0,1\}^{d^{n_d}} : \sum_{j=1}^{n_i} z_{j+d^{i-1}} = 1, \ \forall i = 1, \ldots, n_d\}$. Let $X = [x^\mathsf{T} \ y^\mathsf{T} \ z^\mathsf{T}]^\mathsf{T}$ denote the overall optimization vector. We assume that the vectors x and y of interest are bounded, *i.e.*, $\ell_x \leq x \leq u_x$ and $\ell_y \leq y \leq u_y$, and denote by $\Omega = [\ell_x, u_x] \times ([\ell_y, u_y] \cap \mathbb{Z}) \times \Omega_z$ the domain of X. Let $f : \Omega \mapsto \mathbb{R}$ be the objective function to minimize, that we consider noiseless and expensive to evaluate.

The black-box mixed-variable optimization problem we want to solve can be stated as follows:

$$\begin{aligned}
\text{find } X^* \in \quad & \arg\min_{X \in \Omega} f(X) && \text{(5.1a)}\\
\text{s.t.} \quad & A_{\text{eq}}x + B_{\text{eq}}y + C_{\text{eq}}z = b_{\text{eq}} && \text{(5.1b)}\\
& A_{\text{ineq}}x + B_{\text{ineq}}y + C_{\text{ineq}}z \leq b_{\text{ineq}} && \text{(5.1c)}
\end{aligned}$$

where A_{eq}, B_{eq}, C_{eq} and A_{ineq}, B_{ineq}, C_{ineq} are matrices of suitable dimensions that, together with the right-hand-side column vectors b_{eq}, b_{ineq}, define possible linear equality and inequality constraints on x, y, and z. For example, if $x \in \mathbb{R}$, $Z = [Z^1]$, $Z^1 \in \{\mathsf{red}, \mathsf{blue}, \mathsf{yellow}\}$, the logical constraint $[Z^1 = \mathsf{red}] \to [x \leq 0]$ can be modeled as $x \leq u_x(1 - z_1)$. For modeling more general types of mixed linear/logical constraints, possibly involving the addition of auxiliary real and binary variables, the reader is referred to, e.g., [125–127]. Note that, different from function f, which is assumed to be black-box and expensive to evaluate, we assume the mixed-integer linear constraints on X in (5.1) have known algebraic form and are cheap to evaluate.

5.3 Solution method

We follow the general surrogate-based optimization procedure (see *e.g.*, Figure 2.1) to solve (5.1). The approach consists of an initial (passive) sampling and an active learning stage, in which a surrogate model of

Algorithm 5.1 PWAS: Global optimization using piecewise affine surrogates

Input: Lower and upper bounds ℓ_x, u_x, ℓ_y, u_y; linear constraint matrices A_{eq}, B_{eq}, C_{eq} and A_{ineq}, B_{ineq}, C_{ineq} and right-hand-side vectors b_{eq} and b_{ineq}; number n_d of categorical variables and n_i of possible categories, $i = 1, \ldots, n_d$; initial number K of polyhedral partitions; number $N_{\text{init}} \geq 2$ of initial samples, number $N_{\max} \geq N_{\text{init}}$ of maximum function evaluations; $\delta_1 \geq 0$, $\delta_2 \geq 0$ and $\delta_3 \geq 0$ if solve (5.11) in one step or $\delta \geq 0$ if solve (5.11) in multiple steps; solving strategy for (5.11): {"one-step" or "multi-steps"}.

```
1. Pre-process the optimization variables as described in Section 5.3.1;
2. $N \leftarrow 1$, $N^*_{\text{curr}} \leftarrow 1$, $f^*_{\text{curr}} \leftarrow +\infty$ ;
3. Generate $N_{\text{init}}$ random scaled and encoded samples $\bar{X} = \{\bar{X}_1, \ldots, \bar{X}_{N_{\text{init}}}\}$ using one of the initial sampling methods reported in Section 5.3.5 based on the problem setup;
4. **While** $N \leq N_{\max}$ **do**
    4.1. Scale back and decode $\bar{X}_N$ to $X_N$, *i.e.*, $X_N = S(\bar{X}_N)$, and query $f_N = f(X_N)$;
    4.2. **If** $f_N < f^*_{\text{curr}}$ **then update** $N^*_{\text{curr}} \leftarrow N$, $f^*_{\text{curr}} \leftarrow f_N$;
    4.3. **If** $N \geq N_{\text{init}}$ **then**
        4.3.1. Update and fit the PWA separation function $\phi$ and PWA surrogate function $\hat{f}$ as described in Section 5.3.2;
        4.3.2. Define the acquisition function $a$ as in (5.11);
        4.3.3. Solve the global optimization problem (5.11) and get $\bar{X}_{N+1}$ either in one-step or multi-steps;
    4.4. $N \leftarrow N + 1$;
5. **End**.
```

Output: Best decision vector $X^* = X_{N^*_{\text{curr}}}$ found.

the objective function f is repeatedly learned. In particular, here we propose fitting a *piecewise affine* surrogate of the latent objective function f, to have two main benefits: (i) allow discontinuities introduced by sharp transitions induced by taking values in different classes of the categorical variables. In this case, instead of using one surrogate model for each categorical class as in [120–122], it is possible to adaptively update the number of partitions allowed in the PWA function by analyzing the clusters of the queried samples. For example, one can initiate the surrogate fitting procedure by setting a maximum allowed number of partitions and then discard some partitions if the number of queried samples within these partitions is smaller than some fixed minimum values (cf. [60]); (ii) PWA surrogates have a direct mixed-integer linear reformulation and, therefore, can be minimized by efficient MILP solvers (*e.g.*, Gurobi [128] and GLPK [129]). Also, we can explicitly reformulate and include linear equality and inequality constraints involving integer and one-hot encoded categorical variables in the standard MILP form to maintain their integrality during the solution process. We note that each new sample X_{k+1} is determined by minimizing an *acquisition function*, which combines the surrogate with an *exploration function*, to reach an exploitation/exploration tradeoff. To enable the possibility of making feasible queries during the acquisition step for problems with mixed-variable domain, we will also define a suitable PWA *exploration function* that admits a MILP representation. The resulting approach, which we call PWAS, is summarized in Algorithm 5.1, whose steps will be described in detail in the next sections.

5.3.1 Change of variables: scaling and encoding

Before attempting solving problem (5.1), we first rescale every continuous variable x_i into a new variable $\bar{x}_i \in [-1, 1]$ such that

$$x_i = \frac{u_x^i - \ell_x^i}{2}\bar{x}_i + \frac{u_x^i + \ell_x^i}{2}, \; \forall i = 1, \ldots, n_c.$$

Accordingly, the constraint matrices A_{eq}, A_{ineq} are rescaled to

$$\bar{A}_{\text{eq}} = A_{\text{eq}}\text{diag}\left(\frac{u_x - \ell_x}{2}\right), \quad \bar{A}_{\text{ineq}} = A_{\text{ineq}}\text{diag}\left(\frac{u_x - \ell_x}{2}\right)$$

and the right-hand-side vectors are updated as follows:

$$\bar{b}_{\text{eq}} = b_{\text{eq}} - A_{\text{eq}}\left(\frac{u_x + \ell_x}{2}\right), \quad \bar{b}_{\text{ineq}} = b_{\text{ineq}} - A_{\text{ineq}}\left(\frac{u_x + \ell_x}{2}\right).$$

The intervals $[-1, 1]$ for the continuous variables are possibly further tightened by taking the updated inequality constraints (5.1c) (if they exist) into account (cf. [32]), *i.e.*, for $i = 1, \dots, n_c$, we set

$$\begin{array}{ll} \bar{\ell}_x^i & = \min\limits_{\bar{x},y,z} e_i^\mathsf{T} [\bar{x}^\mathsf{T}\ y^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T} \\ \text{s.t.} & \bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \le \bar{b}_{\text{ineq}} \\ & \bar{x} \in [-1\ 1]^{n_c},\ y \in [\ell_y, u_y] \cap \mathbb{Z},\ z \in \Omega_z \end{array}$$

and, similarly,

$$\begin{array}{ll} \bar{u}_x^i & = \max\limits_{\bar{x},y,z} e_i^\mathsf{T} [\bar{x}^\mathsf{T}\ y^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T} \\ \text{s.t.} & \bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \le \bar{b}_{\text{ineq}} \\ & \bar{x} \in [-1\ 1]^{n_c},\ y \in [\ell_y, u_y] \cap \mathbb{Z},\ z \in \Omega_z \end{array}$$

where e_i denotes the ith column of the identity matrix of the same dimension as vector X. We denote by $\Omega_x = [\bar{\ell}_x^1, \bar{u}_x^1] \times \ldots \times [\bar{\ell}_x^{n_c}, \bar{u}_x^{n_c}]$ the resulting domain of the scaled continuous variables.

Let us assume that only a finite number $N_{\max}$ of queries can be made, which depends on the nature of function f (i.e., how expensive it is to evaluate) and the time available to solve the optimization problem. Moreover, we treat integer variables y differently depending on the relation between $N_{\max}$ and the number $\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}}$ of possible combinations of integer variables, where $n_i^{\text{int}} = \lfloor u_y^i \rfloor - \lceil \ell_y^i \rceil + 1$ is the cardinality of the set $[l_y^i, u_y^i] \cap \mathbb{Z}$, *i.e.*, the number of integer values that variable y_i can take. In particular, as described in Section 5.3.1.1 below, to make the exploration of the search space possibly more efficient, we will treat the vector y of integer variables as categorical when solving (5.1) in case $\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}} < N_{\max}$, *i.e.*, when it may possible to exhaustively list out all the potential combinations of the integer variables within $N_{\max}$ queries if no continuous or categorical variables are present; vice versa, as we will detail in Section 5.3.1.2, we will maintain the optimization variables

y_i integer. We note that this is a general heuristic we applied, which was empirically observed to be more efficient when handling integer variables.

5.3.1.1 Treating integer variables as categorical

The first scenario occurs when the number of possible combinations of integer variables $\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}} < N_{\max}$. In this case, we treat all integer variables y_i as categorical, similarly to vector z, and one-hot encode them into further $d^{n_{\text{int}}}$ binary variables $\bar{y}_j \in \{0,1\}$, $j = 1,\ldots,d^{n_{\text{int}}}$, where $d^{n_{\text{int}}} = \sum_{i=1}^{n_{\text{int}}} n_i^{\text{int}}$. We also define $\Omega_y = \{\bar{y} \in \{0,1\}^{d^{n_{\text{int}}}} : \sum_{j=1}^{n_i^{\text{int}}} \bar{y}_{j+d_y^{i-1}} = 1,\ \forall i = 1,\ldots,n_{\text{int}}\}$, where $d_y^0 = 0$ and $d_y^i = \sum_{j=1}^{i} n_j^{\text{int}}$ for $i = 1,\ldots,n_{\text{int}}$, and set $\bar{y} \in \Omega_y$. The constraint matrix B_{eq} (B_{ineq}) is modified accordingly into a new matrix $\bar{B}_{\text{eq}}$ ($\bar{B}_{\text{ineq}}$) by replacing each scalar entry B_{eq}^{ij} (B_{ineq}^{ij}) with the row vector obtained by multiplying the entry by the vector of integers that variable y_j can take, *i.e.*,

$$\begin{aligned}
B_{\text{eq}}^{ij} &\leftarrow B_{\text{eq}}^{ij}\left[\lceil \ell_y^j \rceil \ \ldots \ \lfloor u_y^j \rfloor\right] \in \mathbb{R}^{1\times n_j^{\text{int}}}, \quad \forall j = 1,\ldots,n_{\text{int}}\\
B_{\text{ineq}}^{ij} &\leftarrow B_{\text{ineq}}^{ij}\left[\lceil \ell_y^j \rceil \ \ldots \ \lfloor u_y^j \rfloor\right] \in \mathbb{R}^{1\times n_j^{\text{int}}}, \quad \forall j = 1,\ldots,n_{\text{int}}.
\end{aligned}$$

The new optimization vector becomes $\bar{X} = [\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T} \in \bar{\Omega}$, where $\bar{\Omega} = \Omega_x \times \Omega_y \times \Omega_z$, and consists of $n = n_c + d^{n_{\text{int}}} + d^{n_d}$ variables. As evaluating the objective function in (5.1) requires the original values in X, we denote by $S : \bar{\Omega} \mapsto \Omega$ the inverse scaling/encoding mapping of $\bar{X}$, *i.e.*, $X = S(\bar{X})$. According to such a change of variables, problem (5.1) is now translated to

$$\begin{aligned}
\text{find } \bar{X}^* \in \arg\min_{\bar{X}\in\bar{\Omega}} & f(S(\bar{X}))\\
\text{s.t. } & \bar{A}_{\text{eq}}\bar{x} + \bar{B}_{\text{eq}}\bar{y} + C_{\text{eq}}z = \bar{b}_{\text{eq}}\\
& \bar{A}_{\text{ineq}}\bar{x} + \bar{B}_{\text{ineq}}\bar{y} + C_{\text{ineq}}z \leq \bar{b}_{\text{ineq}}.
\end{aligned} \tag{5.2a}$$

In the sequel, $D \subseteq \bar{\Omega}$ will denote the set of admissible vectors $\bar{X}$ satisfying the constraints in (5.2a).

5.3.1.2 Scaling integer variables

In the second scenario, $\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}} \geq N_{\max}$, the integer variables are also rescaled and treated as numeric variables $\bar{y}_i \in [-1, 1]$, $i = 1, \ldots, n_{\text{int}}$. In this case, we also keep the original n_{int} integer variables $y_i \in \mathbb{Z}$ in the model for the sole purpose of enforcing integrality constraints, as we link them with $\bar{y}_i$ by the scaling factors

$$y_i = \frac{u_y^i - \ell_y^i}{2}\bar{y}_i + \frac{u_y^i + \ell_y^i}{2}.$$

Similar to the continuous variables, we can also further shrink the bounds on $\bar{y}_i$ by considering the updated inequality constraints (5.1c) (if present)

$$\begin{aligned}
\bar{\ell}_y^i \quad &= \min_{\bar{x},\bar{y},z} e_{n_c+i}^\mathsf{T}[\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T} \\
\text{s.t.} \quad & \quad \bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \leq \bar{b}_{\text{ineq}} \\
& \quad \bar{x} \in \Omega_x,\ \bar{y} \in [-1\ 1]^{n_{\text{int}}},\ z \in \Omega_z
\end{aligned}$$

$$\begin{aligned}
\bar{u}_y^i \quad &= \max_{\bar{x},\bar{y},z} e_{n_c+i}^\mathsf{T}[\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T} \\
\text{s.t.} \quad & \quad \bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \leq \bar{b}_{\text{ineq}} \\
& \quad \bar{x} \in \Omega_x,\ y \in [-1\ 1]^{n_{\text{int}}},\ z \in \Omega_z.
\end{aligned}$$

We denote the domain of $\bar{y}$ after tightening as $\Omega_y = [\bar{\ell}_y^1, \bar{u}_y^1] \times \ldots \times [\bar{\ell}_y^{n_{\text{int}}}, \bar{u}_y^{n_{\text{int}}}]$. Accordingly, problem (5.1) is translated to

$$\begin{aligned}
\text{find } \begin{bmatrix} \bar{X}^* \\ y^* \end{bmatrix} \in \arg & \min_{\bar{X}\in\bar{\Omega}, y\in[\ell_y,u_y]\cap\mathbb{Z}} f(S(\bar{X})) \\
\text{s.t. } & \bar{A}_{\text{eq}}\bar{x} + B_{\text{eq}}y + C_{\text{eq}}z = \bar{b}_{\text{eq}} \\
& \bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \leq \bar{b}_{\text{ineq}}
\end{aligned} \tag{5.2b}$$

where now $\bar{X} = [\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T} \in \bar{\Omega}$ and consists of $n = n_c + n_{\text{int}} + d^{n_d}$ variables, $\bar{\Omega} = \Omega_x \times \Omega_y \times \Omega_z$, and $S : \bar{\Omega} \mapsto \Omega$ is the new inverse scaling mapping. We will denote by $D \subseteq \bar{\Omega}$ the set of admissible vectors $\bar{X}$ such that the constraints in (5.2b) are satisfied for some vector $y \in [\ell_y, u_y] \cap \mathbb{Z}$.

5.3.2 Piecewise affine surrogate function

When fitting a surrogate of the objective function, we treat the modified vector $\bar{X}$ as a vector in $\mathbb{R}^n$. We describe next how to construct a PWA surrogate function $\hat{f} : \mathbb{R}^n \mapsto \mathbb{R}$ such that $\hat{f}(\bar{X})$ approximates $f(S(\bar{X}))$.

Consider N samples $\bar{X}_1, \ldots, \bar{X}_N \in \mathbb{R}^n$ and their corresponding function evaluations $f(S(\bar{X}_1)), \ldots, f(S(\bar{X}_N)) \in \mathbb{R}$. We want to define the PWA surrogate function $\hat{f}$ over a polyhedral partition of $\bar{\Omega}$ into K regions. To this end, we consider the following convex *PWA separation function* $\phi : \mathbb{R}^n \mapsto \mathbb{R}$

$$\phi(\bar{X}) = \omega_{j(\bar{X})}^\mathsf{T} \bar{X} + \gamma_{j(\bar{X})} \tag{5.3a}$$

where $\omega_j \in \mathbb{R}^n$ and $\gamma_j \in \mathbb{R}$, $j = 1, \ldots, K$, need to be determined, and

$$j(\bar{X}) = \arg \max_{j=1,\ldots,K} \{\omega_j^\mathsf{T} \bar{X} + \gamma_j\}, \tag{5.3b}$$

and we define the *PWA surrogate function* $\hat{f}$ as

$$\hat{f}(\bar{X}) = a_{j(\bar{X})}^\mathsf{T} \bar{X} + b_{j(\bar{X})} \tag{5.3c}$$

where $a_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$, $j = 1, \ldots, K$, also need to be determined. Note that $\hat{f}$ is possibly non-convex and discontinuous.

We use the PARC algorithm recently proposed in [60] to fit the PWA separation and surrogate functions and obtain the required coefficients $\omega_j, \gamma_j, a_j, b_j$ for $j = 1, \ldots, K$ (see also in Section 2.4.1 for an overview of PARC). We stress that while the closed-form expression of $f \circ S$ as a function of $\bar{X}$ is generally unavailable and very expensive to evaluate for each given $\bar{X}$, evaluating its surrogate $\hat{f}$ is very cheap and, as we will show in Section 5.3.2.1, admits a simple mixed-integer linear encoding with K binary variables.

We also remark that our purpose for the current study is to obtain a highly accurate approximation of the objective function *around the global optimal solution* and not necessarily over the entire domain of $\bar{X}$, which usually requires much fewer samples as in the illustrative examples discussed in Section 2.4.1. It is because, as the algorithm adaptively queries points to test from the domain, the partitions associated with higher function evaluations, which we are less interested in for optimization purposes, will be sampled less frequently and accurate prediction models for these partitions are not necessary. On the other hand, the regions with promising test points will be more frequently visited, resulting in better (more accurate) PWA surrogates within these partitions.

5.3.2.1 Mixed-integer linear encoding of the surrogate

After learning the coefficients of ϕ and $\hat{f}$ by applying the PARC algorithm, in order to optimize over the surrogate function to acquire a new sample $\bar{X}_{N_1}$ by MILP, as we will describe in Section 5.3.4, we introduce K binary variables $\zeta_j \in \{0,1\}$ and K real variables $v_j \in \mathbb{R}$, $j = 1, \dots, K$, where $\zeta_j = 1$ if and only if $\bar{X}_{N+1}$ belongs to the jth polyhedral region of the partition induced by ϕ. The PWA separation function ϕ can be modeled by the following mixed-integer inequalities via the big-M method [60]:

$$\begin{aligned}&\omega_j^\mathsf{T}\bar{X}_{N+1} + \gamma_j \geq \omega_h^\mathsf{T}\bar{X}_{N+1} + \gamma_h - M_\phi(1-\zeta_j),\ \forall h = 1,\dots,K, h \neq j\\ &\sum_{j=1}^{K}\zeta_j = 1\end{aligned} \tag{5.4}$$

where M_ϕ is a large-enough constant, *i.e.*, satisfies the inequality

$$M_\phi \geq \max_{j,h=1,\dots,K,\ \bar{X}\in D} (\omega_h - \omega_j)^\mathsf{T}\bar{X} + \gamma_h - \gamma_j.$$

The PWA surrogate function $\hat{f}$ can be modeled by setting

$$\hat{f}(\bar{X}) = \sum_{j=1}^{K}\zeta_j(a_j^\mathsf{T}\bar{X}_{N+1} + b_j) = \sum_{j=1}^{K} v_j$$

subject to

$$\begin{aligned}v_j &\leq a_j^\mathsf{T}\bar{X}_{N+1} + b_j - M_{sj}^-(1-\zeta_j)\\ v_j &\geq a_j^\mathsf{T}\bar{X}_{N+1} + b_j - M_{sj}^+(1-\zeta_j)\\ v_j &\geq M_{sj}^-\zeta_j\\ v_j &\leq M_{sj}^+\zeta_j,\end{aligned} \tag{5.5}$$

where M_{sj}^+, M_{sj}^- are large-enough constants satisfying the inequalities

$$M_{sj}^+ \geq \max_{\bar{X} \in D} a_j^\mathsf{T} \bar{X} + b_j, \quad M_{sj}^- \leq \min_{\bar{X} \in D} a_j^\mathsf{T} \bar{X} + b_j$$

for $j = 1, \ldots, K$.

5.3.3 Exploration function

Solely minimizing the surrogate function $\hat{f}$ may easily miss the global optimum. In order to properly explore the admissible set D we introduce an *exploration function* $E : \bar{\Omega} \mapsto \mathbb{R}$ to be coupled with the surrogate function to form the *acquisition function*, which will be optimized to find the next sample to test. Due to the different numerical properties of continuous, integer, and categorical variables, we consider different exploration strategies for each of them that admit a MILP representation. Specifically, we use a distance-based exploration method for continuous and integer variables if the latter are not one-hot encoded (as described in Section 5.3.1.2) and a frequency-based exploration method for one-hot encoded categorical and integer variables (in the alternative scenario described in Section 5.3.1.1). In the following, we discuss the distance-based and frequency-based methods in a general manner, and we will dive into specifics of the exploration functions for our problem of interest when we discuss the acquisition function in Section 5.3.4.

5.3.3.1 Distance-based exploration: "max-box" method

We want to define a function $E_{ct} : \mathbb{R}^{n_{ct}} \mapsto \mathbb{R}$ mapping a generic numeric vector $\bar{x} \in \mathbb{R}^{n_{ct}}$ into a nonnegative value $E_{ct}(\bar{x})$ that is zero at given samples $\bar{x}_1, \ldots, \bar{x}_N$, grows away from them, and admits a PWA representation. To this end, we consider the boxes $B_i(\beta_{ct}) = \{\bar{x} : \|\bar{x} - \bar{x}_i\|_\infty \leq \beta_{ct}\}$ and set $E_{ct}(\bar{x}) = \min\{\beta_{ct} \geq 0 : \bar{x} \in B_i(\beta_{ct}) \text{ for some } i = 1, \ldots, N\}$. Then, maximizing $E_{ct}(\bar{x})$ is equivalent to finding the largest value β_{ct} and a vector $\bar{x}^*$ outside the interior of all boxes $B_i(\beta_{ct})$, a problem that can be solved by the following MILP

$$\begin{aligned}
\bar{x}^* \in \quad & \arg \max_{\bar{x}, \beta_{ct}, \delta^+, \delta^-} \beta_{ct} \\
\text{s.t} \quad & \bar{x}^l - \bar{x}_i^l \geq \beta_{ct} - M_E(1 - \delta_{il}^+), \quad \forall l = 1, \ldots, n_{ct}, \quad \forall i = 1, \ldots, N \\
& -\bar{x}^l + \bar{x}_i^l \geq \beta_{ct} - M_E(1 - \delta_{il}^-), \quad \forall l = 1, \ldots, n_{ct}, \quad \forall i = 1, \ldots, N \\
& \delta_{il}^+ \leq 1 - \delta_{il}^-, \quad \forall l = 1, \ldots, n_{ct}, \quad \forall i = 1, \ldots, N \\
& \sum_{l=1}^{n_{ct}} \delta_{il}^+ + \delta_{il}^- \geq 1, \quad \forall i = 1, \ldots, N \\
& \beta_{ct} \geq 0, \; \bar{x} \in D
\end{aligned} \tag{5.6}$$

where l denotes the lth component of vector $\bar{x}$, $\delta_{il}^-, \delta_{il}^+ \in \{0, 1\}$ are auxiliary optimization variables introduce to model the violation of at least one of the linear inequalities that define the box $B_i(\beta_{ct})$, and M_E is a large-enough constant satisfying the following inequality

$$M_E \geq 2 \left(\max_{l=1,\ldots,n_{ct}} \bar{u}_x^l - \min_{l=1,\ldots,n_{ct}} \bar{\ell}_x^l \right)$$

where $\bar{u}_x^l$ and $\bar{\ell}_x^l$ are the upper and lower bounds, respectively, of the lth component of vector $\bar{x}$.

Figure 5.1 shows an example where we apply the max-box exploration method to $\bar{x} \in \mathbb{R}^2$ and $D = [-3, 9] \times [-2, 8]$. We start with three existing samples $\bar{x}_1, \bar{x}_2, \bar{x}_3$. After 20 iterations, we get the samples reported in the figure, which shows that, indeed, the max-box exploration method effectively explores the feasible region D.

5.3.3.2 Frequency-based exploration: "Hamming distance" method

Unlike the case of continuous variables treated in the previous section, to account for the frequency of occurrence of a particular combination of binary variables we use the Hamming distance, defined as follows: given two binary vectors $z = [z^1, \ldots, z^d]^\mathsf{T} \in \{0, 1\}^d$ and $z_i = [z_i^1 \; \ldots \; z_i^d]^\mathsf{T} \in \{0, 1\}^d$, the Hamming distance between z and z_i is defined by the number

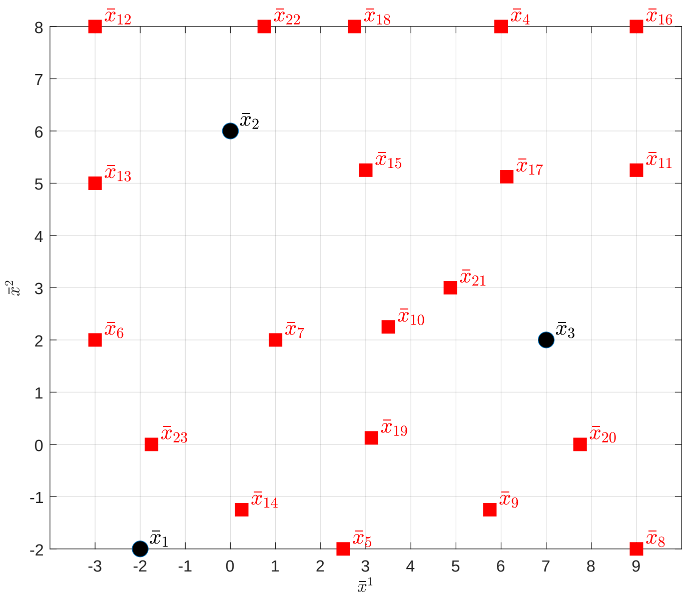


Figure 5.1: Illustrative example of the max-box exploration function in 2D. The black dots denote the initial samples. The red squares denote the samples generated using the max-box exploration method. The subscript number indicates the order of the point generated.

of different components between them

$$d_H(z, z_i) = \sum_{m=1}^{d} |z^m - z_i^m| \tag{5.7}$$

which can be encoded as the following linear expression

$$d_H(z, z_i) = \sum_{m:z_i^m=0} z^m + \sum_{m:z_i^m=1} (1 - z^m). \tag{5.8}$$

We consider the exploration function $E_{dt} : \{0, 1\}^d \mapsto \mathbb{R}$ such that $E_{dt}(z)$ quantifies the average number of different binary components between z and the given N vectors $z_1, \ldots, z_N$

$$E_{dt}(z) = \frac{1}{dN} \sum_{i=1}^{N} d_H(z, z_i).$$

Hence, a binary vector z^* with maximum average Hamming distance $E_{dt}(z^*)$ from the current samples $z_1, \ldots, z_N$ can be determined by solving the following MILP

$$z^* \in \arg\max_{z \in D} \; E_{dt}(z). \tag{5.9}$$

Table 5.1 shows an example in which we have three categorical variables $Z = [Z^1, Z^2, Z^3]$, where $Z^1 \in \{A, B\}$, $Z^2 \in \{A, B, C, D, E\}$, and $Z^3 \in \{A, B, C\}$. We start with three initial samples $Z_1 = [A, E, C]$, $Z_2 = [B, B, B]$, and $Z_3 = [A, D, C]$. First, we binary encode the categorical variables, getting the corresponding vectors z_1, z_2, $z_3 \in \{0, 1\}^{10}$. Then, we solve the optimization problem (5.9) to identify $z_4 = z^*$ and its corresponding decoded form Z_4. The table shows the categorical values $Z_4, \ldots, Z_{23}$ generated in 20 subsequent iterations, which shows that a diverse set of categorical variables are obtained when applying the Hamming distance exploration method.

Table 5.1: Illustrative example of the Hamming distance exploration fun.

Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Z^1	B	A	B	A	B	A	B	A	B	A
Z^2	A	C	D	A	E	B	C	D	A	E
Z^3	A	B	A	C	A	B	C	A	B	C
Iteration	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Z^1	B	A	B	A	B	A	B	A	B	A
Z^2	B	C	D	A	E	B	C	D	A	E
Z^3	A	B	C	A	B	C	A	B	C	A

5.3.4 Acquisition function

The surrogate and exploration functions defined in Sections 5.3.2 and 5.3.3 can be combined into the following *acquisition problem*

$$\text{find } \bar{X}^* \in \arg\min_{\bar{X} \in D} \hat{f}(\bar{X}) - \delta(E_{ct}(\bar{x}) + E_{dt}([\bar{y}^\mathsf{T} \; z^\mathsf{T}]^\mathsf{T})) \tag{5.10a}$$

when integer variables are treated as categorical as described in Section 5.3.1.1, or into

$$\text{find } \begin{bmatrix} \bar{X}^* \\ y^* \end{bmatrix} \in \arg \min_{\bar{X} \in D, y \in [\ell_y, u_y] \cap \mathbb{Z}} \hat{f}(\bar{X}) - \delta(E_{ct}([\bar{x}^\mathsf{T} \; \bar{y}^\mathsf{T}]^\mathsf{T}) + E_{dt}(z)) \tag{5.10b}$$

when the integer vector y is scaled as described in Section 5.3.1.2. In (5.10), the nonnegative scalar δ is called the *exploration parameter* and decides the tradeoff between exploiting the surrogate $\hat{f}(\bar{X})$ and promoting the exploration of the feasible domain D. In the sequel, we will refer to the cost function $a : \bar{\Omega} \mapsto \mathbb{R}$ in (5.10a) or (5.10b) as the *acquisition function*. By construction, Problem (5.10) can be solved by MILP. An optimal vector $\bar{X}^*$ of its solution, once scaled and decoded back, defines the next sample X_{N+1} to query for the corresponding function value $f_{N+1} = f(X_{N+1})$. Note that X_{N+1} satisfies all the constraints in (5.1) since $\bar{X}^* \in D$.

The direct formulation (5.10) can be further improved to ease the selection of δ and make the exploration more homogenous with respect to all types of variables (continuous, integer, or categorical). In fact, the formulation in (5.10) has the following possible drawbacks:

(i) The relative magnitude between $\hat{f}(\bar{X})$ and $E(\bar{X})$ is hard or impossible to estimate a priori, making the value for the exploration parameter δ hard to select.

(ii) By using the same exploration parameter δ for the exploration function of each type of variable, we implicitly assumed that the relative magnitude of each exploration function is comparable, which may not be the case.

(iii) When the integer variables y are not one-hot encoded as described in Section 5.3.1.2, the max-box exploration function is applied to the combined vector $[\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}]^\mathsf{T}$ (see (5.10b)) and two problems can occur. Firstly, as shown in (5.2), even though $\bar{y}_i$ is a continuous variable, because of the presence of the corresponding auxiliary integer variable y_i, it can only be changed in discrete steps, unlike the remaining variables $\bar{x}_j$. As a result, when finding the max box, one unit change in an integer variable can be reflected as a more significant change of its corresponding scaled variable $\bar{y}$, therefore promoting the exploration of directions with more variations in the integer variables $\bar{y}_i$ than in the continuous variables $\bar{x}_j$. Secondly, due to possibly different lower and upper bounds and therefore scaling factors of integer variables, unit changes of them may cause different changes in size of their corresponding scaled variables.

To address the aforementioned issues, given N samples $(\bar{X}_i, f(S(\bar{X}_i))$, $i = 1, \ldots, N$, we reformulate the acquisition problems (5.10), respectively, as follows:

$$\text{find } \bar{X}_{N+1} \in \arg \min_{\bar{X} \in D} \frac{\hat{f}(\bar{X})}{\Delta F} - \delta_1 E_{ct}(\bar{x}) - \delta_2 E_{ct}(\bar{y}) - \delta_3 E_{dt}(z) \tag{5.11a}$$

$$\text{find } \begin{bmatrix} \bar{X}_{N+1} \\ y_{N+1} \end{bmatrix} \in \arg \min_{\bar{X} \in D, y \in [\ell_y, u_y] \cap \mathbb{Z}} \frac{\hat{f}(\bar{X})}{\Delta F} - \delta_1 E_{ct}(\bar{x}) - \delta_2 E_{dt}(\bar{y}) - \delta_3 E_{dt}(z) \tag{5.11b}$$

where

$$\Delta F = \max \left\{ \max_{i=1,\ldots,N} f(X_i) - \min_{i=1,\ldots,N} f(X_i), \epsilon_{\Delta F} \right\}$$

and $\epsilon_{\Delta F} > 0$ is a threshold to prevent division by zero. The scaling factor ΔF eases the selection of the exploration parameters δ_1, δ_2, and δ_3 by making the surrogate term comparable to the exploration terms (cf. [32]).

An alternative to solve the optimization problem (5.11) in one step is to consider only one exploration term at a time, therefore solving the problem in three consecutive steps (that will be referred to as the "multi-step" approach) where, at each step, the problem is only solved with respect to one variable type. The remaining variables are treated as constants at either the value associated with the current best vector $X_{N^*_{\text{curr}}}$ or at the new value optimized during the multi-step operation. The advantage of serializing the optimization is that the relative value of δ_1, δ_2, and δ_3 is no longer relevant, and therefore we set $\delta_1 = \delta_2 = \delta_3 = \delta$, where δ is the only tradeoff hyperparameter to choose.

A further heuristic is applied to restrict the number of binary variables used to encode the max-box exploration function (5.6) and therefore limit them as the number N of samples increases. Specifically, given an upper bound N_{Emax} defined by the user depending on the computational power available, we only consider the most recent N_S samples in the exploration function (we will use $N_S = 20$ in our experiments) when $Nn_c \geq N_{\text{Emax}}$ or $Nn_{\text{int}} \geq N_{\text{Emax}}$ (when integer variables are not one-hot encoded). Since, instead, the surrogate function is approximated using all the existing samples, the rationale behind the heuristic is that, as the number N of queried samples grows, the surrogate itself should already discourage the exploration around the older samples not included in the exploration term, where the surrogate function, most likely, takes large values.

5.3.5 Initial sampling strategies

The values of the initial samples $X_1, \ldots, X_{N_{\text{init}}}$ can significantly impact the final solution X^* obtained after $N_{\max}$ steps. Moreover, one of the main motivations of the proposed method is its ability to handle mixed-integer constraints on the optimization variables. We propose different initial sampling strategies to efficiently acquire N_{init} scattered feasible samples depending on the constraints and types of optimization variables present in the problem:

(i) When only box constraints are present, we use the *Latin Hypercube Sampling* (LHS) [130] method as in [32].

(ii) When both box constraints and linear equality and/or inequality constraints are present, we consider the following alternatives:

- If only continuous variables are present, we use the Double Description Method [131] to generate the n_V vertices of the convex polytope given by the linear and box constraints. If $n_V < N_{\text{init}}$ and only inequality constraints are involved in (5.1), additional feasible samples can be generated via linear combinations of the vertices; if $n_V < N_{\text{init}}$ and equality constraints are also present, the generated n_V vertices can be used to define initial boxes and additional scattered feasible samples are generated by solving MILPs sequentially with the max-box exploration function discussed in Section 5.3.3.1 as the objective function. In this way, the constraints in (5.1) can be enforced in the formulation.

- If integer and/or categorical variables are also present, the algorithm first attempts to generate samples using LHS and filters out the infeasible ones. If the number of generated feasible samples is insufficient, which may happen when the constraints are hard to fulfill by random sampling, we generate scattered samples by solving MILPs sequentially using the exploration functions discussed in Section 5.3.3 as the objective functions and incorporating the given constraints to ensure sample feasibility.

5.4 Preference-based learning

We want to extend the global optimization method introduced in the previous sections to handle cases in which quantifying an objective function $f(X)$ as in (5.1) can be hard, if not impossible. Similar to GLISp [38] (see also in Section 2.3.1), we define the following *preference function* $\pi : \Omega \times \Omega \to \{-1, 0, 1\}$

$$\pi(X_1, X_2) = \begin{cases} -1 & \text{if } X_1 \text{ ``better'' than } X_2 \\ 0 & \text{if } X_1 \text{ ``as good as'' } X_2 \\ 1 & \text{if } X_2 \text{ ``better'' than } X_1. \end{cases} \tag{5.12}$$

In this case, we are interested in finding a feasible optimization vector X^* that wins or ties the pairwise comparisons with any other feasible X according to the preference function π, *i.e.*, the optimization problem (5.1) is replaced by

$$\text{find } X^* \text{ such that } \pi(X^*, X) \leq 0,\ \forall X \in D. \tag{5.13}$$

We describe next a variant of Algorithm 5.1, that we call as PWASp, for solving the preference-based optimization problem (5.13).

Let the optimization vector X be first pre-processed to $\bar{X}$ (*e.g.*, scaling and/or encoding) as described in Section 5.3.1. Given N samples $\bar{X}_1, \ldots, \bar{X}_N$ and M_c preferences $\pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) \in \{-1, 0, 1\}$, for $k = 1, \ldots, M_c$, where $M_c = N - 1$, we aim to fit a PWA surrogate model reflecting the preference relations among different samples. Since function evaluations are not available, here, the preferences $\pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k}))$ are used to shape the surrogate function $\hat{f}(\bar{X})$ by imposing the following constraints:

$$\begin{aligned} \hat{f}(\bar{X}_{1,k}) &\le \hat{f}(\bar{X}_{2,k}) - \sigma \quad \forall k:\ \pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = -1 \\ \hat{f}(\bar{X}_{2,k}) &\le \hat{f}(\bar{X}_{1,k}) - \sigma \quad \forall k:\ \pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = 1 \\ |\hat{f}(\bar{X}_{1,k}) - \hat{f}(\bar{X}_{2,k})| &\le \sigma \quad \forall k:\ \pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = 0 \end{aligned} \tag{5.14}$$

where $(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = (X_{1,k}, X_{2,k})$ are pairs of compared samples, $X_{1,k}$, $X_{2,k} \in \{X_1, \dots, X_N\}$, $k = 1, \dots, M_c$, and $\sigma > 0$ is a given constant, used to avoid the trivial solution $\hat{f}(\bar{X}) \equiv 0$.

To identify the PWA separation function $\phi(\bar{X})$, we first use K-means [132] to cluster the samples, and then use softmax regression [133, 134] to fit the coefficients. The assignment $j(\bar{X})$ of each sample $\bar{X}$ to each region of the partition is then determined. Following that, different from the PARC algorithm, we determine the coefficients a_j, b_j defining the PWA surrogate function $\hat{f}(\bar{X})$ by minimizing the sum $\sum_{k=1}^{M_c} \epsilon_k$ of the violations of the preference constraints (5.14) under an additional ℓ_∞-regularization term. Specifically, the coefficients a_j, b_j are obtained by solving the following linear programming (LP) problem:

$$\begin{aligned} \min_{\epsilon_k, \xi, a, b} \quad & \sum_{k=1}^{M_c} \epsilon_k + \alpha \xi \\ \text{s.t.} \quad & \hat{f}(\bar{X}_{1,k}) + \sigma \le \hat{f}(\bar{X}_{2,k}) + \epsilon_k \quad \forall k:\ \pi(X_{1,k}, X_{2,k}) = -1 \\ & \hat{f}(\bar{X}_{2,k}) + \sigma \le \hat{f}(\bar{X}_{1,k}) + \epsilon_k \quad \forall k:\ \pi(X_{1,k}, X_{2,k}) = 1 \\ & |\hat{f}(\bar{X}_{1,k}) - \hat{f}(\bar{X}_{2,k})| \le \sigma + \epsilon_k \quad \forall k:\ \pi(X_{1,k}, X_{2,k}) = 0 \\ & \xi \ge \pm a_j^l,\ l = 1, \dots, n \\ & \xi \ge \pm b_j \end{aligned} \tag{5.15}$$

where $\alpha > 0$ is the regularization parameter, $\xi \in \mathbb{R}$ is a new optimization variable introduced to linearly encode the ℓ_∞-regularization of the coefficients, and l denotes the lth component of the vector.

After obtaining the surrogate model, the same procedure as in PWAS can be followed to construct the acquisition function, which is then optimized to identify the next sample $X_{N+1} = S(\bar{X}_{N+1})$ to compare with the current best vector $X_{N^*_{\rm curr}}$. The various steps involved in PWASp are summarized in Algorithm 5.2.

Algorithm 5.2 PWASp: Preference-based optimization using piecewise affine surrogates

Input: Lower and upper bounds ℓ_x, u_x, ℓ_y, u_y; linear constraint matrices $A_{\rm eq}$, $B_{\rm eq}$, $C_{\rm eq}$ and $A_{\rm ineq}$, $B_{\rm ineq}$, $C_{\rm ineq}$ and right-hand-side vectors $b_{\rm eq}$ and $b_{\rm ineq}$; number n_d of categorical variables and n_i of possible categories, $i = 1, \ldots, n_d$; initial number K of polyhedral partitions; number $N_{\rm init} \geq 2$ of initial samples to compare, maximum number $N_{\max} - 1$ of comparisons, $N_{\max} \geq N_{\rm init}$; $\delta_1 \geq 0$, $\delta_2 \geq 0$ and $\delta_3 \geq 0$ if solve (5.11) in one step or $\delta \geq 0$ if solve (5.11) in multiple steps; solving strategy for (5.11): {"one-step" or "multi-steps"}.

```
1. Pre-process the optimization variables as described in Section 5.3.1;
2. $N \leftarrow 1, i^* \leftarrow 1$;
3. Generate $N_{\rm init}$ random and encoded samples $\bar{X} = \{\bar{X}_1, \ldots, \bar{X}_{N_{\rm init}}\}$
   using one of the initial sampling methods described in Section 5.3.5
   based on the problem setup;
4. **While** $N < N_{\max}$ **do**
   4.1. **If** $N \geq N_{\rm init}$ **then**
        4.1.1. Update and fit the PWA separation function $\phi$ and PWA
               surrogate function $\hat{f}$ as described in Section 5.4;
        4.1.2. Define the acquisition function $a$ as in (5.11);
        4.1.3. Solve the MILP problem (5.11) and get $\bar{X}_{N+1}$, either in
               one-step or multi-steps;
   4.2. $i(N) \leftarrow i^*, j(N) \leftarrow N + 1$
   4.3. Query preference $\pi(X_{i(N)}, X_{j(N)})$;
   4.4. **If** $\pi(X_{i(N)}, X_{j(N)}) = 1$ **then set** $i^* \leftarrow j(N)$
   4.5. $N \leftarrow N + 1$;
5. **End**.
```

Output: Best vector $X^* = X_{i^*}$ encountered.

5.5 Optimization benchmarks

To illustrate the effectiveness of PWAS and PWASp in solving the target problems (5.1) and (5.13), we have considered various global optimization benchmarks: a set of standard nonlinear programming problems (NLP) [135] with diverse dimensions (up to 10-D with 11 linear inequality constraints), several integer linear problems (up to 30 integer variables), three unconstrained mixed-variable synthetic benchmarks and two unconstrained mixed-variable real-world benchmarks taken from [123], and two constrained mixed-variable synthetic problems.

Computations are performed on an Intel i7-8550U 1.8-GHz CPU laptop with 24GB of RAM. The MILP problem in the acquisition step is formulated with the PuLP library [136] and solved by Gurobi's MILP solver [128]. For space limitations, we report below the performance of PWAS/PWASp only on the mixed-variable benchmarks, referring the reader to the GitHub repository for more detailed results on other tested benchmarks.

For each benchmark, the function evaluations are fed into PWAS to fit the surrogate, while the explicit function expressions remain unknown to PWAS. As in [123] the benchmark problems are solved via maximization, we use the values $-f(X)$ when running PWAS. As for PWASp, the objective function serves as a synthetic decision-maker whose evaluations are only used to express the preference between two decision vectors, namely $\pi(X_1, X_2) = -1$ if $f(X_1) > f(X_2)$, $\pi(X_1, X_2) = 1$ if $f(X_1) < f(X_2)$, or zero otherwise. In other words, PWASp only has access to the queried preferences (5.12) and not the explicit function expressions nor their evaluations $f(X_N)$. Specifically, $N_{\max} - 1$ pairwise comparisons are obtained for each benchmark when solved by PWASp.

The performance of PWAS and PWASp on the unconstrained benchmarks are compared with the following solvers: CoCABO-auto [123], CoCABO-0.5 [123], One-hot BO [118], SMAC [54], TPE [55], and EXP3BO [121] as noted in [123] as well as MISO [53] and NOMAD [56, 119]. Summaries of these optimization methods are noted in Appendix B. CoCABO-auto and CoCABO-0.5 are selected because the authors noted

Table 5.2: Benchmark problem specifications.

Benchmark	n_c	n_{int}	n_d	n_i
Func-2C	2	0	2	$\{3, 3\}$
Func-3C	2	0	3	$\{3, 3, 3\}$
Ackley-5C	1	0	5	$\{17, 17, 17, 17, 17\}$
XG-MNIST	4	1	3	$\{2, 2, 2\}$
NAS-CIFAR10	21	1	5	$\{3, 3, 3\}$
Horst6-hs044-modified	3	4	2	$\{3, 2\}$
ros-cam-modified	2	1	2	$\{2, 2\}$

that they consistently show competitive performance [123]. MISO and NOMAD are selected since they are noted as the best performers among all the solvers tested in [52]. The settings of the first six algorithms compared are available in [123]. As for MISO and NOMAD, we kept their default algorithm settings, with integer and categorical variables declared as referenced by the corresponding solvers [53, 56, 119]. To have fair comparisons, we use the same initial and maximum number of iterations ($N_{\text{init}} = 20$ and $N_{\max} = 100$) as indicated in [123] to obtain a near-optimal value found on the synthetic and real-world benchmarks for all the solvers tested except for NOMAD. NOMAD starts the optimization process with an initial guess [56], which we generate via the LHS sampling method [130]. When solving the problems using PWAS or PWASp, the multi-step solution strategy is applied in the acquisition step with $\delta_1 = \delta_2 = \delta_3 = 0.05$ or $\delta_1 = \delta_2 = \delta_3 = 1$, respectively for PWAS and PWASp. The initial number K of polyhedral partitions is set to 20 for both PWAS and PWASp in all the benchmarks. We stress that here the function evaluations for PWASp are solely reported for performance comparisons and are not attainable to PWASp during optimization. Table 5.2 summarizes the tested benchmark problems, while a detailed description of the benchmarks is reported in Appendix A. The optimal values obtained by CoCaBO-auto, CoCaBO-0.5, One-hot BO, SMAC and TPE after the maximum number of evaluations were read from Figure 4 in [123] using GetData Graph Digitizer [137]. Regarding MISO and NOMAD (version 4), we retrieved their packages from the GitHub repository. We performed 20 random repetitions for the unconstrained synthetic problems and 10 random repetitions for the unconstrained real-world problems as reported in [123].

In general, PWAS and PWASp can solve the benchmark decision problems, with or without constraints, to near-optimal solutions within a

Table 5.3: Optimal value found on benchmark Func-2C [123] (max = 0.2063).

Algorithm	After 100 iterations		After 200 iterations	
	mean	std	mean	std
PWAS	0.2049	0.0022	0.2061	0.0002321
PWASp	0.1813	0.0443	0.1889	0.0449
CoCaBO-auto	0.1219	0.0172	0.2041	0.0057
CoCaBO-0.5	0.1352	0.01620	0.2041	0.0057
One-hot BO	0.009524	0.02158	0.01524	0.02064
SMAC	0.06381	0.01746	0.07714	0.01556
TPE	0.1273	0.0184	0.1743	0.01650
EXP3BO	0.05524	0.01429	0.1105	0.01650
MISO	0.2063	0.0000	0.2063	0.0000
NOMAD	0.1700	0.0736	0.1754	0.07557

Table 5.4: Optimal value found on benchmark Func-3C [123] (max = 0.7221).

Algorithm	After 100 iterations		After 200 iterations	
	mean	std	mean	std
PWAS	0.5282	0.2117	0.6450	0.0972
PWASp	0.4542	0.2078	0.5106	0.1665
CoCaBO-auto	0.4993	0.0299	0.6912	0.0169
CoCaBO-0.5	0.5371	0.0503	0.6991	0.0205
One-hot BO	0.007670	0.04956	0.1076	0.0606
SMAC	0.1084	0.04016	0.1965	0.0339
TPE	0.2672	0.0472	0.4914	0.5308
EXP3BO	0.1784	0.0393	0.2515	0.0330
MISO	0.7221	0.0000	0.7221	0.0000
NOMAD	0.6618	0.1610	0.6860	0.1615

Table 5.5: Optimal value found on benchmark Ackley-5C [123] (max = 0).

Algorithm	After 100 iterations		After 200 iterations	
	mean	std	mean	std
PWAS	-1.1148	0.4077	-0.7108	0.3320
PWASp	-1.8857	0.5795	-1.6462	0.5422
CoCaBO-auto	-2.5120	0.602	-1.9244	0.5512
CoCaBO-0.5	-2.8415	0.0488	-2.0073	0.0488
One-hot BO	-3.076	0.0483	-2.5341	0.3024
SMAC	-3.0073	0.2488	-1.710	0.2393
TPE	-3.4659	0.2000	-2.7976	0.2487
MISO	-1.6389	0.1388	-1.5582	0.06218
NOMAD	-2.0175	0.2015	-1.5467	0.01437

Note: the reported values for CoCaBO-auto, CoCaBO-0.5, One-hot BO, SMAC and TPE are read from Figure 4 in [123] using GetData Graph Digitizer [137]. Statistics are obtained over 10 runs.

small number of function evaluations or comparisons. In fact, as shown in Tables 5.3–5.7, the optimal values achieved by PWAS and PWASp after 100 iterations are often already better or comparable to the results obtained by the other solvers after 200 iterations. We also observe that PWAS performs consistently better than PWASp, due to the fact that it has access to function evaluations, while PWASp only receives pairwise comparisons. Nonetheless, in spite of the more limited information it gets, PWASp outperforms several other solvers in most of the tested benchmarks.

Regarding problems with constraints, we consider the two mixed-variable synthetic problems reported in Appendix A.3. We do not consider other solvers than PWAS and PWASp as they either do not support constraint handling with mixed variables or allow infeasible samples during the optimization process. Thus, a systematic comparison

Table 5.6: Optimal value found on benchmark XG-MNIST [123].

Algorithm	After 100 iterations		After 200 iterations	
	mean	std	mean	std
PWAS	0.9585	0.0030	0.9609	0.0029
PWASp	0.9576	0.0036	0.9615	0.0028
CoCaBO-auto	0.9639	0.0004	0.9653	0.0004
CoCaBO-0.5	0.9731	0.0008	0.9741	0.0008
One-hot BO	0.9541	0.0019	0.9556	0.0015
SMAC	0.9651	0.0012	0.9681	0.0012
TPE	0.9656	0.0007	0.9679	0.0007
EXP3BO	0.9691	0.0005	0.9706	0.0005
MISO	0.9574	0.0071	0.9594	0.0078
NOMAD	0.9528	0.0138	0.9564	0.0146

Table 5.7: Optimal value found on benchmark NAS-CIFAR10 [123].

Algorithm	After 100 iterations		After 200 iterations	
	mean	std	mean	std
PWAS	0.9440	0.0024	0.9462	0.0016
PWASp	0.9409	0.0052	0.9454	0.0019
CoCaBO-auto	0.9446	0.0017	0.9454	0.0017
CoCaBO-0.5	0.9458	0.0014	0.9468	0.0004
One-hot BO	0.9438	0.0006	0.9451	0.0006
SMAC	0.9422	0.0004	0.9436	0.0004
TPE	0.9427	0.0006	0.9443	0.0007
MISO	0.9440	0.0030	0.9452	0.0025
NOMAD	0.9292	0.0331	0.9338	0.0241

Note: the reported values for CoCaBO-auto, CoCaBO-0.5, One-hot BO, SMAC and TPE are read from Figure 4 in [123] using GetData Graph Digitizer [137]. Statistics are obtained over 10 runs.

is not performed for the constrained problems. Instead, the results are compared against the analytic global optimum. Here, we set $N_{\max} = 100$ and $N_{\text{init}} = \lceil N_{\max}/4 \rceil = 25$, $K = 20$ initial clusters, and the exploration parameters $\delta_1 = \delta_2 = \delta_3 = 0.05$ when using PWAS or $\delta_1 = \delta_2 = \delta_3 = 1$ with PWASp. We run PWAS and PWASp 20 times from different random seeds on these two problems. The resulting optimal values identified by PWAS and PWASp after the maximum allowed iterations are shown in Table 5.8. We observe that both PWAS and PWASp can approach the constrained optimum within a relatively small number of iterations. Also, PWAS achieves better results than PWASp regarding the optimal values obtained after the maximum allowed iterations and consistency over multiple repetitions.

Table 5.8: Performance of PWAS/PWASp on constrained mixed-variable synthetic problems.

Algorithm	Horst6-hs044-modified		ros-cam-modified	
	mean	std	mean	std
PWAS	-62.579	3.5275e-08	-1.1151	0.3167
PWASp	-56.5539	8.3454	8.7421	16.2088
Global optimum	-62.579		-1.81	

Note: for both PWAS and PWASp, the optimum is obtained after 100 iterations.
Statistics are obtained with 20 random repetitions.

The average CPU time spent by PWAS and PWASp to fit the surrogate and solve the acquisition problem during a single iteration is reported in Table 5.9 for each tested benchmark problem. Considering that often evaluating the black-box function f or comparing samples involves expensive-to-evaluate simulations or experiments, such a CPU time can be considered negligible in real-life applications.

Table 5.9: CPU time (s) for surrogate fitting and acquisition optimization, averaged over $N_{\max} - N_{\text{init}}$ active sampling iterations.

		Func-2C	Func-3C	Ackley-5C	XG-MNIST	NAS-CIFAR10	Horst6-hs044-modified	ros-cam-modified
Surrogate fitting	PWAS	0.565	0.556	0.889	0.327	0.329	0.211	0.198
	PWASp	0.221	0.289	0.544	0.312	0.422	0.177	0.162
Acquisition optimization	PWAS	0.231	0.196	1.250	0.505	1.871	0.327	0.311
	PWASp	0.270	0.420	1.352	0.589	1.700	0.387	0.364

5.6 Conclusion

The algorithms PWAS and PWASp introduced in this chapter can handle global and preference-based optimization problems involving mixed variables subject to known linear equality and inequality constraints. Tests on different synthetic and real-world benchmark problems show that PWAS and PWASp can obtain better or comparable performance than other existing methods. Although convergence to global optimizers cannot be guaranteed, we observed that PWAS and PWASp could find satisfactory results within a limited number of iterations, despite the presence of integer and categorical variables and mixed-integer linear constraints.

Chapter 6

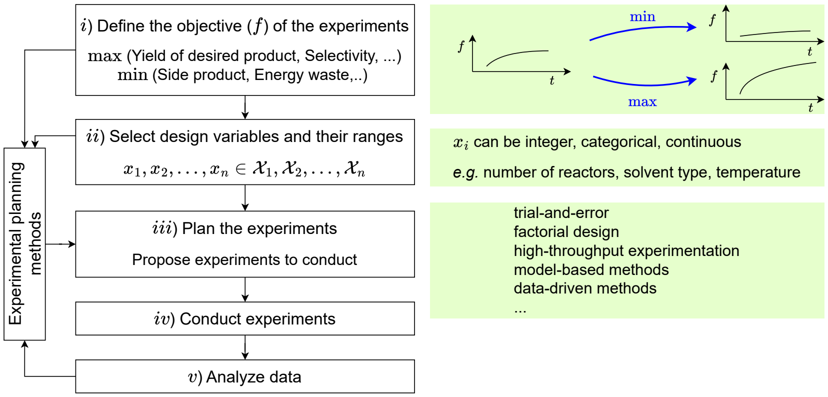
Discrete and mixed-variable experimental design with surrogate-based approach

6.1 Introduction

Experimental design includes five main steps [138] as depicted in Fig. 6.1: i) define the objective of the experiments, for instance, for a chemical reaction, the objective can be maximizing the yield of a desired product; ii) select the relevant variables and their corresponding ranges. The variables may include independent, dependent, and control variables; iii) plan the experiments, for which different strategies can be employed; iv) conduct the experiments; and v) analyze the data obtained from the experiments. Performing chemical and physical experiments is often expensive in terms of the required time, resources, and human labor. Therefore, it is important to plan experiments efficiently to gather pertinent

The content of this Chapter is from M. Zhu, A. Mroz, L. Gui, K. Jelfs, A. Bemporad, EA. del Río Chanona, and Y. Lee, "Discrete and mixed-variable experimental design with surrogate-based approach", *submitted for publication*, 2024.

data with a small number of required experiments. In this chapter, we address the challenges posed by mixed-variable spaces in experimental planning. Specifically, we propose a different framework – the use of mixed-integer surrogates and acquisition functions, where we adopt *PWAS* discussed in Chapter 5 to solve the optimization problem.



i) Define the objective (f) of the experiments
max (Yield of desired product, Selectivity, ...)
min (Side product, Energy waste,..)
ii) Select design variables and their ranges
$x_1, x_2, \ldots, x_n \in \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$
iii) Plan the experiments
Propose experiments to conduct
iv) Conduct experiments
v) Analyze data
Experimental planning methods
min
max
f
t
x_i can be integer, categorical, continuous
e.g. number of reactors, solvent type, temperature
trial-and-error
factorial design
high-throughput experimentation
model-based methods
data-driven methods
...

Figure 6.1: General steps of experimental design.

This chapter is organized as follows. In Section 6.2, we discuss the general problem formulation of the experimental planning optimization problem, focusing on problems with discrete and mixed-variable design space. We then discuss the implementation and performance of *PWAS* through three case studies in Section 6.3. Conclusion are summarized in Section 6.4.

6.2 Problem description

The general mathematical formulation of the targeted problem is:

$$X^* \in \arg\min f(X), \tag{6.1}$$

where $X = [x; y; Z]$ consists of the continuous variable $x \in \mathbb{R}^{n_c}$, integer variable $y \in \mathbb{Z}^{n_{\text{int}}}$, and categorical variable $Z = [Z^1, \ldots, Z^{n_d}]$, with n_i classes in each categorical variable Z^i, $i = 1, \ldots, n_d$. We assume both

x and y are bounded, *i.e.*, $\ell_x \leq x \leq u_x$ and $\ell_y \leq y \leq u_y$, and n_i is finite, for $i = 1, \ldots, n_d$. In (6.1), f is the objective function that maps the optimization vector X to a scalar value in $\mathbb{R}$. Here, we assume an analytic expression of f is not available, and the outcome of $f(X_1)$ can only be measured/recorded post-experimentation/simulation at $X = X_1$. And the goal is to find the X^* that minimizes f.

6.3 Case studies

We assess the applicability and effectiveness of *PWAS* for practical experimental planning problems through three case studies: i) reaction optimization of Suzuki-Miyaura cross-coupling (fully categorical), ii) crossed-barrel design (mixed-integer), and iii) optimal solvent design for Menshutkin reaction (mixed integer and categorical with linear constraints). The case studies are chosen to exhibit a range of complexities, varying in problem size, numerical difficulty, types of variables, and the presence/absence of design constraints.

All the case studies are solved on an Intel i7-8550U 1.8-GHz CPU laptop with 24GB of RAM, with all the results available in the GitHub repository at `https://github.com/MolChemML/ExpDesign`.

6.3.1 Suzuki-Miyaura cross-coupling

6.3.1.1 Problem description

The first case study focused on optimizing the reaction conditions for Suzuki-Miyaura cross-coupling [139, 140]. This reaction is pivotal in medicinal chemistry and materials chemistry, serving as a fundamental process for forming carbon-carbon bonds in the synthesis of various pharmaceuticals and polymers [139–142]. A reaction scheme for the investigated Suzuki-Miyaura coupling is shown in Fig. 6.2, illustrating the coupling of a boronic acid derivative and an aryl halide facilitated by a palladium complex catalyst, a ligand, a base, and a solvent [143–145]. Here, all optimization variables are categorical and the number of possible options for each optimization variable is summarized in Table 6.1.

The full Cartesian product space consists of 3,696 unique reactions. The study looks into the relationship among these categorical variables, and aims to identify optimal combinatorial sets of precursors that can maximize the yield of the desired product within a small number of experiments, and therefore reduce the resources and time required.

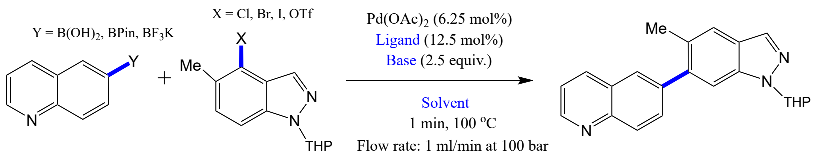


Figure 6.2: Suzuki-Miyaura cross-coupling reaction. The variables - boronic acid derivative (Y), aryl halide (X), ligand, base, and solvent - highlighted in blue represent the experimental design space. All other reaction conditions are fixed and noted in black.

Table 6.1: Reaction design space (fully categorical) for the Suzuki-Miyaura cross-coupling reaction [143–145].

Optimization variables	**# options**
Aryl halide (X)	4
Boronic acid derivative (Y)	3
Base	7
Ligand	11
Solvent	4
Total # of possible combinations	3,696

We employ *PWAS* to solve the optimization problem and benchmark its performance against established optimization libraries: *Genetic* [146, 147] (evolutionary algorithm), *Hyperopt* [148] (BO with Tree-Structured Parzen Estimator (TPE)), *BoTorch* [149] (BO with Gaussian Process (GP)), and *EDBO* [143] (BO with GP specialized for reaction). Additionally, we consider *Random Search* as a baseline. The characteristics of the approaches used in these libraries have been discussed in Appendix B. We note that *Random Search*, *Genetic*, *Hyperopt*, and *BoTorch* have been interfaced in the *Olympus* [145] package; therefore, we use the algorithmic structure implemented in the package for benchmark tests with their default parameter values. A customized forked version tailored for our testing is also available on GitHub at `https://github.com/mjzhu-p/olympus` (Branch "pwas_comp"). Regarding *EDBO*, categorical variables, namely, distinct chemical entities, are one-hot-encoded and the default setting is used with minor changes to the original package to allow customized input for the number of initial samples (see the changes in the forked version at `https://github.com/mjzhu-p/edbo/tree/pwas_comp`). As for *PWAS*, the default setting [3] is used, *i.e.*, the number of initial partitions (K_{init}) is set to 10, with the trade-off parameter between exploitation and exploration (θ_E) set to be 0.5. And the categorical variables are one-hot encoded.

For each optimization method, we conduct 30 repetitions for statistical analysis. Given that the goal of our study is to assess the performance of the algorithms on case studies where only a small number of experiments/simulations can be done due to time and resource constraints, we cap the maximum number of experiments at 50 within each repetition.

6.3.1.2 Results and discussion

The performance comparisons of different methods on Suzuki-Miyaura cross-coupling reaction optimization are shown in Fig. 6.3. Fig. 6.3a illustrates the highest yield achieved (%) so far at different iterations. Since the numerical values of the yields are very close, especially as the number of iterations increases, a zoomed-in panel of the last five iterations is shown for better visualization. In Fig. 6.3b, the corresponding ranks of the yields at different iterations are presented. These ranks are derived from the known yields of all possible combinations (3,696 in total) [144].

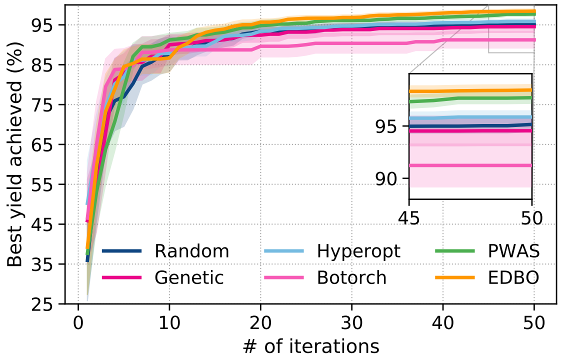
While *EDBO* achieves the highest yield after 50 iterations, *PWAS* demonstrates competitive performance, surpassing all other tested methods in its ability to identify optimal reaction conditions that maximize the reaction yield. It is important to note that *EDBO* is expected to outperform other methods in this case study, given that the GP model used in *EDBO* was pre-trained using the entire Suzuki-Miyaura reaction dataset (3,696 reactions) [143] (see also in Appendix B). To provide a clear demonstration of the efficiency of each method, we present a boxplot in Fig. 6.4. This visualization represents the number of iterations required by each method to achieve a top-20 ranked yield. Each data point on the plot represents the outcome of one specific run, and the statistics presented are derived from 30 repetitions to ensure robustness. On average, both *PWAS* and *EDBO* require significantly fewer iterations to attain a top-20 ranked yield when compared to other methods. This suggests their superior efficiency and potential time and resource savings in practical applications. Overall, the comparable performance of *PWAS* with *EDBO* demonstrated in the case study shows the ability of mixed-integer surrogates to more efficiently optimize the parameter space with no prior knowledge of the system, which has major implications for situations where prior data (literature or otherwise) is not available or difficult/expensive to obtain - a very common scenario in the chemical sciences.

6.3.2 Crossed barrel

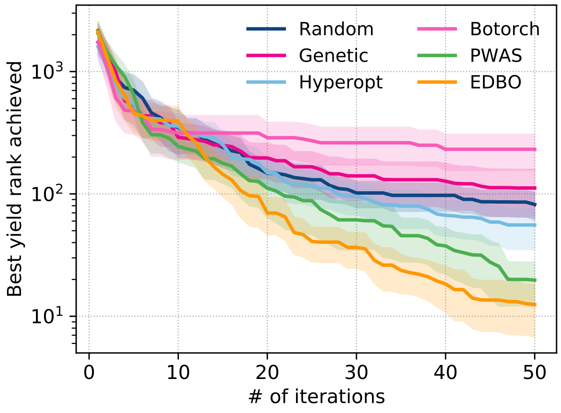
6.3.2.1 Problem description

The second case study explores the optimization of the design of a crossed barrel (see Fig. 6.5) for improved mechanical properties [145, 150]. Specifically, we aim to maximize its toughness while not exceeding a specified force threshold. Here, toughness corresponds to the amount of energy a component can withstand before experiencing failure [150]. Components with a crossed-barrel structure are used to protect more fragile parts within a design while not passing on harmful reactionary forces [150]. For instance, these structures can shield sensitive instrumentation or electronics from mechanical vibrations or impacts.

As depicted in Fig. 6.5, a crossed barrel has n hollow columns with outer radius r and thickness t, twisted at an angle θ. Here, n, r, t and θ are the design variables we want to optimize, whose data types (discrete/continuous) and domains are outlined in Table 6.2. Due to the involvement of continuous variables (n, r, t), exhaustively enumerating all possible combinations is impractical. Hence, an emulator, as recommended by Hickman *et al.* [145], is utilized to simulate the process and therefore make it possible to sample over the whole feasible domain. The



(a) Best yield achieved (%) so far at different iterations.



(b) Best yield rank achieved so far at different iterations.

Figure 6.3: A comparison of the performance of *PWAS* and the benchmark methods on Suzuki-Miyaura cross-coupling reaction optimization. For each method, the solid line represents the mean value, and the filled area comprises the 95% confidence interval, *i.e.*, mean $\pm$ 1.96 std.

emulator was modeled as Bayesian neural nets (BNN) [145] and trained on over 2,500 HTE data points collected by Gongora *et al.* [150]. We note that the trained emulator serves the purpose of method comparisons in

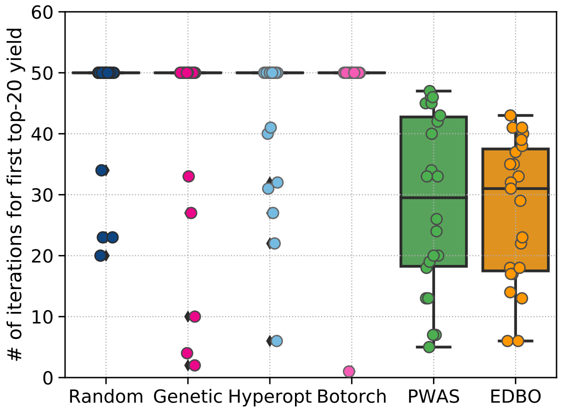


Figure 6.4: Number of iterations each method takes in each run to obtain the first top-20 ranked yield. The results for 30 repetitions are summarized in the boxplot. Each dot represents one run of the repetitions. The diamond-shaped points are the ones classified as outliers by the boxplot.

this case study. Nevertheless, the accuracy of the trained model can be improved if more data could be provided.

The challenges of this case study involve balancing trade-offs between conflicting mechanical properties, such as strength (the ability to resist an applied force without being damaged) and ductility (the ability to stretch without breaking), and incorporating mixed-integer design choices. This case study is selected due to the mixed-integer nature of the problem and the availability of an adequate number of HTE experimental data to train an emulator [145, 150]. Similar procedures can be followed to design chemical-related units, *e.g.*, chemical reactors, if data acquisition is possible via experiments or high-fidelity simulations.

We solve this optimization problem with the same set of methods employed in the Suzuki-Miyaura cross-coupling case study, using the packages interfaced in the *Olymnpus* package for *Random Search*, *Genetic*, *Hyperopt*, and *BoTorch*. As discussed in Appendix B, the method implemented in *EDBO* package requires a pre-defined discrete search space, meaning that the continuous variables, θ, r, and t, need to be discretized.

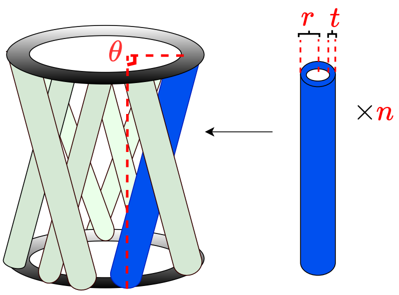


Figure 6.5: Schematic representation of a crossed-barrel design [150], illustrating the optimization variables, where θ is twist angle of the columns [degree], r is outer radius of the columns [mm], n is the number of hollow columns, and t is thickness of the hollow columns [mm].

Table 6.2: Optimization variables for the crossed-barrel design [145, 150].

Optimization variables	**Type**	**Domain**
Number of hollow columns (n)	integer	[6, 12]
Twist angle of the columns (θ) [degree]	continuous	[0.0, 200.0]
Outer radius of the columns (r) [mm]	continuous	[1.5, 2.5]
Thickness of the hollow columns (t) [mm]	continuous	[0.7, 1.4]

Here, we consider three discretization schemes for the search domain, evenly divided and spaced by: 100, 10, and 10 points (*EDBO_1*); 10, 10, and 10 points (*EDBO_2*); and 10, 5, and 5 points (*EDBO_3*), respectively. These configurations yield 70,000, 7,000, and 1,750 possible combinations to form the search domain. As for *PWAS*, two strategies are used to handle integer variables as detailed in the pre-processing step in Section 5.3.1, and the same values introduced in Section 6.3.1 are used for K_{init} and θ_E. Similarly to the Suzuki-Miyaura cross-coupling case study, we run 30 repetitions for statistical analysis. Within each run, we include 10 initial experiments and then allow a maximum of 50 iterations.

6.3.2.2 Results and discussion

The optimization outcomes are summarized in Figs. 6.6, 6.7, and Table 6.3. In achieving the best objective function values within a specified budget, *EDBO_1* outperforms all other methods, while *PWAS* is comparable with that of *Hyperopt* (see Fig. 6.6) and *EDBO_2* (see Fig. 6.7). However, the effectiveness of *EDBO* varies depending on the number of discretization steps considered. As evidenced in Fig. 6.7, the performance of *EDBO* improves with an increase in discretization steps, showing that only *EDBO_1* outperforms *PWAS*. Although the differences in objective function values obtained by *EDBO*s and *PWAS* are marginal, the number of discretization steps required to achieve a particular quality of solution is unknown and there is no systematic method of determining appropriate value. Furthermore, the increase in the number of discretization steps can result in higher computational cost, particularly with a higher number of continuous variables. This is reflected in Table 6.3, where the CPU time for *EDBO* and *BoTorch* are significantly higher than that of other methods. Despite that, we acknowledge that our target problems often involve costly experiments or simulations. The CPU time needed by the methods examined in this study is negligible compared to the time required for conducting many types of experiments or simulations. Nevertheless, we note the necessity for some chemistry applications that may benefit from fast feedback, including flow chemistry [151, 152]. Moreover, when the number of continuous variables and their ranges grow, selecting an appropriate discretization scheme that guarantees improved performance a prior may not be straightforward.

Table 6.3: CPU time (seconds) required by different methods for one run of the optimization for the crossed barrel design. Statistics were obtained from 30 random runs.

	Random	**Genetic**	**Hyperopt**	**BoTorch**	**PWAS**	**EDBO_1**	**EDBO_2**	**EDBO_3**
Average	1.85	1.77	2.80	398.68	35.36	272.54	227.54	212.92
std	0.44	0.35	0.71	260.71	2.00	67.61	2.52	20.38

6.3.3 Solvent design

6.3.3.1 Problem description

In the third case study, we consider the design of solvents for enhanced kinetics of a Menschutkin reaction (see Fig. 6.8), following the computer-

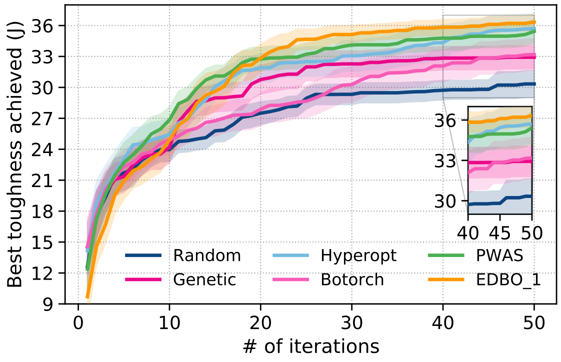


Figure 6.6: Best toughness achieved so far at different iterations for the designed structure at different iterations for crossed barrel design. Results are summarized over 30 repetitions. For each method, the solid line represents the mean value, and the filled area comprises the 95% confidence interval, *i.e.*, mean ± 1.96 std.

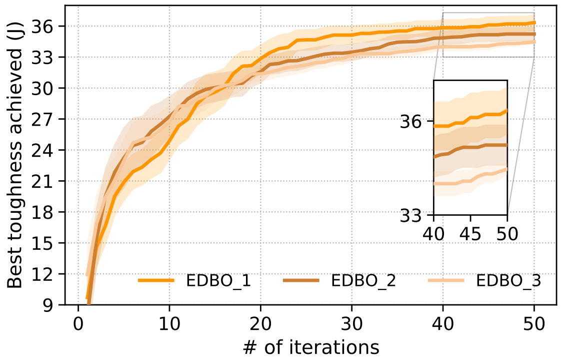


Figure 6.7: Best toughness achieved so far at different iterations for the designed structure at different iterations with *EDBO* method with different discretization steps for crossed barrel design. The trajectory of *PWAS* is also shown for comparison.

aided molecular design (CAMD) formulation of Gui *et al.* [153]. Choosing an appropriate solvent is crucial for liquid-phase reactions, as it can reduce the Gibbs free energy barrier (see Fig. 6.8) and therefore promote fast reaction kinetics. The aim of optimization is to determine the optimal molecular structure of the solvent that maximizes the reaction rate constant k [L mol^{-1} s^{-1}], for which we define the objective function as $f(X) = -\ln k$. We note that (6.1) is formulated as a minimization problem, thereby maximizing $\ln k$ is equivalent to minimizing $-\ln k$). Here, $\ln k$ is used, which is a common practice when developing data-driven models and comparing with experimental data [153], because k can significantly differ across different solvents, sometimes by orders of magnitude.

A set of 46 functional groups were selected as molecular building blocks. The solvent was represented by integer variables to indicate the number of each functional group present in the solvent molecule. To ensure that only chemically feasible combinations of functional groups are generated during the optimization process and to limit the size of the solvent, a set of chemical feasibility and complexity constraints was imposed. For instance, constraints were used to ensure the octet rule [154]. Since the solvent designed must be in the liquid phase at reaction conditions, the normal melting point (T_m) and the boiling point (T_b) of the solvent were added as design constraints. Two physical properties, namely, flash point (T_{fp}) and octanol/water partition coefficient (K_{ow}), as well as the oral rat median lethal dose (LD_{50}) of the solvent were constrained to reduce health, safety, and environmental impact. In total, the problem consists of 115 linear inequality and 5 linear equality constraints, where one auxiliary categorical and 7 binary variables were introduced to formulate the constraints. The types of design variables and the property prediction model used are summarized in Tables 6.4 and 6.5. For a comprehensive description of the mathematical formulation used, the reader is referred to Gui *et al.* [153] and section 2.3 therein.

6.3.3.2 Surrogate model for the rate constant

As discussed in Appendix B, the methods examined in the previous two case studies cannot explicitly handle mixed integer/categorical constraints. While post-hoc screening of infeasible solutions obtained from

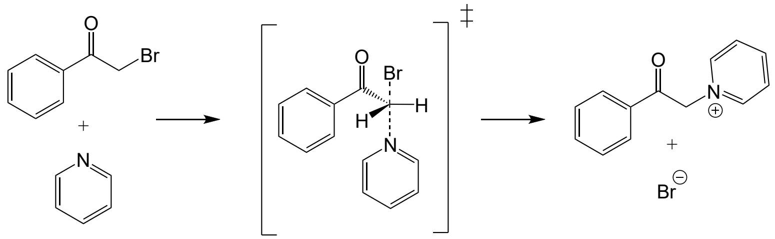


Figure 6.8: The Menschutkin reaction of phenacyl bromide and pyridine. In the illustration, Solvent 2 is preferred which lowers the free energy compared to Solvent 1 [153, 155].

Table 6.4: Optimization variables and problem size for the solvent design [153].

Description	Notes
Number of functional group types	46 (integer)
Number of auxiliary variables introduced for chemical feasibility	1 (categorical) and 7 (binary)
Number of inequality/equality design constraints	115 (linear) / 5 (linear)

Table 6.5: Property constraints for the solvent design. The property prediction method of Hukkerikar *et al.* [156] is used for T_b, T_m, T_{fp}, and K_{ow}, and Hukkerikar *et al.* [157] is used to predict LD_{50}

Physical property	Bounds
T_m (K)	$[10^{-5}, 298.15]$
T_b (K)	$[323.15, 10^5]$
T_{fp} (K)	$[252, 10^5]$
log K_{ow}	$[10^{-5}, 3]$
-log LD_{50} (mol/kg)	$[10^{-5}, 3]$

an unconstrained optimization may appear as a potential approach, the large number of constraints often renders such post-optimization exclusion computationally expensive and potentially inefficient in achieving convergence. Thus, direct comparisons with such methods are not practical. Instead, we benchmark our optimization results against those obtained using *DoE-QM-CAMD* [153] – a CAMD framework tailored to incorporate quantum-mechanical (QM) calculations of rate constant and computational experimental design into the molecular design process.

In the following, we provide an overview of the *DoE-QM-*

CAMD [153] method. *DoE-QM-CAMD* employed a multiparameter solvatochromic equation [158, 159] to correlate solvent properties and the logarithm of the rate constant:

$$\ln k = c_0 + c_A A + c_B B + c_S S + c_\delta \delta + c_H \delta_H^2, \tag{6.2}$$

where A, B, S, δ, and δ_H^2 are the Abraham's overall hydrogen-bond acidity, Abraham's overall hydrogen-bond basicity, dipolarity/polarisability, and Hildebrand solubility parameter, respectively, of the solvent, and $c_0, c_A, c_B, c_S, c_\delta$, and c_H are the coefficients that need to be estimated via multiple linear regression (MLR). Estimating the parameters in the MLR model with high accuracy can be challenging because only a small number of experiments can often be conducted, restricting its predictive capacity. To address this challenge, *DoE-QM-CAMD* first selects an information-rich set of (computer) experiments using the D-optimality criterion. These initial experiments, which are observed to cover a wider range of solvent properties, are then used to train an initial MLR model. Subsequently, to refine the MLR model and enhance its predictability around the optimal solvent region, iterative optimization is undertaken to identify the best solvent (*i.e.*, the one that gives the highest reaction rate) based on the current MLR model. Should a new solvent be identified, the MLR model undergoes re-fitting with the updated experimental set that consists of the newly identified optimal solvent and the solvents in the original set. The iterative process terminates when the best solvent, determined by optimizing the MLR model, has been sampled previously. Upon convergence of the MLR model, the top 10 solvents are determined by re-initializing and optimizing the problem, wherein integer cuts are added to exclude previously identified solutions. We note that the active-learning-like iterative process of *DoE-QM-CAMD* after the initial experiments relies solely on the newly fitted MLR model with limited capabilities for exploration.

In contrast, *PWAS* solves the problem by employing an active-learning technique with exploration capability. It systematically identifies optimal solvents for examination, effectively balancing the trade-off between exploring new possibilities for model improvement and exploiting known knowledge of reaction kinetics. As opposed to *DoE-QM-CAMD*, which assumes linear relationship between the expert-derived solvent properties and $\ln k$, *PWAS* adopts PWA surrogates to represent the correlations between the functional groups within the designed solvent and $\ln k$, where the relationship between solvent properties and $\ln k$ are learned implicitly. As discussed in Chapter 5, *PWAS* leverages *PARC* [160] to fit the surrogates, where *PARC* first clusters samples in K_{init} initial partitions. The initial partitions are then optimized by balancing between enhancing separability, which relies on similarities among different solvents (in this context, functional groups), and improving the predictability of the surrogate function within each partition, in this case, the input-output correlations. Here, the output ($\ln k$) correlates with the properties of the designed solvent and therefore can be implicitly learned during surrogate fitting, offering insights not available when solely considering individual functional groups.

Consistent with previous case studies, default parameters are utilized when solving the problem with *PWAS*, including a maximum of 50 experiments, with an initial set of 10 samples. It is important to highlight that the complete QM reaction constant data were generated by exhaustively enumerating all 326 feasible solvents within the defined design space [161], enabling the sampling of new solvents without the need for additional calculations and providing the true rank of the solvents.

6.3.3.3 Results and discussion

Comparison between PWAS and DoE-QM-CAMD

In the following, we compare the performance of *PWAS* and *DoE-QM-CAMD* based on the top 10 solvents identified by *PWAS* and *DoE-QM-CAMD*, for which we compare their rank alignment with *QM* calculated values. As shown in Tables 6.6 and 6.7, respectively, the top 10 solvents identified by *PWAS* are consistent with the true rank, in contrast to ranks of the optimal solvents obtained from *DoE-QM-CAMD*, which show a large deviation. Also, we observe that the predicted values based on *PWAS* are more accurate to the QM calculated values compared to those of *DoE-QM-CAMD* (The mean squared errors for the top-10 ranked solvents with *QM* are 2.4×10^{-4} log units for *PWAS* and 0.46 log units for *DoE-QM-CAMD*). These observations can be attributed to the inherent nature of *PWAS* and *DoE-QM-CAMD*. The MLR model utilized within the *DoE-QM-CAMD* method primarily serves as a predictive tool across the design space. As previously noted, it generates predictions for the top-ranked samples post-generation of the MLR model, which may not align well with the experimental results. In contrast, *PWAS* operates as an optimization mechanism, focusing on refining the predictive model within the design space where promising solvent candidates exist—those that yield high reaction rates, while also exploring uncovered design space to prevent from being stuck in the local optimum. It achieves this by iteratively proposing new samples for evaluation through active learning, where an acquisition function that trades off between exploitation (finding the solvents with a high reaction rate) and exploration (covering unexplored design space) is minimized. As a result, it is not surprising that *PWAS* performs better in terms of the rank alignment and predictability around the optimal region.

Table 6.6: The top 10 ranked solvents identified by *PWAS* for the solvent design case study. k [L mol^{-1} s^{-1}]: rate constant for the Menschutkin reaction, QM: ln k obtained from quantum-mechanical calculation, pred: ln k predicted by the *PWAS* surrogate.

Rank	**Chemical formula**	**ln k**	
		QM	**pred**
1	CH_3NHCHO	-5.92	-5.92
2	$OHCH_2NO_2$	-6.46	-6.49
3	$CH_2OHCH_2NO_2$	-6.72	-6.69
4	$(CH_3)_2SO$	-6.82	-6.82
5	$(CH_2)_2OHCH_2NO_2$	-6.93	-6.93
6	$CH_3CHOHCH_2NO_2$	-6.96	-6.97
7	$CH_2{=}COHCH_2NO_2$	-6.98	-6.97
8	$CH{=}CHOHCH_2NO_2$	-7.00	-6.98
9	$(CH_2)_3OHCH_2NO_2$	-7.10	-7.10
10	$CHCH_2{=}CHOHCH_2NO_2$	-7.11	-7.11

Analysis of algorithmic exploitation and exploration capabilities

To further analyze the exploitation and exploration capabilities of *PWAS*,

Table 6.7: The top 10 ranked solvents identified by *DoE-QM-CAMD* for the solvent design case study. k [L mol^{-1} s^{-1}]: rate constant for the Menschutkin reaction, QM: ln k obtained from quantum-mechanical calculation, pred: ln k predicted by the *DoE-QM-CAMD* surrogate, *i.e.*, the multiparameter solvatochromic equation (6.2).

Rank	**Chemical formula**	**ln k**	
		QM	**pred**
1	$CH_2OHCH_2NO_2$	-6.72	-5.50
2	$(CH_3)_2SO$	-6.82	-5.59
3	$CH_2OHCH_2NO_2$	-6.72	-6.28
4	$CH_2{=}COHCH_2NO_2$	-6.98	-6.66
5	$(CH_2)_2OHCH_2NO_2$	-6.93	-6.74
6	$CH_3CHOHCH_2NO_2$	-6.96	-6.87
7	$(CH_3)_2COHCH_2NO_2$	-7.23	-6.91
8	$CH{=}CHOHCH_2NO_2$	-7.00	-6.92
9	$CH_2CH_2{=}COHCH_2NO_2$	-7.15	-6.97
10	CH_3NHCHO	-5.92	-7.00

we examine the solvents determined by *PWAS*. Specifically, we aim to show that *PWAS* can exploit the surrogate to implicitly learn the preferred solvent properties that lead to a high reaction rate, and can explore the design space to obtain a set of solvents with diverse structures and chemical properties.

Before examining specific solvents, we first perform a sensitivity analysis using Partial Dependence Plot (PDP) and Individual Conditional Expectation (ICE) plots to investigate the relative influence of solvent properties on the reaction rate constant, which we will then use to assess *PWAS*'s exploitation and exploration capability. Seven representative descriptors considered are refractive index at 298K (n^2), Abraham's overall hydrogen-bond acidity (A), Abraham's overall hydrogen-bond basicity (B), dielectric constant at 298K (ϵ), microscopic surface tension at 298 K (γ), aromaticity, and halogenticity, which are used in a successful quantum mechanical continuum solvation model [162]. The descriptors of all feasible solvents are calculated using the group contribution method of Sheldon *et al.* [163]. As can be seen in Fig. 6.9, the fluctuation in PDPs is most pronounced for ϵ, while the fluctuation in ICEs is most significant for three properties, namely, n^2, B, and ϵ, indicating their strong marginal effect on predicting $\ln k$. This finding aligns with the established results for S_N2 reactions. As reported in the literature [164–166], solvent effects on reaction kinetics stem from the fact that the solvent-solute interactions stabilize the reactant(s) and the transition state to different extents. In general, when the transition state is (partially) ionic by nature and the reactants are neutral, a solvent with larger dielectric constant, indicating greater polarity, or those with stronger hydrogen bond basicity, meaning they are more potent hydrogen bond acceptors, can lower the free energy of the transition state more than that lowered for the reactants, thereby reducing the overall free energy barrier. It is also known that non-basic, polar aprotic solvents are preferred as they do not solvate the nucleophile strongly, making it more reactive and available for the reaction. Regarding the refractive index, although it does not directly reflect the polarity of solvents, it can greatly affect the solvation of reactants and the overall environment. Besides the dominant solvent descriptors, from Fig. 6.9, we can see that the relationship between the reaction rate and the solvent properties is not strictly linear. Additionally, there are dependencies among different properties, explaining why the MLR model (6.2) integrated into the *DoE-QM-CAMD* approach demonstrates inconsistent performance across the entire design space. While the MLR model could be improved by incorporating second-order terms and interaction terms, as discussed by Gui [161], deciding which terms to include often resorts to a trial-and-error approach, which can be time-consuming and potentially hard to justify. In contrast, *PWAS* provides a systematic approach to decompose the solvent design space based on their similarity related to rate constants, making it possible to capture the nonlinear relationship between the solvent properties and the reaction rate without making prior assumptions on the functional form, which we demonstrate in the following.

Focusing on the most significant solvent descriptors, n^2, B, and ϵ, three radar charts are plotted in Figs. 6.10 and 6.11, showing the relevant properties of the initial, first-10 active-learning, and last-10 active learning samples. To facilitate comparison, all features are normalized to a range between 0 and 1 using min-max normalization, except for the

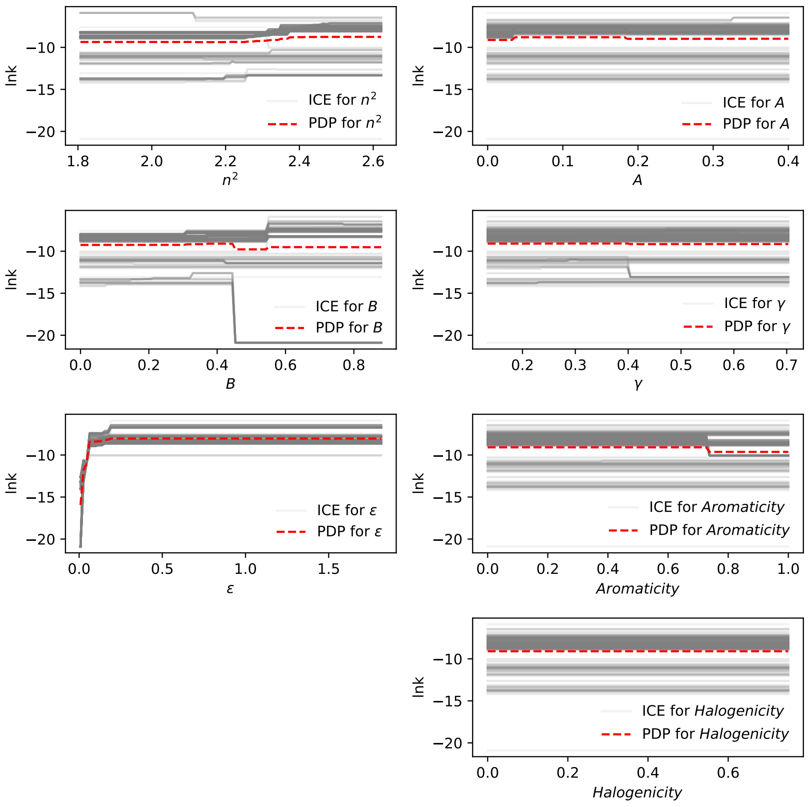


Figure 6.9: Partial Dependence Plots (PDP) and Individual Conditional Expectation (ICE) Plots utilized to assess the influence of diverse solvent properties on the reaction rate across all feasible solvents. n^2: refractive index at 298K, B: Abraham's overall hydrogen-bond basicity, ϵ: dielectric constant at 298K, A: Abraham's overall hydrogen-bond acidity, γ: the macroscopic surface tension at 298K. Solvent properties are calculated based on the property prediction method of Sheldon *et al.* [161, 163].

dielectric constant. Since the dielectric constants of two solvents are significantly higher than the others, the dielectric constant is normalized relative to the remaining solvents, resulting in values of these two solvents exceeding 1. Besides the radar charts, we also depict the structures of the solvents and their categorization based on the constituent functional groups in Fig. 6.12. Fig. 6.12a arranges the solvents in the sequence of optimization steps, distinguishing the initial and subsequent active-learning samples with a black line in, while Fig. 6.12b arranges the solvents into partitions, with orange lines denoting partition boundaries.

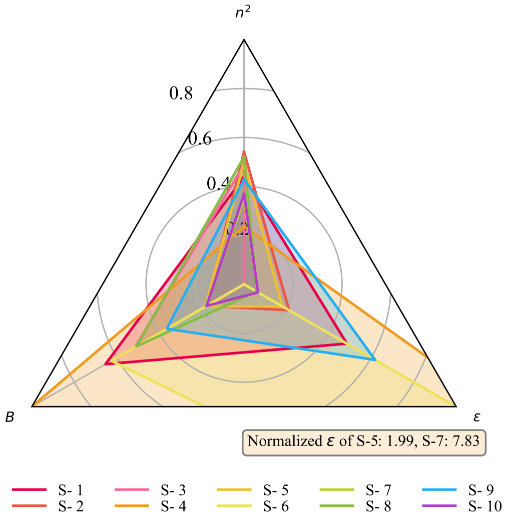
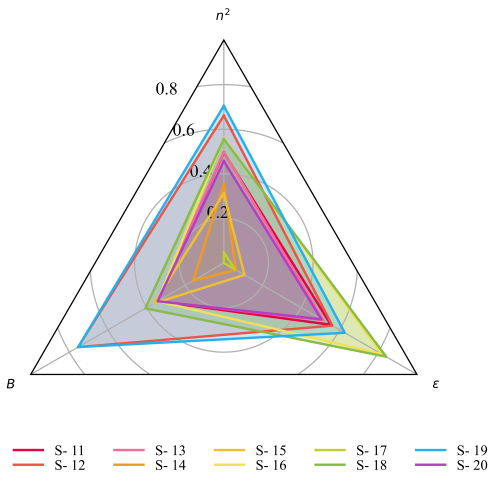
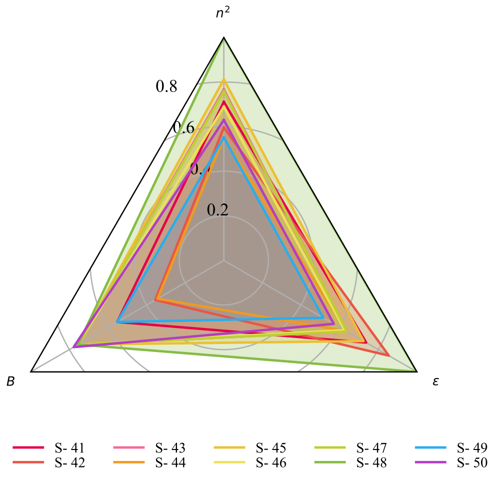


Figure 6.10: Radar chart of the three selected features of the first 10 initial samples. n^2: refractive index at 298K, B: Abraham's overall hydrogen-bond basicity, ϵ: dielectric constant at 298K. All features were normalized to a range between 0 and 1 using min-max normalization, except for the dielectric constant. Since the dielectric constants of S-5 and S-7 were significantly higher than those of the others, the dielectric constant was normalized relative to the remaining solvents. With the relative normalized dielectric constants of S-5 and S-7 denoted in the figure.

By examining Figs. 6.10, 6.11 and 6.12, it can be observed that *PWAS* explores diverse solvent structures, covering large ranges of solvent properties, during the initial sample step, and then gradually converges toward clear patterns while maintaining an exploratory nature. These results demonstrate the effectiveness of *PWAS* at finding a diverse and promising set of solvents over the constrained mixed-integer and cate-

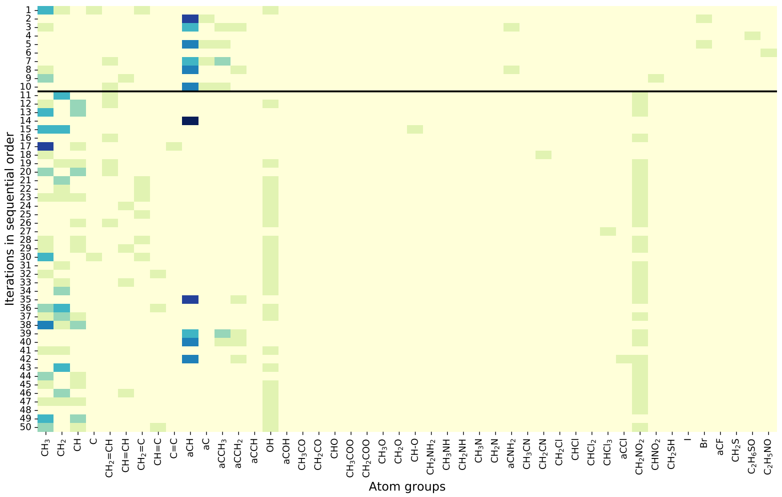


(a) The first-10 active-learning samples.



(b) The last-10 active-learning samples.

Figure 6.11: Radar chart of the three selected features for the first-10 (a) and last-10 (b) active learning samples. n^2: refractive index at 298K, B: Abraham's overall hydrogen-bond basicity, ϵ: dielectric constant at 298K. All features are normalized to a range between 0 and 1.



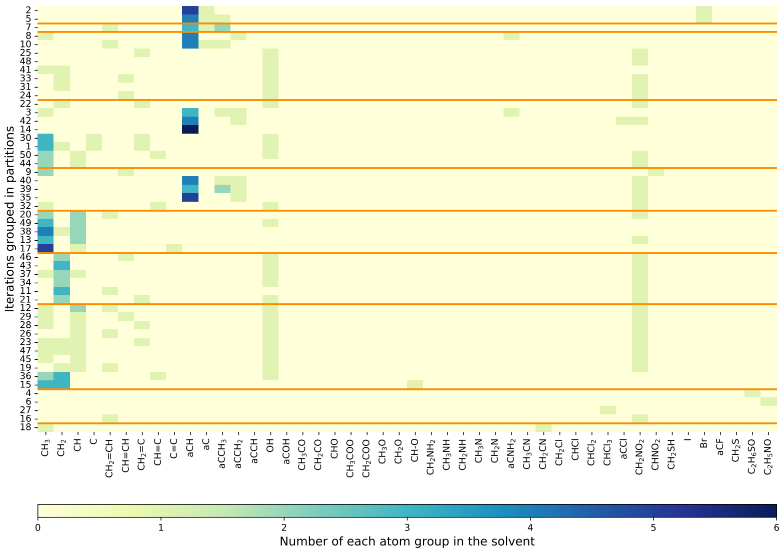


Figure 6.12: Solvents identified by *PWAS* in 50 iterations, whose structures are depicted in functional group representations. (a): in sequential iteration order, with a black line separating the initial and active-learning samples; (b): grouped in partitions, with orange lines representing the boundary of the partitions (in total 10 partitions).

gorical domain. One major advantage of *PWAS* is that it allows one not only to group solvents with similar functional groups into the same partition, but also to place solvents with similar chemical properties into the same partition through the *PARC* mechanism. These similar findings across the functional group- and property-based design spaces highlight the adeptness of *PWAS* in identifying key implicit relationships, demonstrating its capability to effectively discern and utilize the links between functional groups and the ensuing solvent properties.

Preferred solvent properties

We further investigate the implicitly learned solvent properties to gain some chemical insights to derive general conclusions on the preferred solvent properties that can result in a high reaction rate for the Menschutkin reaction. In Fig. 6.13, we plot each solvent in relative-rank order, with x and y axis indicating n^2 and $\log \epsilon$, respectively. B is represented by the size of each bubble whose scales are shown in the legend. The relative ranks are indicated using a colorbar, with the top-10 and last-10 ranked solvents also denoted with texts for clarity. Upon examination of Fig. 6.13, it seems that the dielectric constant emerges as the predominant factor influencing reaction kinetics. This finding aligns with the established results for the Menschutkin reaction, where polar aprotic solvents are typically favored [164, 165]. Also, in scenarios where differences in dielectric constants are small, a higher refractive index tends to correlate with higher reaction rates. Among the top-ranked solvents, there exists a notable uniformity in basicity levels, while no clear trend can be observed for basicity across all identified solvents, which is also consistent with the PDP plot for B (see Fig. 6.9).

In summary, we compared the effectiveness of two approaches for the solvent design case: *PWAS* and *DoE-QM-CAMD*. Our findings reveal the strengths in each method: the *DoE-QM-CAMD* approach, utilizing a MLR model, demonstrates more robust predictive capabilities across the entire design space; while, *PWAS*, employing PWA surrogates, can better predict reaction rates in proximity to optimal regions, which is important for our optimization objective. Furthermore, *PWAS* can learn correla-

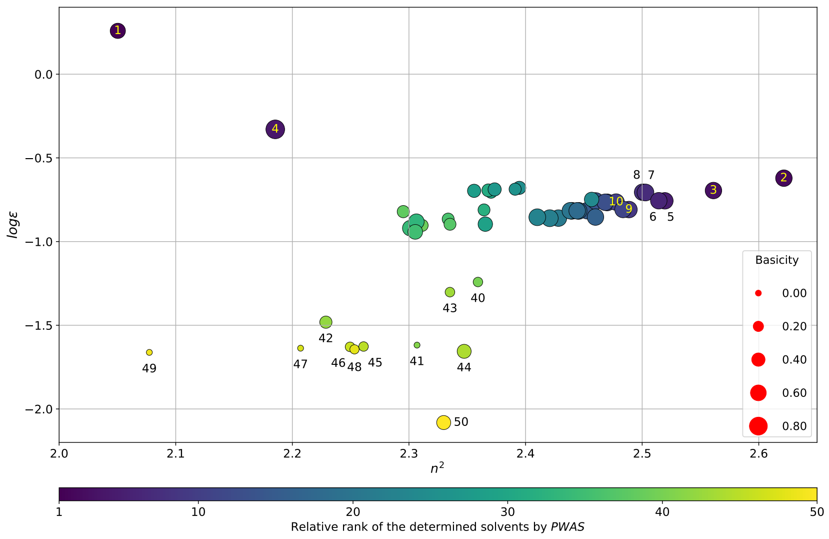


Figure 6.13: Bubble chart of solvent properties of the solvents identified by *PWAS*. n^2: refractive index at 298K, ϵ: dielectric constant at 298K. Abraham's overall hydrogen-bond basicity is represented by the size of each bubble, with the relevant bubble size scale shown in the legend. The relative ranks of each solvent are indicated using a color bar, with the top-10 and last-10 ranked solvents also denoted with texts for clarity.

tions between solvent properties and reaction rates and offer valuable insights.

6.4 Conclusion

In this Chapter, we have shown the effectiveness of mixed-integer surrogates, specifically *PWAS*, in addressing discrete and mixed-variable experimental planning problems. We illustrated the efficiency of *PWAS* through analysis and experimentation across three benchmark case studies, covering problems from different domains. While BO has undeniably revolutionized the landscape of optimization in experimental planning, especially in the chemical domain, it is important to recognize the potential of other surrogate-based approaches within conventional chemistry optimization problems. This is particularly relevant due to the inherent complexity of many chemical problems, that are often characterized by mixed variables and a relatively large number of constraints, making it challenging for conventional BO approaches to obtain feasible samples during the acquisition step while still maintaining exploration capability. In this chapter, we demonstrated that integrating mixed-integer optimization strategies is an effective way to address these challenges.

Chapter 7

Conclusion

In this thesis, we have addressed various challenges in the field of black-box and preference-based optimizations as well as their subset optimization problems involving mixed variables. Our research has led to the development of novel algorithms and methodologies that offer efficient and effective solutions to these complex problems.

In Chapter 3, we began by tackling the calibration of control policy parameters through global optimization. Traditional approaches often require explicit knowledge of the objective function, which can be impractical or difficult to obtain in many real-world scenarios. To overcome this limitation, we introduced a semi-automated calibration approach that relies on pairwise preferences between control policies rather than a quantifiable performance index. This preference-based approach not only simplifies the calibration process but also makes it suitable for tasks with qualitative or subjective performance evaluations. We applied the approach to calibrate MPC for a steady-state switching task in a CSTR and an obstacle avoidance task in automated driving. Our findings revealed that achieving satisfactory results only required a small number of experiments.

Building upon this, in Chapter 4, we extended our research to preference-based global optimization problems with unknown constraints. We proposed an algorithm called C-GLISp, which effectively handles such optimization tasks by leveraging the preferences expressed by the decision-maker as well as the satisfactory and feasibility assessment by the decision-maker. The algorithm's ability to incorporate additional information from the decision-maker improves the probability of proposing feasible samples during the exploration process. The effectiveness of C-GLISp was demonstrated through numerical benchmark problems and a semi-automated model predictive control (MPC) calibration case study, where we find that C-GLISp can find satisfactory solutions within a small number of iterations.

Furthermore, in Chapter 5, we addressed the challenges of optimization problems involving mixed variables subject to linear equality and inequality constraints. We introduced two algorithms, PWAS and PWASp, that utilize piecewise affine surrogates to approximate the objective function in these problems. PWAS efficiently solves global optimization problems with mixed variables, while PWASp extends the approach to handle preference-based optimization tasks. Both algorithms show comparable performance with existing methods in various synthetic and real-world benchmark problems. Despite the presence of integer and categorical variables and mixed-integer linear constraints, PWAS and PWASp consistently found satisfactory solutions within a limited number of iterations.

Next, in Chapter 6, we put into practice PWAS developed in Chapter 5 to tackle experimental design challenges, where we demonstrate the efficacy of PWAS in optimizing experimental design problems through three case studies of varying design space sizes and numerical complexities. These include: i) optimizing reaction conditions for Suzuki–Miyaura cross-coupling (fully categorical), ii) optimizing crossed-barrel design to enhance mechanical toughness (mixed-integer), and iii) designing solvents for improved Menschutkin reaction kinetics (mixed-integer and categorical with linear constraints). Through comparison with conventional optimization methods implemented in established libraries (Genetic, Hyperopt, BoTorch, EDBO), we show that PWAS often performs comparably or better. Additionally, it can address problems with a relatively large number of linear constraints.

In conclusion, this thesis has contributed some insights and methodologies to the field of black-box and preference-based optimization. The developed algorithms and approaches address the challenges posed by black-box functions, unknown constraints, and mixed variables, providing efficient and effective solutions. The semi-automated calibration approach, along with the C-GLISp, PWAS, and PWASp algorithms, have demonstrated their practical applicability and performance through numerical benchmarks and simulation case studies on real-world problems.

7.1 Open questions and future work

Future research can be conducted to further enhance the algorithms' performance and scalability (high-dimensional problems), explore alternative surrogate modeling techniques, and extend the methodologies to handle more complex constraints and problem structures. Specifically, regarding surrogate modeling, since initial samples play an important role, future research can be devoted to finding efficient initial sampling strategies, especially when complex (mixed-variable) constraints are present. Additionally, there are challenges involved in designing the acquisition function. In our thesis, we focus on exploration methods that employ space-filling strategies. However, it is worth noting that in many real-world applications, even if two points have the same spatial distance from existing samples, they can have significantly different impacts on the experiment. For instance, in a chemical reaction, different variations in temperature may cause different shifts in the steady state. To address this issue, one approach is to incorporate prior knowledge to pre-process the data before inputting it into the exploration function. Additionally, for mixed-variable problems, it may be useful to investigate the adoption of acquisition strategies in PWAS to other BO methods, especially the ones with tree-based kernels.

In the following, we note some open questions in preference-based optimization problems, which encompass various aspects, including eliciting and modeling user preferences, handling uncertainty and ambiguity in preferences, and integrating preferences into decision-making processes.

Firstly, accurately capturing and understanding decision-maker's preferences pose a significant challenge. It is because the expressed preferences can be subjective and complex. Furthermore, modeling these preferences to be compatible with the optimization process requires careful consideration. Additionally, we may want to reach a consensus among different users. In such cases, one may extend the preference-based optimization approach to a distributed scheme similar to the approach discussed in [167].

Secondly, decision-makers often encounter uncertainty and ambiguity when expressing their preferences. Preferences can be imprecise, conflicting, or incomplete. In our approach, we allow inconsistent preferences to a certain extent by introducing the slack variables when fitting the surrogate (cf. (2.10)). Other methods may be developed to represent and reason with uncertain preference information.

Lastly, integrating preferences into real-world decision-making processes can be challenging. Decision-makers need to consider various other factors, such as feasibility, cost, and constraints, alongside their preferences (cf. Chapter 4). Some engineering interfacing techniques may be investigated to find effective ways to incorporate preferences into the decision-making process and align them with other decision criteria.

Also, as advancements in large language models (LLMs) continue to unfold rapidly, there emerges the intriguing prospect of entrusting decision-making tasks to LLM agents. For example, in a study by Mao *et al.* [168], the authors suggested employing the Generative Pre-trained Transformer (GPT) agent for the generation of driving scenarios. This overarching concept can be viewed through the lens of "AI/LLM-in-the-loop" optimization as opposed to the conventional "human-in-the-loop" optimization prevalent in many preference-based optimization approaches. This shift signifies a pivotal juncture in the intersection of artificial intelligence and decision-making processes, paving the way for a deeper exploration of the roles LLMs can play in diverse domains beyond traditional human decision-making paradigms, which can be an interesting research area to explore.

Bibliography

[1] Mengjia Zhu, Alberto Bemporad, and Dario Piga. "Preference-based MPC calibration". In: *European Control Conference (ECC)*. IEEE. 2021, pp. 638–645.

[2] Mengjia Zhu, Dario Piga, and Alberto Bemporad. "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration". In: *IEEE Transactions on Control Systems Technology* 30.5 (2022), pp. 2176–2187.

[3] Mengjia Zhu and Alberto Bemporad. "Global and Preference-based Optimization with Mixed Variables using Piecewise Affine Surrogates". In: *arXiv preprint arXiv:2302.04686* (2023).

[4] Mengjia Zhu et al. "Discrete and mixed-variable experimental design with surrogate-based approach". In: *ChemRxiv preprint doi:10.26434/chemrxiv-2024-h37x4* (2024).

[5] Dietmar Maringer and Panos Parpas. "Global optimization of higher order moments in portfolio selection." In: *Journal of Global optimization* 43 (2009).

[6] Stewart Greenhill et al. "Bayesian optimization for adaptive experimental design: A review". In: *IEEE access* 8 (2020), pp. 13937–13948.

[7] Ramkumar Karuppiah, Kevin C Furman, and Ignacio E Grossmann. "Global optimization for scheduling refinery crude oil operations". In: *Computers & Chemical Engineering* 32.11 (2008), pp. 2745–2766.

[8] Jie Li, Ruth Misener, and Christodoulos A Floudas. "Scheduling of crude oil operations under demand uncertainty: A robust optimization framework coupled with global optimization". In: *AIChE journal* 58.8 (2012), pp. 2373–2396.

[9] Graham Kendall et al. “Scheduling in sports: An annotated bibliography”. In: *Computers & Operations Research* 37.1 (2010), pp. 1–19.

[10] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. “SVM parameter optimization using grid search and genetic algorithm to improve classification performance”. In: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 14.4 (2016), pp. 1502–1509.

[11] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012).

[12] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[13] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.

[15] David E Goldberg, Bradley Korb, and Kalyanmoy Deb. “Messy genetic algorithms: Motivation, analysis, and first results”. In: *Complex systems* 3.5 (1989), pp. 493–530.

[16] James Kennedy and Russell Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[17] Thomas Weise. “Global optimization algorithms-theory and application”. In: *Self-Published Thomas Weise* 361 (2009).

[18] Margaret A Oliver and Richard Webster. “Kriging: a method of interpolation for geographical information systems”. In: *International Journal of Geographical Information System* 4.3 (1990), pp. 313–332.

[19] Donald R Jones, Matthias Schonlau, and William J Welch. “Efficient global optimization of expensive black-box functions”. In: *Journal of Global optimization* 13.4 (1998), p. 455.

[20] Charles Audet. *A survey on direct search methods for blackbox optimization and their applications*. Springer, 2014.

[21] Virginia Torczon. “On the convergence of pattern search algorithms”. In: *SIAM Journal on optimization* 7.1 (1997), pp. 1–25.

[22] Enrico Fermi. *Numerical solution of a minimum problem*. Tech. rep. Los Alamos Scientific Lab., Los Alamos, NM, 1952.

[23] Charles Audet and John E Dennis Jr. “Mesh adaptive direct search algorithms for constrained optimization”. In: *SIAM Journal on optimization* 17.1 (2006), pp. 188–217.

[24] Mark A Abramson et al. “OrthoMADS: A deterministic MADS instance with orthogonal directions”. In: *SIAM Journal on Optimization* 20.2 (2009), pp. 948–966.

[25] John A Nelder and Roger Mead. “A simplex method for function minimization”. In: *The computer journal* 7.4 (1965), pp. 308–313.

[26] Robert Hooke and Terry A Jeeves. ““Direct Search”Solution of Numerical and Statistical Problems”. In: *Journal of the ACM (JACM)* 8.2 (1961), pp. 212–229.

[27] Bobak Shahriari et al. “Taking the human out of the loop: A review of Bayesian optimization”. In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[28] Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[29] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. “Ant colony optimization”. In: *IEEE computational intelligence magazine* 1.4 (2006), pp. 28–39.

[30] Zong Woo Geem, Joong Hoon Kim, and Gobichettipalayam Vasudevan Loganathan. “A new heuristic optimization algorithm: harmony search”. In: *simulation* 76.2 (2001), pp. 60–68.

[31] Xin-She Yang and Xingshi He. “Firefly algorithm: recent advances and applications”. In: *International journal of swarm intelligence* 1.1 (2013), pp. 36–50.

[32] A. Bemporad. “Global optimization via inverse distance weighting and radial basis functions”. In: *Computational Optimization and Applications* 77 (2020). Code available at `http://cse.lab.imtlucca.it/~bemporad/glis`, pp. 571–595.

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

[34] Andrzej P Wierzbicki. "The use of reference objectives in multiobjective optimization". In: *Multiple Criteria Decision Making Theory and Application: Proceedings of the Third Conference Hagen/Königswinter, West Germany, August 20–24, 1979*. Springer. 1980, pp. 468–486.

[35] Ralph L Keeney and Howard Raiffa. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993.

[36] Roseanna W Saaty. "The analytic hierarchy process—what it is and how it is used". In: *Mathematical modelling* 9.3-5 (1987), pp. 161–176.

[37] Christian Wirth et al. "A survey of preference-based reinforcement learning methods". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 4945–4990.

[38] Alberto Bemporad and Dario Piga. "Global optimization based on active preference learning with radial basis functions". In: *Machine Learning* 110 (2021), pp. 417–448.

[39] Li Chen and Pearl Pu. *Survey of preference elicitation methods*. Tech. rep. EPFL, 2004.

[40] Louis P Hagopian, Ethan S Long, and Karena S Rush. "Preference assessment procedures for individuals with developmental disabilities". In: *Behavior Modification* 28.5 (2004), pp. 668–677.

[41] Ashish Kapoor et al. "Interactive optimization for steering machine classification". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2010, pp. 1343–1352.

[42] Hendrik S Houthakker. "Revealed preference and the utility function". In: *Economica* 17.66 (1950), pp. 159–174.

[43] Fabio Aiolli and Alessandro Sperduti. "A preference optimization based unifying framework for supervised learning problems". In: *Preference learning*. Springer, 2010, pp. 19–42.

[44] Willem Waegeman and Bernard De Baets. "A transitivity analysis of bipartite rankings in pairwise multi-class classification". In: *Information Sciences* 180.21 (2010), pp. 4099–4117.

[45] Eyke Hüllermeier et al. "Label ranking by learning pairwise preferences". In: *Artificial Intelligence* 172.16-17 (2008), pp. 1897–1916.

[46] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge university press, 2012.

[47] David Kreps. *Notes On The Theory Of Choice*. Westview Press, 1988.

[48] H.M. Gutmann. "A Radial Basis Function Method for Global Optimization". In: *Journal of Global Optimization* 19 (2001), pp. 201–2227.

[49] D.B. McDonald et al. "Global and local optimization using radial basis function response surface models". In: *Applied Mathematical Modelling* 31.10 (2007), pp. 2095–2110.

[50] A.J. Smola and B. Schölkopf. "A tutorial on support vector regression". In: *Statistics and Computing* 14 (2004), pp. 199–222.

[51] M. Stone. "Cross-validatory choice and assessment of statistical predictions". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2 (1974), pp. 111–133.

[52] Nikolaos Ploskas and Nikolaos V Sahinidis. "Review and comparison of algorithms and software for mixed-integer derivative-free optimization". In: *Journal of Global Optimization* (2022), pp. 1–30.

[53] Juliane Müller. "MISO: mixed-integer surrogate optimization framework". In: *Optimization and Engineering* 17 (2016), pp. 177–203.

[54] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. "Sequential model-based optimization for general algorithm configuration". In: *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*. Springer. 2011, pp. 507–523.

[55] James Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24 (2011).

[56] Charles Audet et al. "NOMAD version 4: Nonlinear optimization with the MADS algorithm". In: *arXiv preprint arXiv:2104.11627* (2021).

[57] Margherita Porcelli and Philippe L Toint. "BFO, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables". In: *ACM Transactions on Mathematical Software (TOMS)* 44.1 (2017), pp. 1–25.

[58] Brian M Adams et al. *Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 6.13 user's manual*. Tech. rep. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.

[59] Martin Schlüter, Jose A Egea, and Julio R Banga. "Extended ant colony optimization for non-convex mixed integer nonlinear programming". In: *Computers & Operations Research* 36.7 (2009), pp. 2217–2229.

[60] Alberto Bemporad. "A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems". In: *IEEE Transactions on Automatic Control* (2022). Code available at `http://cse.lab.imtlucca.it/bemporad/parc`.

[61] Laurence Charles Ward Dixon. "The global optimization problem: an introduction". In: *Towards Global Optimiation 2* (1978), pp. 1–15.

[62] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*. Vol. 2. Nob Hill Publishing Madison, WI, 2017.

[63] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[64] Manfred Morari and Jay H Lee. "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667–682.

[65] Daniele Masti and Alberto Bemporad. "Learning nonlinear state–space models using autoencoders". In: *Automatica* 129 (2021), p. 109666.

[66] Marco Forgione and Dario Piga. "Continuous-time system identification with neural networks: Model structures and fitting criteria". In: *European Journal of Control* 59 (2021), pp. 69–81.

[67] Alessandro Alessio and Alberto Bemporad. "A survey on explicit model predictive control". In: *Nonlinear Model Predictive Control: Towards New Challenging Applications* (2009), pp. 345–369.

[68] Corentin Briat. "Linear parameter-varying and time-delay systems". In: *Analysis, observation, filtering & control* 3 (2014), pp. 5–7.

[69] Marcelo M Morato, Julio E Normey-Rico, and Olivier Sename. "Model predictive control design for linear parameter varying systems: A survey". In: *Annual Reviews in Control* 49 (2020), pp. 64–80.

[70] Alberto Bemporad and Claudio Rocchi. "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles". In: *2011 50th IEEE conference on decision and control and European control conference*. IEEE. 2011, pp. 7488–7493.

[71] Marco Forgione, Dario Piga, and Alberto Bemporad. "Efficient Calibration of Embedded MPC". In: *Proc. of the 21st IFAC World Congress*. Berlin, Germany, 2020.

[72] A. Lucchini et al. "Torque vectoring for high-performance electric vehicles: an efficient MPC calibration". In: *IEEE Control Systems Letters* (2020).

[73] Dario Piga et al. "Performance-oriented model learning for data-driven MPC design". In: *IEEE Control Systems Letters* 3.3 (2019), pp. 577–582.

[74] Somil Bansal et al. "Goal-driven dynamics learning via Bayesian optimization". In: *Proc. of the IEEE 56th Annual Conference on Decision and Control*. 2017, pp. 5168–5173.

[75] Marcello Fiducioso et al. "Safe Contextual Bayesian Optimization for Sustainable Room Temperature PID Control Tuning". In: *Proc. of the 28th IJCAI*. Macao, China, 2019, pp. 5850–5856.

[76] Alonso Marco et al. "Automatic LQR tuning based on Gaussian process global optimization". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 270–277.

[77] Loris Roveda, Marco Forgione, and Dario Piga. "Robot control parameters auto-tuning in trajectory tracking applications". In: *Control Engineering Practice* 101 (2020).

[78] Danny Drieß, Peter Englert, and Marc Toussaint. "Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 902–907.

[79] Roberto Calandra et al. "Bayesian gait optimization for bipedal locomotion". In: *International Conference on Learning and Intelligent Optimization*. Springer. 2014, pp. 274–290.

[80] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. "Safe controller optimization for quadrotors with Gaussian processes". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 491–496.

[81] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4299–4307.

[82] Marcel Menner et al. "Inverse learning for human-adaptive motion planning". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 809–815.

[83] Sascha Rosbach et al. "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving". In: *arXiv preprint arXiv:1905.00229* (2019).

[84] Wenshuo Wang et al. "Driving style classification using a semisupervised support vector machine". In: *IEEE Transactions on Human-Machine Systems* 47.5 (2017), pp. 650–660.

[85] Sébastien Gros et al. "From linear to nonlinear MPC: bridging the gap via the real-time iteration". In: *International Journal of Control* 93.1 (2020), pp. 62–80.

[86] Moritz Diehl, Hans Georg Bock, and Johannes P. Schlöder. "A real-time iteration scheme for nonlinear optimization in optimal feedback control". In: *SIAM Journal on Control and Optimization* 43.5 (2005), pp. 1714–1736.

[87] A.I.F. Vaz and L.N. Vicente. "PSwarm: A hybrid solver for linearly constrained global derivative-free optimization". In: *Optimization Methods and Software* 24 (2009). `http://www.norg.uminho.pt/aivaz/pswarm/`, pp. 669–685.

[88] M.D. McKay, R.J. Beckman, and W.J. Conover. "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code". In: *Technometrics* 21.2 (1979), pp. 239–245.

[89] The MathWorks. *Statistics and Machine Learning Toolbox*. Natick, Massachusetts, United State, 2019. URL: `https://www.mathworks.com/products/statistics.html`.

[90] A. Bemporad, L. Ricker, and M. Morari. *Model Predictive Control Toolbox*. Natick, Massachusetts, United State, 2019. URL: `https://www.mathworks.com/products/mpc.html`.

[91] B. Wayne Bequette. *Process dynamics: modeling, analysis, and simulation*. Prentice hall PTR New Jersey, 1998.

[92] G. Matheron. "Principles of geostatistics". In: *Economic geology* 58.8 (1963), pp. 1246–1266.

[93] H.J. Kushner. "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise". In: *Journal of Basic Engineering* 86.1 (1964), pp. 97–106.

[94] J. Sacks et al. "Design and analysis of computer experiments". In: *Statistical Science* (1989), pp. 409–423.

[95] D.R. Jones, M. Schonlau, and W.J. Matthias. "Efficient global optimization of expensive black-box functions". In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[96] Gianluca Savaia et al. "Experimental automatic calibration of a semi-active suspension controller via Bayesian Optimization". In: *Control Engineering Practice* 112 (2021).

[97] Gang Luo. "A review of automatic selection methods for machine learning algorithms and hyper-parameter values". In: *Network Modeling Analysis in Health Informatics and Bioinformatics* 5.1 (2016), pp. 1–16.

[98] Loris Roveda et al. "Human–robot collaboration in sensorless assembly task learning enhanced by uncertainties adaptation via Bayesian Optimization". In: *Robotics and Autonomous Systems* 136 (2021), p. 103711.

[99] Wei Chu and Zoubin Ghahramani. "Extensions of Gaussian processes for ranking: semisupervised and active learning". In: *Learning to Rank* 29 (2005).

[100] Eric Brochu, Nando De Freitas, and Abhijeet Ghosh. "Active Preference Learning with Discrete Choice Data." In: *NIPS*. 2007, pp. 409–416.

[101] Javier González et al. "Preferential Bayesian optimization". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1282–1291.

[102] Majid Abdolshah et al. “Multi-objective Bayesian optimisation with preferences over objectives”. In: *Advances in neural information processing systems* 32 (2019).

[103] Yanan Sui et al. “Stagewise Safe Bayesian Optimization with Gaussian Processes”. In: *Proc. of the 35th ICML*. PMLR. 2018, pp. 4781–4789.

[104] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. “Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics”. In: *Machine Learning* (2021), pp. 1–35.

[105] Michael A. Gelbart, Jasper Snoek, and Ryan P. Adams. “Bayesian Optimization with Unknown Constraints”. In: *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. Arlington, VA, USA, 2014, pp. 250–259.

[106] Antonio Candelieri. “Sequential model based optimization of partially defined functions under unknown constraints”. In: *Journal of Global Optimization* (2019), pp. 1–23.

[107] V Roshan Joseph and Lulu Kang. “Regression-based inverse distance weighting with applications to computer experiments”. In: *Technometrics* 53.3 (2011), pp. 254–265.

[108] Alberto Costa and Giacomo Nannicini. “RBFOpt: an open-source library for black-box optimization with costly function evaluations”. In: *Mathematical Programming Computation* 10 (2018), pp. 597–629.

[109] Rommel G Regis and Christine A Shoemaker. “Constrained global optimization of expensive black box functions using radial basis functions”. In: *Journal of Global optimization* 31.1 (2005), pp. 153–171.

[110] L.M. Rios and N.V. Sahinidis. “Derivative-free optimization: a review of algorithms and comparison of software implementations”. In: *Journal of Global Optimization* 56.3 (2013), pp. 1247–1293.

[111] Sudhanshu K Mishra. “Some new test functions for global optimization and performance of repulsive particle swarm method”. In: *Available at SSRN 926132* (2006).

[112] Phenix Integration. *Bird Problem (Constrained)*. https://web.archive.org/web/20161229032528/http://www.phoenix-int.com/software/benchmark_report/bird_constrained.php. Accessed: 2021-04-18. 2016.

[113] Momin Jamil and Xin-She Yang. "A literature survey of benchmark functions for global optimisation problems". In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013), pp. 150–194.

[114] Charles Audet, Edward Hallé-Hannan, and Sébastien Le Digabel. "A general mathematical framework for constrained mixed-variable blackbox optimization problems with meta and categorical variables". In: *Operations Research Forum*. Vol. 4. Springer. 2023, p. 12.

[115] Sun Hye Kim and Fani Boukouvala. "Surrogate-based optimization for mixed-integer nonlinear problems". In: *Computers & Chemical Engineering* 140 (2020), p. 106847.

[116] Miten Mistry et al. "Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded". In: *INFORMS Journal on Computing* 33.3 (2021), pp. 1103–1119.

[117] Kenneth Holmström. "An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization". In: *Journal of Global Optimization* 41 (2008), pp. 447–464.

[118] GPyOpt. *GPyOpt: A Bayesian Optimization framework in Python*. http://github.com/SheffieldML/GPyOpt. 2016.

[119] Mark A Abramson et al. *The NOMAD project*. Software available at http://www.gerad.ca/nomad. 2011.

[120] Laura P Swiler et al. "Surrogate models for mixed discrete-continuous variables". In: *Constraint Programming and Decision Making* (2014), pp. 181–202.

[121] Shivapratap Gopakumar et al. "Algorithmic assurance: An active approach to algorithmic testing using bayesian optimisation". In: *Advances in Neural Information Processing Systems* 31 (2018).

[122] Vu Nguyen. "Bayesian optimization for accelerating hyper-parameter tuning". In: *2019 IEEE second international conference on artificial intelligence and knowledge engineering (AIKE)*. IEEE. 2019, pp. 302–305.

[123] Binxin Ru et al. "Bayesian optimisation over multiple continuous and categorical inputs". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8276–8285.

[124] Theodore P Papalexopoulos et al. "Constrained discrete blackbox optimization using mixed-integer programming". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 17295–17322.

[125] H.P. Williams. *Model Building in Mathematical Programming*. 5$^{\text{th}}$. John Wiley & Sons, 2013.

[126] J.N. Hooker and M.A. Osorio. "Mixed logical/linear programming". In: *Discrete Applied Mathematics* 96–97 (1999), pp. 395–442.

[127] F.D. Torrisi and A. Bemporad. "HYSDEL — A Tool for Generating Computational Hybrid Models". In: *IEEE Trans. Contr. Systems Technology* 12.2 (Mar. 2004), pp. 235–249.

[128] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2023. URL: `https://www.gurobi.com`.

[129] Andrew Makhorin. *GNU Linear Programming Kit. Reference Manual. Version 5.0*. 2020.

[130] Michael D McKay, Richard J Beckman, and William J Conover. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code". In: *Technometrics* 21.2 (1979), pp. 239–245.

[131] Theodore S Motzkin et al. "The double description method". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 51–73.

[132] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[133] David Roxbcc Cox. "Some Procedures Connccted With the Logistic Qualitative Response Curve". In: *Research Papers in Statistics* (1951), pp. 55–71.

[134] Henri Theil. "A multinomial extension of the linear logit model". In: *International economic review* 10.3 (1969), pp. 251–259.

[135] Linas Stripinis and Remigijus Paulavičius. "DIRECTGO: A new DIRECT-type MATLAB toolbox for derivative-free global optimization". In: *ACM Transactions on Mathematical Software* 48.4 (2022), pp. 1–46.

[136] Stuart Mitchell, Michael OSullivan, and Iain Dunning. "PuLP: a linear programming toolkit for python". In: *The University of Auckland, Auckland, New Zealand* 65 (2011).

[137] Getdata Graph Digitizer. *GetData Graph Digitizer: Version 2.26*. 2013. URL: `http://getdata-graph-digitizer.com/`.

[138] Riccardo Leardi. "Experimental design in chemistry: A tutorial". In: *Analytica chimica acta* 652.1-2 (2009), pp. 161–172.

[139] Norio Miyaura, Kinji Yamada, and Akira Suzuki. "A new stereospecific cross-coupling by the palladium-catalyzed reaction of 1-alkenylboranes with 1-alkenyl or 1-alkynyl halides". In: *Tetrahedron Letters* 20.36 (1979), pp. 3437–3440.

[140] Norio Miyaura and Akira Suzuki. "Palladium-catalyzed cross-coupling reactions of organoboron compounds". In: *Chemical reviews* 95.7 (1995), pp. 2457–2483.

[141] Timothy E Barder et al. "Catalysts for Suzuki- Miyaura coupling processes: scope and studies of the effect of ligand structure". In: *Journal of the American Chemical Society* 127.13 (2005), pp. 4685–4696.

[142] Alastair JJ Lennox and Guy C Lloyd-Jones. "Selection of boron reagents for Suzuki–Miyaura coupling". In: *Chemical Society Reviews* 43.1 (2014), pp. 412–443.

[143] Benjamin J Shields et al. "Bayesian reaction optimization as a tool for chemical synthesis". In: *Nature* 590.7844 (2021), pp. 89–96.

[144] Damith Perera et al. "A platform for automated nanomole-scale reaction screening and micromole-scale synthesis in flow". In: *Science* 359.6374 (2018), pp. 429–434.

[145] Riley Hickman et al. *Olympus, enhanced: benchmarking mixed-parameter and multi-objective optimization in chemistry and materials science*. en. May 2023. DOI: `10.26434/chemrxiv-2023-74w8d`. (Visited on 06/21/2023).

[146] Félix-Antoine Fortin et al. "DEAP: Evolutionary algorithms made easy". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 2171–2175.

[147] François-Michel De Rainville et al. "Deap: A python framework for evolutionary algorithms". In: *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. 2012, pp. 85–92.

[148] James Bergstra, Daniel Yamins, and David Cox. “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures”. In: *International conference on machine learning*. PMLR. 2013, pp. 115–123.

[149] Maximilian Balandat et al. “BoTorch: A framework for efficient Monte-Carlo Bayesian optimization”. In: *Advances in neural information processing systems* 33 (2020), pp. 21524–21538.

[150] Aldair E Gongora et al. “A Bayesian experimental autonomous researcher for mechanical design”. In: *Science advances* 6.15 (2020), eaaz1708.

[151] Matthew B Plutschack et al. “The hitchhiker's guide to flow chemistry‖”. In: *Chemical reviews* 117.18 (2017), pp. 11796–11893.

[152] Brandon J Reizman and Klavs F Jensen. “Feedback in flow for accelerated reaction development”. In: *Accounts of chemical research* 49.9 (2016), pp. 1786–1796.

[153] Lingfeng Gui et al. “Integrating model-based design of experiments and computer-aided solvent design”. In: *Computers & Chemical Engineering* 177 (2023), p. 108345.

[154] Gaia Franceschini and Sandro Macchietto. “Model-based design of experiments for parameter precision: State of the art”. In: *Chemical Engineering Science* 63.19 (2008), pp. 4846–4872.

[155] Heiko Struebing et al. “Computer-aided molecular design of solvents for accelerated reaction kinetics”. In: *Nature chemistry* 5.11 (2013), pp. 952–957.

[156] Amol Shivajirao Hukkerikar et al. “Estimation of environment-related properties of chemicals for design of sustainable processes: development of group-contribution+ (GC+) property models and uncertainty analysis”. In: *Journal of chemical information and modeling* 52.11 (2012), pp. 2823–2839.

[157] Amol Shivajirao Hukkerikar et al. “Group-contribution+ (GC+) based estimation of properties of pure components: Improved property estimation and uncertainty analysis”. In: *Fluid Phase Equilibria* 321 (2012), pp. 25–43.

[158] Mortimer J Kamlet, Jose Luis Abboud, and RW Taft. “The solvatochromic comparison method. 6. The. pi.* scale of solvent polarities”. In: *Journal of the American Chemical Society* 99.18 (1977), pp. 6027–6038.

[159] Michael H Abraham et al. “Linear solvation energy relationships. Part 37. An analysis of contributions of dipolarity–polarisability, nucleophilic assistance, electrophilic assistance, and cavity terms to solvent effects on t-butyl halide solvolysis rates”. In: *Journal of the Chemical Society, Perkin Transactions 2* 7 (1987), pp. 913–920.

[160] Alberto Bemporad. “A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems”. In: *IEEE Transactions on Automatic Control* 68.6 (2023), pp. 3194–3209.

[161] Lingfeng Gui. “Solvent design assisted by mechanistic insights: methods and application to peptide synthesis”. PhD thesis. Imperial College London, 2024.

[162] Aleksandr V Marenich, Christopher J Cramer, and Donald G Truhlar. “Universal solvation model based on solute electron density and on a continuum model of the solvent defined by the bulk dielectric constant and atomic surface tensions”. In: *The Journal of Physical Chemistry B* 113.18 (2009), pp. 6378–6396.

[163] TJ Sheldon, CS Adjiman, and JL Cordiner. “Pure component properties from group contribution Hydrogen-bond basicity, hydrogen-bond acidity, Hildebrand solubility parameter, macroscopic surface tension, dipole moment, refractive index and dielectric constant”. In: *Fluid Phase Equilibria* 231.1 (2005), pp. 27–37.

[164] Christian Reichardt and Thomas Welton. *Solvents and solvent effects in organic chemistry*. John Wiley & Sons, 2011.

[165] James Sherwood et al. “N-Butylpyrrolidinone as a dipolar aprotic solvent for organic synthesis”. In: *Green Chemistry* 18.14 (2016), pp. 3990–3996.

[166] Haydar Taylan Turan, Sebastian Brickel, and Markus Meuwly. “Solvent effects on the Menshutkin reaction”. In: *The Journal of Physical Chemistry B* 126.9 (2022), pp. 1951–1961.

[167] Loris Cannelli et al. “Multi-agent active learning for distributed black-box optimization”. In: *IEEE Control Systems Letters* (2023).

[168] Jiageng Mao et al. “GPT-Driver: Learning to Drive with GPT”. In: *NeurIPS 2023 Foundation Models for Decision Making Workshop*. 2023.

[169] S. Surjanovic and D. Bingham. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved February 1, 2023, from `http://www.sfu.ca/~ssurjano`.

[170] Marcin Molga and Czesław Smutnicki. "Test functions for optimization needs". In: *Test functions for optimization needs* 101 (2005), p. 48.

[171] Evelyn Martin Lansdowne Beale. *On an iterative method for finding a local minimum of a function of more than one variable*. 25. Statistical Techniques Research Group, Section of Mathematical Statistics, Department of Mathematics, Princeton University, 1958.

[172] Jorge J Moré, Burton S Garbow, and Kenneth E Hillstrom. "Testing unconstrained optimization software". In: *ACM Transactions on Mathematical Software (TOMS)* 7.1 (1981), pp. 17–41.

[173] David H Ackley. "The model". In: *A Connectionist Machine for Genetic Hillclimbing*. Springer, 1987, pp. 29–70.

[174] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[175] Yann LeCun, Corinna Cortes, and Chris Burges. *MNIST handwritten digit database*. 2010. URL: `http://yann.%20lecun.%20com/exdb/mnist`.

[176] Chris Ying et al. "Nas-bench-101: Towards reproducible neural architecture search". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7105–7114.

[177] Aaron Klein and Frank Hutter. "Tabular benchmarks for joint architecture and hyperparameter optimization". In: *arXiv preprint arXiv:1905.04970* (2019).

[178] Kevin K Yang, Zachary Wu, and Frances H Arnold. "Machine-learning-guided directed evolution for protein engineering". In: *Nature methods* 16.8 (2019), pp. 687–694.

[179] Reiner Horst, Panos M Pardalos, and Nguyen Van Thoai. *Introduction to global optimization*. Springer Science & Business Media, 2000.

[180] Hoai An Le Thi, A Ismael F Vaz, and LN Vicente. "Optimizing radial basis functions by DC programming and its use in direct search for global derivative-free optimization". In: *Top* 20 (2012), pp. 190–214.

[181] Momin Jamil and Xin-She Yang. "A literature survey of benchmark functions for global optimisation problems". In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013), pp. 150–194.

[182] Peter Auer et al. "The nonstochastic multiarmed bandit problem". In: *SIAM journal on computing* 32.1 (2002), pp. 48–77.

[183] Yevgeny Seldin et al. "Evaluation and analysis of the performance of the EXP3 algorithm in stochastic environments". In: *European Workshop on Reinforcement Learning*. PMLR. 2013, pp. 103–116.

[184] Derek T Ahneman et al. "Predicting reaction performance in C–N cross-coupling using machine learning". In: *Science* 360.6385 (2018), pp. 186–190.

[185] Hirotomo Moriwaki et al. "Mordred: a molecular descriptor calculator". In: *Journal of cheminformatics* 10.1 (2018), pp. 1–14.

[186] Shuhei Watanabe and Frank Hutter. "c-TPE: tree-structured parzen estimator with inequality constraints for expensive hyperparameter optimization". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 2023, pp. 4371–4379.

[187] Sébastien Le Digabel et al. *NOMAD user guide version 3.9. 1*. 2019.

[188] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[189] Kalyanmoy Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197.

[190] Ahmed Fawzy Gad. "Pygad: An intuitive genetic algorithm python library". In: *Multimedia Tools and Applications* (2023), pp. 1–14.

[191] Diego Giacomelli. *GeneticSharp*. `https://https://github.com/giacomelli/GeneticSharp`. 2017.

[192] Max Halford. *eaopt: Evolutionary optimization library for Go (genetic algorithm, particle swarm optimization, differential evolution)*. `https://github.com/MaxHalford/eaopt`. 2016.

[193] Austin Tripp. *mol_ga: Simple, lightweight package for genetic algorithms on molecules*. `https://github.com/AustinT/mol_ga`. 2023.

[194] Jan H Jensen. "A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space". In: *Chemical science* 10.12 (2019), pp. 3567–3572.

[195] Nathan Brown et al. "GuacaMol: benchmarking models for de novo molecular design". In: *Journal of chemical information and modeling* 59.3 (2019), pp. 1096–1108.

[196] Kashif Hussain et al. "Metaheuristic research: a comprehensive survey". In: *Artificial intelligence review* 52 (2019), pp. 2191–2233.

[197] Yaochu Jin. "Surrogate-assisted evolutionary computation: Recent advances and future challenges". In: *Swarm and Evolutionary Computation* 1.2 (2011), pp. 61–70.

[198] Linqiang Pan et al. "A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization". In: *IEEE Transactions on Evolutionary Computation* 23.1 (2018), pp. 74–88.

[199] Carlos A Coello Coello. "Constraint-handling techniques used with evolutionary algorithms". In: *Proceedings of the genetic and evolutionary computation conference companion*. 2022, pp. 1310–1333.

Appendix A

Benchmark

Note: the unconstrained mixed-variable synthetic and real-world problems are adopted from [123] for our comparisons. Note that the objective function is maximized in [123], so we consider the minimization of $-f(X)$ in PWAS and PWASp.

A.1 Unconstrained mixed-variable synthetic benchmarks

Func-2C [61, 123, 169–172]: $n_c = 2$, $n_{\text{int}} = 0$, and $n_d = 2$ with $n_i = 3$ for each categorical variable (denoted as n_{di}, for $i = 1, 2$). Each categorical variable is in $\{0, 1, 2\}$. The bounds are $\ell_x = [-1.0 \ -1.0]^\mathsf{T}$, $u_x = [1.0 \ 1.0]^\mathsf{T}$. The global maximum $f(X) = 0.20632$ is attained at $X = [0.0898 \ -0.7126 \ 1 \ 1]^\mathsf{T}$ and $[-0.0898 \ 0.7126 \ 1 \ 1]^\mathsf{T}$.

$$
\begin{aligned}
f(X) &= \begin{cases} f_1 + f_{\text{ros}}(x) & n_{d2} = 0 \\ f_1 + f_{\text{cam}}(x) & n_{d2} = 1 \\ f_1 + f_{\text{bea}}(x) & n_{d2} = 2 \end{cases} \\
\text{where} \quad f_1(x, y) &= \begin{cases} f_{\text{ros}}(x) & n_{d1} = 0 \\ f_{\text{cam}}(x) & n_{d1} = 1 \\ f_{\text{bea}}(x) & n_{d1} = 2 \end{cases} \\
f_{\text{ros}}(x) &= -(100(x_2 - x_1^2)^2 + (x_1 - 1)^2)/300 \\
f_{\text{cam}}(x) &= -(a_1 + a_2 + a_3)/10 \\
a_1 &= (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 \\
a_2 &= x_1 x_2 \\
a_3 &= (-4 + 4x_2^2)x_2^2 \\
f_{\text{bea}}(x) &= -((1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 \\
&\quad + (2.625 - x_1 + x_1 x_2^3)^2)/50
\end{aligned}
\tag{A.1}
$$

Func-3C [61, 123, 169–172]: $n_c = 2$, $n_{\text{int}} = 0$, and $n_d = 3$ with $n_i = 3$ for each categorical variable (denoted as n_{di}, for $i = 1, 2, 3$). Each categorical variable is in $\{0, 1, 2\}$. The bounds are $\ell_x = [-1.0\ -1.0]^\mathsf{T}$, $u_x = [1.0\ 1.0]^\mathsf{T}$. The global maximum $f(X) = 0.72214$ is attained at $X = [0.0898\ -0.7126\ 1\ 1\ 0]^\mathsf{T}$ and $[-0.0898\ 0.7126\ 1\ 1\ 0]^\mathsf{T}$.

$$
\begin{aligned}
f(X) &= \begin{cases} f_2 + 5f_{\text{cam}}(x) & n_{d3} = 0 \\ f_2 + 2f_{\text{ros}}(x) & n_{d3} = 1 \\ f_2 + n_{d2}f_{\text{bea}}(x) & n_{d3} = 2 \end{cases} \\
\text{where} \quad f_2(x, y) &= \begin{cases} f_1 + f_{\text{ros}}(x) & n_{d21} = 0 \\ f_1 + f_{\text{cam}}(x) & n_{d2} = 1 \\ f_1 + f_{\text{bea}}(x) & n_{d2} = 2 \end{cases} \\
f_1(x, y) &= \begin{cases} f_{\text{ros}}(x) & n_{d1} = 0 \\ f_{\text{cam}}(x) & n_{d1} = 1 \\ f_{\text{bea}}(x) & n_{d1} = 2 \end{cases} \\
& f_{\text{ros}}(x), f_{\text{cam}}(x), \text{ and } f_{\text{bea}}(x) \text{ are defined in (A.1)}
\end{aligned}
\tag{A.2}
$$

Ackley-5C [123, 169, 173]: $n_c = 1$, $n_{\text{int}} = 0$, and $n_d = 5$ with $n_i = 17$ for each category (denoted as n_{di}, for $i = 1, \ldots, 5$). Each categorical variableis in $\{0, 1, \ldots, 16\}$. The bounds are $\ell_x = -1.0$, $u_x = 1.0$. The global maximum $f(X) = 0$ is attained at $X = [0\ 8\ 8\ 8\ 8\ 8]^\mathsf{T}$.

$$
\begin{aligned}
f(X) &= a \exp\left(-b\left(\frac{s_1}{n}\right)^{\frac{1}{2}}\right) + \exp\left(\frac{s_2}{n}\right) - a - \exp(1) \\
\text{where} \quad & a = 20, b = 0.2, c = 2\pi \\
& s_1 = x^2 + \sum_{i=1}^{5} z_i^2(n_{di}) \\
& s_2 = \cos(cx) + \sum_{i=1}^{5} \cos(cz_i(n_{di})) \\
& z_i(n_{di}) = -1 + 0.125n_{di}
\end{aligned}
\tag{A.3}
$$

A.2 Unconstrained mixed-variable real-world benchmarks

XG-MNIST [123, 174, 175]: $n_c = 4$, $n_{\text{int}} = 1$, and $n_d = 3$ with $n_i = 2$, for $i = 1, 2, 3$. Each categorical variable (n_{di}) can be either 0 or 1. The bounds are $\ell_x = [10^{-6}, 10^{-6}\ 0.001\ 10^{-6}]^{\mathsf{T}}$, $u_x = [1\ 10\ 1\ 5]^{\mathsf{T}}$; $\ell_y = 1$, $u_y = 10$.

Notes on the optimization variables:
The 0.7/0.3 stratified train/test split ratio is applied as noted in [123]. The *xgboost* package is used [174] on MNIST classification [175]. The optimization variables in this problem are the parameters of the *xgboost* algorithm. Specifically, the continuous variables x_1, x_2, x_3, and x_4 refer to the following parameters in *xgboost*, respectively: `'learning_rate'`, `'min_split_loss'`, `'subsample'`, and `'reg_lambda'`. The integer variable y stands for the `'max_depth'`. As for the categorical variables, n_{d1} indicates the booster type in *xgboost* where $n_{d1} = \{0, 1\}$ corresponding to {`'gbtree'`, `'dart'`}. n_{d2} represents the `'grow_policy'`, where $n_{d2} = \{0, 1\}$ corresponding to {`'depthwise'`, `'lossguide'`}. n_{d3} refers to the `'objective'`, where $n_{d3} = \{0, 1\}$ corresponding to {`'multi: softmax'`,`'multi:softprob'`}.

Notes on the objective function:
The classification accuracy on test data is used as the objective function.

NAS-CIFAR10 [123, 176, 177]: $n_c = 21$, $n_{\text{int}} = 1$, and $n_d = 5$ with $n_i = 3$ for each category (denoted as n_{di} for $i = 1, \ldots, 5$). Each categorical variable is in $\{0, 1, 2\}$. The bounds are $\ell_x^i = 0$, $u_x^i = 1$, $\forall i = 1, \ldots, n_c$; $\ell_y = 0$, $u_y = 9$.

Notes on the optimization problem:
The public dataset NAS-Bench-101 [178] is used. This dataset maps

convolutional neural network (CNN) architectures to their trained and evaluated performance on CIFAR-10 classification. As a result, we can quickly look up the validation accuracy of the proposed CNN architecture by PWAS/PWASp. The same encoding method for the CNN architecture topology as noted in [123, 178] is used.

Notes on the optimization variables:
The CNN architecture search space is described by a directed acyclic graph (DAG) which has 7 nodes with the first and the last nodes being the input and output nodes. The continuous variables represent the probability values for the 21 possible edges in the DAG. The integer variable y is the number of edges present in the DAG. The categorical variables are the operations for the 5 intermediate nodes in the DAG, for which $n_{di} = \{0, 1, 2\}$ corresponding to {`'3x3 conv'`, `'1x1 conv'`, `'3x3 max-pool'`}. Within the 21 possible edges, only y edges with the highest probability are activated. The DAG that leads to invalid CNN architecture topology specifications will result in zero validation accuracy.

Notes on the objective function:
The validation accuracy of the defined CNN architecture topology on CIFAR-10 classification is used as the objective function.

A.3 Constrained mixed-variable synthetic problems

Horst6-hs044-modified [179, 180]: $n_c = 3$, $n_{\text{int}} = 4$, and $n_d = 2$ with $n_1 = 3$ and $n_2 = 2$ (the first (n_{d1}) and the second (n_{d2}) categorical variable are in $\{0, 1, 2\}$ and $\{0, 1\}$, respectively. $\ell_x = [0\ 0\ 0]^\mathsf{T}$, $u_x = [6\ 6\ 3]^\mathsf{T}$; $\ell_y = [0\ 0\ 0\ 0]^\mathsf{T}$, $u_y = [3\ 10\ 3\ 10]^\mathsf{T}$. The global minimum $f(X) = -62.579$ is attained at $X = [5.21066\ 5.0279\ 0\ 0\ 3\ 0\ 4\ 2\ 1]^\mathsf{T}$.

$$
\begin{aligned}
f(X) &= \begin{cases} |f_1| & n_{d2} = 0 \\ f_1 & n_{d2} = 1 \end{cases} \\
\text{s.t} \quad & A_{\text{ineq}} x + B_{\text{ineq}} y \le b_{\text{ineq}} \\
\text{where} \quad f_1(x,y) &= \begin{cases} f_{\text{Horst6}}(x) + f_{\text{hs044}}(y) & n_{d1} = 0 \\ 0.5 f_{\text{Horst6}}(x) + f_{\text{hs044}}(y) & n_{d1} = 1 \\ f_{\text{Horst6}}(x) + 2 f_{\text{hs044}}(y) & n_{d1} = 2 \end{cases} \\
f_{\text{Horst6}}(x) &= x^T Q x + p x \\
Q &= \begin{bmatrix} 0.992934 & -0.640117 & 0.337286 \\ -0.640117 & -0.814622 & 0.960807 \\ 0.337286 & 0.960807 & 0.500874 \end{bmatrix} \\
p &= \begin{bmatrix} -0.992372 & -0.046466 & 0.891766 \end{bmatrix} \\
f_{\text{hs044}}(y) &= x_0 - x_1 - x_2 - x_0 x_2 + x_0 x_3 + x_1 x_2 - x_1 x_3 \\
A_{\text{ineq}} &= \begin{bmatrix} 0.488509 & 0.063565 & 0.945686 \\ -0.578592 & -0.324014 & -0.501754 \\ -0.719203 & 0.099562 & 0.445225 \\ -0.346896 & 0.637939 & -0.257623 \\ -0.202821 & 0.647361 & 0.920135 \\ -0.983091 & -0.886420 & -0.802444 \\ -0.305441 & -0.180123 & -0.515399 \end{bmatrix} \\
B_{\text{ineq}} &= \begin{bmatrix} 1 & 2 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\
b_{\text{ineq}} &= [2.86506,\ -1.49161,\ 0.51959,\ 1.58409,\ 2.19804,\ -1.30185, \\
&\qquad -0.73829,\ 8,\ 12,\ 12,\ 8,\ 8,\ 5]^{\mathsf{T}}
\end{aligned}
\tag{A.4}
$$

ros-cam-modified [32, 181]: $n_c = 2$, $n_{\text{int}} = 1$, and $n_d = 2$ with $n_i = 2$ for each categorical variable (denoted as n_{di}, for $i = 1, 2$). Each categorical variable is in $\{0, 1\}$. $\ell_x = [-2.0 \ -2.0]^{\mathsf{T}}$, $u_x = [2.0 \ 2.0]^{\mathsf{T}}$; $\ell_y = 1$, $u_y = 10$. The global minimum $f(X) = -1.81$ is attained at $X = [0.0781 \ 0.6562 \ 5 \ 1 \ 1]^{\mathsf{T}}$.

$$\begin{aligned}
f(X) &= \begin{cases} f_1 + f_{\text{ros}}(x, y) & n_{d2} = 0 \\ f_1 + f_{\text{cam}}(x, y) & n_{d2} = 1 \end{cases} \\
\text{s.t} \quad & A_{\text{ineq}} x \le b_{\text{ineq}} \\
\text{where} \quad & f_1(x, y) = \begin{cases} f_{\text{ros}}(x, y) & n_{d1} = 0 \\ f_{\text{cam}}(x, y) & n_{d1} = 1 \end{cases} \\
& f_{\text{ros}}(x, y) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2 + (y - 3)^2 \\
& f_{\text{cam}}(x, y) = a_1 + a_2 + a_3 + (y - 5)^2 \\
& \qquad a_1 = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 \\
& \qquad a_2 = x_1 x_2 \\
& \qquad a_3 = (-4 + 4x_2^2)x_2^2 \\
& A_{\text{ineq}} = \begin{bmatrix} 1.6295 & 1 \\ 0.5 & 3.875 \\ -4.3023 & -4 \\ -2 & 1 \\ 0.5 & -1 \end{bmatrix} \\
& b_{\text{ineq}} = [3.0786, \ 3.324, \ -1.4909, \ 0.5, \ 0.5]^T
\end{aligned} \tag{A.5}$$

Appendix B

Optimization methods and relevant implementations

B.1 Bayesian Optimization methods

Bayesian Optimization (BO) [33] is a technique often used to optimize expensive black-box functions with limited data. It involves constructing a surrogate model, typically a probabilistic model such as a Gaussian process, to approximate the unknown function. BO follows a general procedure of surrogate-based optimization methods outlined in Section 2.2. To guide the optimization process, various acquisition functions are commonly employed, including expected improvement, upper confidence bound, and probability improvement. These acquisition functions help in selecting the most promising points to evaluate, trading-off exploitation of the surrogate and exploration of the search space, and iteratively refining the surrogate model to find the optimal solution.

B.1.1 CoCaBO

Continuous and Categorical Bayesian Optimization (CoCaBO) [123] is a technique proposed to optimize box-constrained expensive black-box problems with both continuous and categorical variables, specifically for problems with *multiple* categorical variables with *multiple* possible values. CoCaBO model the input space with a Gaussian Process kernel, which is designed to allow information sharing across different categorical variables to enhance data efficiency.

B.1.2 EXP3BO

EXP3BO [121] is a technique proposed to optimize box-constrained, expensive black-box problems, which is modified from BO via the EXP3 algorithm [182, 183]. It can deal with mixed categorical and continuous input spaces by utilizing multi-armed bandits. Specifically, EXP3BO constructs a Gaussian process surrogate specific to each chosen category, making it unsuitable for handling problems with multiple categorical classes.

B.1.3 One-hot BO

One-hot BO [118] handles the input space with continuous and categorical variables by one-hot encoding the categorical variables and treating these encoded variables as continuous. Then, the standard BO noted in Section B.1 is used to solve the optimization problem on the transformed input space.

B.1.4 BoTorch

BoTorc (BO in PyTorch) [149] is a package that implements BO based on GPs. In this framework, categorical variables are one-hot encoded, and users can select different kernels. For instance, for fully-categorical design space, the Hamming distance kernel is commonly used. For mixed-integer cases within the reaction optimization domain, it is recommended to use Matérn5/2 kernel for both continuous and integer (discrete) variables [143, 145]. For mixed-integer cases, *BoTorch* finds the next point to test by iterating through all possible integer values in an outer loop and optimizing the remaining continuous variables while keeping the integer value fixed. Subsequently, it selects the integer value that returns the best evaluation on the acquisition function. This step can cause the computational time to increase significantly, especially when the number of integer (discrete) variables increases and/or the number of possible options for these variables increases. Additionally, constraint handling for mixed-integer cases has not been implemented [149] as conventional approaches, such as trust regions, cannot be trivially integrated with the current framework.

B.1.5 EDBO

EDBO (Experimental Design via Bayesian Optimization) [143] is a package that implements BO based on GPs, where it considers three descriptors to encode the categorical variables, which are density functional theory [184], Mordred [185], and one-hot encoding. Different from *BoTorch*, *EDBO* pre-trains the GP model with data from the literature for the following two reactions: Suzuki-Miyaura reaction [144], consisting of 3,696 reactions in the dataset, and the Buchwald-Hartwig reaction, whose training dataset consists of 3,960 unique reactions [184]. We note that *EDBO* is tailored to solve problems for reaction condition optimization, which only involve categorical variables with finite options. As a result, its workflow involves pre-generating all the possible combinations. When searching the next sample to test, rather than searching within defined bounds, *EDBO* exhaustively enumerates the entire domain, selecting the point with the lowest acquisition function evaluation. For mixed-integer cases, continuous variables first need to be discretized. The enumeration procedure in the acquisition step can become computationally expensive as the number of discretization steps increases, highlighting the trade-off between computational time and achieving a better representation of the original domain. Regarding constraint handling, there is currently no implementation integrated into *EDBO*.

B.1.6 TPE

Tree-structured Parzen Estimator (TPE) [55] is a black-box optimization algorithm that uses tree-structured Parzen density estimators. TPE maintains historical data of “good“ and “bad“ configurations and builds probability density functions based on these data. Notably, TPE is capable of handling both continuous and categorical spaces due to the nature of its kernel density estimators. TPE is implemented in *Hyperopt*, which offers a framework specifically designed to facilitate the application of BO for hyperparameter selection [55]. TPE is computationally cheap and simple compared to many other algorithms within the BO framework [148]. However, incorporating a relatively large number of constraints is challenging and is not currently implemented in *Hyperopt* [148, 186]. Watanabe and Hutter [186] attempted to address this challenge by integrating the acquisition function of constrained BO by Gardner *et al.* [105]. However, the proposed approach considers the probability of constraint improvement and still allows infeasible samples [105, 186], which may be selected and tested in simulations but can not be queried for real experiments.

B.2 Other methods

B.2.1 Random Search

Samples are randomly selected within the search domain without any encoding for categorical variables or optimization steps.

B.2.2 MISO

Mixed-Integer Surrogate Optimization (MISO) [53] is a technique that targets expensive black-box functions with mixed-integer variables. It constructs a surrogate model using the radial basis function to approximate the unknown function. MISO follows a general procedure of surrogate-based optimization methods outlined in Section 2.2. Additionally, MISO combines different sampling strategies (*e.g.*, coordinate perturbation, random sampling, expected improvement, target value, and surface minimum) and local search to obtain high-accuracy solutions.

B.2.3 NOMAD

NOMAD [56, 119] is a C++ implementation of the Mesh Adaptive Direct search (MADS). It is designed to solve difficult black-box optimization problems. In particular, it can handle nonsmooth, nonlinearly constrained, single or bi-objetive, and mixed variable optimization problems. It handles the categorical variable using the extended poll, which is defined as following [187]:

> The extended poll first calls the user-provided procedure defining the neighborhood of categorical variables. The procedure returns a list of points that are neighbors of the current best point (incumbent) such that categorical variables are changed and the other variables may or may not be changed. These points are called the extended poll points and their dimension may be different than the current best point, for example when a categorical variable indicates the number of continuous variables.

B.2.4 SMAC

Sequential Model-based Algorithm Configuration (SMAC) [54] is a surrogate-based black-box optimization method originally proposed to tackle *algorithm configuration* problems with continuous and categorical variables. Its model is based on random forests [188], so it can handle categorical variables explicitly. SMAC uses empirical mean and variance within a tree ensemble to identify uncertain search space regions and optimizes the acquisition function by combing local and random search.

B.2.5 Genetic

Different genetic algorithm implementations are available [146, 147, 189–195]. In Chapter 6, we compare PWAS with the evolutionary algorithm implemented in the Distributed Evolutionary Algorithm in Python (DEAP) package [146, 147]. DEAP handles categorical variables by label encoding them. Next sample to test is generated through crossover, mutation, or a combination of both, depending on whether the randomly generated probabilities for the execution of crossover or mutation exceed the default threshold. Evolutionary algorithms often balance exploitation and exploration through crossover and mutation, without explicitly utilizing the input-output correlations. Therefore, when the design space is large, it may require a large number of experiments or simulations to attain desired outcomes, making it not suitable for expensive-to-evaluate problems [196]. To address this issue, different surrogate-assisted evolutionary algorithms have been proposed [197, 198]. Nevertheless, surrogate model selection and relevant parameter tuning remain challenging [198]. Moreover, incorporating constraints within the framework can be non-trivial [199].

B.2.6 PSO

Particle Swarm Optimization (PSO) [16] is a population-based technique often used to optimize black-box, nonlinear, and multi-modal functions with continuous variables. It mimics the behavior of a swarm of birds or fish. In PSO, potential solutions are represented as particles that move through a search space. At each iteration, particles adjust their positions based on their own historical best position and the best position found by any particle in the swarm. This information influences their movement toward promising areas of the search space. The algorithm iteratively refines the particle positions until a stopping criterion is met.