Model-assisted robust optimization for continuous black-box problems

Ullah, S.

Citation

Ullah, S. (2023, September 27). *Model-assisted robust optimization for continuous black-box problems*. Retrieved from https://hdl.handle.net/1887/3642009

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3642009

Note: To cite this publication please use the final published version (if applicable).

Model-Assisted Robust Optimization for Continuous Black-Box Problems

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op woensdag 27 september 2023
klokke 16:15 uur

door

Sibghat Ullah
geboren te Dera Ghazi Khan, Pakistan
in 1995

Promotores:
Prof.dr. T.H.W. Bäck
Prof.dr. B. Sendhoff (TU Darmstadt, Germany)

Co-promotor:
Dr. H. Wang

Promotiecommissie:
Prof.dr. Y. Jin (Universität Bielefeld, Germany)
Dr. K. Li (University of Exeter, UK)
Prof.dr. A. Plaat
Prof.dr. M.M. Bonsangue
Dr. A.V. Kononova

Copyright © 2023 Sibghat Ullah All Rights Reserved.

This research has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement number 766186.

Figures and diagrams are generated using Flowchart Maker and matplotlib.

Abstract

Uncertainty and noise are frequently-encountered obstacles in real-world applications of numerical optimization, e.g., mechanics, engineering, economics and finance. Due to various reasons, various types of uncertainties and noise can emerge in optimization problems. These uncertainties and noise can alter the problem landscape, and affect the practical applicability of the optimal solutions found by the algorithms. The practice of optimization that deals with uncertainties and noise is commonly referred to as robust optimization. This thesis concentrates on robust optimization with respect to the parametric uncertainties in the search variables. These parametric uncertainties are assumed to be structurally symmetric, additive in nature, and can be modeled in a deterministic or a probabilistic fashion.

Despite its significance, achieving robustness in real-world applications is quite challenging. One of the major reasons is the computational cost involved to find the robust solution. The computational cost mainly depends on problem landscape, dimensionality, type and structure of the uncertainty, and the robustness formulation or criterion among others. To achieve robustness in an efficient manner, this thesis utilizes surrogate modeling. For this purpose, several attempts are made to implement and apply surrogate modeling in robust optimization. One research stream (Chapter 3) focuses on the fundamental research questions with the help of a "one-shot optimization" strategy based on surrogate modeling. The main questions targeted in this research stream deal with the impact of factors, such as sample size, modeling technique, design of experiments, data pre-processing, structure and scale/severity of the uncertainty, robustness aim/criterion, dimensionality, and problem landscape among others. By and large, this research stream targets the practical applicability of surrogate modeling to find robust solutions and the related difficulties thereof. To be able to answer these questions in a comprehensive manner, two empirical studies are designed.

The key findings from these studies reveal the promising nature of Kriging, Polynomials, and Support Vector Machines to construct a good quality surrogate model based on a reasonable sample size. Moreover, it is found that in the majority of the cases, surrogate modeling yields a reasonably good solution, which is very close to the baseline. Another observation from the empirical results in this context affirms the suitability of Principal Component Analysis and Autoencoders to perform dimensionality reduction in the case of high dimensional problems, albeit with some performance deterioration.

The second research stream (Chapters 4 and 5) targets more advanced research questions, such as the practicality of the Bayesian optimization approach to find robust solutions, as well as how to choose the robustness criterion/merit in practical scenarios. To investigate the applicability of the Bayesian optimization approach in this context, it is extended to account for parametric uncertainties in the search variables. Moreover, the validity of the Bayesian optimization approach to find robust solutions is investigated.

The key findings from this investigation indicate the suitability of the extended Bayesian optimization algorithm to find robust solutions in an efficient manner. Furthermore, it is found that dimensionality, and consequently the computational budget, plays a significant role in determining the performance of our approach. Lastly, we find that the “Expected Improvement” criterion and the “Moment-Generating Function of Improvement” prove to be excellent choices as the sampling infill criterion for robust optimization.

As part of the second research stream (Chapters 5), an attempt is made to answer a crucial yet unanswered question, namely how to select a robustness criterion/merit in practical scenarios with regards to computational efficiency? Another empirical investigation is carried out to answer this question in a comprehensive manner. This empirical investigation computes the running cpu time of the Bayesian optimization algorithm for five of the most common robustness criteria. The key findings from this investigation indicate the promising nature and practical applicability of “mini-max robustness” to find solutions under uncertainty with regards to computational efficiency.

The last part of the thesis (Chapter 6) deals with benchmarking the performance of the surrogate modeling approaches, introduced earlier in the thesis, on a real-world engineering case study. To this end, a case study focusing on the optimization of car hood designs is investigated in detail for both, the “one-shot optimization” strategy and the Bayesian optimization algorithm. The results from this case study indicate the promising nature of Kriging and Ensemble methods, e.g., Random Forest, to effectively model the objective function in practical scenarios. Furthermore, it is found that the “Moment-Generating Function of Improvement” and the “Lower Confidence Bound” are excellent choices for the sampling infill criterion in Bayesian optimization. A short summary of the major contributions in the thesis is provided in Chapter 7, which also encompasses the list of major challenges and opportunities pertaining to robust optimization.

Contents

1 Introduction 1

1.1 Robust Optimization 3

1.2 Organization and Contributions 5

2 Background 9

2.1 Black-Box Optimization 9

2.2 Robust Optimization 12

2.2.1 Uncertainties and Noise in Black-Box Optimization 13

2.2.2 Sources of Uncertainty and Noise 14

2.2.3 Modeling Uncertainty and Noise 17

2.2.4 Cases of Uncertainty and Noise 19

2.2.5 Scope of Robust Optimization 20

2.3 Surrogate Modeling 22

2.3.1 Introduction 23

2.3.2 Response Surface Models 24

2.3.3 Kriging 25

2.3.4 Surrogate Modeling in Practice 26

2.4 Summary and Discussion 29

3 Surrogate-Assisted Robust Optimization 32

3.1 Robust Optimization via Surrogate Modeling 33

3.1.1 Robust Counterpart Approach 37

3.1.2 Design of Experiment 40

3.1.3 Preparing Data and Choosing a Modeling Approach 43

3.1.4 Appraising the Surrogate Model 45

3.1.5 Hyper-parameter Optimization 47

3.1.6 Empirical Investigation 47

3.1.7 Results 51
3.2 The “Curse of Dimensionality” 59
3.2.1 Principal Component Analysis 61
3.2.2 Kernel Principal Component Analysis 62
3.2.3 Autoencoders 63
3.2.4 Variational Autoencoders 64
3.2.5 Dimensionality Reduction 66
3.2.6 Empirical Investigation 66
3.2.7 Experimental Setup 67
3.2.8 Results 71
3.3 Summary and Discussion 76

4 Robust Bayesian Optimization **79**
4.1 Bayesian Optimization 80
4.1.1 Sampling Infill Criteria 81
4.2 Robustness in Bayesian Optimization 84
4.2.1 Robust Infill Criteria 86
4.3 Empirical Investigation 88
4.3.1 Results 91
4.4 Summary and Discussion 95

5 Computational Cost of Robustness **100**
5.1 Cost of Robustness 101
5.2 Empirical Investigation 102
5.2.1 Results 104
5.2.2 Analysis 108
5.3 Summary and Discussion 109

6 Engineering Applications **112**
6.1 Car Hood Design 113
6.1.1 Data Set 114
6.1.2 Data Wrangling 114
6.1.3 Tasks 117
6.2 One-shot Optimization 118
6.2.1 Experimental Setup 118
6.2.2 Results 120
6.3 Bayesian Optimization 123

6.3.1 Experimental Setup 124
6.3.2 Results 126
6.4 Summary and Discussion 129

7 Conclusion and Outlook **132**
7.1 Challenges and Opportunities 134

Bibliography 138

Index 154

English Summary 158

Nederlandse Samenvatting 163

Acknowledgements 167

About the Author 169

CHAPTER

1

Introduction

Solving a real-world optimization problem entails dealing with uncertainties and noise within the system, or a model of the system, for which optima are sought. Due to various reasons, various types of uncertainties and noise can emerge in optimization problems. These uncertainties and noise can alter the problem landscape, and affect the practical applicability of the optimal solutions found by the algorithms. Hence, for practical scenarios, optimization methods are needed which can deal with these uncertainties, and solutions have to be found which take into account the impact of the unexpected drifts and changes in the optimization setup. The practice of optimization that accounts for uncertainties and noise is referred to as *robust optimization* (Ben-Tal et al., 2009).

In real-world engineering applications, e.g., automobile manufacturing, building construction, and steel production, finding a robust solution, i.e., a solution whose performance is not greatly affected by the uncertainties in the optimization setup, is crucial due to the potentially serious impact in case of a failure. Despite the significance, however, achieving robustness in modern engineering applications is quite challenging. Some of the most important reasons for that include the variety of problem landscapes, high dimensionality, the type and structure of the uncertainty, and the robustness formulation or criterion among others (Gabrel et al., 2014). In practice, the optimization scenarios in these applications are treated as *black-box problems*, which need to be efficiently solved in the face of uncertainty and noise.

Most of the approaches to efficiently solve black-box problems fall under the category of *direct-search methods* (Lewis et al., 2000), such as *evolutionary algorithms* (EAs) (Bäck and Schwefel, 1993), and *surrogate-assisted optimization* (SAO) (Forrester et al., 2008). This thesis emphasizes on SAO to efficiently solve numerical black-box problems, subject to uncertainty and noise. Note that SAO refers to the utilization of statistical models while solving expensive to evaluate black-box problems. These statistical models are referred to as the *surrogate models* or the *meta-models* (Keane et al., 2008). The basic idea behind SAO is to replace the actual (expensive) function evaluations by the predictions of these statistical models, which is desirable if the optimization problem under consideration is hard to solve directly. The abstraction provided by the surrogate models is useful in multiple situations. For instance, it can simplify the task to a great extent in simulation based modeling and optimization, i.e., where a non-deterministic simulator replaces the actual (physical) system (Sóbester et al., 2014). Surrogate models can also provide practically useful insights about the *search space*, e.g., space visualization and comprehension (Forrester et al., 2008).

It is worthwhile to note that SAO was initially utilized to find the nominal solution of an optimization problem (Schmit Jr and Farshi, 1974; Barthelemy and Haftka, 1993), without taking into account the unexpected drifts and changes in the optimization setup. However, this is problematic for many real-world scenarios, since uncertainty can alter the practical applicability of the optimal solutions. Therefore, a natural question arises on the suitability of SAO to find optimal solutions which are still useful in the face of uncertainty and noise. This thesis focuses on the applicability of SAO in this context. The most important research questions that we address in this thesis are:

1. Is surrogate modeling suitable to find robust solutions efficiently[1]?
2. How can one select the modeling approach and the sampling plan to find robust solutions via surrogate modeling?
3. What is the impact of external factors, such as the noise level – the scale of the uncertainty, the problem landscape, and the dimensionality, on the applicability of surrogate modeling in this context?
4. What is the impact of robustness formulation/criterion in efficiently solving black-box problems subject to uncertainty and noise, and which robustness formulations are recommended to practitioners with regards to computational efficiency?

---

[1] The notion of efficiency is based on the utilization of computational resources, and would be further discussed in Chapter 2.

1.1 Robust Optimization

In the first half of the twentieth century, Sir Ronald A. Fisher made efforts to grow larger crops in the face of varying weather and soil conditions (Fisher, 1936). His work comprised the basic techniques of *design of experiments* (DoE) and *analysis of variance* (ANOVA), which were later enhanced by several statisticians (Plackett and Burman, 1946; Rao, 1946; Cox and Cochran, 1957). The Japanese engineer Taguchi employed similar techniques for quality improvement of industrial products and processes in 1950s and 1960s. Taguchi's work, referred to as *robust design*, was virtually unknown outside Japan until the 1980s when he traveled to the United States and introduced his concept, which became popular afterwards (Taguchi and Phadke, 1989; Parr, 1989).

In Taguchi's framework of robust design, three different types of parameters can be distinguished. The first type of parameters are referred to as the controllable parameters, since they can be chosen or controlled by the designer during the process of optimization. The second category of parameters are known as the noise parameters, which serve as the source of variation in system's performance. Note, however, that, while the variation in these parameters is beyond the designer's control, they can be known or describable in the form of probability density functions. The third category of parameters are known as the system constants. The overall goal of the robust design in Taguchi's methods is to determine the optimal settings of the control parameters, such that the resulting process or product performance is insensitive to the variations originating from the noise parameters (Taguchi, 1995).

This manifestation of robust design was based on classic DoE techniques, where all control variables were altered according to an orthogonal array (Rao, 1946), which was referred to as the *inner array*. At each control variable setting, the noise variables were altered according to a second orthogonal array, which was referred to as the *outer array*. Based on the combinations of the inner and the outer arrays, the response data was used to estimate the process mean and variance. Both of these statistics, namely the mean and the variance, were then combined to give rise to a single quantitative measure, which was referred to as the *signal-to-noise-ratio* (SNR) (Johnson, 2006). SNR was further used to perform a standard ANOVA, and those control variable settings were identified which yielded the most stable performance. Taguchi's work started a process which made aware the importance of parameters variations to engineers and designers. His work has been reviewed, criticized, and enhanced throughout the years (Pignatiello Jr, 1988; Pignatiello Jr and Ramberg, 1991; Goh, 1993).

Different from Taguchi's work, the problem of dealing with uncertainties and noise consisted of a number of variants in *operations research*. Studies that considered uncertainty in the optimization model date back to the work of Dantzig in 1955 (Dantzig, 1955), and Wets in 1966 (Wets, 1966). Today, approaches dealing with uncertainties can be found in various settings in the scope of *mathematical programming*, such as in the form of stochastic programming (Kall et al., 1994), under the term robust optimization (Mulvey et al., 1995; Ben-Tal et al., 2004; Bertsimas et al., 2011), and in the scope of fuzzy programming (Bellman and Zadeh, 1970), which includes the two types of flexible programming (Zimmermann, 1975; Tanaka et al., 1973), and possibilistic programming (Tanaka and Asai, 1984). A survey of different mathematical programming classes in the context of robust optimization is provided by Sahinidis (Sahinidis, 2004).

Within the scope of numerical black-box optimization, and particularly in the field of surrogate modeling, there has been an increasing interest for methods that deal with uncertainty and noise. Earlier work by Jurecka (Jurecka, 2007) focused on the application of surrogate modeling for structural optimization problems, whereas the work of Rehman (Rehman, 2016) emphasized on the application of integrated electronics. Both of these works contributed extending the *Bayesian optimization* approach (Jones et al., 1998) to the robust scenario with a particular focus on computational efficiency. Some of the most important challenges and opportunities highlighted in the literature (Rehman, 2016; Jurecka, 2007; Beyer and Sendhoff, 2007; Kruisselbrink, 2012) form the starting point of this thesis. Our research questions, introduced earlier, are based on these points. These points are summarised in the following.

- Surrogate modeling was initially utilized to find the nominal solution of a black-box problem. Its validity to find a robust solution needs further empirical evidence. In particular, the impact of some of the most important factors, e.g., the type, structure, and the scale of uncertainty, the corresponding robustness formulation, the choice of the modeling technique, the dimensionality, and the problem landscape, should be considered (Jurecka, 2007; Rehman, 2016).

- Regarding the computational tractability of surrogate modeling to find robust solutions, empirical evidence on some of the most important details, e.g., the sample size, and the appropriate computational budget, is lacking (Jurecka, 2007).

- Solving high dimensional black-box problems with surrogate modeling is quite challenging due to the computational complexity involved (Shan and Wang, 2010).

- Bayesian optimization is a global-search strategy designed for expensive to evaluate black-box problems, and has been extended to the robust scenario. Within the scope of *robust Bayesian optimization*, an important contribution in the literature would be to propose, evaluate, and compare the sampling plans to sequentially update the surrogate model in a region of interest based on different robustness formulations/criteria (ur Rehman et al., 2014).

- Finding a robust solution requires additional computational resources as opposed to finding a nominal solution, since the optimizer has to take into account the impact of uncertainty and noise as well. This need for additional computational resources is referred to as the “computational cost of robustness” (CCoR) in this thesis. CCoR depends on the formulation of the robustness, and could be an important factor in efficiently solving the problem. Ranking and evaluating some of the widely applied robustness formulations based on CCoR would be a nice contribution to the literature, as it would help practitioners choose a suitable robustness criterion with regards to computational efficiency.

1.2 Organization and Contributions

The organization of this thesis is as follows. The motivation, research questions, and major contributions of each chapter are briefly introduced, followed by a list of publications resulting from this research.

Chapter 2 provides the technical background and context for robust optimization. Starting with black-box optimization and related material in Section 2.1, we provide a concise overview on uncertainty and noise in Section 2.2. Next, surrogate modeling is defined in Section 2.3.

Chapter 3 deals with the applicability of surrogate modeling to find robust solutions with the help of a “one-shot optimization” strategy. In this chapter, Section 3.1 answers the question regarding the training sample size, modeling techniques, effect of the type and structure of the uncertainty, and quality of the robust solution. In the following section, we discuss how to alleviate the “Curse of Dimensionality” in surrogate modeling. To answer our questions in this chapter, two empirical studies are conducted and presented. The results of these studies are published as:

> Ullah, S., H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2019). An Empirical Comparison of Meta-Modeling Techniques for Robust Design Optimization. In *2019 IEEE Symposium Series on Computational Intelligence* (SSCI), 2019, pp. 819-828.
>
> Ullah, S., D. Anh Nguyen, H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2020). Exploring Dimensionality Reduction Techniques for Efficient Surrogate-Assisted optimization. In *2020 IEEE Symposium Series on Computational Intelligence* (SSCI), 2020, pp. 2965-2974.

Chapter 4 deals with the applicability of the Bayesian optimization algorithm to the robust scenario. In particular, Section 4.1 provides an overview of the existing literature for BO and the so-called infill criteria. Section 4.3 empirically compares the performance of the so-called *Moment-Generating Function of the Improvement*, which is an infill criterion extended to the robust scenario. For the baseline, the so-called *Expected Improvement* criterion is chosen, which has already been extended to find robust solutions. The publication reports the results in this chapter:

> Ullah, S., H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2021). A New Acquisition Function for Robust Bayesian Optimization of Unconstrained Problems. In *2021 Genetic and Evolutionary Computation Conference Companion* (GECCO 21 Companion), New York, NY, USA, pp. 1344-1345.

Chapter 5 provides a novel perspective on the computational cost for achieving robustness. Note that it has been observed in the literature that finding a robust solution is computationally more expensive than finding a nominal solution. The needs for additional computational resources are determined by the robustness formulation/criterion among others. Because of this, Section 5.2 conducts an empirical study which measures the running (cpu) time for some of the widely adopted robustness formulations on a wide range of test scenarios. Based on the findings in this section, these robustness formulations are ranked with respect to each other. These rankings provide a new perspective to practitioners for choosing the robustness formulations in practical scenarios with regards to computational efficiency. The results of this investigation have been published as:

> Ullah, S., H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2022). A Systematic Approach to Analyze the Computational Cost of Robustness in Model-Assisted Robust Optimization. In *Seventeenth International Conference on Parallel Problem Solving from Nature* (PPSN 2022), pp. 63-75.

Chapter 6 focuses on benchmarking the performance of the surrogate modeling techniques, introduced earlier in the thesis, on a real-world engineering case study. To this end, a case study focusing on the optimization of car hood designs is investigated for both the “one-shot optimization” strategy and the Bayesian optimization algorithm. The results from this case study validate some of the earlier findings in the thesis and provide a novel perspective regarding the applicability of surrogate modeling in robust optimization.

Chapter 7 provides the overall summary of the thesis, alongside major challenges and opportunities pertaining to robust optimization.

CHAPTER 2

Background

This chapter provides the necessary background and context for robust optimization. Starting with an overview of black-box optimization and related material in Section 2.1, we delineate some of the most important concepts related to robust optimization in Section 2.2. Section 2.3 provides an overview on surrogate modeling, followed by two of the widely adopted modeling techniques, namely the *response surface models* and *Kriging*. Lastly, we provide a short summary of the chapter in Section 2.4.

2.1 Black-Box Optimization

We start with an abstract system[1], which takes some input $\mathbf{x}$, and produces some output $\mathbf{y}$. The goal of optimization is to find such setting(s) of the input $\mathbf{x}$ accepted by the system, which produce the best possible output $\mathbf{y}$. We refer to such a system as a black-box, since no further information about the system is assumed (Alarie et al., 2021; Conn et al., 2009). This refers to the fact that the internal dynamics and mechanism of the system are unknown to the designer. Such a system can represent a wide range of optimization problems in practice, such as finding the optimal control parameters of an industrial production line.

In the following, we define some of the most important concepts related to the optimization of such a system.

Domain It may also be referred to as the *search space*, and contains the set of all inputs accepted by the system. We denote it with symbol $\mathcal{S}$ throughout this thesis. Examples of some important domains include Discrete spaces (Korte et al., 2011), Hilbert spaces of functions (Balakrishnan, 2012), and Mixed-integer spaces (Belotti et al., 2013). In this thesis, the discussion is always restricted to the search space $\mathcal{S} \subseteq \mathbb{R}^D$ with Euclidean metric, where D denotes the dimensionality. The resulting optimization problems are known as *real parameter* optimization problems. The practice of optimization that deals with real parameter problems is referred to as *continuous optimization* (Wright et al., 1999).

---

[1] The notion of “abstract” refers to the fact that no particularities are assumed on the input, output, and the internal functionality of the system.

Objective Function In optimization, the purpose of the objective function is to assign score to each input based on the quality of the output. In this thesis, we deal with real-valued black-box functions, which means that the objective function represents a black-box system. Furthermore, no additional analytical properties, e.g., continuity, differentiability, and smoothness, are assumed on the objective function, and the only available information about the objective function is taken to be the evaluation of points in its domain (Audet and Hare, 2017).

$$f \,:\, \mathcal{S} \subseteq \mathbb{R}^D \rightarrow \mathbb{R}^M, \tag{2.1}$$

where the domain $\mathcal{S}$ is assumed to be a subset of the D-dimensional Euclidean space, and its image is $\mathbb{R}^M$.

Single-Objective Optimization Problem A *real-valued single-objective optimization problem* is a special problem in continuous optimization which has exactly one objective, i.e., $M = 1$ in Eq. (2.1). Without loss of generality, the optimization of such a problem can be defined as the problem of determining a *global minimum* $\mathbf{x}^*$ as:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{S}} \; f(\mathbf{x}), \tag{2.2}$$

where the definition of global minimizer is provided later in this section.

Multi-Objective Optimization Problem A *real-valued multi-objective optimization problem* is also a special case of an optimization problem with at least two objectives, i.e., $M > 1$ in Eq. (2.1). For this class of optimization problems, the definition of optimality is often based on the notion of Pareto dominance. Note that dominance can be defined by introducing a partial order on the space of objective function values, and can result in weak dominance, strict dominance, or in-comparability.

The majority of the current work in this context emphasizes on obtaining a representative subset of Pareto optimal solutions, which are based on the notion of non-dominance introduced by Edgeworth, and later independently by Vlifredo Pareto (Pareto et al., 1971). Within the scope of Pareto methods, nature inspired heuristics have been successfully integrated, resulting in famous algorithms, such as Non-dominated Sorting Genetic Algorithm-II (NSGA-II) (Deb et al., 2002), Strength Pareto Evolutionary Algorithm2 (SPEA2) (Kim et al., 2004), and Multi-objective particle swarm optimization (MOPSO) (Coello and Lechuga, 2002), among others. For a technical overview on multi-objective optimization, please refer to the work of Emmerich and Deutz (Emmerich and Deutz, 2018).

Definition 2.1 (Black-Box Optimization). An objective function f as defined in Eq. (2.1) is called a black-box if no prior knowledge about f is available, and the only accessible information about f is the objective value: $f(\mathbf{x}),\ \forall\ \mathbf{x} \in \mathcal{S}$. The optimization of such a function is referred to as *black-box optimization* in this thesis.

Definition 2.2 (Global Minimum). For a given single-objective optimization problem, a candidate solution: $\mathbf{x}^* \in \mathcal{S}$, is said to be a global minimum of f if: $\forall\, \mathbf{x} \in \mathcal{S},\ f(\mathbf{x}^*) \leq f(\mathbf{x})$.

Finding a global minimum is generally a difficult task due to multi-modality, discontinuity, and ill-conditioning. In practice, it is only possible to guarantee the convergence to the *local minimizer* (Wright et al., 1999).

Definition 2.3 (Local Minimum). For a given single-objective optimization problem, a candidate solution: $\mathbf{x} \in \mathcal{S}$, is said to be a local minimum of f if there is a neighborhood $N_{\mathbf{x}}$ of $\mathbf{x}$, such that: $\forall\, \mathbf{x}^{'} \in N_{\mathbf{x}},\ f(\mathbf{x}) \leq f(\mathbf{x}^{'})$.

As the search space is a subset of the metric space $\mathbb{R}^D$, in principle, any metric on $\mathbb{R}^D$ can be used to define the neighborhood. For instance, in the case of Euclidean metric, the neighborhood can be defined as a subset of the search space $\mathcal{S}$, which contains an open Euclidean ball around $\mathbf{x}$ as: $B_\epsilon(\mathbf{x}) = \{\mathbf{x}^{'} \in \mathcal{S} : ||\mathbf{x} - \mathbf{x}^{'}|| < \epsilon\}$, for any $\epsilon > 0$.

Definition 2.4 (Constrained Optimization Problems). When solving a black-box optimization problem in practice, we might encounter a set of *constraint functions*: $\mathcal{G} = \{g_1, \ldots, g_p\},\ p \in \mathbb{N}_0$. A real-valued black-box optimization problem with a set of constraint functions $\mathcal{G}$ is referred to as a *constrained optimization problem.*

Note that the set of constraint functions can include two types of constraints, namely inequality and equality constraints. In this thesis, we only present the inequality constraints for convenience since equality constraints can be easily transformed into inequality ones. In the presence of inequality constraints, the global minimizer $\mathbf{x}^*$ must belong to the set of feasible solutions $\mathcal{A}$, which can be defined as:

$$\mathcal{A} = \{\forall\, \mathbf{x} \in \mathcal{S} \mid g_i(\mathbf{x}) \geq 0\ ,\ i = 1, \ldots, p\}. \tag{2.3}$$

Definition 2.5 (Practical Goal of Optimization). Given a black-box optimization problem with an optimization goal and a finite amount of computational resources, the practical goal of optimization is to use these resources in an optimal way to find as good a solution as possible.

In an alternative sense, one can also aim for finding solution(s) that are an improvement with respect to the previously known best solution(s). Based on these reasons, one can also define a *global optimization algorithm* as the algorithm, that, given an infinite amount of computational resources, would get arbitrarily close to the global optimum.

2.2 Robust Optimization

The traditional view on black-box optimization as presented in the previous section does not account for the unexpected drifts and changes in the optimization setup. However, this is unrealistic for many real-world optimization scenarios. For instance, often in engineering applications, a non-deterministic simulator replaces the actual (physical) system (Bhosekar and Ierapetritou, 2018). The goal of optimization in these scenarios is to find solutions, such that the real-world realizations of these solutions are also of a good quality, even if they are subject to perturbations (Kruisselbrink, 2012).

It is intuitive to believe that we can face several types of uncertainties and noise in real-world applications (Beyer and Sendhoff, 2007). For instance, we can think of the uncertainty because an approximate model substitutes the actual (physical) system. Furthermore, the model itself may be non-deterministic/stochastic in nature. We can also think of the uncertainty in this situation because the real-world manufacturing of different parts of the system is precise only to a certain degree. Therefore, when these parts are assembled together, the system may not perform as expected. Taking these observations into consideration, we are faced with three issues as:

- How can uncertainties and noise arise in black-box optimization?
- In what ways can we classify such uncertainties based on their common characteristics?
- How can we mitigate the effects of such uncertainties in practical scenarios?

In the following, we summarize the state-of-the-art to answer these questions.

2.2.1 Uncertainties and Noise in Black-Box Optimization

Uncertainties and noise comprise one of the most challenging areas in black-box optimization. They are encountered frequently in real-world optimization problems (Jurecka, 2007). Below, we provide a few reasons why they can appear in practical scenarios.

- The search variables, also referred to as the decision variables, can not be controlled with unlimited precision in reality, e.g., manufacturing tolerances.
- The operational or environmental conditions for an industrial product or process can only be known to a certain extent.
- The output of the (physical) system, or a model of the system, is intrinsically stochastic.
- An approximate model may replace the real-world system within the optimization loop.
- The objective and the constraint functions can be fuzzy in nature, e.g., a degree of vagueness on the objective and constraint functions might exist.

Because of these reasons, we can establish that uncertainties and noise surround the black-box system in practical scenarios, since they can emerge in the input and output of the system, in addition to the modeling and evaluation of the system (Beyer and Sendhoff, 2007). Therefore, it is intuitive to believe that the common assumptions for solving real-world black-box problems can be significantly compromised in the face of uncertainty and noise. But in order to effectively account for these uncertainties and noise, a nomenclature is needed. To this end, we follow the categorization of Beyer and Sendhoff to a large degree (Beyer and Sendhoff, 2007), in combination with the work of Kruisselbrink (Kruisselbrink, 2012).

2.2.2 Sources of Uncertainty and Noise

Here, we first categorize uncertainty and noise by looking at their origin within the general loop of black-box optimization. For this purpose, an elaborated version of the black-box optimization loop is provided in Fig. 2.1, which highlights the potential sources of uncertainty and noise. In this figure, an optimizer[1] is coupled to the system, or a model of the system, for which optima are sought. The optimizer generates some candidate solution(s), which is/are fed to the system. The system evaluates this/these solution(s), and provides a quality score of this/these candidate solution(s). Based on this feedback, the optimizer generates a new set of candidate solution(s), which is/are fed to the system again for evaluation (Pošík et al., 2012). This loop is repeated until either a satisfactory solution is found, or a predefined computational budget, or other termination criterion is reached (Audet and Hare, 2017).

In Fig. 2.1, we can identify five regions of interest where uncertainty or noise can arise and affect the black-box optimization loop.

(I) Uncertainties and/or noise in the search/decision variables, denoted as $\mathbf{x}$.

(II) Uncertainties and/or noise in the environmental or operating conditions (generally referred to as the environmental variables, and denoted as α).

(III) Uncertainties and/or noise in the evaluation(s) of the candidate solution(s), denoted as $\mathbf{y}$.

(IV) Vagueness when modeling the constraints.

---

[1] The notion of “optimizer” is used to refer to a particular solution, as well as an optimization algorithm, in this thesis.

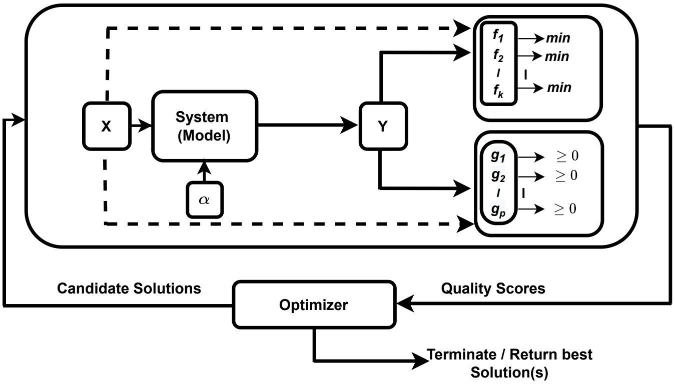


Figure 2.1: The general black-box optimization loop with five different sources of uncertainty. These sources include the decision variables $\mathbf{x}$, the environmental variables α), the evaluation of the system $\mathbf{y}$, the objectives f_i, and the constraint functions g_i.

(V) Preference uncertainty in the objectives, if the optimization problem has more than one objective.

The effect of these sources of uncertainty is presented in several different ways in black-box optimization.

(I) **Uncertainties and/or noise in the decision variables**

This type of uncertainty arises in practical scenarios because the real-world realizations of the candidate solutions differ arbitrarily much from their nominal values, which are used to find the optimal solutions. For instance, in the area of product engineering, we might encounter this uncertainty due to manufacturing tolerances, i.e., realizing a candidate solution to its nominal value might be too costly, and may not make an economic sense (Beyer and Sendhoff, 2007). With this type of uncertainty, we can usually face either of the following two scenarios.

Scenario 1:

An approximate model replaces the actual (physical) system within the black-box optimization loop (presented in Fig. 2.1). Thus, although the model might accept inputs with unlimited precision, the real-world (physical) system can only realize these inputs to a certain degree, e.g., in automobile design optimization, physical parts of the vehicle can only be manufactured with a limited precision (Chowdhury and Taguchi, 2016).

Scenario 2:

In case the real-world (physical) system is enclosed within the black-box optimization loop, the uncertainty in the inputs can propagate through the output, to the set of objective and constraint functions. Generally, the motif of the uncertainty in the inputs is unknown in advance. Hence, one can only make very general assumptions on the structure of the uncertainty (Rehman, 2016), e.g., additive vs multiplicative, deterministic vs stochastic, symmetric vs non-symmetric (Averbakh and Zhao, 2008).

The effect of the additive uncertainty $\Delta_{\mathbf{x}}$, in the search variables, can be represented by reformulating the objective function as:

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}, \Delta_{\mathbf{x}}) = f(\mathbf{x} + \Delta_{\mathbf{x}}), \tag{2.4}$$

For constraint functions, similar formulation can be adopted:

$$\tilde{g}_j(\mathbf{x}) = g_j(\mathbf{x}, \Delta_{\mathbf{x}}) = g_j(\mathbf{x} + \Delta_{\mathbf{x}}),\ j = \{1, \ldots, p\}. \tag{2.5}$$

In Eqs. (2.4)-(2.5), we are not making any assumption on the way in which the uncertainty $\Delta_{\mathbf{x}}$ is mathematically modeled, which would be the topic of interest later in this chapter.

(II) **Uncertainties and/or noise in the environmental variables**

In design optimization, the uncontrollable environmental variables are generally assumed to be system constants. In practical scenarios, however, it is found that they fluctuate, and can affect the performance of anotherwise stable system (Beyer and Sendhoff, 2007). As such, it is useful to think that these fluctuations can also affect the objective and constraint functions, similar to the uncertainty in the search variables (Jin and Branke, 2005).

(III) **Uncertainties and/or noise in the output**

This class of uncertainty is formed in the evaluation of the candidate solutions. Here, we can distinguish between two different scenarios.

Scenario 1:

The system is inherently non-deterministic in nature. Therefore, the precise evaluation of the candidate solutions is impossible, and the resulting output is noisy and stochastic (Nissen and Propach, 1998).

Scenario 2:

The system produces a deterministic output. However, this output can not be realized in practice, e.g., due to the manufacturing imprecisions and tolerances, or similar issues.

(IV) **Uncertainty in the constraints**

Another class of uncertainty is the ambiguity and the vagueness when mathematically formulating the set of constraint functions. This is due to the fact that there are several different types of constraints, e.g., soft vs hard constraints, and probabilistic vs deterministic. The requirements for the satisfaction of these constraints can therefore be represented in various different ways in black-box optimization (Shahraki and Noorossana, 2014).

(V) **Preference uncertainty in the objectives**

Intrinsically, when dealing with a black-box optimization problem with multiple conflicting objectives, a source of uncertainty lies in the trade-off of the objective functions. This is due to the fact that the quality of the candidate solutions can only be known a posteriori. Hence, regarding the importance and trade-off of different objective functions, highly subjective decisions have to be made. This type of uncertainty deals with multi-objective optimization, and can be compensated for by introducing a partial order on the space of objective function values, such as Pareto dominance (Kruisselbrink, 2012).

2.2.3 Modeling Uncertainty and Noise

To properly account for the uncertainties and noise in black-box optimization, we have to describe the mathematical ways in which they can be modeled. However, for that, we first have to make an important, albeit an informal distinction between uncertainty and noise.

Uncertainty vs Noise In the existing literature on robust optimization, there does not appear to be a clear distinction between uncertainty and noise[1]. Here, we provide an informal distinction between both concepts, which are also related to *aleatory* and *epistemic* uncertainties respectively. The most basic understanding of aleatory uncertainty is that it is fundamentally irreducible, completely random, and almost certainly unavoidable (Der Kiureghian and Ditlevsen, 2009). This type of uncertainty can informally be thought of as the *additive noise* in the context of black-box design optimization (Beyer and Sendhoff, 2007).

Epistemic uncertainty on the other hand, is, in principle, due to the lack of understanding, knowledge, or information on the optimization problem (Der Kiureghian and Ditlevsen, 2009). The effect of this kind of uncertainty, can, therefore, be minimized by representing the optimization problem in another way, and/or with more data. Epistemic uncertainty can also be referred to as just the uncertainty, albeit in an informal setting. Similar view is also adopted by Cornell (Paté-Cornell, 1996). For a thorough discussion on the differences between uncertainty and noise, please refer to the work of Kruisselbrink (Kruisselbrink, 2012).

In the following, we review different ways of mathematically representing the uncertainty.

1. **Deterministic**

   One of the most important ways to mathematically describe the uncertainty is with the help of deterministic *crisp sets*, which describe the crisp possibility of the states of the uncertain variables (Ionescu-Bujor and Cacuci, 2004). Here, an uncertain variable is usually modeled as a pair: (A, m_A), where A serves as the crisp set, and m_A describes the membership function. Note that the membership function is usually of the form: $m_A : A \rightarrow \{0, 1\}$. A particular design $\mathbf{x} \in A$ can then take one of the two forms:

   - $\mathbf{x}$ is a member of the set A, if $m_A(\mathbf{x}) = 1$.
   - $\mathbf{x}$ is not a member of the set A, if $m_A(\mathbf{x}) = 0$.

   Note that crisp sets may also be referred to as the *classical set* or *full membership sets* in the literature (Ben-Tal et al., 2009; Beyer and Sendhoff, 2007).

---

[1] In this thesis, we use both terms interchangeably, which refer to the unexpected drifts and changes in the optimization setup.

2. **Probabilistic**

   In a probabilistic setting, an uncertain variable is assumed to be of a stochastic nature. A probabilistic measure can be established by measuring the probabilistic frequency of the events that may occur. Uncertainties of this type can be represented by the *probability (density) functions*. This refers to the fact that a function: $p : A \rightarrow \mathbb{R}_0$ maps every event $\mathbf{x} \in A$, to a probability value, which quantifies the likeliness of that event (Kruisselbrink, 2012).

3. **Possibilistic**

   In this setting, the uncertainty is formulated with fuzzy statements, which describe the possibility (or degree of membership) about the states of the uncertain variables of interest. As opposed to crisp sets in the deterministic setting, here we make use of the *fuzzy sets*. An uncertain variable is modeled as a pair: (A, m_A), where A serves as the fuzzy set, and m_A describes the membership function (Kruisselbrink, 2012). Note that the membership function in this setting is usually of the form: $m_A : A \rightarrow [0, 1]$. Thus, the degree of membership is a real value between 0 and 1. The degree of membership increase as we get close to 1 (Bagheri et al., 2016).

2.2.4 Cases of Uncertainty and Noise

So far, we have seen five major classes of uncertainty in black-box optimization (based on their origins), along side three common ways to mathematically represent them. This gives rise to a total of 15 scenarios in which we can encounter uncertainty in practical situations. It is intuitive that not all of these scenarios are equally important. Therefore, a question arises as to which of these scenarios should be given more consideration over the others for achieving robustness? Answering this question will limit the scope of this thesis to a well-defined class of uncertainty, which will then be the focus for the rest of the thesis. In Table 2.1, we provide a summary of different classes of uncertainty based on their conceptual distinction, mathematical representation, and common characteristics.

It is pertinent to note that the first two classes of uncertainty discussed in this section, namely the Class (I) and (II), are related to the so-called "sensitivity

Table 2.1: A summary of different categorizations of uncertainty in black-box optimization as described by Kruisselbrink (Kruisselbrink, 2012).

Conceptual Classification	**Modeling**	**Characteristics**
Epistemic (uncertainty)	Possibilistic	Domain unknown Probabilities unknown
	Deterministic	Domain known Probabilities unknown
Aleatory (noise)	Probabilistic	Domain known Probabilities known

robustness[1]" (Beyer and Sendhoff, 2007). Furthermore, Class (IV) and (V) are related to each other in that they do not directly affect the output of the black-box system. Instead, they influence the search space $\mathcal{S}$, and the set of feasible solutions $\mathcal{A}$. Uncertainty of Class (IV) is also related to another important concept – the so-called "reliability-based robustness" (Shahraki and Noorossana, 2014).

2.2.5 Scope of Robust Optimization

Given five different classes of uncertainty in black-box optimization, alongside three mathematical approaches to model them, one can identify several different scenarios of robust optimization as presented in Table 2.2. For the scope of robust optimization in this thesis, however, we limit ourselves to a few of these scenarios. This is due to the fact that not all of these cases are considered to belong to robust optimization in the literature. For instance, Bertsimas (Bertsimas et al., 2010, 2011) only considers the uncertainty in the decision variables to define robust optimization.

In this work, we only consider the first two types of uncertainties – Class (I) and (II) – to represent robust optimization. This refers to the fact that we only deal with sensitivity robustness – robustness which is associated with the sensitivity of the objective function with respect to the specific changes in the decision and environmental variables. The most important reasons for limiting the scope of this work to only the first two types of uncertainties.

---

[1] "Sensitivity robustness" refers to the sensitivity of the objective function with respect to the specific changes in the optimization setup.

Table 2.2: A summary of different cases of uncertainty and noise in black-box optimization as described by Kruisselbrink (Kruisselbrink, 2012). Bold types of modeling and algorithmic approaches are more common in the literature.

Class	**Modeling**	**Major Approaches**
Class (I)	(1) **Deterministic**	(1) **Evolutionary Algorithms**
	(2) **Probabilistic**	(2) Surrogate Modeling
	(3) Possibilistic	(3) Quasi-Newton Methods
Class (II)	(1) **Deterministic**	(1) Evolutionary Algorithms
	(2) **Probabilistic**	(2) Surrogate Modeling
	(3) Possibilistic	(3) Mathematical Programming
Class (III)	(1) Deterministic	(1) **Evolutionary Algorithms**
	(2) **Probabilistic**	(2) Surrogate Modeling
	(3) **Possibilistic**	(3) **Mathematical Programming**
Class (IV)	(1) Possibilistic	(1) **Fuzzy Logic**
	(2) Probabilistic	(2) **Monte-Carlo Methods**
Class (V)	(1) Deterministic	(1) **Evolutionary Algorithms**
	(2) Possibilistic	(2) Surrogate Modeling

- As indicated earlier, uncertainties of Class (IV) and (V) do not directly affect the candidate solutions. Instead, they affect the search space $\mathcal{S}$ and its image $\mathbb{R}^M$ when formulating the optimization problem. For this reason, they have been considered separately from robust optimization in the literature (Jurecka, 2007).

- Uncertainties of type (I) and (II) are most frequent in design optimization, and can also determine the practical applicability of the optimal solutions to a large degree (Rehman, 2016). Accounting for these types of uncertainties is therefore critical (Jurecka, 2007; Kruisselbrink, 2012).

- Accounting for the uncertainty of type (III) refers to optimizing a noisy objective function, instead of finding robust optima. This formulates another scenario of optimization under uncertainty. Although important, this scenario is often considered separately from robust optimization, where the focus is to find optimal solutions, which are practically applicable despite varying conditions. In his work, Kruisselbrink (Kruisselbrink, 2012) also considers this type of uncertainty into robust optimization, but treats it differently from the first two types.

Based on the information provided so far, we can now extend the practical goal of optimization in Definition 2.5 to formulate the general goal of robust black-box optimization.

Definition 2.6 (Practical Goal of Robust Black-Box Optimization). Given a black-box optimization problem with uncertainty and/or noise in the decision and environmental variables, alongside an optimization goal, and a limited amount of computational resources. the *practical goal of robust black-Box optimization* is to use these resources to find as good as possible solutions despite uncertainty and/or noise, which are also optimal and useful in the face of uncertainties/noise.

In the remainder of this thesis, we will only deal with single-objective numerical optimization problems, which are subject to uncertainty and noise in the decision/search variables. We will further assume that the uncertainty/noise is additive and structurally symmetric in nature, and can only be represented in a deterministic, or a probabilistic fashion.

2.3 Surrogate Modeling

Continuous optimization problems in real-world application domains, e.g., mechanics, engineering, economics and finance, can encompass some of the most complicated optimization setups. Principal obstacles in solving the optimization tasks in these areas involve multi-modality (Beasley et al., 1993), high dimensionality (Shan and Wang, 2010), and unexpected drifts and changes in the optimization setup (Kruisselbrink, 2012; Beyer and Sendhoff, 2007). Due to these obstacles and the black-box assumption on the optimization setup, traditional numerical optimization schemes, e.g., gradient descent and Newton methods, are rendered inapplicable. The majority of the optimization schemes applied in these areas now focus on utilizing direct-search methods (Lewis et al., 2000; Beyer and Sendhoff, 2007), in particular EAs (Bäck et al., 2018), and SAO (Keane et al., 2008).

The use of direct-search methods, also referred to as derivative-free methods (Audet and Hare, 2017), in numerical black-box optimization, can be attributed to the following reasons.

- Direct-search methods perform well in practice, since many of them are based on sound heuristics. Recent analysis demonstrates the global convergence behavior for some of these methods, similar to the results known for the globalized quasi-Newton methods (Lewis et al., 2000).

- Some of the most important characteristics of direct-search methods, e.g., no evaluation of the derivative, dictate the practical applicability of these methods, where more sophisticated techniques fail to perform (Audet and Kokkolaras, 2016).

Within the scope of direct search methods, EAs and SAO formulate two of the most important classes of techniques to solve non-linear black-box problems. This thesis deals with SAO since we further assume that the black-box problem is expensive to evaluate.

In the following, we provide a brief introduction to SAO.

2.3.1 Introduction

Surrogate-Assisted Optimisation (SAO) refers to solving the optimisation problem with the help of a surrogate model, also referred to as the meta-model, which replaces the actual function evaluations by the model prediction (Keane et al., 2008). The surrogate model estimates the true values of the objective function under consideration. This is desirable if the objective function is too costly to evaluate. The abstraction provided by the surrogate model is useful in a variety of situations. For instance, it simplifies the task to a great extent in simulation-based modeling and optimisation, by providing the opportunity to evaluate the objective function indirectly if the exact computation is intractable. Surrogate models can also provide practically useful insights, e.g., space visualization and comprehension, about the search space (Forrester et al., 2008).

The idea of SAO was proposed as early as 1974 (Schmit Jr and Farshi, 1974). This line of research was particularly useful in structural optimization with the name of *response surface approximation*. Some of the most important contributions, concerning the applicability of SAO for structural optimization, include the initial work of Svanberg (Svanberg, 1987), Toropov (Toropov et al., 1993), Roux (Roux et al., 1998), Box (Box and Draper, 1987), and Myers (Myers et al., 2016). The first attempt to classify such methodologies, based on their accuracy, in the context of structural engineering, was made in 1993 (Barthelemy and Haftka, 1993).

In the context of optimization under uncertainty, surrogate models were investigated by Rehman (Rehman, 2016), Jurecka (Jurecka, 2007), Jin (Jin et al., 2003), and Persson (Persson and Ölvander, 2013). For a detailed review of surrogate modeling and its applications in structural engineering, please refer to the work of Jurecka (Jurecka, 2007).

In the following, we provide a brief overview for *response surface models* and *Kriging* models, two of the widely utilized modeling techniques.

2.3.2 Response Surface Models

The term response surface models (RSM) can be somewhat misleading, since all types of surrogate models construe a "surface", which enables the designer to estimate the function response at untried locations. For this reason, the term RSM has also been used as a synonym for surrogate models in the literature. A different understanding of the term, however, points to the "polynomial regression models" (Bishop, 2007), which were initially utilized for the analysis of physical experiments (Santner et al., 2003; Jurecka, 2007).

The basic idea behind RSM is to establish an explicit functional relationship (response surface) between the input and the output variables (Hastie et al., 2009). We start with the design data of N vectors of co-variate values $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}^\top$, where each one of these vectors denotes a sample point in the search space $\mathcal{S} \subseteq \mathbb{R}^D$. The corresponding response values of the function f are denoted as $\mathbf{y} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)]^\top$. Then, a polynomial approximation of the function f, of degree M, at an untried location $\mathbf{x}$, can be written as:

$$\hat{f}(\mathbf{x}, M, \boldsymbol{\gamma}) = \gamma_0 + \gamma_1 \mathbf{x} + \gamma_2 \mathbf{x}^2 + \dots + \gamma_M \mathbf{x}^M = \sum_{j=0}^{M} \gamma_j \mathbf{x}^j, \tag{2.6}$$

where the free parameters: $\boldsymbol{\gamma} = \{\gamma_0, \gamma_1, \dots, \gamma_M\}^\top$ can be estimated through the maximum likelihood principle as: $\boldsymbol{\Phi}\boldsymbol{\gamma} = \mathbf{y}$, and $\boldsymbol{\Phi}$ is the Vandermonde matrix (Kalman, 1984) defined as:

$$\boldsymbol{\Phi} = \begin{pmatrix} 1 & \mathbf{x}_1 & \mathbf{x}_1^2 & \cdots & \mathbf{x}_1^M \\ 1 & \mathbf{x}_2 & \mathbf{x}_2^2 & \cdots & \mathbf{x}_2^M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{x}_N & \mathbf{x}_N^2 & \cdots & \mathbf{x}_N^M \end{pmatrix}. \tag{2.7}$$

The maximum likelihood estimate of $\boldsymbol{\gamma}$ can be proven to be:

$$\boldsymbol{\gamma} = \boldsymbol{\Phi}^+ \mathbf{y}, \tag{2.8}$$

where $\mathbf{\Phi}^{+} = (\mathbf{\Phi}^{\top}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\top}$, is the Moore-Penrose pseudo-inverse (Golub and Van Loan, 2013) of $\mathbf{\Phi}$. Using Eq. (2.8), we can estimate the value of the free parameters.

Note that the polynomial approximation $\hat{f}$ of the objective function f, based on M degrees is essentially, a Taylor series expansion of f truncated after $M + 1$ terms. From this observation, it follows that a greater value of M will yield a better approximation. However, with greater number of terms, the approximation also becomes too flexible, and there might be over-fitting and poor generalization capability. We can prevent this phenomenon by restricting the value of M to be small (Forrester et al., 2008; Bishop, 2007; Hastie et al., 2009).

One of the ways to do it is through cross-validation, which can determine the optimal setting of M for a given problem. We can also prevent over-fitting with the help of regularization, such as Lasso and Ridge regularization (Hastie et al., 2009; Bishop, 2007). Throughout this thesis, we utilize a RSM with degree $M = 2$, combined with the so-called *elastic-net penalty*, which linearly combines the Lasso and Ridge regularization terms. It is also pertinent to mention that a higher value of M will result in a more (computationally) expensive approximation of f, since computing the inverse of $\mathbf{\Phi}$ in Eq. (2.7) will become much costlier.

2.3.3 Kriging

Kriging is an interpolation technique based on geostatistics (Woodard, 2000; Rasmussen and Williams, 2006), and has been widely utilized as a surrogate modeling tool in Design and Analysis of Computer Experiments (Sacks et al., 1989; Santner et al., 2003), Surrogate-Assisted Evolutionary Algorithms (Emmerich, 2005), Global Optimization (Jones et al., 1998), and Algorithm Configuration (Hutter et al., 2011). Similar to RSM, we start with the data of N vectors of co-variate values as: $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}^{\top}$, and the corresponding functions responses as: $\mathbf{y} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^{\top}$. Kriging formulates that the function response at any untried search point $\mathbf{x}$ can be described as a normally distributed random variable $Y(\mathbf{x})$ with mean μ and variance σ^2. Furthermore, for any pair $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$, the correlation between $f(\mathbf{x})$ and $f(\mathbf{x}')$ is modeled by a kernel function (Rasmussen and Williams, 2006). Here, we describe the popular Matérn 3/2 kernel:

$$k(\mathbf{x}, \mathbf{x}') = \left(1 + \sqrt{3}l\right) e^{-\sqrt{3}l},\ l = \sqrt{\sum_{i=1}^{D} \theta_j (\mathbf{x}_j - \mathbf{x}'_j)^2}, \tag{2.9}$$

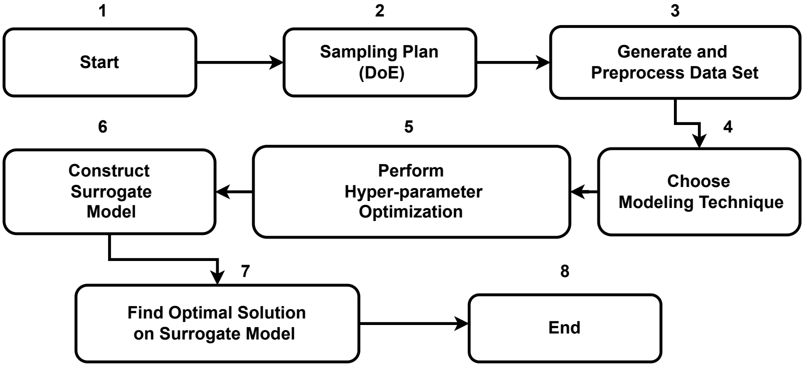

1
Start
2
Sampling Plan (DoE)
3
Generate and Preprocess Data Set
4
Choose Modeling Technique
5
Perform Hyper-parameter Optimization
6
Construct Surrogate Model
7
Find Optimal Solution on Surrogate Model
8
End

Figure 2.2: Flowchart showing the implementation of a "one-shot optimization" strategy in this thesis. The "one-shot optimization" strategy is based on surrogate modeling, to find the optimal solutions in an efficient manner.

where D represents the dimensionality of the problem, and θ_j measures the influence of the j-th dimension with respect to the search domain. Then, the Kriging prediction of the function response at any untried point $\mathbf{x}$ can be shown to be:

$$\hat{f}(\mathbf{x}) = \hat{\mu} + \mathbf{c}^\top \mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}), \tag{2.10}$$

where $\mathbf{c}$ is the vector of correlations between $\mathbf{x}$ and each of the N sample points, $\hat{\mu}$ is the generalized least square estimator of μ, $\mathbf{\Sigma}$ is the $N \times N$ correlation matrix between N sample points with elements defined by Eq. (2.9), and $\mathbf{1}$ is a vector of 1's.

An estimated mean squared error (MSE) of $\hat{f}$ arises naturally from Kriging's theoretical setup:

$$s^2(\mathbf{x}) := \mathrm{E}\{Y(\mathbf{x}) - \hat{f}(\mathbf{x})\}^2 = \sigma^2 \left[1 - \mathbf{c}^\top \mathbf{\Sigma}^{-1}\mathbf{c} + \frac{1 - \mathbf{1}^\top \mathbf{\Sigma}^{-1}\mathbf{c}}{\mathbf{1}^\top \mathbf{\Sigma}^{-1}\mathbf{1}}\right]. \tag{2.11}$$

The MSE is zero at the sample points since the true response of the function is known at these locations.

2.3.4 Surrogate Modeling in Practice

In this thesis, we implement SAO in two different ways: in the framework of a "one-shot optimization" (OSO) strategy (Ta'asan et al., 1992), and with the help of a "sequential model-based optimization" (SMBO) framework (Jones et al., 1998). As we shall see, the former is more desirable with regards to practicality (Chapter 3), due to the potential difficulties and pitfalls of extending the latter to the robust scenario (ur Rehman et al., 2014). However, we note that the latter is more stringent in nature, due to the fact that it updates the surrogate model in an iterative manner, according to a sampling infill criterion (Jurecka, 2007). The sampling infill criterion encodes the search behavior, i.e., balances the trade-off of exploration and exploitation, and can be utilized to find a globally optimal solution on the model surface (Jones et al., 1998).

The working mechanism of OSO strategy in this thesis is described as follows. We start by generating an initial design data set $\mathcal{D} = (\mathrm{X}, \mathbf{y})$, on the objective function f. The locations $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ can be determined by the DoE methodologies, such as the Latin Hyper-cube Sampling (LHS) scheme (Montgomery, 2017). After this, objective function values $\mathbf{y} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^\top$ are computed on these locations. The next step involves constructing the surrogate model based on the available data set $\mathcal{D}$. Note that before constructing the surrogate model, we perform Hyper-parameter Optimization (HPO) to estimate the best configuration of the corresponding hyper-parameters, in order to achieve the best quality surrogate model based on the available function evaluations (Hutter et al., 2009). Once the surrogate model is constructed, we utilize a benchmark numerical optimization algorithm (Wright et al., 1999) to find the optimal solution on the model surface, and the process comes to a halt (Ullah et al., 2019). Since we do not perform an adaptive sampling in this case, the sampling infill criterion does not need to be extended to care for robustness. This allows us to be much more thorough and comprehensive in our approach, as we can take into account the variability in external factors without involving the prohibitively high computational costs (Bossek et al., 2019).

Similar to the previous case, in the SMBO approach, we also construct the initial design data set $\mathcal{D} = (\mathrm{X}, \mathbf{y})$ (Jurecka, 2007). After this, we construct the surrogate model based on the available data set. Following this, the next query point $\mathbf{x}_{\text{new}}$ (to sample the function) is determined with the help of a sampling infill criterion, such as the "Expected Improvement" criterion (Jones et al., 1998). The function response $f(\mathbf{x}_{\text{new}})$ is computed at this location, and the data set $\mathcal{D}$ is extended by appending the pair $(\mathbf{x}_{\text{new}}, f(\mathbf{x}_{\text{new}}))$ to it. The surrogate model is then updated based on the extended data set (Močkus, 2012). This process is repeated until either a satisfactory solution is obtained, or a predetermined computational budget,

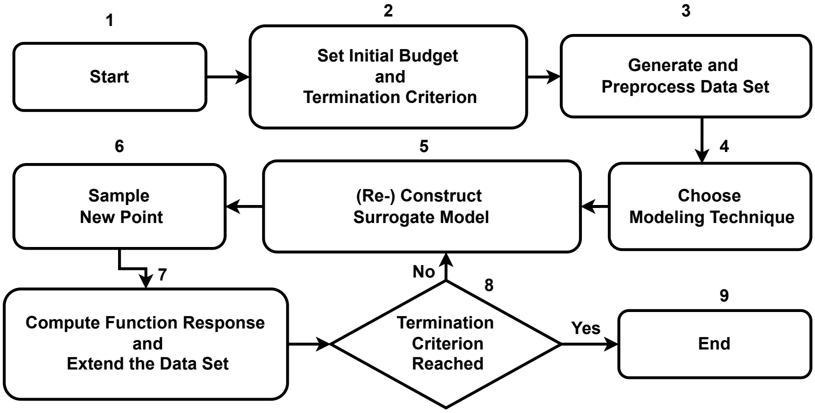

1
Start
2
Set Initial Budget and Termination Criterion
3
Generate and Preprocess Data Set
4
Choose Modeling Technique
5
(Re-) Construct Surrogate Model
6
Sample New Point
7
Compute Function Response and Extend the Data Set
8
Termination Criterion Reached
No
Yes
9
End

Figure 2.3: Flowchart shows the implementation of "sequential model-based optimization" framework in this thesis. The "sequential model-based optimization" framework updates the surrogate model based on a sampling infill criterion, which encodes the search behavior to find the optimal solution on the model surface.

or another termination criterion is reached. Since at each iteration, the surrogate model is updated according to an infill criterion, the optimal solution can be obtained in an efficient manner (Jones et al., 1998). While the SMBO approach is deemed a powerful heuristic to find a globally optimal solution on the model surface, extending it to the robust scenario is a much more difficult task. The potential difficulties and pitfalls of extending the SMBO approach to the robust scenario are explained in detail in Chapter 4.

In this thesis, we attempt to answer some research questions with the help of a OSO strategy (Chapter 3). These research questions target the potential of surrogate modeling to find robust solutions, and the related difficulties thereof. The potential of surrogate modeling in this context is evaluated by varying sample size, modeling technique, problem landscape, dimensionality, robustness formulation, and noise level among others. The OSO strategy is readily applicable to answer these questions in an empirical fashion (Ullah et al., 2019). However, we also assume that SMBO is a a powerful heuristic, and it is possible to answer more advanced research questions with it (Ullah et al., 2021). These research questions deal with the impact of the sampling infill criterion, computational cost of robustness, and the choice of a robustness criterion in practical scenarios (Chapters 4 and 5).

Important Remarks

In the remainder of the thesis, the terms “efficient global optimization”, “sequential model-based optimization”, and “Bayesian optimization” are used interchangeably to refer to the same concept – surrogate-assisted optimization, where the surrogate model is iteratively updated according to a criterion/merit to find the optimal solution. The chosen criterion/merit is referred to as the “sampling infill criterion”, or the “acquisition function”, which controls the search behavior of the algorithm, i.e., balances the trade-off between exploration and exploitation.

2.4 Summary and Discussion

This chapter provides a concise overview on three different but related topics, namely black-box optimization, robust optimization, and surrogate modeling, respectively. Section 2.1 provides a short description of black-box optimization, as well as definitions for some of the most important and related concepts. This section also emphasizes on the practical goal of optimization with regards to computational tractability, i.e., computational resources/budget available. Note that the practical goal of optimization proposes to use the computational resources in an optimal way to find better solutions, which are an improvement with respect to the previously known best solutions (Wright et al., 1999).

Section 2.2 introduces the notion of uncertainty and noise in black-box optimization. Based on their origins, five different classes of uncertainty and noise are identified. The sources of uncertainty include decision/search variables, environmental variables, output/evaluation of the system, constraints, and objectives, respectively. Furthermore, three mathematical ways of modeling these uncertainties: deterministic, probabilistic, and possibilistic, are presented. Based on the combinations of different classes of uncertainties alongside their mathematical representations, we limit the scope of this thesis to only deal with the uncertainties of the first two types, which can be represented in a deterministic or a probabilistic fashion (Ullah et al., 2019).

In Section 2.3, we provide a short overview of surrogate modeling, which utilizes the empirical models to substitute the expensive function evaluations. Surrogate modeling can be helpful in multiple different ways in black-box optimization (Forrester et al., 2008). We describe the working mechanism of two of the most important modeling techniques, namely the RSM and Kriging. We also describe two different manifestations of surrogate modeling in this thesis, namely the “one-shot optimization” approach, and the “sequential model-based optimization” approach respectively.

CHAPTER 3

Surrogate-Assisted Robust Optimization

This chapter mainly focuses on the applicability of surrogate modeling to find robust solutions, which are still optimal and useful in the face of uncertainty and noise in the decision variables. Uncertainty in the decision variables is frequently-encountered in engineering, where an approximate model replaces the real-world (physical) system (Beyer and Sendhoff, 2007). Note that the model can take arbitrarily precise values of the decision variables, whereas the actual (physical) system cannot be set arbitrarily precise. As such, the (nominal) optimal solutions returned by the model may not be useful in practice (Kruisselbrink, 2012; Jurecka, 2007). Therefore, the designer has to aim for robust solutions, which would still be optimal and useful, even if the real-world (physical) system differs from the (simulation) model.

Pertaining to find robust solutions via surrogate modeling, following are some of the key research questions which need to be answered. Note that these questions are manifestations of the foundational questions introduced in the first chapter.

1. How to choose the sampling plan and data preparation approach for constructing a surrogate model?
2. Which modeling technique to prefer for constructing the surrogate?
3. How to assess the quality of the surrogate model?
4. How to deal with the issue of high dimensionality?
5. How can the structure and scale of the uncertainty, the problem landscape, the dimensionality, and the robustness formulation, impact the quality of the surrogate model?

To answer these questions in a comprehensive manner, we will first provide an optimization framework, whereby we can perform empirical research in surrogate modeling. Note that in real-world scenarios, the issue of high dimensionality can also affect the practical applicability of surrogate modeling (Shan and Wang, 2010). Consequently, we devote the later part of this chapter towards dimensionality reduction in surrogate-assisted optimization.

3.1 Robust Optimization via Surrogate Modeling

We propose an optimization framework, based on OSO strategy (Ta'asan et al., 1992), which can be employed to find robust solutions via surrogate modeling. The proposed framework is presented in Fig. 3.1 in the form of a flowchart. Note that this framework is different from the one proposed by Jurecka (Jurecka, 2007), which is rooted in the SMBO (Jones et al., 1998) approach. SMBO seeks to sequentially update the surrogate model based on the so-called “acquisition function” (Wang, 2018), in order to find the optimal solution. It is an important optimization framework and a topic later in this thesis (Chapters 4 and 5). However, in this chapter, we deviate from SMBO to examine the practicality of surrogate modeling to find robust solutions. This is due to two main reasons.

- As stated earlier, SMBO requires to sequentially update the surrogate model based on the acquisition function, which also needs to be extended to care for robustness, in order to find robust solutions (Rehman, 2016). Extending the acquisition function to the robust scenario is a difficult task, since it depends on the way the uncertainty and robustness formulations are specified. Existing work only focuses on the so-called “Expected Improvement” criterion for only one robustness formulation, namely the so-called “mini-max robustness” (ur Rehman et al., 2014).

  Note that even this approach is limited, as it does not provide a rigorous mathematical formulation for extending the “Expected Improvement”criterion to the robust scenario, but rather focuses on computational tractability to get a reasonably good solution. As we shall see in the next chapter, this approach is a practical compromise, since modeling the true robust response of the function within surrogate-assisted robust optimization is computationally intractable. Furthermore, we find that there are no systematic studies that deal with other types of uncertainties, robustness formulations, and acquisition functions. Therefore, we note that studying the applicability of surrogate modeling to find robust solutions in a comprehensive manner with the help of the SMBO approach is extremely difficult. Due to this reason, we utilize a different optimization framework based on the OSO strategy (Ta'asan et al., 1992) to answer the questions outlined earlier.

- Implementing surrogate-assisted robust optimization based on our proposed framework is straightforward, as it does not require extending the acquisition function, or prohibitively high computational budget. Since the goal here is to assess the practical applicability of surrogate models based on a number of criteria[1], it makes sense to utilize the proposed framework, as it allows us to be much more thorough and comprehensive in our empirical approach (Ullah et al., 2019).

The first step in our proposed framework is the clear formulation of problem description, alongside uncertainty and robustness specification (Kruisselbrink, 2012). Note that uncertainty specification deals with two issues, namely the mathematical modeling of the uncertainty – deterministic vs probabilistic, and the scale/severity of the uncertainty. It is also important to note that in practical situations, the structure and the scale of the uncertainty cannot be completely described in advance (Beyer and Sendhoff, 2007). Nonetheless, the designer, with the help of a domain expert, can make a few general assumptions about the quality of the system at hand, and hence specify uncertainty to some extent. An alternative approach would be to first find a deterministic solution $\mathbf{x}$, and employ the practical applicability of the nominal solution as an indicator for uncertainty assessment.

Robustness specification in the first step refers to choosing a robustness formulation/criterion to find robust solutions – solutions that are not greatly impacted by the uncertainties in the search variables (Jurecka, 2007). The choice of robustness is one of the most critical decisions for the designer, since it can determine the quality of the robust solution, as well as the efficiency of the optimization (measured in terms of computational resources), to a large degree (Gregory et al., 2011). The aim for robustness can be achieved from two different schools of thought, which are often conflicting (Kruisselbrink, 2012).

---

[1] These criteria involve the computational budget/sample size, the modeling technique, the noise level, the robustness formulation, the dimensionality, and the problem landscape, among others.

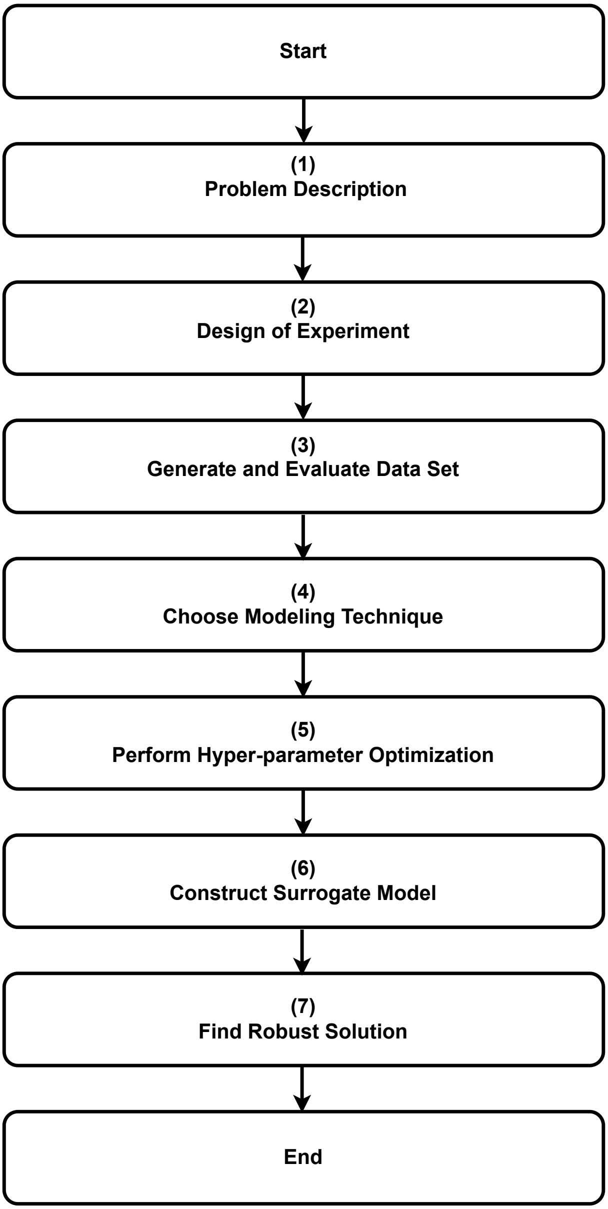

Start
(1)
Problem Description
(2)
Design of Experiment
(3)
Generate and Evaluate Data Set
(4)
Choose Modeling Technique
(5)
Perform Hyper-parameter Optimization
(6)
Construct Surrogate Model
(7)
Find Robust Solution
End

Figure 3.1: Flowchart shows the proposed framework for surrogate-assisted robust optimization in this chapter.

- **Performance/Quality**

  Robustness of a solution is measured from the perspective of the overall performance, i.e., function value, under the variation of the uncertain parameters of the solution.

- **Robustness/Stability**

  Robustness of a solution is measured from the perspective of minimal performance variation under the variation of the uncertain parameters of the solution.

In Section 3.1.1, we describe some of the most common ways to represent robustness mathematically.

The second step of our framework emphasizes on the specification of sampling plans (Montgomery, 2017; Santner et al., 2003). Ideally, we want to have the maximum information about the search space, in order to achieve a high quality approximation. In practice, however, we only have a finite amount of computational resources, and thus cannot afford to observe the function response at each search point. Therefore, we must come up with a plan to extract maximum information about the search space with a finite (usually small) number of samples. To this end, we can select a sampling plan, which according to a criterion/merit, and available computational budget, completely specifies the sampling points (Gramacy, 2020). Section 3.1.2 describes some of the most common sampling plans based on DoE approaches.

The third step in our framework involves generating the initial design data, i.e., computing function responses, and pre-processing it, if deemed necessary. Note that pre-processing the data is a common practice in statistical learning (Hastie et al., 2009), since many modeling techniques make a few general assumptions on the structure of the data, in order to effectively model it. Pre-processing the design data may also help with better generalization capability of the model (Bishop, 2007). Despite its importance, however, the decision on the pre-processing should be carefully made based on the problem at hand, as well as the choice of the modeling technique, among others.

After the completion of the first three steps, we can construct the surrogate model based on the modeling technique chosen, e.g., Kriging. Note that the computational complexity involved to find the robust solution may also be affected by the choice of the modeling technique (Ullah et al., 2020a). After constructing the surrogate model, we perform hyper-parameter optimization (HPO) (Hutter et al., 2011, 2009). The purpose of HPO is to get the best quality surrogate model based on the available function evaluations, i.e., HPO can improve the quality of a surrogate model by optimizing the corresponding hyper-parameters.

Following this, we utilize the model to find robust solutions. This refers to the fact that we employ a benchmark numerical optimization algorithm (Wright et al., 1999) to find a robust solution[1], which utilizes the predictions of the surrogate model as an approximation to the actual function evaluations. When the algorithm converges, we return the robust solution, and the process comes to a halt. In the following, we describe some of the most important concepts related to our optimization framework in further detail.

3.1.1 Robust Counterpart Approach

We deal with the uncertainty in the decision variables, which can be represented in a deterministic or a probabilistic fashion. When facing this type of uncertainty, the objective function is transformed as well. Reformulating the objective function to account for robustness is referred to as the *robust counterpart approach* (RCA) (Ben-Tal et al., 2009). The new objective function depends on the choice of the robustness formulation, which in turn is based on the anticipated structure of the uncertainty, as well as other criteria, e.g., application domain, computational budget.

In the following, we describe some of the most common robustness formulations, employed in this thesis.

Mini-max Robustness

"Mini-max robustness" (MMR) deals with the uncertainty which is modeled with a deterministic set (Rehman, 2016). Given a real-parameter objective function: $f(\mathbf{x})$, and the additive uncertainty in the decision variables: $\Delta_{\mathbf{x}}$, the "mini-max" treatment minimizes the worst-case scenario for each search point $\mathbf{x}$, where the worst-case is defined by taking into account all possible perturbations to $\mathbf{x}$, which are restricted in a compact set $\mathrm{U} \subseteq \mathbb{R}^D$ (containing a neighborhood of $\mathbf{x}$).

---

[1] The robust solution is based on the robustness formulation/criterion chosen in the first step of the framework.

Effectively, this is to minimize the following objective function:

$$f_{\text{eff}}(\mathbf{x}) = \max_{\Delta_{\mathbf{x}} \in \mathrm{U}} f(\mathbf{x} + \Delta_{\mathbf{x}}). \tag{3.1}$$

The worst-case scenario refers to the fact that we consider the maximal value of the function under additive uncertainty at each search point, and try to minimize that (Ben-Tal et al., 2009; Ullah et al., 2019). As Kruisselbrink notes (Kruisselbrink, 2012), this type of uncertainty handling is also referred to as the *least upper bound* (LUB). Note that the bounds of the compact set U are based on the anticipated scale/severity of the uncertainty – based on the maximum anticipated deviation of the decision variables from their nominal values. Throughout this thesis, we assume that the deterministic uncertainty is symmetric around zero.

Note that $f_{\text{eff}}(\mathbf{x})$ in Eq. (3.1) is the robust (or effective) counterpart of the original objective function, which ensures we have a minimal performance variation under uncertainty. Therefore, this robustness criterion has a great appeal, when dealing with highly sensitive applications – where the designer cannot afford to accept a slight deviation in the performance (McIlhagga et al., 1996; El Ghaoui and Lebret, 1997; Herrmann, 1999).

Mini-max Regret Robustness

"Mini-max Regret Robustness" (MMRR) (Jurecka, 2007) focuses on minimizing the maximum regret under uncertainty. The regret can be defined as the difference between the best obtainable value of the function f^* for an uncertainty event $\Delta_{\mathbf{x}}$, and the actual function value under that uncertainty event $f(\mathbf{x} + \Delta_{\mathbf{x}})$. The best obtainable response f^* of the function under an uncertainty event $\Delta_{\mathbf{x}}$ can be defined as:

$$f^*(\Delta_{\mathbf{x}}) = \min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x} + \Delta_{\mathbf{x}}), \tag{3.2}$$

and the robust counterpart for MMRR can be defined as:

$$f_{\text{eff}}(\mathbf{x}) = \max_{\Delta_{\mathbf{x}} \in \mathrm{U}} (f(\mathbf{x} + \Delta_{\mathbf{x}}) - f^*(\Delta_{\mathbf{x}})). \tag{3.3}$$

Minimizing Eq. (3.3) refers to the fact that firstly, the best achievable response value for each uncertainty event: $\Delta_{\mathbf{x}} \in \mathrm{U}$, is subtracted from the actual outcome: $f(\mathbf{x} + \Delta_{\mathbf{x}})$. Then, the worst-case is determined similar to the MMR. As a conclusion, the optimal solution is identified as the one for which the worst-case has a minimal deviation from f^* as defined in Eq. (3.2) (Jiang et al., 2013).

One of the most beneficial aspects of employing MMRR is that it ensures that even in the worst-case scenario, we are still not very far from the nominal optimum, and therefore not compromising significantly in terms of optimality. Hence, it can be argued that it deals with the concerns of both schools of thought in an elegant manner. The biggest challenge, however, is the prohibitively high computational demand. Notice that solving Eq. (3.3) inside an iterative optimization framework (such as SMBO) implies a quadrupled nested loop, which is computationally infeasible even for a modest setting of dimensionality.

Expectation-based Robustness

Different from the first two robustness formulations, the expected output of a noisy function can also serve as a robustness criterion (Kruisselbrink, 2012; Jurecka, 2007; Beyer and Sendhoff, 2007). The focus of this robustness criterion is the overall good performance, rather than the minimal deviation of the optimal solution under uncertainty. Note, however, that, this robustness formulation (RF) requires the uncertainty to be defined in a probabilistic manner. The uncertainty can be modeled according to a continuous uniform probability distribution, if no prior information is available.

The (effective) robust counterpart of the original function based on "Expectation-based Robustness" (EBR) is defined as:

$$f_{\text{eff}}(\mathbf{x}) = \mathbb{E}_{\Delta_{\mathbf{x}} \sim \mathcal{U}(a,b)}[f(\mathbf{x} + \Delta_{\mathbf{x}})], \tag{3.4}$$

where the bounds a and b can be set according to the anticipated scale of the uncertainty.

Dispersion-based Robustness

Minimizing the performance variance under variation of uncertain search variables (Jurecka, 2007; Kruisselbrink, 2012) is an important criterion to achieve robustness in several applications (Das, 2000). In this case, the original objective function: $f(\mathbf{x})$, can be remodeled into a robust objective function: $f_{\text{eff}}(\mathbf{x})$, by minimizing the variance as:

$$f_{\text{eff}}(\mathbf{x}) = \sqrt{\mathrm{Var}_{\Delta_{\mathbf{x}} \sim \mathcal{U}(\mathrm{a},\mathrm{b})}[f(\mathbf{x} + \Delta_{\mathbf{x}})]}. \tag{3.5}$$

Note that this RF also requires the uncertainty to be defined in a probabilistic manner, similar to the previous case.

Composite Robustness

Different from the robustness criteria mentioned above, practitioners may also optimize the expected output of a noisy function, while minimizing the dispersion simultaneously. We refer to this formulation as the “Composite Robustness” (CR). CR requires the uncertainty to be specified in the form of a probability distribution. The expectation and dispersion of the noisy function are combined at each search point $\mathbf{x} \in \mathcal{S}$ to produce a robust output.

The optimization goal thus becomes to find a point $\mathbf{x}^* \in \mathcal{S}$, which minimizes:

$$f_{\text{eff}}(\mathbf{x}) := \mathbb{E}_{\Delta_\mathbf{x} \sim \mathcal{U}(a,b)}[f(\mathbf{x} + \Delta_\mathbf{x})] + \sqrt{\mathrm{Var}_{\Delta_\mathbf{x} \sim \mathcal{U}(a,b)}[f(\mathbf{x} + \Delta_\mathbf{x})]}. \tag{3.6}$$

Note that this approach also tries to balance the concerns of both schools of thought, e.g., performance and stability. This approach may also be adapted as a bi-objective optimization problem, and the two terms in Eq. (3.6) may also be weighted (Lee and Park, 2001).

3.1.2 Design of Experiment

The surrogate modeling techniques rely on training data, which has to be collected by evaluating the function responses at well-specified sampling points. The mechanism to choose these points is described with the help of DoE techniques (Gramacy, 2020). As stated earlier, it is often advantageous to minimize the number of sampling points in order to reduce the simulation effort. On the other hand, it is crucial to extract as much information as possible about the major characteristics of the system under investigation. Note that the choice of the modeling technique, e.g., Kriging, can also influence the sampling plan (Montgomery, 2017).

In this thesis, we describe an *experimental design* as a set of N experiments, each of which is based on D input variables, which may also be referred to as the *factors* or *co-variates* in this context. An experimental design can be represented with a matrix X, where the rows distinguish between different experiments, and the columns represent different factors. As Jurecka notes (Jurecka, 2007), the choice for a particular DoE depends on the following factors.

- The intended utilization of the model, e.g., to construct a surrogate model for numerical optimization (Wright et al., 1999), vs space visualization and comprehension (Forrester et al., 2008).

- Available information concerning the problem at hand, e.g., the complexity, the dimensionality, and the relevant domain information (Rehman, 2016).
- Additional constraints such as the limitation of the computational resources, and the choice of the modeling technique (Jurecka, 2007).

In the following, we describe two major classes of DoE techniques.

Full Factorial Design

In *full factorial design*, we start with the assumption that we have D factors (or variables) describing our system, which can only take a finite number of values/levels, which are denoted as L. These levels are completely specified before observing the system response at any of the locations. Then, the full factorial design can be defined as the design, which contains all factor-level combinations (Montgomery, 2017). The total number of experiments to be performed results from the product of the respective number of discrete levels for each factor as:

$$N = \prod_{k=1}^{D} L_k. \tag{3.7}$$

Note that a full factorial design with D factors, each evaluated at L levels, is symbolized as L^D. Therefore, an increasing number of factors or levels rapidly raises the experimental effort (Jurecka, 2007).

Fractional Factorial Design

A *fractional factorial design* consists of a subset of a full factorial design (Gramacy, 2020; Montgomery, 2017). These experimental designs are typically represented as L^{D-R}, where R defines the reduction compared to the corresponding full factorial design. The total number of sampling points in a fractional factorial design is only a fraction of the corresponding full factorial design, where the fractional portion is $(\frac{1}{L})^R$ of the full design (Jurecka, 2007).

Note that the full and fractional factorial designs have been utilized for performing *screening experiments*, where the aim is to identify either especially significant factors, or factors with negligible effect on the response. In this context, they can be thought of being useful for reducing the dimensionality of the problem. Examples of some other DoE techniques, besides full and fractional factorial design, include Orthogonal Arrays (Parr, 1989), Plackett-Burman designs (Myers et al., 2016), and Box-Behnken designs (Ferreira et al., 2007).

Space Filling Design

For (surrogate) modeling techniques which rely on interpolation, e.g., Kriging, the quality of the estimation depends on the distance to the nearest sampling point, as they, i.e., the distance and the quality, are correlated with each other. Due to this reason, the so-called *space-filling property* is emphasized a lot in the context of model-assisted optimization (Rasmussen and Williams, 2006; Gramacy, 2020; Jurecka, 2007). This property ensures that the sampling points are evenly spread over the entire factor space (search space in the context of continuous optimization). Consequently, for an arbitrary untried point $\mathbf{x}$, the distance to the nearest sampling point does not become too large, and a good quality of the estimation is ensured.

There are two major classes of DoE based on the space-filling property. They are explained in the following.

Maxi-min Designs

The so-called *maxi-min design* scheme emphasizes on maximizing the minimum distance between all pairs of sampling points to ensure space-filling (Johnson et al., 1990; Tan, 2013). Note that in this context, the choice of metric for measuring the distance is crucial. If we choose the (squared) Euclidean metric, we can define the distance as:

$$D(\mathbf{x}, \mathbf{x}^{'}) = ||\mathbf{x} - \mathbf{x}^{'}||^2 = \sum_{j=1}^{D} (\mathbf{x} - \mathbf{x}^{'})^2. \tag{3.8}$$

Then, a design: $\mathrm{X}^* = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, which maximizes the minimum distance between all pairs of points,

$$\mathrm{X}^* = \mathrm{argmax}_{\mathrm{X}} \ \min \{D(\mathbf{x}_i, \mathbf{x}_k) \ : \ i = 1, \ldots, N \ \wedge i \neq k\}, \tag{3.9}$$

is called a maxi-min design. Note that the opposite way around, the so-called mini-max design, also leads to sampling points which are all spread out (Gramacy, 2020).

Latin Hyper-cube Sampling

The aim in *latin hyper-cube sampling* (LHS) (Gramacy, 2020; Olsson et al., 2003) is to guarantee a certain degree of spread in the design, while otherwise enjoying the properties of a random uniform sampling scheme. LHS accomplishes that by dividing the search space into equal-sized cubes/segments, and ensuring that each such segments contains just one sample point, which is drawn uniformly within that cube. If we have only two factors, the pattern of the selected cube containing a sample resembles *latin squares* (Jurecka, 2007). Since the location of the sampling point within the selected cube is allowed to be random, the LHS does not preclude two points located nearby one another. For instance, two points may reside near a corner in common between a cube from an adjacent row and column. Nonetheless, it is important to know that LHS does limit the number of such adjacent cases, guaranteeing a certain amount of spread.

Formally, a latin hype-cube design X in the search space $[0,1]^D$ is an $N \times D$ matrix, whose ij-th entry can be defined as

$$\mathbf{x}_{ij} = \frac{l_{ij} + (N-1)/2 + \mu_{ij}}{N}, \; i = 1, \ldots, N \;\wedge j = 1, \ldots, D, \tag{3.10}$$

where μ_{ij} are independent uniform random samples in $[0,1]$, and l_{ij} represents the i-th sample and j-th factor from the latin hyper-cube of N designs and D factors. Note that the denominator N normalizes such that $\mathbf{x}_{ij}$ is mapped into $[0,1]^D$. For a detailed overview of LHS and related space-filling designs, please refer to the work of Gramacy (Gramacy, 2020), and Montgomery (Montgomery, 2017).

3.1.3 Preparing Data and Choosing a Modeling Approach

After generating the training data set with the help of DoE techniques, it is important to prepare and present it in a meaningful way to the learning technique to ensure a good generalization capability of the model. We refer to this step as the pre-processing of the initial (design) data. Pre-processing the data may involve dimensionality reduction, screening, space visualization, and scaling/standardization among others (Bishop, 2007; Hastie et al., 2009). Note that dimensionality reduction and screening could be important for pre-processing if we are dealing with a large number of design variables.

In general, standardizing the decision variables is important when we compare measurements that have different units. Decision variables that are measured at different scales do not contribute equally to the construction of the surrogate models, and might end up creating a bias. For instance, a decision variable that ranges between 0 and 1000 may outweigh another decision variable that only ranges between 0 and 1. Employing these variables without standardization thus may give the variable with the larger range, a weight of 1000 in the analysis (Rehman, 2016).

Transforming the design data to a comparable scale can prevent issues like this. Typical data standardization procedures equalize the range, and/or data variability. Similar to standardization, the goal of normalization is to change the values of the numerical decision variables in the data set to a common scale, without distorting the differences in the ranges of values. Note that in the context of statistical learning (Hastie et al., 2009; Bishop, 2007), not every data set requires normalization. Rather, data normalization is emphasized only when variables have different ranges.

Although pre-processing can be crucial, it is hard to specify the exact procedure for it. This is since the aim is to get the best quality surrogate model for each test problem with minimum computational effort. In the following, we describe some general guidelines for pre-processing.

- In the face of a large number of decision variables, dimensionality reduction becomes important (Shan and Wang, 2010). To this end, one may utilize the classical dimensionality reduction techniques, e.g., Principal Component Analysis (Jolliffe, 1986), as well as screening and visualization of the search space.
- In case the decision variables have different units of measurement, e.g., meter vs seconds, standardization and/or normalization becomes crucial (Jolliffe and Cadima, 2016).
- If the chosen modeling technique assumes certain characteristics of the data-generating process, e.g., standard normal distribution, consider converting the original design data into a form, which is closer to the assumed behavior (Hastie et al., 2009).

After the data pre-processing, we face another important question, which deals with the choice of the modeling technique. The selection of the modeling technique depends on the following factors.

- **Approximation Quality**

  The quality of approximation of the original search space is one of the most important factors in determining the modeling technique. Since the problem landscape may exhibit certain undesirable characteristics, e.g., multi-modality, our modeling technique has to be able to model the function responses in an effective manner (Jurecka, 2007). Examples of such techniques involve Polynomial Regression, Kriging, Support Vector Regression, and Artificial Neural Networks, among others. Note that we also have to avoid over-fitting in addition to maximizing the quality of the approximation, when choosing a modeling approach (Bishop, 2007; Goodfellow et al., 2016; Hastie et al., 2009; Vapnik, 1999). This is since some techniques, e.g., Artificial Neural Networks, are more prone to over-fitting than others, which may result in poor generalization capability of such models (Ying, 2019).

- **Computational Budget**

  Similar to approximation quality, computational budget is also an important factor. This is since even if we choose a modeling technique with a good (approximation) quality, we may not have enough computational resources to construct the model based on that (Forrester et al., 2008). On the other hand, it is possible that we may have to choose a less promising modeling technique in practice simply because of the computational resources available (Keane et al., 2008).

- **Utilization in Optimization**

  The choice of the modeling technique may also be based on its potential utilization in the optimization pipeline. For instance, if we want to sequentially update the surrogate model, we need a measure of the uncertainty in the prediction, which makes Kriging an ideal candidate for the choice of the modeling technique (Rasmussen and Williams, 2006). On the other hand, if our purpose is only space visualization and comprehension, we may end up using the Polynomial Regression (Bishop, 2007).

3.1.4 Appraising the Surrogate Model

An important aspect in model-assisted optimization is the evaluation of the surrogate model (Forrester et al., 2008). An appropriate evaluation criterion can ensure the quality of our model, and therefore its subsequent usage (Queipo et al., 2005). There are different criteria that apply to the surrogate models based on their utilization. In this work, we only deal with two criteria. These are referred to as *the modeling accuracy* and *the quality of the solution* respectively. There are explained in the following.

Modeling Accuracy

We measure the global error in modeling, which is referred to as the Relative Mean Absolute Error (RMAE) (Shcherbakov et al., 2013). The RMAE measures the error in the overall approximation of the search space by the surrogate model, with the help of a testing data set of M instances.

The mathematical formulation of the RMAE is given as

$$\mathrm{RMAE} = \frac{1}{M}\sum_{i=1}^{M} 100 \cdot \left(\frac{|f_i - \hat{f}_i|}{|f_i|}\right), \tag{3.11}$$

where f_i and $\hat{f}_i$ are the target and the predicted values for the i-th test data point. Note that the RMAE is averaged, normalized, and measured in percentage. The RMAE indicates the overall precision of the surrogate model. The benefit of employing RMAE as an assessment criterion is that the quality of the approximation is always determined relative to the corresponding ground truth/baseline, which is desirable since different robustness criteria can affect the scale and structure of the original landscape in different ways (Kruisselbrink, 2012).

Optimality Criteria

Apart from the modeling accuracy, another important criterion employed in this thesis is based on the difference in the quality ($\mathcal{DQ}$) of the optimal solutions. In this case, the quality of the optimal solution (obtained from the surrogate model) is compared to that of a baseline for multiple independent runs. Often, the baseline is measured by solving the optimization problem with a benchmark (numerical) optimization algorithm (Wright et al., 1999; Boggs and Tolle, 1995). Then, the average[1] difference between the two solutions is computed, which acts as a criterion for assessing the performance of the model.

$$\mathcal{DQ} = \frac{1}{R}\sum_{i=1}^{R} |S^* - S_i|, \tag{3.12}$$

where S_i and R denote the optimal solution (obtained from the surrogate model), and number of independent runs respectively, and S^* serves as the baseline[2].

---

[1] Note that in some empirical studies in this thesis (Ullah et al., 2020a), we take the median in lieu of the average difference. In such situations, it is explicitly stated that the difference in the quality is measured according to the median.

[2] The difference between the two solutions can be computed in the search space, as well as the space of the objective function values. Therefore, S_i and S^* in Eq. (3.12) represent both of

3.1.5 Hyper-parameter Optimization

Hyper-parameter optimization (HPO) refers to estimating the optimal configuration (settings) of the hyper-parameters involved in constructing the surrogate model (Hutter et al., 2009). HPO can improve the performance of the surrogate model to a certain degree. Recent advances in computer hardware technology, as well as the increase in the number of hyper-parameters in algorithm configuration, have led the researchers and practitioners to employ HPO as a powerful heuristic to ensure good quality of the approximation in surrogate modeling (Ullah et al., 2019; Hutter et al., 2011). We employ HPO in our proposed framework to ensure the best quality surrogate models based on the available function evaluations.

For performing HPO in this thesis, we do not stick to a particular method, but rather choose an approach which is practically viable for the situation at hand. This includes the utilization of a grid of values, as well as random and LHS schemes for HPO. We also employ the Tree Parzen Estimator algorithm (TPE) (Bergstra et al., 2011) to perform HPO. For the related material on how to perform HPO in algorithm configuration, please refer to the works of Bergstra (Bergstra et al., 2011, 2013a), and Hutter (Hutter et al., 2009, 2011).

3.1.6 Empirical Investigation

So far in this chapter, we have provided a basic workflow, whereby we can solve RO problems via surrogate modeling. The workflow starts with the problem description, which involves uncertainty and robustness specifications. We assume that the uncertainty in this work can only be specified by a probability density function, or a compact deterministic set. The latter describes the potential range of uncertainty with a set of numeric values in which all values are equally likely. The robustness formulation is therefore based on the chosen representation of the uncertainty. Section 3.1.1 describes some of the widely utilized robustness criteria such as MMR, MMRR, EBR, DBR, and CR (Jurecka, 2007). In Chapter 5, we will extensively deal with the issue of choosing the robustness criteria based on the computational efficiency.

The second step in our framework deals with two issues, namely the determination of the total number of samples, and the corresponding locations in the search space. We will investigate the first issue – the total number of samples – in the following

---

these situations with one generalized notation.

empirical study (Ullah et al., 2019). For choosing the sampling locations, we can make use of the DoE techniques mentioned in the literature (Gramacy, 2020; Montgomery, 2017; Jurecka, 2007). Note that while each of the DoE techniques has its own merits and demerits, it is recommended to employ a space filling DoE for model-based optimization. Throughout this work, we utilize the LHS scheme to determine sampling points.

The next step in our framework is to compute the function responses at the chosen sampling locations, and pre-process the resulting data set, if deemed necessary. Pre-processing is an important step in surrogate modeling, since it can help us in two different ways.

- Pre-processing may help us achieve a higher quality surrogate model, since it can alleviate some of the problems related to poor generalization of the model, e.g., multi-dimensional scaling issues, improper representation of the training data (Hastie et al., 2009).
- Pre-processing may help us construct the surrogate model in an efficient manner. This is due to the fact that we can employ screening or dimensionality reduction techniques in pre-processing, which can provide us with a representative subset of the most important decision variables (Jolliffe and Cadima, 2016; Bishop, 2007).

After the pre-processing, we construct the surrogate model based on the modeling technique chosen. The choice of the modeling technique is based on a number of factors, such as the ability to model the complex behavior of the system, as well as the samples size required to train the model adequately (Keane et al., 2008). In the following empirical investigation, we will attempt to find which of the widely utilized modeling techniques is most suitable for RO. Note that we also perform HPO to get the best quality surrogate model. Following this, we perform the "one-shot optimization" (Ta'asan et al., 1992) to find the robust solution on the surrogate model.

In the following, we describe the aim, the experimental setup, and the key findings of our empirical investigation (Ullah et al., 2019).

Aim of the Empirical Investigation

Through this study, we aim to answer the following questions.

- How many samples are required to construct a surrogate with good (approximation) quality?
- Which modeling technique is most suitable for finding the robust solutions?
- How does the scale of the uncertainty, i.e., noise level, affect the quality of the approximation, and the robust solution?
- What is the impact of problem landscape and dimensionality on the quality of the approximation, and the robust solution?
- Is the quality of the approximation, and the robust solution, affected by the choice of the robustness criterion?
- Is surrogate modeling applicable to find robust solutions, i.e., is the quality of the robust solution (obtained from surrogate modeling), deemed satisfactory, when compared with the baseline?

Answering these questions in a comprehensive manner is important because of the associated practical reasons, as it will enable us to find robust solutions in an efficient manner, with the help of surrogate modeling.

Experimental Setup

In this section, we first describe the optimization problems considered in our study. We then outline the three levels of noise investigated in our experiments. In particular, we will specify the context of the noise level for the robustness formulations employed. Lastly, we will describe the (surrogate) modeling techniques, and the evaluation criteria to appraise the surrogate models.

We choose six unconstrained black-box optimization problems in our study. Each of these problems is uniquely identified based on the choice of the test function, and the dimensionality: $D = \{2, 5, 10\}$. The selected test functions are known as Ackley, Branin, Sphere and Rastrigin. Among these test functions, Branin is only defined for $2D$, Sphere for $5D$, Rastrigin for $10D$, and Ackley is tested for all three settings of the dimensionality. This results in a total of six optimization problems. Note that each one of these problems is investigated on three levels of additive noise – 5, 10 and 20 % perturbation in the nominal values of the decision variables, and two robustness formulations – mini-max robustness and composite robustness, as described in Section 3.1.1. All six optimization problems are presented in Table 3.1, including the box constraints, and key landscape characteristics.

Table 3.1: All six optimization problems with test functions, key landscape characteristics, dimensions, and box constraints.

Function	**Landscape**	**Dimensionality**	**Bounds**
Ackley	Multi-Modal	$D = \{2, 5, 10\}$	$x_i \in [-32.768, 32.768]$
Branin	Multi-Global	$D = \{2\}$	$x_1 \in [-5, 10], x_2 \in [0, 15]$
Sphere	Isotropic	$D = \{5\}$	$x_i \in [-5, 5]$
Rastrigin	Multi-Modal	$D = \{10\}$	$x_i \in [-5.12, 5.12]$

In our setup, we consider three levels of additive noise. The effect of the additive noise in the decision variables has already been presented in Chapter 2. Let $R = |U_b - L_b|$ be the absolute range of the decision variables, where U_b and L_b serve as the upper and lower limits of the box constraints for the decision variables. Further, let L be the additive noise level. In the case of mini-max robustness, this means having a neighbourhood of each design $\mathbf{x}$, whose scale is defined by the parameters range R and noise level L. As an example, the Ackley function is defined from $L_b = -32.768$ to $U_b = 32.768$, having an absolute range of $R = 65.536$.

Considering the first noise level, i.e., $L = 5$ % in Eq. (3.1), this means the size of the neighborhood is defined as: $L \times R = 0.05 \times 65.536 = 3.2768$. Throughout this thesis, we assume the noise is symmetric around zero, hence a neighbourhood: $\mathrm{U} = [-3.2768, 3.2768]$, of design $\mathbf{x}$ is constructed, and Eq. (3.1) can be solved. For composite robustness in our setup, we employ a normal distribution as: $\Delta_{\mathbf{x}} \sim \mathcal{N}(0, \sigma^2)$, where the variance is defined as: $\sigma^2 = \frac{(L \times R)}{6}$, and L and R serve as the noise level and the absolute range of the decision variables, same as above. Once the noise is specified in the form of a probability distribution, the CR in Eq. (3.6) can be solved.

We choose six (surrogate) modeling techniques in our study, namely Kriging (Rasmussen and Williams, 2006), Support Vector Machines (SVM) (Cristianini and Shawe-Taylor, 2004), Radial Basis Function Network (RBFN) (Orr et al., 1996), Random Forest (RF), K-Nearest Neighbors (KNN), and Polynomial Regression with Elastic-net penalty (ELN) (Bishop, 2007), for each of the six optimization problems with three levels of noise. More specifically, for each case – for each unique combination of the test problem and the noise level, we train these modeling techniques on ten different training sample sizes as: $N = D \times K$, where $D \in \mathscr{D}$

is the corresponding setting of the dimensionality, and K is a factor that maps the dimensionality to the sample size. The resulting surrogate model – for each setting of the sample size – is evaluated with respect to the modeling accuracy on a testing data set with size: $M = 75 \times D$. The sampling points to train and test the surrogate models are generated using LHS scheme. To achieve the best results in model training, we perform HPO with cross-validation, based on a grid of values.

The criteria to evaluate the surrogate models in our study are based on the modeling accuracy, and the quality of the obtained (robust) solutions. To find the robust solutions on the surrogate models, a benchmark optimization algorithm is run on the model surface. To this end, the Sequential Least Square Programming (SLSQP) (Boggs and Tolle, 1995) is chosen as the benchmark. To evaluate the surrogate models for the second criterion, each model is first constructed using HPO on a training sample of $N = 50 \times D$. An optimization run with SLSQP is then performed on the constructed model to minimize Eqs. (3.1) and (3.6). This process is repeated a 100 times, and the difference in the quality $\mathcal{DQ}$ of the solutions is computed according to Eq. (3.12). Note that the difference in the quality $\mathcal{DQ}$ in this context is based on the objective function values.

3.1.7 Results

Graphs showing the (modeling) accuracy of the surrogate models, by varying the training sample size N, evaluated on the basis of RMAE, are presented in Figs. 3.2 – 3.7. Note that each curve in these plots is accompanied with standard error (SE) (in the computation of RMAE for a particular choice of the training sample size). The modeling accuracy results for the two dimensional Ackley function are presented in Fig. 3.2, whereas the results for the two dimensional Branin function are illustrated in Fig. 3.3. We show the modeling accuracy of the surrogates on five dimensional Ackley and Sphere functions in Figs. 3.4 and 3.5 respectively. Lastly, the modeling accuracy results for the ten dimensional functions are presented in Figs. 3.6 and 3.7.

To find the best surrogate model for each of the 36 test scenarios – owing to the combination of two robustness criteria, three noise levels, and six test problems, we rank the surrogates based on the RMAE, which is averaged over all ten settings of the training sample size N. Note that the ranking is based on an ascending order (lowest to highest average RMAE for each test scenario). Then, we perform a non-parametric test, namely the so-called Mann-Whitney U test, to find if there is a statistical difference in the performance of the top two surrogate models, i.e, to verify that the top performing surrogate model is indeed significantly better than the nearest opponent. The resulting p-values for all 36 test scenarios are presented in Fig. 3.8. Additionally, the frequencies of the modeling techniques, having the top performing surrogates, for all optimization problems, are presented in Fig. 3.9. Note that this figure follows the similar evaluation criteria to find the top performing surrogates, i.e., lowest average RMAE. Finally, the results concerning the performance of the surrogate models based on the optimality criterion – the difference in the quality $\mathcal{DQ}$ of the robust solution, for all thirty six cases, are presented in Fig. 3.10.

In the following, we report the major findings from our investigation.

- **Sample Size**

  From the graphs in Figs. 3.2 – 3.7, we find that even a small setting of K results in a good (approximation) quality for most of the test scenarios. The only exception we find is related to the Branin function, where the quality of the approximation is not adequate, even for a higher value of K. Our observation validates the generally employed heuristic in model-assisted optimization, which states that the initial sample size can be set linearly in terms of dimensionality (Forrester et al., 2008; Jurecka, 2007).

- **Modeling Technique**

  As Fig. 3.9 suggests, in most cases, we obtain the best quality[1] surrogate models from Kriging, SVMs, and Polynomials. This observation validates the capability of Kriging and Polynomials to model complex system behaviors in an effective manner (Santner et al., 2003; Gramacy, 2020). Besides, it also indicates the promising nature and practical applicability of SVMs in surrogate modeling.

- **Impact of Noise Level**

  From these results, we find that the noise level, in general, does not have a detrimental impact on the quality of the approximation, as we mostly observe similar curves across columns of subplots in Figs. 3.2 – 3.7. We also observe that the quality of the optimal solution is not significantly deteriorated with increasing noise level (cf. Fig. 3.10). However, it is pertinent to mention that these findings cannot be generalized to other optimization frameworks, e.g., SMBO, since there is a possibility that the noise can propagate through the acquisition function (which is to be extended to care for uncertainty and noise).

---

[1] Note that in this case, the quality is solely based on the modeling accuracy.

- **Impact of Dimensionality and Problem Landscape**

  From these results, we observe that dimensionality affects the quality of the optimal solutions for CR. The techniques affected by the dimensionality in this context include Kriging, RBFN, KNN, and RF. Furthermore, we find that in terms of the quality of the solution, the problem landscape can have a significant impact, i.e., notice that highly multi-modal functions, such as Rastrigin, seem extremely difficult to optimize, as opposed to the isotropic, and multi-global landscapes.

- **Robustness Formulation**

  We find that the problem landscape, induced by CR, seems difficult to optimize in most cases. One of the possible reasons for that is that unlike MMR, it relies heavily on numerical approximations, i.e., monte-carlo samples to estimate mean and variance of the noisy objective function, which can deteriorate the quality of the solution to a certain extent.

- **Applicability of Surrogate Modeling**

  Based on the overall performance of the surrogate models in terms of modeling accuracy, and quality of the robust optimal solutions, we conclude that surrogate modeling is applicable for efficiently solving optimization problems under uncertainty. This is due to the fact that in most cases, the quality of the approximation obtained from Kriging, SVMs, and Polynomials is good enough to employ a surrogate to find robust solution. The quality of the optimal solutions in most cases is also satisfactory[1], since the optimal function value found on the model surface is close to the baseline/ground truth (Ullah et al., 2019).

---

[1] Notice that in some cases, the difference in quality is non-existent.

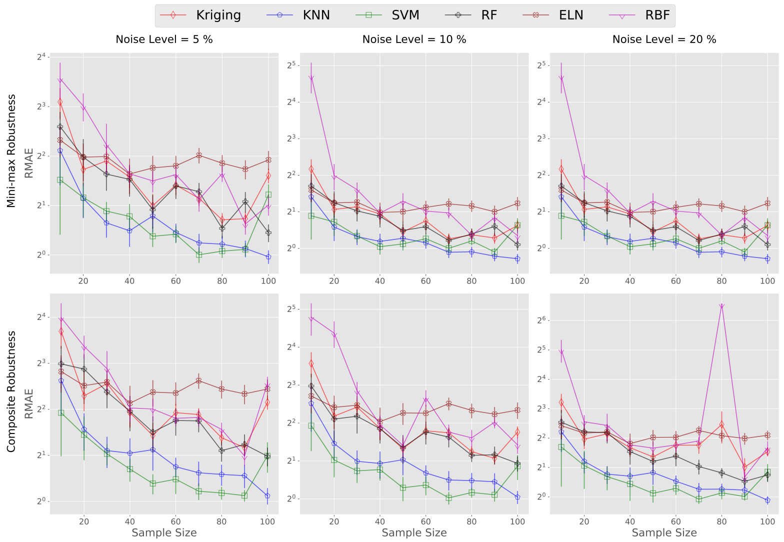


Figure 3.2: Modeling accuracy of the surrogates for the two dimensional Ackley function. The modeling accuracy is evaluated on six test scenarios, owing to the combination of three noise levels, and two robustness criteria.

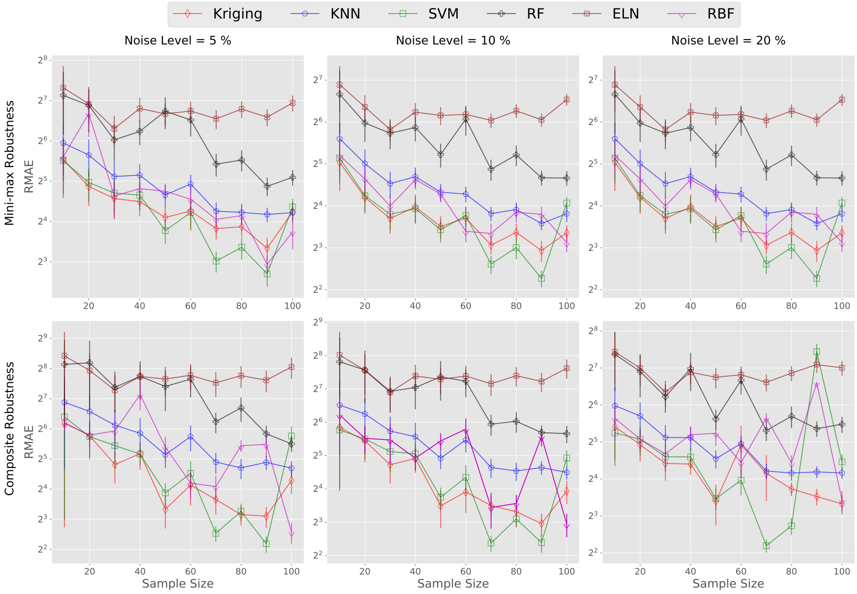


Figure 3.3: Modeling accuracy of the surrogates for the two dimensional Branin function.

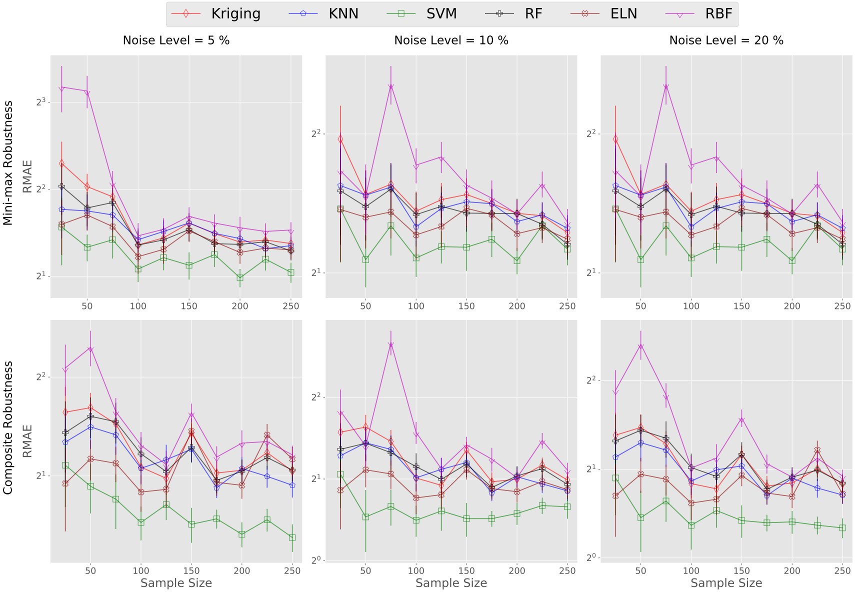


Figure 3.4: Modeling accuracy of the surrogates for the five dimensional Ackley function.

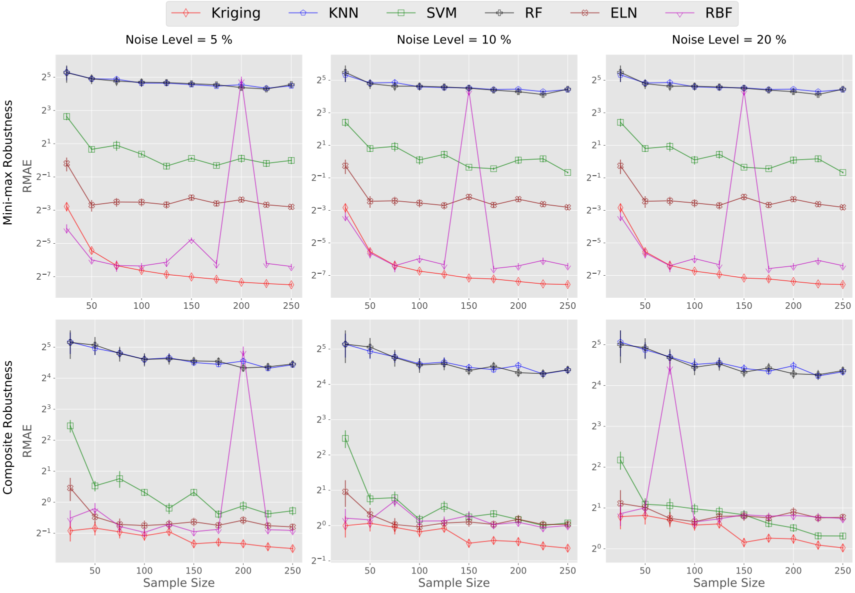


Figure 3.5: Modeling accuracy of the surrogates for the five dimensional Sphere function.

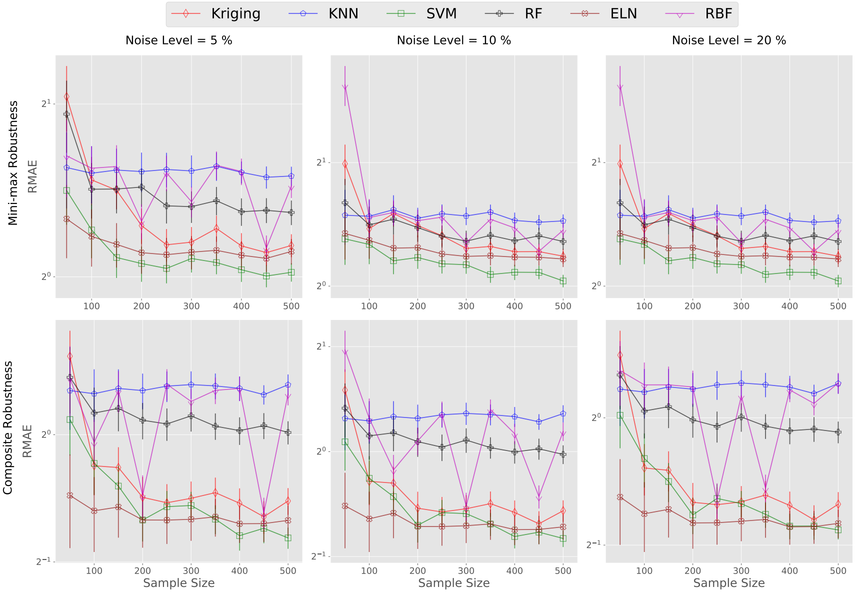


Figure 3.6: Modeling accuracy of the surrogates for the ten dimensional Ackley function.

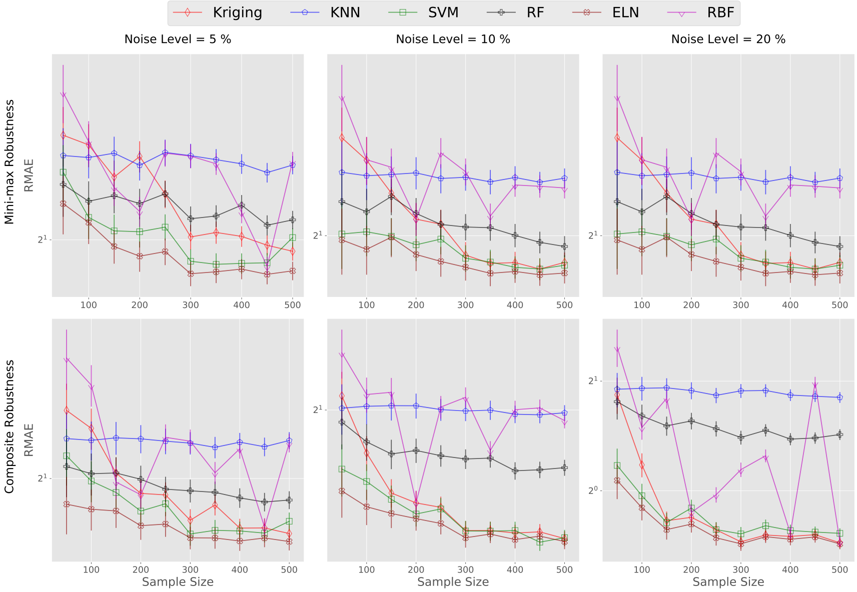


Figure 3.7: Modeling accuracy of the surrogates for the ten dimensional Rastrigin function.

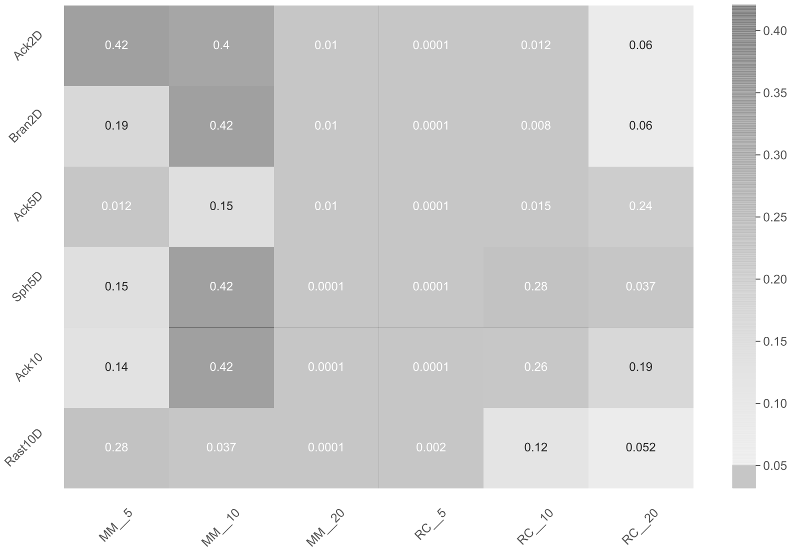


Figure 3.8: *p*-values for all thirty-six test scenarios. The test scenarios are based on six optimization problems (Y-axis), and the combination (X-axis) of three noise level, and two robustness criteria.

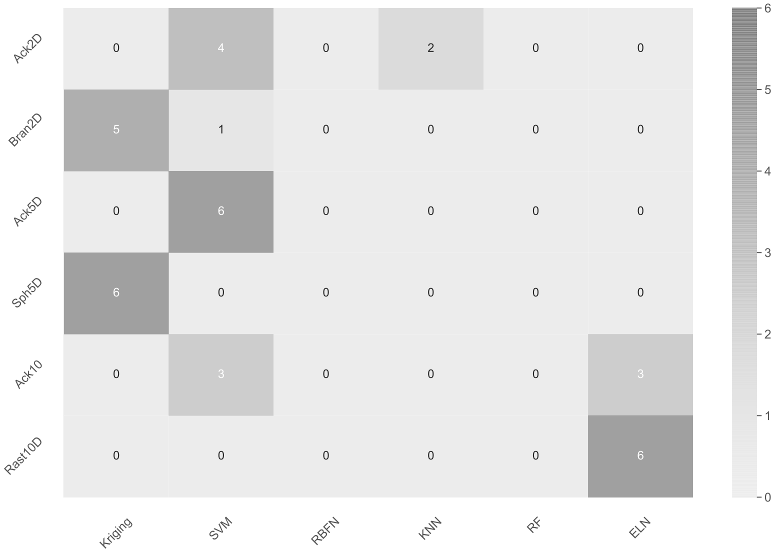


Figure 3.9: Number of occurrences of the modeling techniques, producing the best quality surrogate models. Note that the quality in this context is based on the RMAE values, averaged over all ten settings of the sample size N.

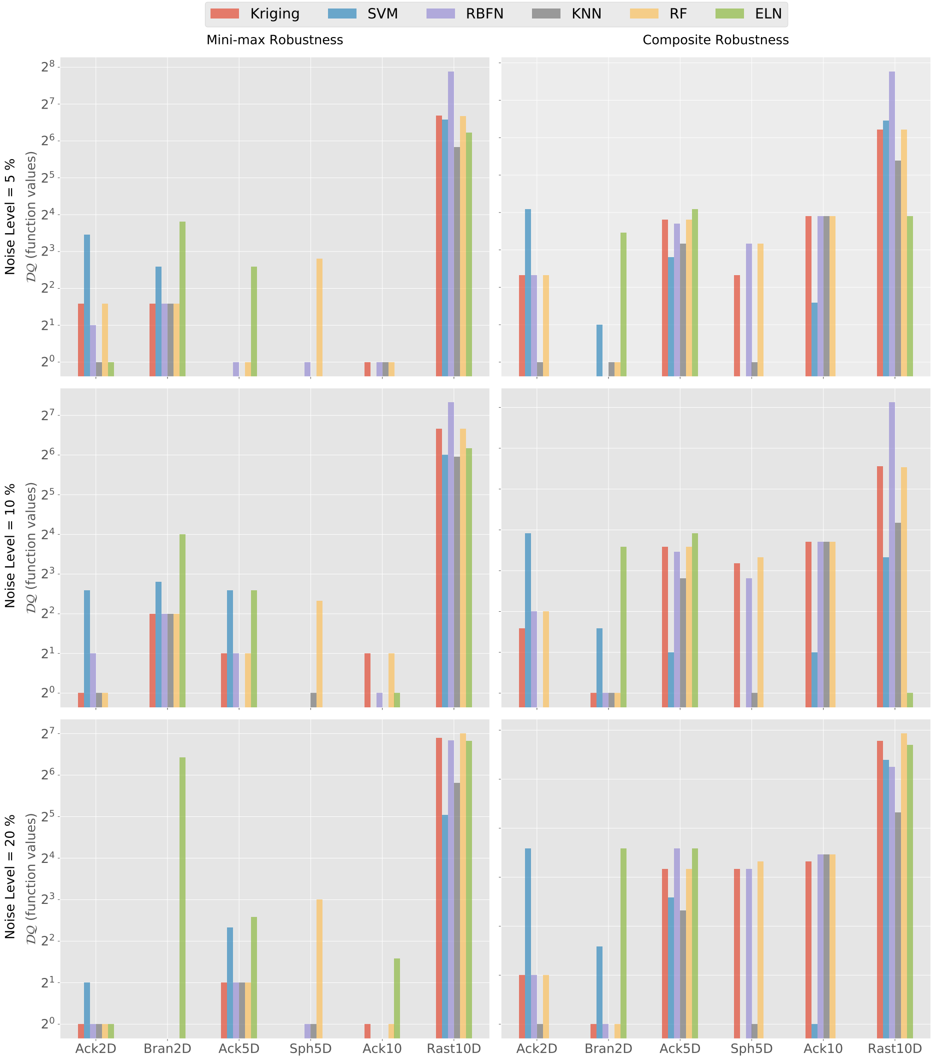


Figure 3.10: The difference in the quality $\mathcal{DQ}$ of the robust solutions (obtained from surrogate modeling), for all thirty six test scenarios. Note that the quality in this context is based on the objective function values, measured according to Eq. (3.12). Furthermore, the number of independent runs in this context are set to be 100.

3.2 The “Curse of Dimensionality”

So far in this chapter, we have exclusively dealt with the issue of uncertainty and noise in continuous optimization. We have proposed a modified optimization framework (as opposed to SMBO), based on surrogate modeling, which enables us to solve robust optimization problems in an efficient manner. Note that the notion of “efficiency” in this context emphasizes on the computational time taken to solve the optimization problem. We believe to have achieved efficiency, since we replaced the actual (expensive) function evaluations by the model predictions. Note, however, that continuous optimization problems in real-world application domains, e.g., mechanics, engineering, economics and finance, can also have an additional obstacle, namely the obstacle of high dimensionality, which can hinder the computational efficiency (Chen et al., 2015).

Modeling high dimensional optimization problems with surrogates is challenging (Shan and Wang, 2010), due to two main reasons.

- More training data is required to achieve a comparable level of modeling accuracy, as the dimensionality increases (Forrester et al., 2008), which results in a considerable upsurge in the computational budget to solve the problem (due to the expensive function evaluations to generate the training data set).
- Algorithmic complexity to construct the surrogate model also increases rapidly with respect to the dimensionality of the problem, and the number of training data points (Stork et al., 2020; Forrester et al., 2008).

Therefore, constructing the surrogate model becomes much costlier, as the dimensionality increases, which is referred to as the “Curse of Dimensionality” (Bishop, 2007). To highlight this issue, upper bounds on the time complexities of the most common surrogate models are presented in Table 3.2. Note that in Table 3.2, D, N, N_{trees}, N_{sv}, and K, stand for the dimensionality of the problem, the number of training data points, the number of trees in RF, the number of support vectors in SVMs, and the number of neighbours in KNNs respectively. From Table 3.2, it can be deduced that higher dimensionality can severely affect the computational budget in model-assisted optimization in two different ways: directly – by a higher value of D, and indirectly – by a higher value of N, N_{trees}, N_{sv}, and K.

Various methodologies have been proposed to deal with the issue of high dimensionality in model-assisted optimization, including divide-and-conquer (Yang et al.,

Table 3.2: Training and prediction time complexities of the most common surrogate models. Notation: N_{trees} is the number of trees in Random Forest, N_{sv} is the number of support vectors in Support Vector Machines, and K is the number of neighbours in K-Nearest Neighbours.

Model	**Training**	**Prediction**
Quadratic Regression	$O(D^4N + D^9 + D^2)$	$O(D^2)$
Random Forest	$O(N^2DN_{\text{trees}})$	$O(NN_{\text{trees}})$
Support Vector Machines	$O(N^2D + N^3)$	$O(DN_{\text{sv}})$
K-Nearest Neighbours	$O(1)$	$O(KD)$
Kriging	$O(N^3D)$	$O(ND)$

2017), variable screening (Wang, 2009), and mapping the data space to a lower dimensional space (Robinson et al., 2008) using dimensionality reduction techniques (DRTs). One of the most common DRTs is Principal Component Analysis (PCA) (Jolliffe and Cadima, 2016). PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the "principal subspace", such that the variance of the projected data is maximized (Bishop, 2007). Various generalized extensions of PCA have been established in the literature, such as Kernel PCA (Schölkopf et al., 1998), Probabilistic PCA (Tipping and Bishop, 1999b; Roweis, 1998), and Bayesian PCA (Tipping and Bishop, 1999a).

On the other hand, Autoencoders (AEs) (Goodfellow et al., 2016; Hinton and Zemel, 1994) have been contemplated as feed-forward neural networks (FFNNs) trained to attempt to copy their input to their output, so as to learn the useful low dimensional encoding of the data. Like PCA, AEs have also been extended over the years by generalized frameworks, such as Sparse Autoencoders (Ranzato et al., 2007), Denoising Autoencoders (Bengio et al., 2013), Contractive Autoencoders (Rifai et al., 2011), and Variational Autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014; Kingma et al., 2019). Besides PCA and AEs, other important DRTs include Isomap (Tenenbaum et al., 2000), Locally-Linear Embedding (Roweis and Saul, 2000), Laplacian Eigenmaps (Belkin and Niyogi, 2002), Curvilinear component analysis (Demartines and Hérault, 1997), and t-distributed stochastic neighbor embedding (Maaten and Hinton, 2008).

In this thesis, we tackle the issue of high dimensionality by performing dimensionality reduction with the help of DRTs. To this end, we face a question on which

DRT to choose for? Ideally, we want to choose a technique which can help us construct a low dimensional surrogate model, which is still useful and approximates the original search space similar to the corresponding (original) higher dimensional surrogate. To find the most suitable DRT in this context, we perform yet another empirical investigation, which takes into account the impact of external factors as well.

In our study, we evaluate and compare the potential of four of the most important DRTs mentioned above, namely PCA, Kernel PCA, AEs and VAEs. We choose PCA and AEs due to their historical significance, since both have been employed regularly for dimensionality reduction, lossy data compression, feature learning, and data visualization (Bishop, 2007; Goodfellow et al., 2016) in machine learning. We incorporate Kernel PCA due to the generalized non-linear extension of the classical PCA algorithm (Schölkopf et al., 1998). Similarly, we consider VAEs in this paper due to the presence of the non-linear stochastic encodings (Kingma and Welling, 2014; Ullah et al., 2020a) of the data space, which can be utilized for constructing the surrogate models efficiently. The focal point of our study is to provide a novel perspective on the applicability of these DRTs in model-assisted optimization. This is accomplished by performing an extensive quality assessment of the corresponding low dimensional surrogate models (LDSMs) on a broad spectrum of of test cases. For a comprehensive overview on DRTs, please refer to the works of Sánchez and Maaten (Van Der Maaten et al., 2009).

3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a data pre-processing method, which is commonly employed for dimensionality reduction, feature engineering, and data visualization (Jolliffe and Cadima, 2016; Bishop, 2007). PCA learns a linear map $\mathbb{R}^D \rightarrow \mathbb{R}^D$, that transforms the original data set into a centered and uncorrelated one, meaning after the transformation, the sample mean is zero, and the sample co-variance matrix is diagonal.

Denoting by $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}^\top$ the design matrix, PCA starts with calculating its sample mean as:

$$\boldsymbol{\mu} = \left[\frac{1}{N}\sum_{i=1}^{N} \mathrm{X}_{i1}, \ldots, \frac{1}{N}\sum_{i=1}^{N} \mathrm{X}_{iD}\right]^\top, \tag{3.13}$$

and centering the original design matrix as:

$$\bar{\mathrm{X}} = \mathrm{X} - \mathbf{1}_N \boldsymbol{\mu}, \tag{3.14}$$

where $\mathbf{1}_N$ is a vector containing N 1's, and each element inside $\boldsymbol{\mu}$ represents the mean of a particular co-variate (decision variable) for the initially generated design data. Then, the first principal component (PC) $\mathbf{u}_1$ can be identified by maximizing the variance of $\bar{\mathrm{X}}$ projected onto $\mathbf{u}_1$ as:

$$\mathbf{u}_1 = \underset{||u|| \leq 1}{\operatorname{argmax}} \ \mathrm{Var}\{\bar{\mathrm{X}}\mathbf{u}\}. \tag{3.15}$$

Similarly, further principal components can be obtained by removing the already-computed PCs from $\bar{\mathrm{X}}$, and then solving the same maximization problem.

Note that the variance of the data projections onto each PCs is monotonically decreasing concerning the order of PCs. It is not difficult to verify that all PCs are necessarily the eigenvectors of the sample co-variance matrix: $\bar{\mathrm{X}}^\top \bar{\mathrm{X}}/N$, sorted with respect to the decreasing order of their corresponding eigenvalues. Using PCA for dimensionality reduction, we select a subset of computed PCs according to a user-specific criterion, e.g., to keep the first several PCs such that the variance of the data projections onto them sum up to a satisfying percentage of the total variability of the original data. In this work, we shall select the first L PCs, where L is the size of the latent space[1].

3.2.2 Kernel Principal Component Analysis

Kernel Principal Component Analysis (KPCA) utilizes the so-called "kernel trick", which transforms the original data points into a high dimensional (usually infinite dimensional) feature space using a non-linear feature map, and performs the linear PCA therein (Schölkopf et al., 1998).

Formally, we denote the feature map as:

$$\phi \colon \mathbb{R}^D \to \mathcal{H}, \tag{3.16}$$

where the feature space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) (Berlinet and Thomas-Agnan, 2011; Bertsimas and Koduri, 2022), equipped with a reproducing kernel such as:

$$k(\mathbf{x}, \cdot) := \phi(\mathbf{x}), \tag{3.17}$$

---

[1] The term "latent space" refers to the newly created feature space, induced by the chosen PCs.

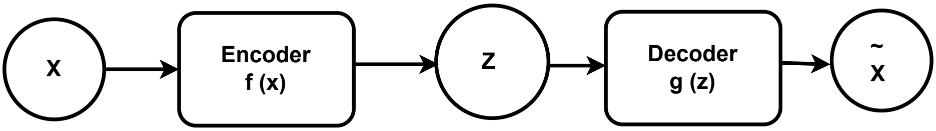


Figure 3.11: A schematic diagram of an under-complete Autoencoder. The Autoencoder receives some data **x**, which is encoded to produce a code **z**. The decoder takes this code to reproduce the original data, albeit with some deterioration.

and an inner product defined as:

$$\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{x}'). \tag{3.18}$$

For the transformed data points: $\{\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)\}^\top$, we first calculate their sample mean:

$$\bar{\phi} = \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i), \tag{3.19}$$

and the pairwise inner product, which is defined as:

$$\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}. \tag{3.20}$$

Then, by seeking a point: $u_1 \in \mathcal{H}$, that maximizes the variability of the data projections onto it, we identify the first PC as:

$$u_1 = \underset{\|u\|_{\mathcal{H}} \leq 1}{\text{argmax}} \frac{1}{N} \sum_{i=1}^{N} \langle u, \phi(\mathbf{x}_i) - \bar{\phi} \rangle_{\mathcal{H}}. \tag{3.21}$$

Without further derivations, we state that the solution to this problem is:

$$u_1 = \sum_{i=1}^{N} \alpha_i^{(1)} (\phi(\mathbf{x}_i) - \bar{\phi}), \tag{3.22}$$

where $\alpha_i^{(1)}$ is the eigenvector that corresponds to the largest eigenvalue of the matrix: **HKH** ($\mathbf{H} = \mathbf{I}_{N \times N} - N^{-1}\mathbf{1}_N$). As with linear PCA, we proceed to compute further components in the same way after removing the data projections onto the already-computed PCs. In essence, all PCs take the same form of u_1, and the coefficients thereof are determined by eigenvectors of **HKH**.

3.2.3 Autoencoders

Autoencoders (AEs) (Goodfellow et al., 2016; Hinton and Zemel, 1994) are a family of deep generative models, which approximate the data distribution by constructing a lower dimensional representation of the data points. An AE consists of two feed-forward neural networks (FFNNs) for the encoder and the decoder respectively. The encoder: $\mathbf{z} = f(\mathbf{x})$, takes the input $\mathbf{x}$, and produces a code $\mathbf{z}$ used to represent the input. The decoder takes this code, and produces a reconstruction: $\mathbf{x}' = g(\mathbf{z})$, of the original data. AEs are restricted in ways that allow them to learn the most salient features of the data. We employ the so-called "Under-complete Autoencoders" (UAEs) in our study, which are constrained to have smaller dimensions for $\mathbf{z}$, when compared with the original data $\mathbf{x}$. The learning process in UAEs focuses on minimizing a loss function: $\mathcal{L}(\mathbf{x}, g(f(\mathbf{x})))$, such as mean squared error (MSE). A graphical illustration of a standard AE is presented in Fig. 3.11.

3.2.4 Variational Autoencoders

Variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) are AEs, which provide a principled methodology for employing deep latent-variable models, and can be used to learn complex data distributions in a generative fashion. The data: $\mathbf{x} \in \mathbb{R}^D$, and the latent variables: $\mathbf{z} \in \mathbb{R}^L$, are jointly related by the chain rule:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}), \tag{3.23}$$

where $L < D$, and θ denote the associated set of parameters. More specifically, a VAE consists of two coupled but independently parameterized models: an inference (also called an encoder or recognition) model, and a generative (also called a decoder) model; both of which are implemented by non-linear functions, such as neural networks (Goodfellow et al., 2016; Kingma et al., 2019). The encoder encodes the input data $\mathbf{x}$ to the set of latent variables $\mathbf{z}$, and the decoder maps these latent variables $\mathbf{z}$ back to reproduce $\mathbf{x}$.

The VAE treats the conditional probability distribution $p_\theta(\mathbf{x}|\mathbf{z})$ as a function approximation of $\mathbf{x}$. However, the non-linear mapping from $\mathbf{z}$ to $\mathbf{x}$ can not be implemented directly because of the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$ on the latent-variable. The VAE thus introduces the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$, parameterized by ϕ to approximate the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The parameters for the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ are generated by the encoder network. Lastly, the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ of $p_\theta(\mathbf{z}|\mathbf{x})$ enables the use of Evidence Lower

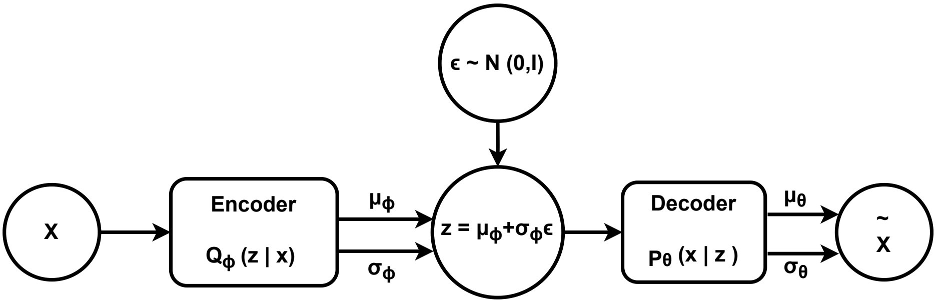


Figure 3.12: The schematic diagram of a vanilla Variational Autoencoder, where the code $\mathbf{z}$ is produced with the help of the reparameterization trick.

Bound (ELBO) as:

$$\log p_\theta(\mathbf{x}) \geq -\operatorname{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})], \tag{3.24}$$

where $\operatorname{KL}(Q||P)$ is the Kullback-Leibler divergence between two probability distributions Q and P. In the original work of Kingma and Welling (Kingma and Welling, 2014), the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is modeled by a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}^2))$, where the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the outputs of the inference network, and diag corresponds to the diagonal co-variance structure of the Gaussian distribution. The prior $p(\mathbf{z})$ is assumed to be a standard Gaussian distribution. The training process focuses on maximizing ELBO, which yields the optimal parameters for the inference and generative networks. A low variance estimator can be substituted with the help of the reparameterization trick: $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon$; where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a vector of standard Gaussian variables, and $\odot$ denotes the element-wise product:

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\log p_\theta(\mathbf{x}|\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon)]. \tag{3.25}$$

In summary, VAEs are AEs, which provide a principled framework to learn deep latent-variable models efficiently by combining the Variational Inference (VI) and the reparameterization trick. Due to this reason, they have become a very important tool for dimensionality reduction, lossy data compression, representation learning, and generative modeling. The graphical representation of a VAE is presented in Fig. 3.12.

3.2.5 Dimensionality Reduction

As previously discussed, high dimensionality poses major difficulties in the applicability of model-assisted optimization (Ullah et al., 2019). Consequently, we are interested in performing dimensionality reduction to efficiently construct the low dimensional surrogate models. To this end, however, we face an important question concerning the choice of the DRT. To answer this question, we conduct an empirical investigation, which emphasizes on some of the most important DRTs, such as PCA, KPCA, AEs, and VAEs. We are interested in evaluating the efficacy of the low dimensional representations of the data, obtained after performing dimensionality reduction by these techniques. The low dimensional surrogate models constructed from these representations/encodings can be represented as:

$$\hat{f}_l : \mathcal{T} \subseteq \mathbb{R}^L \to \mathbb{R}, \tag{3.26}$$

where $\mathcal{T}$ is the low dimensional feature space of the original search space.

Building the surrogate models in this lower dimensional space is beneficial for two main reasons. Firstly, we can alleviate the curse of dimensionality by controlling the size of the latent space. Secondly, with some tolerance, the low dimensional representations from the latent space contain enough information to reconstruct the original features in $\mathcal{S}$. Together, these rationales allows us to make a compromise on the quality of the surrogate model, and the computational budget (Roweis and Saul, 2000). Intuitively, if the size of the latent space is very close to the original search space, the quality of the LDSM is expected to be similar to the baseline surrogate with dimensionality D. On the other hand, if the size of the latent space is very small, i.e., $L \ll D$, the quality of the LDSM is expected to deteriorate, due to a considerable loss in information (Shan and Wang, 2010). Overall, we believe it is crucial to utilize the DRTs for constructing the surrogate models in high dimensional cases with limited computational resources.

3.2.6 Empirical Investigation

Through this empirical study (Ullah et al., 2020a), we aim to answer the following research questions.

- Which DRT is most suitable to efficiently construct the low dimensional surrogate models, appraised on the basis of criteria introduced earlier in this chapter?

- What is the impact of problem landscape on the performance of the low dimensional surrogate models?
- How does the size of the original search space affect the quality of the low dimensional surrogate models?
- How does the size of the latent feature space (obtained after the dimensionality reduction), affect the quality of the low dimensional surrogate models?
- Is the choice of the modeling technique, e.g., Kriging and Polynomials, an important factor in this context?

Answering these questions in a comprehensive manner is the aim of our study. In the following, we describe the experimental setup of our study, which describes the test problems, the data generating scheme, and HPO. A flowchart of the entire experimental setup is provided in Fig. 3.13 for further clarification.

3.2.7 Experimental Setup

For our study, we select ten unconstrained, noiseless, single-objective optimization problems from the continuous benchmark function test-bed, known as “Black-Box-Optimization-Benchmarking” (BBOB) (Hansen et al., 2021). Note that BBOB provides a total of twenty-four such functions divided in five different categories, namely “Separable Functions”, “Functions with low or moderate conditioning”, “Functions with high conditioning and unimodal”, “Multi-modal functions with adequate global structure”, and “Multi-modal functions with weak global structure” respectively. We select two functions from each of these categories to cover a broad spectrum of test cases. The set of selected test functions is given as: $\mathscr{F} = \{f_2, f_3, f_7, f_9, f_{10}, f_{13}, f_{15}, f_{16}, f_{20}, f_{24}\}$. An important thing to note is that each of the test functions in $\mathscr{F}$ is evaluated on three different settings of dimensionality as: $D = \{50, 100, 200\}$.

After the selection of the test cases, we move forward to the data generation, and pre-processing. For the purpose of data generation, the choice of the training sample size N is problem-dependent. The practical advice, however, is to begin with $N = K \times D$ (Forrester et al., 2008; Ullah et al., 2019), where K is usually a low valued scalar, that maps the dimensionality to the sample size. In this study, we proceed with $K = 20$. Choosing $K = 20$ is based on the previous empirical evidence (Ullah et al., 2019), as this results in a training data set of moderate size, which is neither too small to train, nor too big to hinder the computational

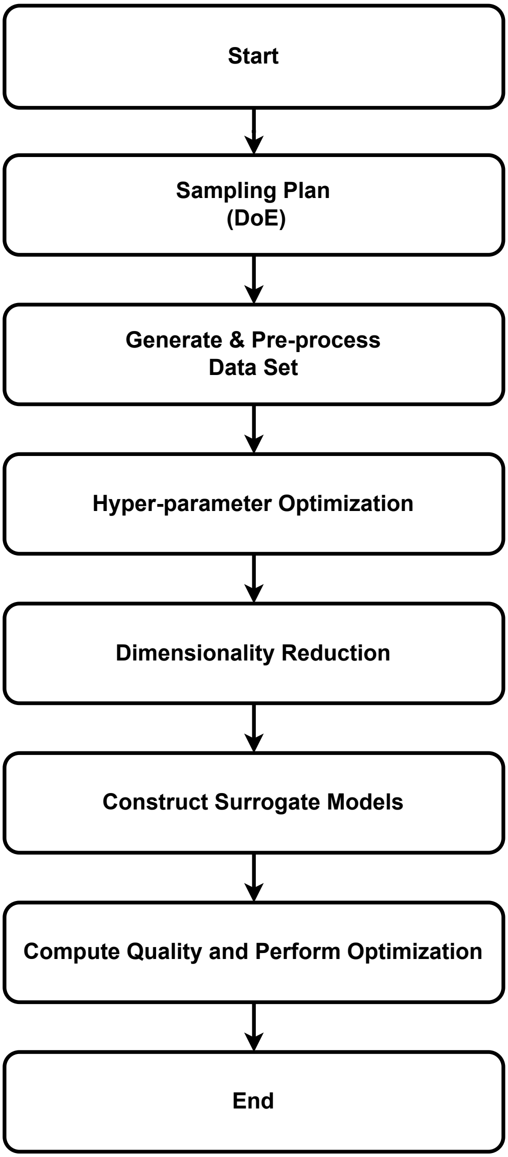

Start
Sampling Plan (DoE)
Generate & Pre-process Data Set
Hyper-parameter Optimization
Dimensionality Reduction
Construct Surrogate Models
Compute Quality and Perform Optimization
End

Figure 3.13: A schematic diagram of the experimental setup. Each step of the process is shown in grey rectangles. The central rectangle indicates the HPO loop based on the modeling accuracy of the surrogates.

efficiency. Additionally, the testing data set with size $M = 0.2K \times D$ is generated to evaluate the modeling accuracy of the LDSMs. Notably, we also make sure that the training and testing data sets are completely disjoint, i.e., no data point is shared between the two sets. The sampling locations for both data sets are chosen using a LHS scheme (Gramacy, 2020). The data pre-processing in this study is a rather straightforward task, involving only the re-scaling of the features between 0 and 1 (ur Rehman et al., 2014).

We consider four DRTs in our study, which have been explained earlier. For each of these techniques, specifying L – the size of the latent feature space – is crucial, since it may affect the quality of the corresponding LDSM (Kingma et al., 2019). Therefore, for each distinct setting of the original dimensionality D, we choose three corresponding values for L as: $L \in \{0.7, 0.4, 0.1\} \times D$. As an example, $L \in \{35, 20, 5\}$ when $D = 50$. An important thing to note is that in AEs and VAEs, both, the encoder and the decoder, have four hidden layers each with hyperbolic tangent non-linearity (Sharma et al., 2017).

For PCA and KPCA, we perform a linear transformation of the original features, before performing the dimensionality reduction (Ullah et al., 2020a). In our study, we only consider two (surrogate) modeling techniques, namely the Kriging, and Polynomial Regression (degree=2 with elastic-net penalty) (Zou and Hastie, 2005). Notably, both sets of techniques, i.e., the dimensionality reduction and the surrogate modeling techniques, have some hyper-parameters. Therefore, it is crucial to tune these hyper-parameters to get the best quality surrogate models (Hutter et al., 2009).

At this stage, however, we have a total of 720 experiments, due to four DRTs, two (surrogate) modeling techniques, ten test functions, three values of the original dimensionality D, and three different values for L – the size of the latent feature space. We deem performing HPO for each of these 720 cases infeasible. Hence, we reduce the number of cases to a total of 72, by aggregating the performance of the LDSMs on all ten test functions. This implies that we optimize the hyper-parameters for each of the 72 cases, defined on the combinations of three values of the original dimensionality D, three values of L, two surrogate modeling techniques, and four DRTs. In each of these 72 cases, we optimize the hyper-parameters for both, the dimensionality reduction and the surrogate modeling techniques together, based on the aggregated quality of the corresponding LDSMs on all ten test functions. The quality assessment for an individual LDSM, i.e., for a particular test function such as $f2$, is measured by taking the RMAE (cf. Eq. (3.11)).

For HPO, we measure this RMAE for all ten test functions by specifying D, L, the dimensionality reduction, and the (surrogate) modeling technique. After this, we take the median of the RMAE values on all ten test functions. The goal of the HPO then becomes to find the best configuration of the hyper-parameters, which minimizes this median. This process is repeated for all 72 cases. Overall, this approach makes the HPO feasible, and ensures that the configuration of the hyper-parameters generalizes well across all ten test functions (Ullah et al., 2020a).

We employ the TPE (Bergstra et al., 2011, 2013b) algorithm to perform the HPO for each of the 72 cases discussed above by specifying D, L, the dimensionality reduction, and the surrogate modeling technique. The number of function evaluations are restricted to 150 for finding the best configuration of the hyper-parameters using TPE, as the maximum number of hyper-parameters in any of the 72 cases is six.

In our study, we choose two criteria to evaluate and compare the LDSMs. The first criterion is that of the modeling accuracy. To compare the LDSMs on this criterion, we first construct the LDSMs in all 720 cases, after performing the HPO. This implies that we construct and compare four LDSMs, for each distinct value of D, L, the surrogate modeling technique, and the test function. These four LDSMs are based on PCA, KPCA, AEs, and VAEs respectively. Note that in this context, the LDSMs which share the same test function, will also share the same configuration of the hyper-parameters as an implication of the HPO procedure discussed before. Since we vary the (surrogate) modeling technique, the test function, and the values for D and L, we can perform a comprehensive analysis of the modeling accuracy of the LDSMs based on a particular DRT. We employ RMAE (cf. Eq. (3.11), as the performance measure for this criterion (Ullah et al., 2019).

The second criterion to compare the LDSMs is the quality of the optimal solutions. To compare the LDSMs for this criterion, we proceed with the same setup as before. This implies that we construct the LDSM in each of the 720 cases, based on the best configuration of the hyper-parameters, and employ the corresponding LDSM to substitute the exact function evaluations within the optimization loop of the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm (Morales and Nocedal, 2011) for global optimization. To this end, maximum function evaluations are restricted to be under $1000 \times D$. We perform a total of 30 independent runs of the L-BFGS algorithm, for each LDSM, by varying the starting position, i.e., the initial guess (Wright et al., 1999).

After this, we evaluate and compare the difference in the quality $\mathcal{DQ}$ of the optimal solutions based on two aspects, i.e., the difference in the objective function values, and the difference in the search space (cf. Eq. (3.12)). Note that the difference in the quality $\mathcal{DQ}$ in this context is measured by taking the median, rather than the average, of the 30 independent runs of the L-BFGS algorithm (Ullah et al., 2020a).

3.2.8 Results

We first share the results concerning the criterion of the modeling accuracy. For this, we share the graphs illustrating the modeling accuracy of the LDSMs, based on the two modeling techniques chosen – Kriging and Polynomial Regression, in Figs. 3.14 and 3.15 respectively. Both of these figures contain a total of nine subplots each, based on three distinct values of D and L. Each of these subplots contains ten bar charts, corresponding to the ten test functions discussed. Furthermore, each bar chart shares the RMAE values, for the LDSMs based on the DRTs involved in our study.

Next, we report the results concerning the difference in the quality $\mathcal{DQ}$ of the optimal solutions. For this, we first report $\mathcal{DQ}$ for all 720 test cases, measured on the basis of the objective function values, and presented in Figs. 3.16 and 3.17. As opposed to this, $\mathcal{DQ}$ (measured on the basis of the difference in the search space), is shared in Figs. 3.18 and 3.19.

In the following, we report the major findings from these results.

- **Dimensionality Reduction Technique**

  From the results concerning the criterion of the modeling accuracy, we find the AEs as the most promising modeling technique. This is due to the fact that in 132/720 test scenarios, the LDSMs obtained from the latent feature space of the AEs, achieve the highest modeling accuracy. In most of the remaining cases, the performance of the LDSMs is analogous. On the other hand, appraising the quality of the LDSMs based on $\mathcal{DQ}$, we find the AEs perform poorly as opposed to the other competing DRTs in most cases. In this context, we deem that PCA and KPCA perform excellently. An interesting observation here is that the KPCA does not perform superior to PCA in most cases, which is computationally less expensive to evaluate.

- **Problem Landscape**

  Based on the results, we observe the functions $f2$, $f9$, $f10$, and $f20$ to be extremely challenging to approximate and optimize. Out of these four test functions, we observe the worst performance on $f2$ and $f10$, which are ill-conditioned functions with smooth local irregularities (Hansen et al., 2021). The other two test functions, namely $f9$ and $f20$, exhibit highly multi-modal and rugged structure in high dimensions (Merkuryeva and Bolshakovs, 2011), and are consequently difficult to model and optimize with dimensionality reduction (loss of information). An important thing to note here is that we see significant variation in the performance of the LDSMs for the functions belonging to the same class. For instance, both $f2$ and $f3$ belong to the class of "separable functions", and yet exhibit a significant difference in the quality.

- **Size of the Search Space**

  We observe that the quality of the approximation of the LDSMs is improved with a higher dimensionality. We believe this might be due to the fact that our sample size is more than sufficient to maintain the modeling accuracy, as the dimensionality increases, i.e., the true sample size required to maintain the modeling accuracy may not be linear, but rather a fraction of the linear sample size.

- **Size of the Latent (Feature) Space**

  For an extremely small size of the latent feature space, i.e., $\mathrm{L} = 0.1 \times D$, we observe a significant deterioration in the quality of the LDSMs in most cases (measured on the basis of $\mathcal{DQ}$) (cf. Fig. 3.18 and Fig. 3.19).

- **Modeling Technique**

  We observe that in terms of $\mathcal{DQ}$, the performance of the LDSMs (based on Kriging) is sensitive.

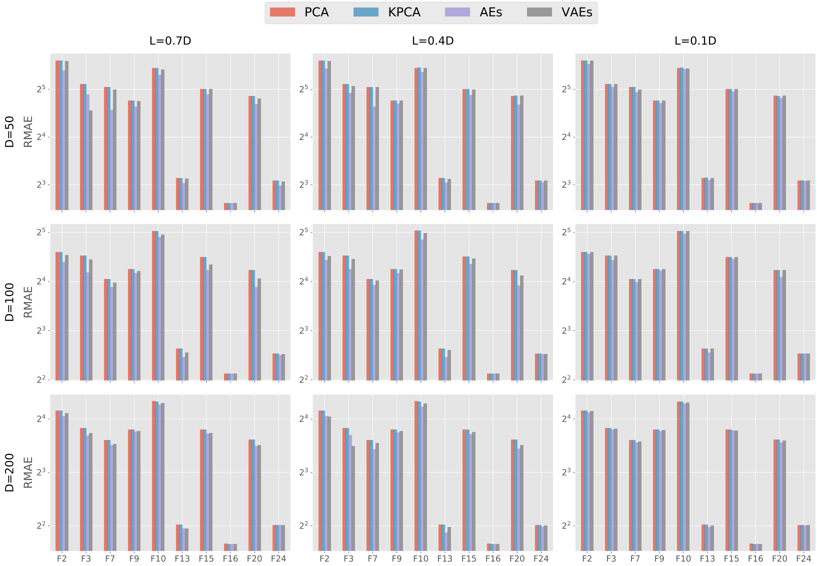


Figure 3.14: Modeling accuracy of the low dimensional Kriging surrogates. The accuracy is measured with Relative Mean Absolute Error (lower is better).

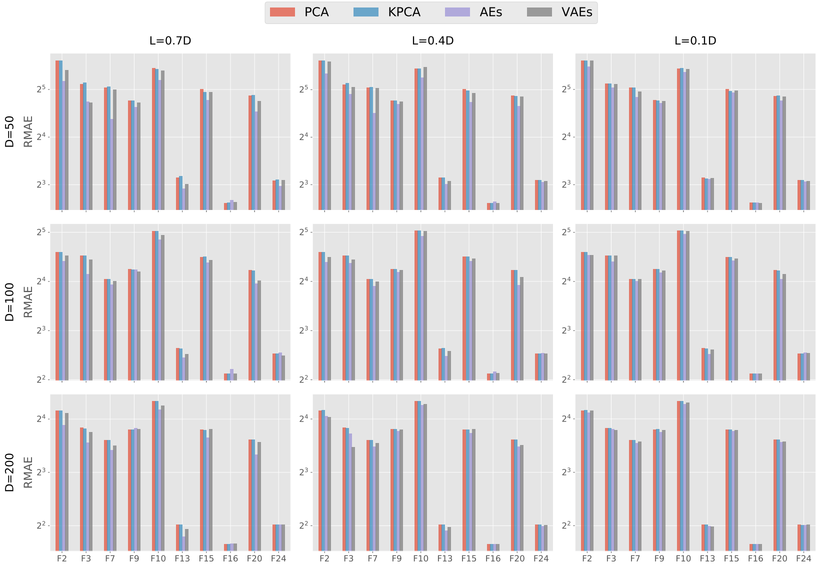


Figure 3.15: Modeling accuracy of the low dimensional Polynomial surrogates. The accuracy is measured with Relative Mean Absolute Error (lower is better).

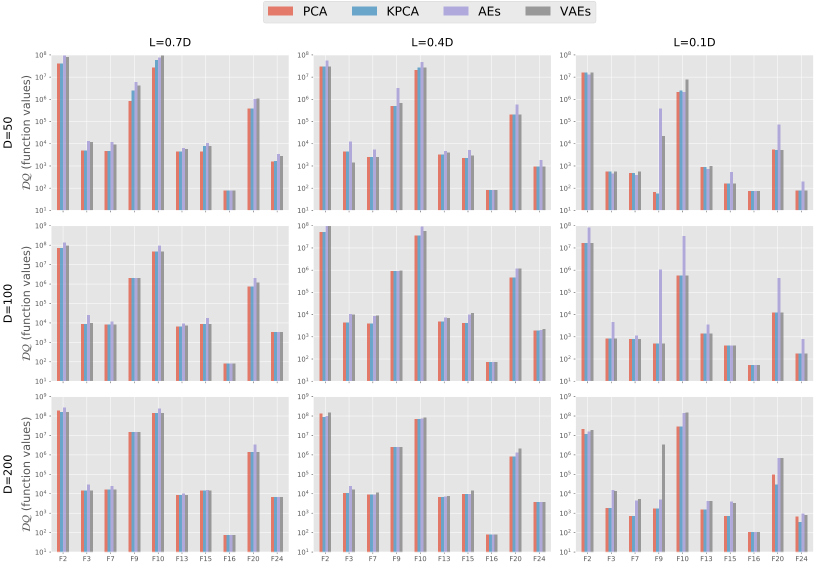


Figure 3.16: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Kriging surrogates, and the corresponding baseline. The difference in the quality is based on the objective function values.

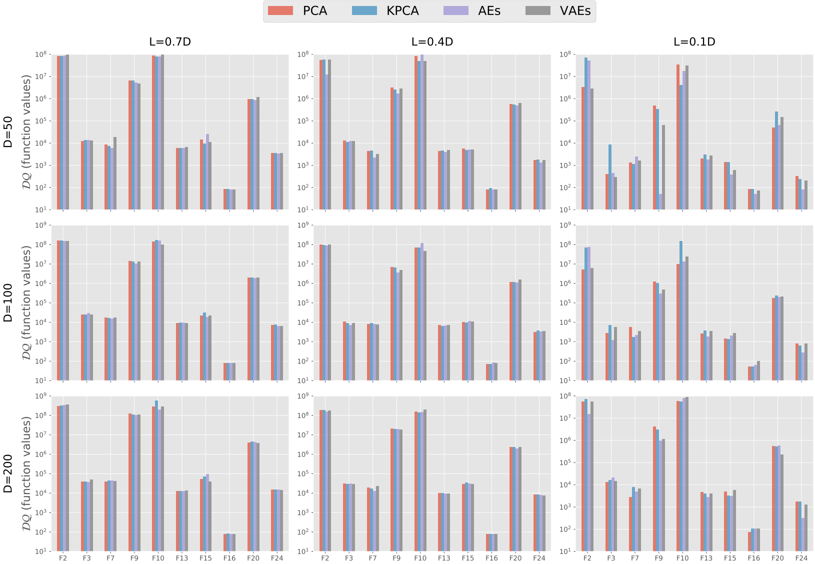


Figure 3.17: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Polynomial surrogates, and the corresponding baseline. The difference in the quality is based on the objective function values.

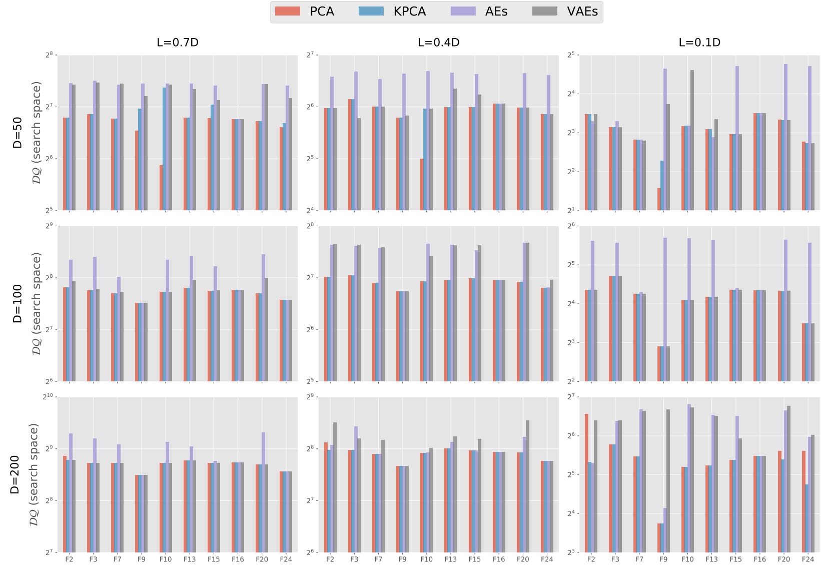


Figure 3.18: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Kriging surrogates, and the corresponding baseline. The difference in the quality is based on the distance in the search space.

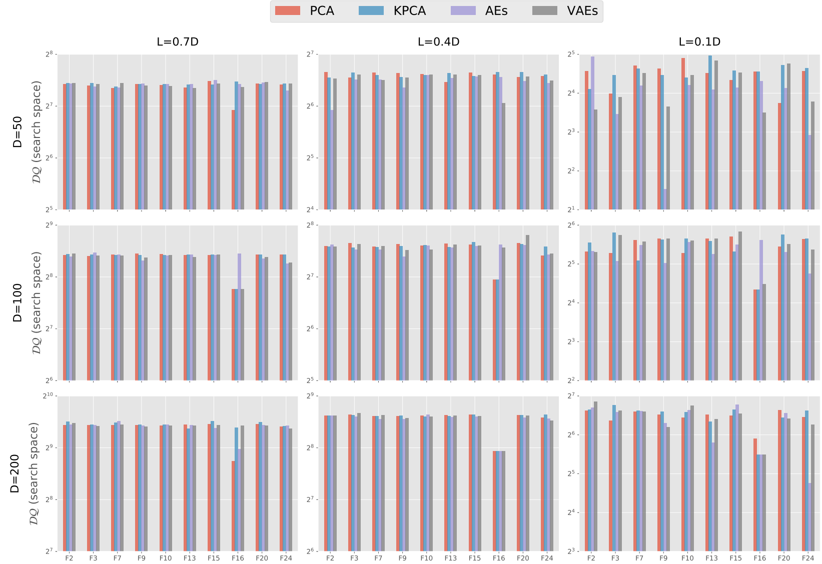


Figure 3.19: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Polynomial surrogates, and the corresponding baseline. The difference in the quality is based on the distance in the search space.

3.3 Summary and Discussion

In this chapter, we focused on the applicability of surrogate modeling to find robust solutions, which are solutions that are still optimal and useful in the face of uncertainty and noise. Note that the uncertainty and noise in this context refer to the deviation in the nominal values of the decision variables. Our aim is to find such solutions in an efficient manner, for which we employ an optimization framework based on surrogate modeling.

To answer the key research question outlined earlier in the chapter, we perform two empirical investigations. Through our investigations, we find that in general, a linear sample size[1] is sufficient to construct a good quality surrogate model based on Kriging, Polynomials, and Support Vector Machines. These modeling techniques are promising, and provide a good quality approximation in most test scenarios considered. Furthermore, we observe that noise level does not have a detrimental impact on the quality of the surrogate models. Note, however, that it is quite possible that noise level can affect the quality of solutions in other optimization frameworks, e.g., the SMBO approach where noise can propagate through the acquisition function. Our results also indicate that dimensionality can have a significant impact on the quality of the solution, especially for "composite robustness". Similarly, we find that "composite robustness" has a higher variation/sensitivity in performance. Overall, we conclude that surrogate modeling is a viable approach to find robust solutions for black-box optimization problems.

In practical scenarios of continuous optimization, problems may also exhibit the issue of high dimensionality, which can hinder the computational efficiency. This issue is based on the so-called "curse of dimensionality", which refers to the fact that constructing a surrogate model becomes much costlier as the dimensionality increases. This is due to the fact that more training data is needed to construct the surrogate model, if the dimensionality increases. The algorithmic complexity for constructing the surrogate model depends on the training data, and the number of search variables, among others.

To address the issue of high dimensionality, we propose to perform dimensionality reduction. For choosing the most suitable dimensionality reduction in this context, we empirically evaluate and compare four of the most important DRTs, namely

---

[1] Linearity is defined in terms of the dimensionality of the problem.

PCA, KPCA, AEs, and VAEs. The results from our study indicate the promising aspects of AEs and PCA to perform dimensionality reduction in surrogate modeling. Note that the LDSMs based on AEs provide good approximation of the search space, whereas the ones based on PCA find better solutions as opposed to their competitors. From these results, we also observe that Kriging is more sensitive to performance variation in high dimensions than RSMs.

CHAPTER 4

Robust Bayesian Optimization

This chapter is devoted to the applicability of the so-called "Bayesian optimization" algorithm (Močkus, 2012; Jones et al., 1998) to efficiently solve expensive to evaluate blck-box problems, which are subject to uncertainty and noise in the search variables. The Bayesian optimization algorithm is based on the so-called "sequential model-based optimization" approach, which updates the surrogate model in an iterative manner, in order to find a globally optimal solution on the model surface (cf. Section 2.3.4). We consider the scenario of finding robust solutions via Bayesian optimization. Note that in this context, the standard (nominal) Bayesian optimization algorithm cannot be utilized directly, and must be extended to care for robustness, in order to find robust solutions (Rehman, 2016; Ullah et al., 2021). Pertaining to find robust solutions via Bayesian optimization, we attempt to answer the following questions in this chapter.

1. How can we extend the Bayesian optimization algorithm to find robust solutions: solutions which are still optimal and useful in the face of parametric uncertainties in the search variables?
2. What is the performance of the Bayesian optimization algorithm in this context, and which factors[1] influence its performance?

In Section 4.1, we introduce the Bayesian optimization algorithm, which is followed by three of the most important sampling infill criteria considered in this chapter: the so-called "Lower Confidence Bound", "Expected Improvement" criterion, and the "Moment-Generating Function of the Improvement" (Wang et al., 2017). Following this, we extend the Bayesian optimization algorithm to care for parametric uncertainties in the search variables. Note that this section also describes the practical difficulties and potential pitfalls for extending the Bayesian optimization algorithm to the robust scenario (ur Rehman et al., 2014; Jurecka, 2007). We then move forward to benchmark the empirical performance of the Bayesian optimization algorithm to find robust solutions. Lastly, we provide the discussion on the empirical results and summary of the chapter.

---

[1] Note that the factors considered in this context include scale/severity of the uncertainty, dimensionality, sampling infill criterion, and computational budget among others.

4.1 Bayesian Optimization

Bayesian optimization (BO) is a global search strategy, designed to optimize expensive to evaluate black-box problems in an efficient manner (Močkus, 2012; Jones et al., 1998). The basic idea behind BO is to treat the objective function as a random function, and place a prior over it (Močkus, 1975). The prior information captures our beliefs about the anticipated behavior of the function, e.g., smoothness. After observing the function response at well-specified sampling locations, the prior is updated to form the posterior distribution over the objective function (Močkus, 1975). The posterior distribution, in turn, is used to construct a utility function, which determines the next query point (where the function response is to be observed). Note that the utility function, also referred to as the acquisition function (AF) or the sampling infill criterion (SIC), quantifies the potential "gain" in the objective value, by evaluating the potential of each new solution (Liu et al., 2012). It therefore selects the next query point which maximizes this gain. Once the next query point is determined, the function response is observed at that location, and the posterior distribution is updated (Frazier, 2018).

The BO algorithm is based on the SMBO approach, which is already described in Chapter 2 (cf. Fig. 2.3). The main points of the BO algorithm are summarized as follows. We start by generating an initial design data set: $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$, on the objective function f. The next step involves constructing the Kriging model $\mathcal{K}_f$, based on the available data set $\mathcal{D}$. Once the Kriging model is constructed, we can utilize the strategy of adaptive sampling (based on the AF), to estimate the global optimum of the objective function f (ur Rehman et al., 2014).

The AF is constructed by assuming that the function response at any untried position $\mathbf{x}$ can be modeled in terms of a normally distributed random variable $Y(\mathbf{x})$, whose mean is given by the predicted value: $\hat{f}(\mathbf{x})$, and the variance is given by the MSE: $s^2(\mathbf{x})$ (as described in Eq. (2.11)) (Rasmussen and Williams, 2006; Woodard, 2000).

The potential improvement to query the position $\mathbf{x}$ with respect to the best-so-far observed valued of the function: $f_{\min}$, can be described as:

$$\mathcal{I}(\mathbf{x}) = \max\{0, f_{\min} - Y(\mathbf{x})\}. \tag{4.1}$$

The utility function of the improvement is denoted as $\mathscr{A}$, and can be employed to find the next query point $\mathbf{x}_{\text{new}}$:

$$\mathbf{x}_{\text{new}} = \underset{\mathbf{x}\in\mathcal{S}}{\operatorname{argmax}} \ \mathscr{A}(\mathbf{x}). \tag{4.2}$$

Once the next query point is determined, the data set $\mathcal{D}$ is extended by appending the pair $(\mathbf{x}_{\text{new}}, f(\mathbf{x}_{\text{new}}))$ to it (Jones et al., 1998). The Kriging model $\mathcal{K}_f$[1] is then reconstructed based on the extended data set. This process is repeated until either a satisfactory solution is obtained, or a predefined computational budget, or other termination criterion is reached. Since at each iteration, the next query point $\mathbf{x}_{\text{new}}$ brings the maximum anticipated improvement to the current solution according to the chosen infill criterion, the algorithm can find the optimal solution in an efficient manner (Wang, 2018).

4.1.1 Sampling Infill Criteria

When employing the surrogate model to perform optimization, it is important to determine how the search should be balanced with respect to exploration and exploitation (Snoek et al., 2012). To this end, we can introduce the notion of “gain” to assess the potential of untried points, i.e., to assess the potential improvement with respect to the current best known solution. Since in BO, the surrogate model is stochastic in nature, the resulting “gain” function also becomes stochastic (Wang, 2018). Consequently, it is important to use some statistical properties, e.g., the expectation, of the this function to assess the potential of untried locations. Utilizing such a function, we can determine the location of the next query point (to observe the function response).

---

[1] While other modeling techniques, e.g., Random Forest, Support Vector Machines, can also be employed, the theoretical quantification of the uncertainty in the Kriging prediction makes it an ideal candidate in this context (cf. Eq. (2.11)). Furthermore, Kriging arises naturally in the context of non-parametric Bayesian inference, and therefore has a natural Bayesian interpretation (Rasmussen and Williams, 2006; Wang, 2018).

In the literature, several different types of AFs exist, each with its own merits and demerits (Hoffman et al., 2014). Examples of some of the most important AFs, based on the notion of "improvement", include "Expected Improvement" criterion (Jones et al., 1998), "Bootstrapped Expected Improvement" criterion (Kleijnen et al., 2012), "Probability of Improvement" (Žilinskas, 1992), "Weighted Expected Improvement" (Sóbester et al., 2005), "Generalized Expected Improvement", and "Multiple Generalized Expected Improvement" (Ponweiser et al., 2008) among others. This chapter, however, only concentrates on "Upper Confidence Bound", "Expected Improvement" criterion, and the "Moment-Generating Function of the Improvement", to find robust solutions in an efficient manner.

The "Upper Confidence Bound" (Srinivas et al., 2010; Parr et al., 2010), also referred to as the "Lower Confidence Bound" (LCB) in the case of minimization, is defined as:

$$\text{LCB}\,(\mathbf{x};\beta) = \hat{f}(\mathbf{x}) - \sqrt{\beta s^2(\mathbf{x})}, \tag{4.3}$$

where β is a carefully chosen learning rate, which explicitly controls the trade-off between exploitation and exploration (Auer, 2002). Note that a high setting of β concentrates more on model uncertainty ($s^2(\mathbf{x})$), and thus performs exploration (Bubeck et al., 2009).

"Expected Improvement" (EI) criterion is a widely utilized sampling infill criterion in BO (Močkus, 2012; Jones et al., 1998). This infill criterion is based on the first moment, i.e., the expectation, of the improvement. In the context of Gaussian processes (where the Kriging response can be represented as a Gaussian random variable: $Y(\mathbf{x}) \sim (\hat{f}(\mathbf{x}), s^2(\mathbf{x}))$), the expectation of the improvement has a closed form expression[1]:

$$\mathbb{E}[\mathcal{I}(\mathbf{x})] = (f_{\min} - \hat{f}(\mathbf{x}))\Phi\left(\frac{f_{\min} - \hat{f}(\mathbf{x})}{s}\right) + s\phi\left(\frac{f_{\min} - \hat{f}(\mathbf{x})}{s}\right), \tag{4.4}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are cumulative distribution function and probability density function of the standard normal random variable respectively.

"Moment-Generating Function of the Improvement" (MGFI) (Wang et al., 2017) is another important infill criterion discussed in this chapter, where all the moments of the improvement are linearly combined. MGFI is based on the intuition of utilizing the higher moments of the improvement, and can be thought of as an alternative way of defining its probability distribution.

---

[1] Recall that the potential "gain" or "improvement", which is defined with respect to the best known function value: $f_{\min}$ (cf. Eq. (4.1)), is a stochastic process over the search space $\mathcal{S}$, as it depends on the stochastic modeling of the function value.

Formally speaking, MGFI can be defined as:

$$\forall t \in \mathbb{R}, \quad \mathcal{M}(\mathbf{x}, t) := \mathbb{E}[e^{t\mathcal{I}(\mathbf{x})}] = \int_{-\infty}^{\infty} e^{tu}\; PI(u; \mathbf{x})\; du, \tag{4.5}$$

where $u = (f_{\min} - \hat{f}(\mathbf{x}))/s$, PI indicates the probability density function of the improvement, and t is a real-valued parameter which controls the behavior of the search, i.e., balances the trade-off of exploration and exploitation. Note that in this context, the parameter t is referred to as the "temperature", similar to the simulated annealing algorithm (Kirkpatrick et al., 1983), and can be updated for each iteration of the BO algorithm based on a "linear" or an "exponential" cooling strategy (Wang et al., 2018).

The MGFI can also be calculated using the density function of $\mathcal{I}(\mathbf{x})$ as:

$$\mathcal{M}(\mathbf{x}, t) = 1 + \Phi\left(\frac{f_{\min} - \hat{f}'(\mathbf{x})}{s}\right) \exp\left((f_{\min} - \hat{f})t + \frac{s^2 t^2}{2}\right) - \Phi\left(\frac{f_{\min} - \hat{f}}{s}\right), \tag{4.6}$$

where $\hat{f}' = \hat{f} - s^2 t$, and MGFI is well-defined for all $t \in \mathbb{R}$ in this context.

From a different perspective, the Taylor expansion of the MGFI is:

$$\mathcal{M}(\mathbf{x}, t) = 1 + t\mathbb{E}[\mathcal{I}(\mathbf{x})] + \frac{t^2}{2!}\mathbb{E}[\mathcal{I}^2(\mathbf{x})] + \frac{t^3}{3!}\mathbb{E}[\mathcal{I}^3(\mathbf{x})] + \cdots = \sum_{n=0}^{\infty} \frac{t^n}{n!}\mathbb{E}[\mathcal{I}^n(\mathbf{x})]. \tag{4.7}$$

As Wang notes (Wang, 2018), for an arbitrary distribution, this series might not converge for all $t \in \mathbb{R}$, even if all the moments exist. The functional form in Eq. (4.7) can be considered a linear combination of all the moments, where each moment $\mathbb{E}[\mathcal{I}^n(\mathbf{x})]$ is weighted by $\frac{t^n}{n!}$. In this context, the weight of each moment can be controlled with parameter t. Note that these weights can also be normalized, since $\sum_{n=0}^{\infty} \frac{t^n}{n!} = e^t$. Normalizing the weights in this manner leads to the convergence for all $t \in \mathbb{R}$.

Finally, by incorporating the probability of improvement $P\mathcal{I}(\mathbf{x})$ as the "zero-order" moment, and replacing the constant 1 by it in Eq. (4.6), we have:

$$\begin{aligned}\mathcal{M}(\mathbf{x};t) &= \frac{\mathcal{M}(\mathbf{x},t) - 1 + P\mathcal{I}(\mathbf{x})}{e^t} \\ &= P\mathcal{I}(\mathbf{x}) + \frac{t}{e^t}\mathbb{E}[\mathcal{I}(\mathbf{x})] + \frac{t^2}{2!e^t}\mathbb{E}[\mathcal{I}^2(\mathbf{x})] + \frac{t^3}{3!e^t}\mathbb{E}[\mathcal{I}^3(\mathbf{x})] + \dots \\ &= \Phi\left(\frac{f_{\min} - \hat{f}^{'}(\mathbf{x})}{s}\right)\exp\left((f_{\min} - \hat{f} - 1)t + \frac{s^2t^2}{2}\right).\end{aligned} \tag{4.8}$$

4.2 Robustness in Bayesian Optimization

In the previous chapter, we defined five of the most common robustness criteria (cf. Section 3.1.1), which can be employed to achieve robustness in practical scenarios. When aiming to find a robust solution based on these robustness criteria, we note that the standard BO algorithm cannot be utilized.

As Rehman notes (ur Rehman et al., 2014), there are two main reasons for that.

- The potential "improvement", which is defined in the nominal scenario (cf. Eq. (4.1)), renders inapplicable in the context of RO. This is due to the fact that this improvement is defined with respect to the "best-so-far" observed value of the function: $f_{\min}$, which has no clear meaning and usage when aiming for a robust solution. Rather, in the case of RO, the improvement must be defined with respect to the current best known "robust" value of the function: $\hat{f}^*(\mathbf{x})$, which by implication can only be estimated on the Kriging surface (as opposed to observed or fully known in the nominal case).
- The posterior process: $Y(\mathbf{x}) \sim \mathcal{N}(\hat{f}(\mathbf{x}), s^2(\mathbf{x}))$, does not model the robust (effective) response of the function[1], which is desirable when aiming for a robust solution.

Therefore, the standard BO approach must be extended to the robust scenario, which is henceforth referred to as "Robust Bayesian optimization" (RBO) in this thesis. Following the approach of Rehman (ur Rehman et al., 2014), the adaptation of the BO algorithm to RBO is done in the following manner.

---

[1] The robust or effective function response has already been defined in Section 3.1.1 for five of the most common robustness criteria.

- We substitute the "best-so-far" observed value of the function: $f_{\min}$, with its robust Kriging counterpart: $\hat{f}^*(\mathbf{x})$, which is defined as:

$$\hat{f}^*(\mathbf{x}) = \min_{\mathbf{x} \in \mathcal{S}} \quad \hat{f}_{\text{eff}}(\mathbf{x}), \tag{4.9}$$

  where $\hat{f}_{\text{eff}}(\mathbf{x})$ is the robust (effective) Kriging response of the function, which depends on the robustness criterion chosen. Note that $\hat{f}_{\text{eff}}(\mathbf{x})$ is the approximation of the true robust response of the function: $f_{\text{eff}}(\mathbf{x})$. In the context of deterministic uncertainty: MMR and MMRR, this estimation merely refers to the substitution of true function responses with their Kriging predictions in Eqs. (3.1) – (3.3). On the other hand, in the context of probabilistic uncertainty: EBR, DBR, and CR, it also encompasses the monte-carlo approximations for the corresponding statistical quantities of interests, e.g., in Eq. (3.4), $\hat{f}_{\text{eff}}(\mathbf{x})$ is approximated with monte-carlo samples based on the Kriging prediction at each search point $\mathbf{x} + \Delta_{\mathbf{x}}$.

- We extend the nominal posterior process: $Y(\mathbf{x}) \sim \mathcal{N}(\hat{f}(\mathbf{x}), s^2(\mathbf{x}))$ to model the true robust response of the function: $f_{\text{eff}}(\mathbf{x})$, by assuming that the true robust response of the function at each search point is also normally distributed with mean $\hat{f}_{\text{eff}}(\mathbf{x})$ and variance $s^2_{\text{eff}}(\mathbf{x})$: $Y_{\text{eff}}(\mathbf{x}) \sim \mathcal{N}(\hat{f}_{\text{eff}}(\mathbf{x}), s^2_{\text{eff}}(\mathbf{x}))$. Note that the assumption that $Y_{\text{eff}}(\mathbf{x})$ is normally distributed is not entirely rigorous, but rather a practical compromise (ur Rehman et al., 2014). Ideally, we should have attempted to estimate the true posterior distribution of the robust Kriging response of the function: $\hat{f}_{\text{eff}}(\mathbf{x})$, which would require additional assumptions on the joint distribution of all search points. However, the computational costs of finding this generally non-Gaussian distribution several times on the original (nominal) Kriging surface $\mathcal{K}_f$ are prohibitively high. Additionally, numerically computing the integral for the expectation of the improvement for this generally non-Gaussian distribution would also be computationally expensive. To add to that, we note that the Kriging surface $\mathcal{K}_f$ only ever provides an approximation, and hence the true distribution of the robust response of the function for each robustness criterion can never be described with certainty in BO.

Modeling the true robust response of the function with a normally distributed random variable: $Y_{\text{eff}}(\mathbf{x})$, we note that in the context of deterministic uncertainty, the value $s^2_{\text{eff}}(\mathbf{x})$ merely refers to the Kriging MSE at point $\mathbf{x} + \Delta^*_{\mathbf{x}}$, where $\Delta^*_{\mathbf{x}}$

indicates the worst setting of the uncertainty, i.e., which maximizes Eq. (3.1) or (3.3), as the case may be.

In the context of EBR, $s^2_{\text{eff}}(\mathbf{x})$ has a closed form expression as:

$$s^2_{\text{eff}} = \frac{1}{J^2}\sum_{i,j}^{J}\mathcal{C}, \tag{4.10}$$

where $\mathcal{C}$ is a co-variance matrix with elements $C(\mathbf{x}_i^{'}, \mathbf{x}_j^{'})$. The entries $C(\mathbf{x}_i^{'}, \mathbf{x}_j^{'})$ in the matrix $\mathcal{C}$ are computed with the help of posterior Kernel (with optimized hyper-parameters), and the point $\mathbf{x}_j^{'}$ is defined as: $\mathbf{x}_j^{'} = \mathbf{x}+\Delta_{\mathbf{x}}^{j}$, where $\Delta_{\mathbf{x}}^{j}$ indicates the j-th sample for $\Delta_{\mathbf{x}}$. In the context of DBR and CR, $s^2_{\text{eff}}(\mathbf{x})$ does not have a closed form expression, and should be computed numerically.

After substituting the "best-so-far" observed value of the function: $f_{\min}$, with its robust Kriging counterpart: $\hat{f}^{*}(\mathbf{x})$, and modeling the true robust response of the function with a normally distributed random variable: $Y_{\text{eff}}(\mathbf{x}) \sim \mathcal{N}(\hat{f}_{\text{eff}}(\mathbf{x}), s^2_{\text{eff}}(\mathbf{x}))$, we can define the improvement in the robust scenario as:

$$\mathcal{I}_{\text{eff}}(\mathbf{x}) = \max\{0, \hat{f}^{*}(\mathbf{x}) - Y_{\text{eff}}(\mathbf{x})\}, \tag{4.11}$$

In the following, we extend the LCB, EIC, and MGFI to the robust scenario based on the improvement in Eq. (4.11).

4.2.1 Robust Infill Criteria

The adaptation of the LCB to the robust scenario is referred to as LCB_{eff}, and can be formulated to be:

$$\text{LCB}_{\text{eff}}\ (\mathbf{x}; \beta) = \hat{f}_{\text{eff}}(\mathbf{x}) - \sqrt{\beta s^2_{\text{eff}}(\mathbf{x})}, \tag{4.12}$$

where $\hat{f}_{\text{eff}}(\mathbf{x})$ and $s^2_{\text{eff}}(\mathbf{x})$ describe the robust Kriging response of the function, and the uncertainty therein. An important thing to note here is that the search point induced by the uncertainty: $\mathbf{x} + \Delta_{\mathbf{x}}$, can become infeasible with respect to the original search space $\mathcal{S}$, if $\mathbf{x}$ is already close to the boundary of $\mathcal{S}$ (Ullah et al., 2021). In this case, we simply clip the infeasible point with the boundary it breaks, similar to the approach of Rehman (ur Rehman et al., 2014).

Like LCB, the adaptation of the EI criterion to the robust scenario can be written as:

$$\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})] := (\hat{f}^*(\mathbf{x}) - \hat{f}_{\text{eff}}(\mathbf{x}))\Phi\left(\frac{\hat{f}^*(\mathbf{x}) - \hat{f}_{\text{eff}}(\mathbf{x})}{s_{\text{eff}}(\mathbf{x})}\right) + s_{\text{eff}}(\mathbf{x})\phi\left(\frac{\hat{f}^*(\mathbf{x}) - \hat{f}_{\text{eff}}(\mathbf{x})}{s_{\text{eff}}(\mathbf{x})}\right), \tag{4.13}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ in Eq. (4.13) represent the cumulative distribution function and probability density function of the standard normal random variable respectively.

Lastly, the MGFI is extended to the robust scenario as (Ullah et al., 2021):

$$\mathcal{M}_{\text{eff}}(\mathbf{x};t) = \Phi\left(\frac{\hat{f}^*(\mathbf{x}) - \hat{f}''(\mathbf{x})}{s_{\text{eff}}}\right)\exp\left((\hat{f}^*(\mathbf{x}) - \hat{f}_{\text{eff}}(\mathbf{x}) - 1)t + \frac{s_{\text{eff}}^2 t^2}{2}\right), \tag{4.14}$$

where $\hat{f}''(\mathbf{x}) = \hat{f}_{\text{eff}}(\mathbf{x}) - s_{\text{eff}}^2 t$, and $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal random variable, same as above.

Algorithm 1: Robust Bayesian Optimization

```
procedure $(f, \mathcal{S}, \mathscr{A}_{\text{eff}}, \Delta_{\mathbf{x}})$ ▷ $f$: objective function, $\mathcal{S}$: search space, $\mathscr{A}_{\text{eff}}$: robust acquisition function, $\Delta_{\mathbf{x}}$: uncertainty in the search variables
    Generate the initial data set $\mathcal{D} = \{\text{X}, \mathbf{y}\}$ on the objective function.
    Construct the Kriging model $\mathcal{K}_f$ on $\mathcal{D} = \{\text{X}, \mathbf{y}\}$.
    while the stop criteria are not fulfilled do
        Find robust optimum on the Kriging surface $\mathcal{K}_f$ as:
            $\hat{f}^*(\mathbf{x}) = \min_{\mathbf{x} \in \mathcal{S}} \hat{f}_{\text{eff}}(\mathbf{x})$.
        Choose a new sample $\mathbf{x}_{\text{new}}$ by maximizing the robust (effective) acquisition function:
            $\mathbf{x}_{\text{new}} \leftarrow \operatorname{argmax}_{\mathbf{x} \in \mathcal{S}} \mathscr{A}_{\text{eff}}(\mathbf{x})$.
        Compute function response $f(\mathbf{x}_{\text{new}})$.
        Extend the data set $\mathcal{D}$ by appending the pair $(\mathbf{x}_{\text{new}}, f(\mathbf{x}_{\text{new}}))$ to $\mathcal{D} = \{\text{X}, \mathbf{y}\}$.
        Reconstruct the Kriging model $\mathcal{K}_f$ on $\mathcal{D} = \{\text{X}, \mathbf{y}\}$.
    end while
end procedure
```

4.3 Empirical Investigation

So far in this chapter, we have provided the basic working mechanism of the BO algorithm, alongside three of the most important AFs: LCB, EI criterion, and the MGFI. Furthermore, we have extended the BO algorithm to the robust scenario, to account for parametric uncertainties in the search variables (Ullah et al., 2021). Extending the BO algorithm in this context is a rather difficult task, since the Kriging model only ever provides an approximation to the nominal response of the function, making the modeling of the true robust (effective) response of the function computationally intractable (Rehman, 2016). This is due to the fact that modeling the true robust response of the function requires additional assumptions on the joint probability distribution of all search points, which are induced by the uncertainty (ur Rehman et al., 2014). Furthermore, computing the utility function, e.g., the expectation, of this generally non-Gaussian distribution would also require us to evaluate analytically intractable integrals, which would result in prohibitively high computational demand.

Practically, following the approach by Rehman (ur Rehman et al., 2014), we model the true robust (effective) response of the function with a Gaussian process over the search points induced by the uncertainty: $\mathbf{x} + \Delta_{\mathbf{x}}$, similar to the nominal scenario. This approach enables us to study the performance of the BO algorithm in a comprehensive manner, as we can take into account the variability in external factors, such as severity of the uncertainty, robustness criterion, and infill criterion among others. The BO algorithm extended in this context is presented in Algorithm 1.

We are now interested in benchmarking the performance of the extended BO algorithm (cf. Algorithm. 1) to find robust solutions. We follow an empirical approach, based on a broad spectrum of test cases, to assess the performance of this algorithm. Following are the most important research question which we aim to answer with our study.

- Is the extended BO algorithm suitable to find robust solutions in an efficient manner?
- What factors influence the performance of the BO algorithm in this context?
- What impact does the infill criterion have on the quality of the robust solutions?

- How does the noise level and dimensionality affect the quality of the robust solutions?
- Which infill criterion is recommended to practitioners for practical scenarios, i.e., with regards to computational efficiency?

Answering these questions in a comprehensive manner is important because of the associated practical reasons, as it will enable us to find robust solutions in an efficient manner, with the help of the BO algorithm.

In the following, we describe the experimental setup of our study.

Experimental Setup

We select ten multi-modal test functions: $\mathscr{F} = \{f15 - f24\}$, from BBOB (Hansen et al., 2021) for our study. The uni-modal functions in BBOB are skipped because the BO algorithm is designed for multi-modal functions, and utilizing a high temperature in MGFI (high explorative effect) usually leads to inefficient convergence on the uni-modal function (Wang, 2018). All test functions are subject to minimization, and are evaluated on three different settings of dimensionality as: $D = \{2, 5, 10\}$.

Apart from the test functions and dimensionality, we also vary the uncertainty level based on two distinct settings as: $\mathscr{L} = \{0.05, 0.1\}$, which indicate the maximum % deviation in the nominal values of the search variables. For the deterministic setting of the uncertainty, i.e., MMR and MMRR, the compact set U is defined as: $\mathrm{U} = [-(L \times R), (L \times R)]$, where $L \in \mathscr{L}$ denotes the choice of the uncertainty level, and R serves as the absolute range of the search variables. For the test functions in $\mathscr{F}$, the absolute range of the search variables is 10, since all test functions are defined from -5 to 5. For the probabilistic setting of the uncertainty, i.e., EBR, DBR and CR, the uncertainty is modeled according to a continuous uniform probability distribution: $\Delta_{\mathbf{x}} \sim \mathcal{U}(a, b)$, where the boundaries a and b are defined similar to the boundaries of the the set U in the deterministic case.

In our study, the size of the initial training data is set to be $2 \times D$, where $D \in \mathscr{D}$ denotes the corresponding setting of the dimensionality. Likewise, the maximum number of iterations for BO is set to be $50 \times D$. Note that our Kriging surrogate is based on the popular Matérn 3/2 kernel (Rasmussen and Williams, 2006), and we standardize the function responses: $\mathbf{y} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^\top$, before constructing the Kriging surrogate $\mathcal{K}_f$. In addition, we utilize the three robust

AFs discussed in our study: LCB_{eff}, $\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})]$, and $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$, as the infill criteria for our experiments.

The hyper-parameters β and t in LCB_{eff} and $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$ respectively, are set in a similar fashion, as we monotonically decrease them with increasing number of iterations of the BO algorithm. This is due to the fact that we mainly want to emphasize on exploration at the beginning of the search. As the search progresses, we want to be more and more exploitative to be able to retain the good candidate solutions. To monotonically decrease β and t, we perform a linear cooling strategy (Wang et al., 2018) as:

$$t_{i+1} = t_i - \eta, \tag{4.15}$$

and

$$\eta = \frac{t_0 - t_f}{N_{\max}}, \tag{4.16}$$

where t_0 and t_f indicate the initial and final temperature settings respectively, and $N_{\max}$ serves as the maximum number of iterations of the BO algorithm.

We set the parameter β by adapting the Eqs. (4.15) and (4.16) for LCB_{eff}. In our experiments, β_0 and β_f are set to be 25 and 1 respectively, whereas t_0 and t_f are set to be 2 and 0.1, following the setup of Wang (Wang, 2018). For the parallel execution of RBO for each of the 360 test cases considered, we utilize the Distributed ASCI Supercomputer 5 (DAS-5) (Bal et al., 2016), where each standard node has a dual 8-core 2.4 GHz (Intel Haswell E5-2630-v3) cpu configuration and 64 GB memory. We implement our experiments in python 3.7.0 with the help of "scikit-learn" module (Pedregosa et al., 2011). The performance assessment of the robust solutions in our experiments is based on 15 independent runs $\mathscr{R}$ of the RBO algorithm for each of the 360 test cases considered. Note that for each trial, i.e., the unique combination of the independent run and the test case, we ensure the same configuration of hardware and software to account for fairness. Furthermore, in each trial, we measure the cpu time for all iterations of the RBO algorithm to measure the efficiency.

After the successful parallel execution of all trials, we evaluate the quality difference of our robust solutions from the baseline (cf. Eq. (3.12))). Note that $\mathcal{DQ}$ in this case is based on the space of objective function values[1]. After this, we perform six different analyses to answer the questions outlined earlier. The first two type of analyses are referred to as the fixed cpu time analysis, and the fixed iteration analysis respectively. In fixed cpu time analysis, we fix 50 different settings of the cpu time, and report the best $\mathcal{DQ}$ (the lowest) for each trial. The $\mathcal{DQ}$ in this context is averaged over all 50 settings of the cpu time. For fixed iteration analysis, we fix 30 different settings of the iterations (checkpoints) to report the best $\mathcal{DQ}$ (the lowest) for each trial. The $\mathcal{DQ}$ in this context is also averaged over all 30 checkpoints.

---

[1] In this study, we do not divide $\mathcal{DQ}$ with the number of independent runs $\mathscr{R}$ as Eq. (3.12) suggests, but rather report all trials.

After fixed cpu time and fixed iteration analysis, we perform a fixed target analysis. The fixed target analysis is also based on two different settings: by fixing a target $\mathcal{DQ}$ and reporting the cpu time as well as the number of iterations taken to reach that target. We fix ten different settings for the target in this context, and the corresponding cpu time and iterations are averaged over these target values. Note that each target describes the minimum desirable quality threshold of the robust solution. If such a quality is never achieved, we report the penalized cpu time and penalized number of iterations respectively. The penalized cpu time is set to be $D \times T_{\max}$, whereas penalized number of iterations is set to be $D \times N_{\max}$. Here D is the corresponding setting of the dimensionality, and $N_{\max}$ and $T_{\max}$ indicate the maximum number of iterations of the BO algorithm and the cpu time taken to execute it. After the fixed budget and fixed target analyses, we also report the average cpu time per iteration for the BO algorithm. In addition, we also report $T_{\max}$, the accumulated cpu time at the last iteration of the BO algorithm, for each trial.

4.3.1 Results

We share the results originating from our study in Figs. 4.1 – 4.6. Each of these figures contains the graphs for a particular type of analysis. In particular, Fig. 4.1 shares the results based on fixed cpu time analysis. The figure contains six different plots corresponding to two noise levels, and three different settings of the dimensionality. Each plot shares the empirical cumulative distribution function (ecdf) of $\mathcal{DQ}$ for three different robust AFs considered. Note that each ECDF curve (for each AF in a plot) is based on 300 data points, owing to the combination of ten test functions, two robustness criteria, and fifteen independent runs of the algorithm.

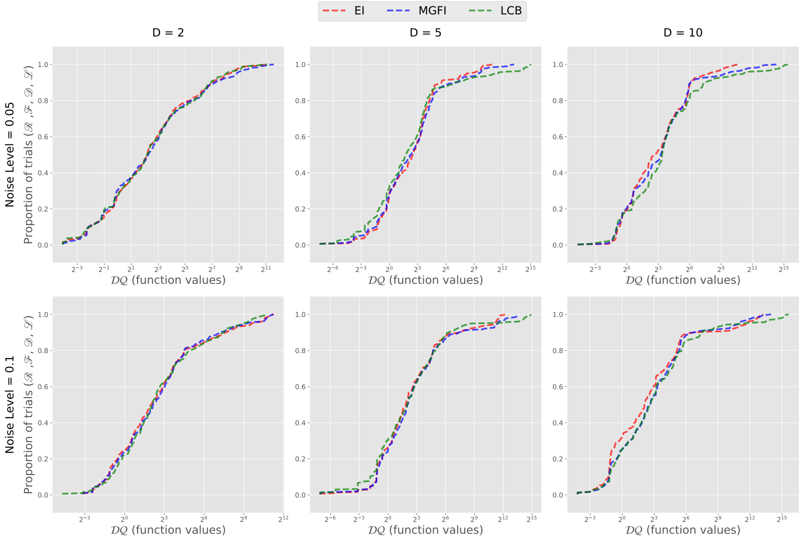


Figure 4.1: Fixed cpu time analysis. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

Likewise, Fig. 4.2 shares the ECDF plots corresponding to fixed iteration analysis, whereas the analyses based on fixed targets are presented in Figs. 4.3 and 4.4. The average cpu time per iteration of the BO algorithm to find robust solutions is presented in Fig. 4.5. Lastly, we present the maximum accumulated cpu time: $T_{\max}$, for each trial in the form of box plots in Fig. 4.6.

In the following, we report the major findings of these results.

- **Applicability of the Bayesian Optimization**

  Based on the results presented in Figs. 4.1 – 4.2, we deem BO as a promising heuristic to find robust solutions in an efficient manner. This is due to the fact that the empirical success rate of the BO algorithm is high. For instance, if we cut-off the $\mathcal{DQ}$ values at 8, the empirical success rate is around 60 %.

- **Factors with Significant Influence**

  Based on the results presented in Figs. 4.1 – 4.4, we find that dimensionality significantly affects the quality of the robust solutions. Furthermore, we observe that this affect is much clearer to notice for LCB_{eff} and $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$,

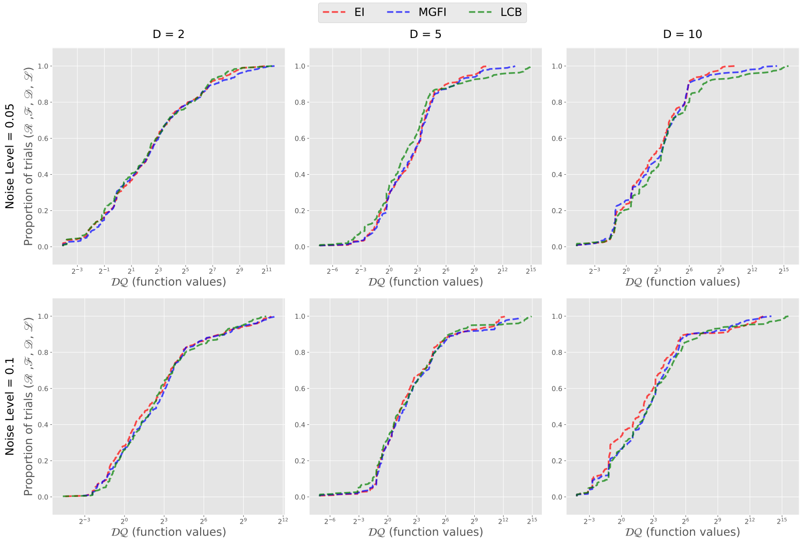


Figure 4.2: Fixed iteration analysis. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

unlike $\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})]$ whose performance is not significantly compromised in the face of higher dimensionality. Because of the dimensionality, the computational budget, i.e., whether measured in cpu time or number of iterations, also affects the quality of the robust solutions in a significant manner. For instance, in Fig. 4.3, we see that the empirical success measured at 2^{10} seconds (cpu time) is more than 85 % for trials belonging to two-dimensional problems. On the other hand, the empirical success rate drops to under 40 % when dimensionality is increased from 2 to 5. If, on the other hand, the dimensionality is further increased to 10, the observed empirical success rate drops below 20 %. When measuring the impact of noise level, i.e., the scale/severity of the uncertainty, on the performance of the BO algorithm, we do not observe any clear patterns. However, in some individual cases, the performance of the BO algorithm is compromised with a higher settings of the noise level.

- **Impact of Infill Criterion**

In the context of fixed budget analyses, the performance of all three AFs

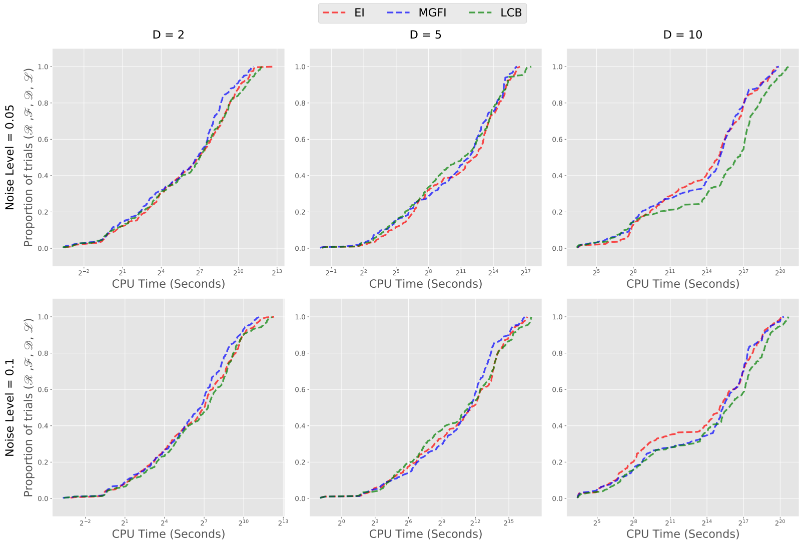


Figure 4.3: Fixed target analysis based on cpu time. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

is comparable in most trials. For a higher setting of the dimensionality, i.e., $D = 10$, however, we observe a higher variance in the performance of $\mathrm{LCB}_{\mathrm{eff}}$ and $\mathcal{M}_{\mathrm{eff}}(\mathbf{x};t)$. In the context of fixed target analyses, we observe similar patterns, i.e., for most trials, we do not observe a significant difference in the performance. Hence, we cannot find a clear winner in this case, albeit we can say that $\mathbb{E}[\mathcal{I}_{\mathrm{eff}}(\mathbf{x})]$ is better suited for higher dimensionality.

- **Infill Criterion for Practical Scenarios**

For choosing an AF for practical scenarios, we emphasize on the average running cpu time per iteration (ARCTPI), as well as the maximum cpu time required for an independent run: $T_{\max}$, in addition to the fixed budget and fixed target analyses. In the context of ARCTPI, i.e.,Figs. 4.5, we find $\mathcal{M}_{\mathrm{eff}}(\mathbf{x};t)$ as clearly superior to its competitors in most trials. Likewise, in the context of $T_{\max}$, i.e., we find $\mathcal{M}_{\mathrm{eff}}(\mathbf{x};t)$ as clearly superior to its competitors. Combining the performance for all type of analyses, we find $\mathbb{E}[\mathcal{I}_{\mathrm{eff}}(\mathbf{x})]$ and $\mathcal{M}_{\mathrm{eff}}(\mathbf{x};t)$ as suitable AF to be employed in the BO algorithm to find robust solutions.

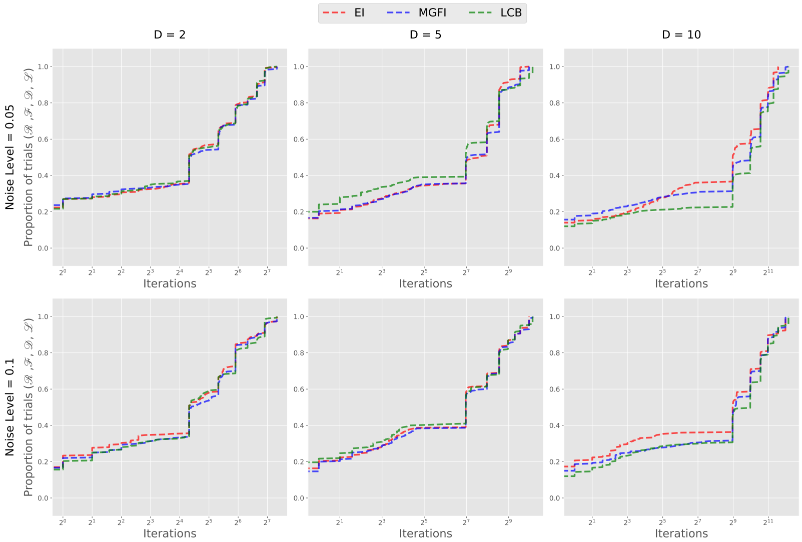


Figure 4.4: Fixed target analysis based on number of iterations. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

4.4 Summary and Discussion

To employ the Bayesian optimization algorithm to find robust solutions, we face several technical issues. Chief among them is the issue that the “best-so-far” observed value of the function, which acts as a baseline to compute “improvement/gain” in nominal Bayesian optimization algorithm, renders inapplicable, when we are interested in robust solutions. This is due to the fact that this value has no clear meaning/usage in the context of robust solutions. Therefore, we substitute this value with the current best known “robust” value of the function, which by implication can only be estimated on the Kriging surface (as opposed to observed or fully known in the nominal case).

The second issue that we face is that the Kriging model only ever provides an approximation to the nominal function response, and therefore cannot be utilized directly to model the “robust” function response, without which we cannot proceed. To solve this issue, we assume that the true “robust” response of the function is also normally distributed with Kriging prediction and MSE acting as the pa-

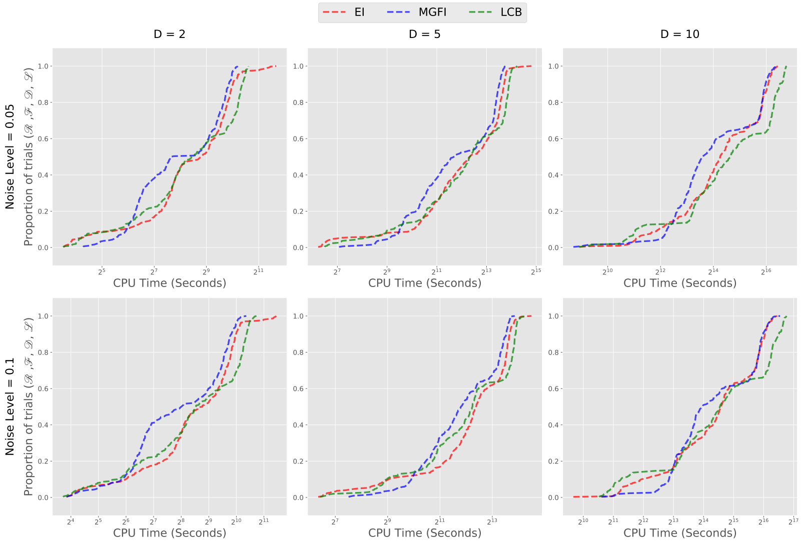


Figure 4.5: Average cpu time per iteration for the BO algorithm. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

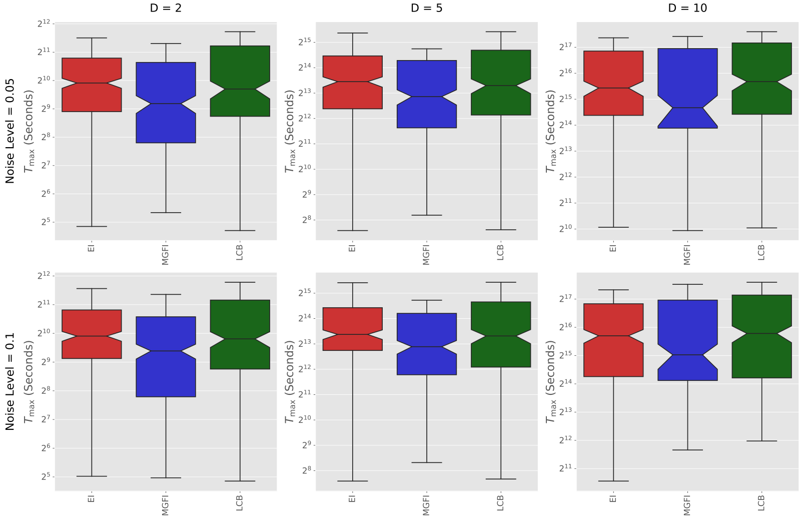


Figure 4.6: Maximum accumulated cpu time: $T_{\max}$ for each trial. Rows distinguish between noise levels, whereas columns help identify different settings of dimensionality.

rameters of the distribution. Note that the assumption that the true “robust” response of the function is normally distributed is not a rigorous one, but a practical compromise. This is due to the fact that modeling the true robust response of the function with a non Gaussian distribution is computationally intractable in our opinion.

After solving these issues, we extend the Bayesian optimization algorithm to find robust solutions. We consider three sampling infill criteria in this chapter: the “Lower Confidence Bound”, the “Expected Improvement” criterion, and the “Moment-Generating Function of the Improvement”, which are also extended to care for robustness, in order to find robust solutions. Following this, we perform a comprehensive empirical investigation to answer fundamental research questions on this topic. These questions deal with the applicability of the Bayesian optimization algorithm to find robust solutions, the factors that influence its performance, the impact of the sampling infill criterion, and the preferred choice of the sampling infill criterion in practical scenarios.

The key findings from our study provide new insights on this topic. For instance, we find that the Bayesian optimization algorithm is suitable to find robust solutions, which implies that our adaptation of the Bayesian optimization algorithm works well in practice. This is an important aspect to know since we are unaware of any empirical investigation which answers this question in a comprehensive manner, i.e., by taking into account the variability in external factors such as dimensionality, robustness criterion, and uncertainty level.

We also find that dimensionality, and consequently the computational budget, plays a significant role in the performance of the Bayesian optimization algorithm. This finding validates our understanding on the so-called “Curse of Dimensionality” discussed in Chapter 3, and the dimensionality reduction techniques discussed therein become even more important. Apart from that, we also validate that the noise level, i.e. the scale/severity of the uncertainty, does not directly affect the quality of the robust solution in an adverse manner.

Lastly, we find that the performance of the “Expected Improvement” criterion, and the “Moment-Generating Function of the Improvement” enables them to be employed in practical scenarios. While the performance of the “Lower Confidence Bound” is deemed satisfactory in most cases, it does not show promising aspects with respect to a higher setting of the dimensionality, and the average cpu time per iteration is also higher.

CHAPTER 5

Computational Cost of Robustness

Solving a robust optimization problem entails re-formulating the original optimization problem into a robust counterpart, e.g., by taking an average of the function values over different perturbations to a specific input (Chapter 3) (Kruisselbrink, 2012). Consequently, solving the robust counterpart is much more costlier as opposed to the original optimization problem. We refer to this aspect as the "computational cost of robustness". The "computational cost of robustness" has been overlooked in the literature to the best of our knowledge. Such an extra cost brought by robust optimization might depend on the problem landscape, the dimensionality, the severity of the uncertainty, and the formulation of the robust counterpart itself. This chapter devotes to such an extra cost brought by robustness in practical scenarios, as it can hinder the aim of computational efficiency set by the designer (Ullah et al., 2022).

Some of the most important questions that we will attempt to answer in this chapter are:

1. How can we rank commonly employed robustness criteria with respect to the trade-off of the "computational cost of robustness" and the quality of the robust solutions?
2. Which robustness criteria are recommended to practitioners for practical scenarios, in order to find robust solutions in an efficient manner?

Note that these questions are based on the fourth foundational question introduced in the first chapter. We will attempt to answer these questions with the help of an empirical investigation, which will focus on a broad spectrum of test cases. The test cases concentrate on the variability in problem landscape, dimensionality, severity of the uncertainty, and robustness criteria, among others.

The five robustness criteria discussed in this chapter have already been defined in Chapter 3. These include “mini-max robustness”, “mini-max regret robustness”, “expectation-based robustness”, “dispersion-based robustness”, and “composite robustness” respectively.

5.1 Cost of Robustness

Solving a numerical black-box problem subject to uncertainty and noise is challenging due to several reasons. These reasons include high dimensionality, problem landscape, and robustness criterion among others. We have already seen the impact of high dimensionality and problem landscape in Chapters 3 and 4 (Ullah et al., 2019, 2020a, 2021). This Chapter concentrates on the choice of the robustness criterion, which can also have a significant impact in efficiently solving the problem.

The choice of the robustness criterion is important for two main reasons.

- The chosen robustness criterion can have a significant impact on the efficiency. This is due to the fact that obtaining a robust solution requires additional computational resources as opposed to a nominal one, since the optimizer has to take into account the impact of uncertainty and noise as well. This need for additional computational resources is based on the robustness criterion[1] chosen, e.g., RO based on the “mini-max” principle requires an internal optimization loop to compute the worst impact of the uncertainty, whereas RO based on the “expectation” of a noisy function requires computing an integral (Beyer and Sendhoff, 2007; Ullah et al., 2019).
- The chosen robustness criterion can also have a significant impact on the performance, i.e., optimality. Recall that the aim for robustness can be achieved from two different schools of thought, which are often conflicting: Performance/Quality and Robustness/Stability (Kruisselbrink, 2012) (cf. 3.1.1). The former measures the robustness of a solution from the perspective of the overall performance, i.e.,optimality, under the variation of the uncertain parameters of the solution. As opposed to that, the latter measures the

---

[1] An underlying assumption in this context is the non-existence of hard constraints on the choice of RF. In some practical scenarios of RO, this assumption is not valid. Note, however, that, there are plenty of situations where the assumption is valid, and the lack of information on the computational aspects of the RFs makes it harder for practitioners to choose a suitable robustness criterion.

robustness of a solution from the perspective of the minimal performance variation under the variation of the uncertain parameters of the solution. Consequently, when choosing a robustness criteria, it is important to know that we maybe compromising on optimality to ensure robustness/stability, e.g., in the case of "mini-max" robustness.

In this Chapter, we only focus on the aspect of computational efficiency, based on the choice of robustness criterion. This is due to the fact that the "computational cost of robustness" (CCoR), i.e., the need for additional computational resources to find the "robust" instead of a "nominal" solution, depends on the robustness criterion chosen. Consequently, it is desirable to evaluate and compare commonly-employed robustness criteria with regards to computational cost, where the computational cost is mainly based on the cpu time taken to find the robust solution.

5.2 Empirical Investigation

Our aim in this Chapter is to understand the impact of robustness criterion in RBO with regards to computational efficiency. Intuitively, robustness criterion can have a significant impact on the performance of the RBO algorithm, since steps 5 and 6 in Algorithm 1 (cf. 1) require much more computational resources as opposed to the nominal BO algorithm (Jones et al., 1998). This need for additional computational resources is based on the chosen robustness criterion. Through our empirical investigation, we aim to better understand this impact for each of the five robustness criteria discussed in the thesis. To make our setup comprehensive, we take into account the variability in external factors such as problem landscape, dimensionality, and the scale/severity of the uncertainty.

For our study, we select ten unconstrained, noiseless, single-objective optimization problems from BBOB (Hansen et al., 2021). The set of selected test functions is given as: $\mathscr{F} = \{f_2, f_3, f_7, f_9, f_{10}, f_{13}, f_{15}, f_{16}, f_{20}, f_{24}\}$. An important thing to note is that each of the test functions in $\mathscr{F}$ is subject to minimization, and is evaluated on three different settings of dimensionality as: $D = \{2, 5, 10\}$. Apart from the test functions and dimensionality, we also vary the uncertainty level based on two distinct settings as: $\mathscr{L} = \{0.05, 0.1\}$, which indicate the maximum % deviation in the nominal values of the search variables.

For the deterministic setting of the uncertainty: MMR and MMRR, the compact set U is defined as: $\mathrm{U} = [-(L \times R), (L \times R)]$, where $L \in \mathscr{L}$ denotes the choice of the uncertainty level, and R serves as the absolute range of the search variables. For the test functions in $\mathscr{F}$, the absolute range of the search variables is 10, since all test functions are defined from -5 to 5. For the probabilistic setting of the uncertainty: EBR, DBR and CR, the uncertainty is modeled according to a continuous uniform probability distribution: $\Delta_{\mathbf{x}} \sim \mathcal{U}(a, b)$, where the boundaries a and b are defined similar to the boundaries of the the set U in the deterministic case.

In our study, the size of the initial training data is set to be $2 \times D$, where $D \in \mathscr{D}$ denotes the corresponding setting of the dimensionality. Likewise, the maximum number of iterations for the BO algorithm is set to be $50 \times D$. Note that our Kriging surrogate is based on the popular Matérn 3/2 kernel (Rasmussen and Williams, 2006), and we standardize the function responses: $\mathbf{y} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^\top$, before constructing the Kriging surrogate. In addition, we utilize the robust EI (cf. Eq. (4.13)) as the sampling infill criterion for our experiments.

For the parallel execution of the BO algorithm for each of the 300 test cases considered, we utilize the DAS-5 supercomputer (Bal et al., 2016), similar to the experimental setup discussed in the previous chapter. The performance assessment of the solutions in our experiments is based on 15 independent runs $\mathscr{R}$ of the RBO algorithm for each of the 300 test cases considered. Note that for each trial, i.e., the unique combination of the independent run and the test case, we ensure the same configuration of hardware and software to account for fairness. Furthermore, in each trial, we measure the cpu time for all iterations of the RBO algorithm.

After the successful parallel execution of all trials, we evaluate the performance of our robust solutions based on the quality difference $\mathcal{DQ}$ from the baseline (cf. Eq. (3.12))). Note that $\mathcal{DQ}$ in this case is based on the space of objective function values[1]. After this, we perform six different analyses to answer the questions outlined earlier. The first two type of analyses are referred to as the fixed cpu time analysis, and the fixed iteration analysis respectively. In fixed cpu time analysis, we fix 50 different settings of the cpu time, and report the best $\mathcal{DQ}$ (the lowest) for each trial. The $\mathcal{DQ}$ in this context is averaged over all 50 settings of the cpu time. For fixed iteration analysis, we fix 30 different settings of the iterations (checkpoints) to report the best $\mathcal{DQ}$ (the lowest) for each trial. The $\mathcal{DQ}$ in this context is also averaged over all 30 checkpoints.

---

[1] In this study, we do not divide $\mathcal{DQ}$ with the number of independent runs $\mathscr{R}$ as Eq. (3.12) suggests, but rather report all trials.

After fixed cpu time and fixed iteration analysis, we perform a fixed target analysis. The fixed target analysis is also based on two different settings: by fixing a target $\mathcal{DQ}$ and reporting the cpu time as well as the number of iterations taken to reach that target. We fix ten different settings for the target in this context, and the corresponding cpu time and iterations are averaged over these target values. Note that each target describes the minimum desirable quality threshold of the robust solution. If such a quality is never achieved, we report the penalized cpu time and penalized number of iterations respectively. The penalized cpu time is set to be $D \times T_{\max}$, whereas penalized number of iterations is set to be $D \times N_{\max}$. Here D is the corresponding setting of the dimensionality, and $N_{\max}$ and $T_{\max}$ indicate the maximum number of iterations of the BO algorithm and the cpu time taken to execute it. After the fixed budget and fixed target analyses, we also report the average cpu time per iteration for the BO algorithm. In addition, we also report $T_{\max}$, the accumulated cpu time at the last iteration of the BO algorithm, for each trial.

5.2.1 Results

We share the results originating from our study in Figs. 5.1 – 5.5. Each of these figures contains the graphs for a particular type of analysis. In particular, Fig. 5.1 shares the results based on fixed cpu time analysis. The figure contains six different plots corresponding to two noise levels, and three different settings of the dimensionality. Each plot shares the empirical cumulative distribution function (ecdf) of $\mathcal{DQ}$ for five different robustness criteria considered. Note that each ecdf curve (for each robustness criterion in a plot) is based on 150 data points, owing to the combination of ten test functions and fifteen independent runs of the algorithm.

Likewise, Fig. 5.2 shares the ecdf plots corresponding to fixed iteration analysis, whereas the analyses based on fixed target is presented in Figs. 5.3. The average cpu time per iteration of the BO algorithm to find robust solutions is presented in Fig. 5.4. Lastly, we present the maximum accumulated cpu time: $T_{\max}$, for each trial in the form of box plots in Fig. 5.5.

In the following, we discuss the major findings of these results.

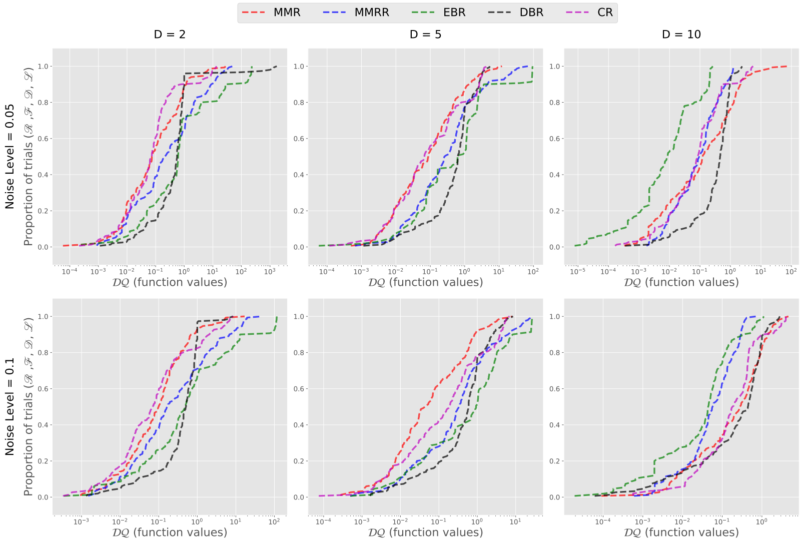


Figure 5.1: Fixed cpu time analysis. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains five Ecdf curves based on five robustness criteria discussed.

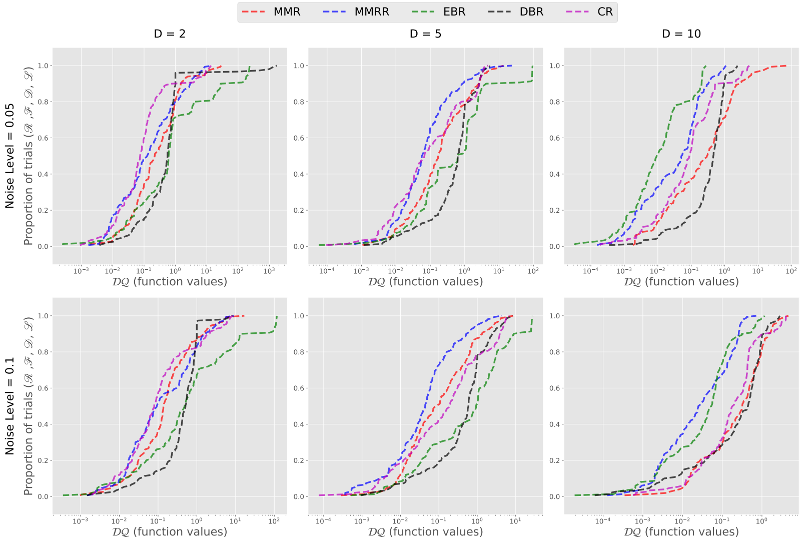


Figure 5.2: Fixed iteration analysis. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains five Ecdf curves based on five robustness criteria discussed.

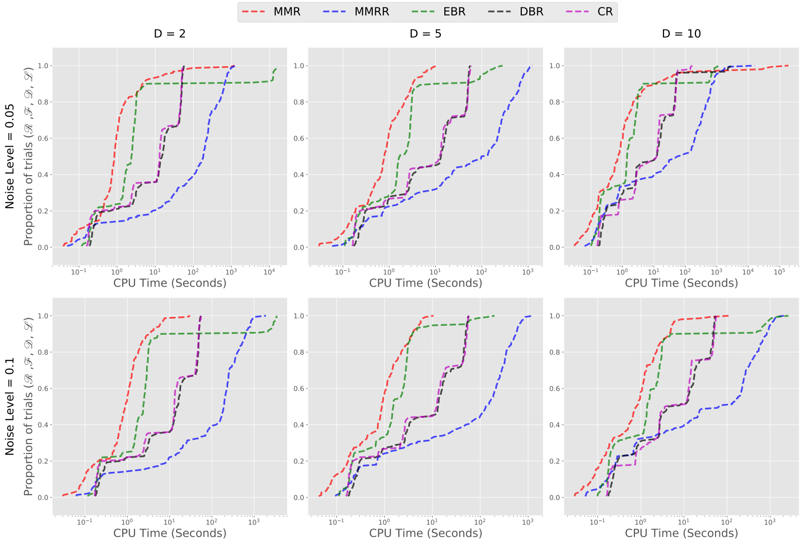


Figure 5.3: Fixed target analysis based on cpu time. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains five Ecdf curves based on five robustness criteria discussed.

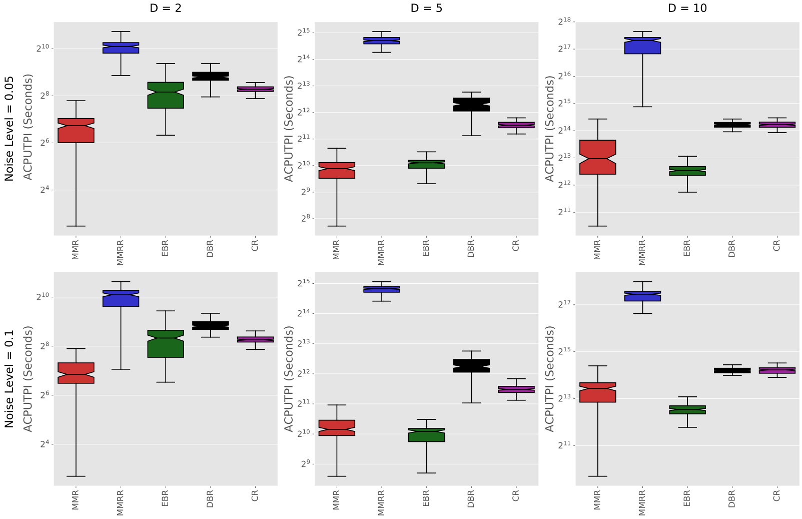


Figure 5.4: Average cpu time per iteration for the BO algorithm. Rows distinguish between noise levels, whereas columnsidentify different settings of dimensionality.

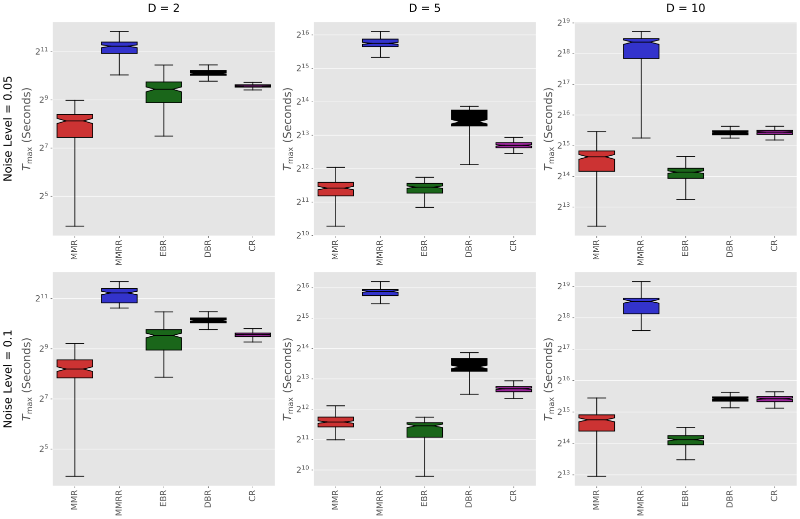


Figure 5.5: Maximum accumulated cpu time $T_{\max}$ for each trial. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality.

In terms of performance comparison with respect to the fixed cpu time analysis, we note the promising nature of all RFs except DBR, which performs poorly compared to its competitors in most trials. Furthermore, we also note the highest variation in quality ($\mathcal{DQ}$) for DBR. Although no RF is deemed a clear winner for this analysis, we note that MMR, MMRR and CR have high empirical success rates. Likewise, we note the highest variation in quality for DBR also in the context of fixed iteration analysis. In this case, MMRR and CR perform superior to the other RFs as we observe a high empirical success rate for both. For the performance measure with respect to the fixed target analysis, we observe that MMR outperforms the competitors, albeit the variation in the running cpu time for MMR is also deemed higher. Here, we note a clear distinction in the empirical success rate between MMRR and other RFs. For instance, if we cut-off the running cpu time at 100 seconds, we observe that MMRR has an empirical success rate of around 40 %, whereas MMR, DBR, and CR achieve a success rate of more than 80 %.

In terms of average cpu time per iteration (ACPUTPI), we note that MMR and EBR perform excellently in most cases, whereas MMRR has the worst performance. In this case, none of the MMR and EBR can be deemed a clear winner, although both perform superior to other RFs in most trials.

When comparing the performance of RFs in the context of maximum cpu time spent: $T_{\max}$, we note that MMR and EBR in general perform superior to other RFs, whereas MMRR performs the worst for each setting of dimensionality. Furthermore, we deem that $T_{\max}$ increases rapidly with respect to dimensionality in the context of deterministic uncertainty, i.e., MMR and MMRR, when compared with the probabilistic uncertainty, i.e., EBR, DBR, and CR. Lastly, we note that in general, the variance in $T_{\max}$ for the probabilistic setting is also significantly lower when compared to the deterministic case.

5.2.2 Analysis

Based on the observations from the fixed budget analyses – fixed cpu time and fixed iteration analyses, we deem MMR, MMRR, EBR and CR to be suitable RFs with regards to the computational cost involved to find the robust solution. This validates their applicability in practical scenarios where the computational resources are limited, and the designer cannot spend more than a fixed amount of computational budget (whether measured in terms of cpu time or the number of iterations). Note that MMR appears to be the most promising RF also in the scenarios where the designer aims for a fixed quality solution – where the designer cannot compromise on the quality below a certain threshold. In those situations, MMR can yield the desired quality robust solution with considerably less cpu time. Apart from these analyses, we note that the ACTPI and $T_{\max}$ for MMR are also excellent alongside EBR.

In terms of performance, we find that MMRR poses an interesting situation as it performs competitively in the context of fixed budget analyses. However, its performance is significantly worse to other RFs in the context of fixed target analysis, the ACTPI, and the maximum cpu time $T_{\max}$ for running KB-RO. We believe this is aligned with the intuition of MMRR (as discussed in Chapter 3), since within an iterative optimization framework, we have to employ a quadrupled nested loop to find the robust solution based on MMRR, which in turn exponentially increases the computational cost per iteration. The MMRR, therefore, has the highest CCoR, and takes much more cpu time to reach the same target value as opposed to other RFs.

In terms of performance variance, we note that stochastic RFs, in particular EBR and DBR, have a higher variance in quality – when measured in terms of NMAE, and a comparatively lower variance in computational cost – when measured in terms of the ACTPI and $T_{\max}$. This can mainly be attributed to their intrinsic stochastic nature as they rely on numerical approximations. Since the sample size of the numerical approximations is fixed with respect to the corresponding setting of the dimensionality, we observe relatively lower variance in the cpu time. However, since we only ever approximate the robust response of the function, the quality of the solution may be deteriorated.

5.3 Summary and Discussion

This chapter analyzes the computational cost of robustness in Bayesian optimization for five of the most commonly employed robustness criteria. In a first approach of such kind, we attempt to evaluate and compare the robustness formulations with regards to the associated computational cost, where the computational cost is based on the cpu time taken to find the optimal solution under uncertainty. Our experimental setup constitutes 300 test cases, which are evaluated for 15 independent runs of Bayesian optimization.

A fixed budget analysis on our experimental results suggests the applicability of “mini-max robustness”, “mini-max regret robustness”, “expectation-based robustness”, and “composite robustness” in practical scenarios where the designer cannot afford the computational budget beyond a certain threshold. On the other hand, a fixed target analysis deems the ‘mini-max robustness” as the most efficient robustness criterion in the scenario where the designer cannot compromise the quality of the optimal solution below a certain threshold. The analysis on the average cpu time per iteration and maximum cpu timer per run also determines “mini-max robustness” as one of the most efficient robustness criteria. Overall, “mini-max robustness” is understood to be the most suitable robustness criterion with regards to the associated computational cost.

A limitation of our study is that we fix the internal computational budget for each robustness formulation in Kriging-based robust optimization. Visualizing the impact of variability in the internal computational budget, e.g., the internal optimization loop in the context of “mini-max robustness”, is also an important thing to do. Additionally, we note that each robustness formulation is intrinsically associated with another cost, namely the cost of compromising on optimality to ensure robustness or stability (cf. 5.1). Focusing on this cost of robustness will advance the state-of-the-art in this area, and help practitioners choose the most suitable formulation with regards to optimality.

CHAPTER 6

Engineering Applications

So far in this thesis, we have focused on two different but related research streams. The first one (Chapter 3) deals with the applicability of surrogate modeling to find robust solutions. The manifestation of surrogate modeling focused in this research stream is based on "one-shot optimization" strategy (Ullah et al., 2019). In this research stream, we emphasize on the fundamental questions regarding the applicability of surrogate modeling in robust optimization, and the related difficulties thereof, e.g., high dimensionality (Ullah et al., 2020a).

The second research stream (Chapters 4 and 5) emphasizes on the applicability of the Bayesian optimization algorithm, and the related difficulties thereof (Ullah et al., 2021). As part of the second research stream, we made an attempt to find a suitable robustness criterion in practical scenarios with regards to the associated computational cost (Ullah et al., 2022). We are now interested in benchmarking our earlier findings for both research streams on a real-world engineering application.

To this end, we consider a benchmark engineering case study based on the design of car hood frames. The associated data set contains over 10,000 3D mesh geometries for variants of card hood frames. This data set is generated through an automated, industry-grade Computer Aided Design (CAD) workflow, described in (Ramnath et al., 2022), and further benchmarked in (Wollstadt et al., 2022). The data set provides realistic designs of car hood frames, which were validated by experts with respect to realism, manufacturability, variability, and performance. Each geometry is described by a subset of design variables, such as cut-outs and ribs on the hood frame as well as their properties, for example, rib location and height, or cut-out location.

Starting in Section 6.1, we provide an overview on the case study with the most important details, such as the description of the data set, the data pre-processing, and the targeted tasks. In Section 6.2, we apply the data set to validate our findings for the first research stream. This is followed by benchmarking the performance of the Bayesian optimization algorithm on the data set. Lastly, we provide a short summary and discussion on the results.

6.1 Car Hood Design

We consider a case study inspired from a real-world design optimization scenario (Ramnath et al., 2022), where the aim is to optimize the design of a car hood frame with respect to three performance metrics. These performance metrics are the "Geometry Mass (kg)", "Directional Deformation Maximum (mm)", and "Equivalent Stress Maximum (MPa)". Each geometry is represented as a surface mesh (STL file), and is described by a subset of 38 design variables, such as "Rear Rib Depth", "Rear Rib Offset", "Pocket Offset", and "Front Curve Height" among others.

The data set in this context was generated using a feature-based modeling approach (Ramnath et al., 2019). In the context of automotive car hoods, features describe components that contribute to desirable properties of the design, e.g., ribs to add stiffness during driving or impact, or cut-outs and pockets to reduce overall weight. Real car hood designs were simplified by removing features that were irrelevant for the hood's performance. Remaining features were created independent of the base surface to allow for the generation of a sufficiently large variety of hoods by combining features and feature patterns with a set of 100 base geometries. Features were parameterized and generated using an automated workflow in computer-aided three-dimensional interactive application (CATIA) v5 (König and Wintermantel, 2004). It is important to note that some parametrizations led to invalid geometries, such that in total 10,478 unique hood geometry files were generated. CAD models were converted to watertight STL surface meshes in the STL format (Ramnath et al., 2022).

For each car hood, structural mechanics performance values were simulated using finite element analysis (FEA) (Szabó and Babuška, 2021). FEA was performed for a hood lift load case under driving conditions, which is an important structural requirement when designing car hood frames (Vyas et al., 2020). The obtained

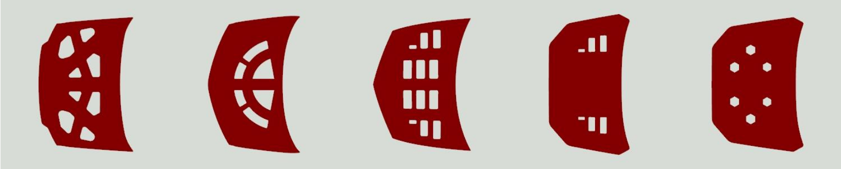


Figure 6.1: Example of variability in car hood designs in the data set. The variability is due to the introduction of features, e.g., pockets, cut outs, and ribs.

performance values are "Maximum Equivalent Stress (MPa)" and "Maximum Directional Deformation (mm)". Additionally the "Geometry Mass (kg)" is provided for each design. FEA was performed using a standardized setup over all geometries to allow for automated generation of simulation results. Fig. 6.1 illustrates the variability in geometries considered in our case study.

6.1.1 Data Set

We start with the data set provided by Ramnath et al. (Ramnath et al., 2022), which includes geometries, i.e., surface meshes, for 10,070 different designs for car hood frames. Each geometry contains values of the design variables, e.g., "Rear Rib Depth", used to run the FEA simulation for assessing the performance of the corresponding design with respect to structural mechanics indicators, e.g., "Maximum Equivalent Stress (MPa)" and "Maximum Directional Deformation (mm)". Note that the entire data set contains 38 unique design variables, but each geometry is accompanied with a subset of these variables. The average number of design variables per geometry is found to be 14, whereas the maximum number of design variables is found to be 27.

6.1.2 Data Wrangling

Since the geometries in the data set have irregular and asynchronous design schema[1], our first task is to identify the most common design variables. For this purpose, we count the frequency of each design variable in the entire data set, and select the top five most commonly appearing design variables. They are "Rib Depth", "Rear Rib Width", "Rear Rib Offset", "Rear Rib Depth", and "Rear Rib

---

[1] Irregular in this context refers to the fact that geometries are provided with varying number of design variables. Asynchronous refers to the fact that not all design variables are present in each geometry (Ullah et al., 2020b)

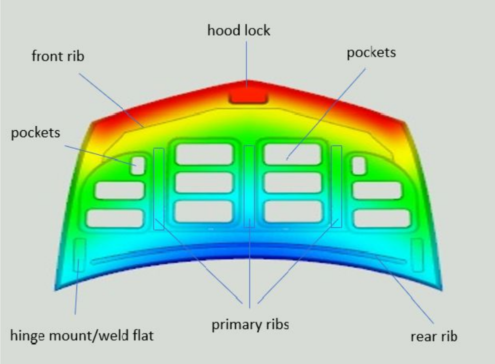


Figure 6.2: Some of the most commonly appearing features considered in the optimization of car hood design. The design variables considered in our study indicate the properties for these features, e.g., "Rear Rib Depth" and "Pocket Offset".

End Point Y". We then scan the entire data set to identify designs where these five variables appear together. As a results, we are left with 1176 geometries where these five variables appear together. We then extend this data set by scanning these 1176 designs so as to search for other design variables, which maybe present in all of these 1176 geometries. This increases the number of design variables to 18.

We are interested in benchmarking our previous findings on a real-world engineering case study. For this purpose, we have to formulate optimization scenarios with three settings of dimensionality as: $\mathscr{D} = \{2, 5, 10\}$. This means we have to select two, five, and ten design variables among the set of available design variables. For this purpose, we construct a benchmark Kriging surrogate model (Rasmussen and Williams, 2006) with all 18 variables and 1176 training instances, for all three performance indicators: Mass (kg), Deformation (mm), and Stress (MPa). Then, we remove each one of the 18 variables in the model, and see the potential impact on the accuracy of the model. Based on this, we rank all 18 variables for all three performance indicators, and select the top two, five, and ten variables that have the most significant effect on the model accuracy (Fan, 2007). These variables for all three performance indicators are presented in Table 6.1. Furthermore, we present an example of some of the most important features for car hood designs in Fig. 6.2 for further clarification.

Table 6.1: A summary of the selected design variables and tasks to be performed. “Performance Indicators” indicates the three output variables (tasks), which are to be minimized in optimization. “Variables” indicates the design variables which are included for a particular choice of task and dimensionality (based on the data wrangling discussed earlier). The abbreviations for these design variables are presented in Table 6.2.

Performance Indicators	**Dimensionality**	**Variables**
Mass (kg)	2	“RRO”, “RRD”
	5	“RRO”, “RRD”
		“P1O”, “P2O”
		“P3O”
	10	“RRO”, “RRD”
		“P1O”, “P2O”
		“P3O”, “RREPY”
		“P2R”, “P3R”
		“P4O”, “SRW”
Deformation (mm)	2	“RRO”, “RCH”
	5	“RRO”, “RCH”
		“RREPY”, “P4O”
		“RRW”
	10	“RRO”, “RCH”
		“PREPY”, “P4O”
		“RRW”, “1SRL”
		“P2O”, “P3O”
		“ARW”, “P3R”
Stress (MPa)	2	“RRO”, “ARW”
	5	“RRO”, “ARW”
		“RCH”, “P4O”
		“SRW”
	10	“RRO”, “ARW”
		“RCH”, “P4O”
		“SRW”, “P2O”
		“P3O”, “RRD”
		“1SRL”, “2SRL”

Table 6.2: Abbreviations of the 18 design variables discussed in Section 6.1.2. Some of these variables are presented in Table 6.1 to formulate the optimization tasks with three settings of the dimensionality.

Design Variable	Abbreviation	Range
“1SRL”	“1st Subsidiary Rib Length”	[140, 200]
“2SRL”	“2nd Subsidiary Rib Length”	[120, 240]
“ARW”	“Angled Rib Width”	[30, 180]
“MRW”	“Middle Rib Width”	[40, 200]
“P1O”	“Pocket1 Offset”	[0, 10]
“P1R”	“Pocket1 Radius”	[18, 35]
“P2O”	“Pocket2 Offset”	[0, 10]
“P2R”	“Pocket2 Radius”	[18, 45]
“P3O”	“Pocket3 Offset”	[0, 10]
“P3R”	“Pocket3 Radius”	[13, 45]
“P4O”	“Pocket4 Offset”	[0, 10]
“RCH”	“Rear Curve Height”	[20, 120]
“RRD”	“Rear Rib Depth”	[8, 14]
“RREPY”	“Rear Rib End Point Y”	[400, 620]
“RRO”	“Rear Rib Offset”	[-50, 10]
“RRW”	“Rear Rib Width”	[20, 30]
“RD”	“Rib Depth”	[15, 30]
“SRW”	“Subsidiary Rib Width”	[25, 50]

6.1.3 Tasks

We are interesting in design optimization scenarios with three performance indicators illustrated in Table 6.1. For each one of the indicators, we consider three settings of dimensionality as described earlier. This gives rise to a total of 9 optimization tasks. Furthermore, we consider two different goals of robust optimization for these 9 tasks. These two goals emphasize on benchmarking “one-shot optimization” strategy (Chapter 3) and Bayesian optimization algorithm (Chapter 4) (Ullah et al., 2019, 2021).

It is pertinent to mention that we try to maintain the same experimental setup, wherever possible, for these two goals, as discussed previously in the thesis (Chapter 3 and Chapter 4), to account for fairness. Nonetheless, the nature of the industrial data, as well as the realism, manufacturability, and variability associated with the data generation process means we might have to compromise on some settings of our previous experimental setups. This would be explained in further details in experimental setup, wherever applicable.

6.2 One-shot Optimization

We begin with the goal of benchmarking the "one-shot optimization" strategy based on our data set. This refers to the fact that we consider a surrogate model, which after construction, is directly utilized by a benchmark numerical optimization algorithm, e.g., L-BFGS-B (Wright et al., 1999), without any adaptive sampling, i.e., updating the surrogate model (Ta'asan et al., 1992). This strategy has been explained in detail in Chapter 2 (cf. Fig. 2.2). This goal has two objectives: evaluating the surrogate model based on the modeling accuracy, and the quality of the optimal solution obtained from surrogate modeling. Experimental setup as well as results for both of these manifestations are provided in the following.

6.2.1 Experimental Setup

In the context of modeling accuracy, our aim is to determine which of the popular modeling techniques (Bishop, 2007) is most suitable to model the objective function effectively. In this experimental setup, we consider five modeling techniques: Kriging, Polynomial (degree=2), K Nearest-Neighbour (KNN), Random Forest (RF), and Support Vector Machines (SVMs). Furthermore, for each one of these techniques, we consider ten different sample sizes as: $K \times D$, where D refers to the corresponding setting of the dimensionality, and $K \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$. This gives rise to a total of 450 test cases, owing to the unique combinations of 9 optimization tasks, 10 different sample sizes, and 5 modeling techniques. For each one of these cases, we measure the modeling accuracy according to the RMAE criterion introduced earlier (cf. Eq. (3.11)).

In terms of data pre-processing, we first extract the information about the quantity of interest, e.g., Mass (kg), as per the task, and the corresponding design variables, e.g., "RRD", according to the setting of the dimensionality. We then identify the duplicates in the newly formed data set and remove them. Following this, we construct a design for each setting of the sample size according to the LHS scheme (Montgomery, 2017). This, however, poses a practical problem, since we do not have direct access to the FEA simulation, but rather a pre-computed evaluation of the FEA simulation for the corresponding quantity of interest. Therefore, for each location in the LHS design, we look for the nearest pre-computed evaluation available. The nearest evaluation is identified based on cosine similarity of the design variables. In this way, we sample the search space according to the LHS scheme based on the nearest available point. Note, however, that, this might also give rise to duplicates, since a pre-computed evaluation could be nearest to more than one location retrieved from LHS. In this case, we do not allow a duplicate, and rather select the second nearest point available point, from the data set. After the generation of the training data for a particular choice of sample size, we look for testing data points in the remaining data set. These data points are randomly selected based on a size, which is half of the training data size. We then feed the corresponding training and testing data set to all five modeling techniques, and report the RMAE.

In the context of the quality of the optimal solutions, we consider 180 test cases, owing to the combinations of 5 modeling techniques described above, 9 optimization tasks discussed, 2 noise levels, and 2 robustness formulations. An important thing to note in this context is that all design variables take integer values. Therefore, we employ the *Mixed-Integer Surrogate Models*, where-ever possible, to find robust solutions, similar to the approach by (Garrido-Merchán and Hernández-Lobato, 2020)[1]. The two noise levels in this context characterize 0.5 and 1 % (max) deviation in the nominal values of the design variables as: $\mathscr{L} = \{0.005, 0.01\}$, whereas the two robustness formulation considered are MMR and CR.

For the deterministic setting of the uncertainty, i.e., MMR, the compact uncertainty set U is defined as: $\mathrm{U} = [-(L \times R), (L \times R)]$, where $L \in \mathscr{L}$ denotes the choice of the noise level, and R serves as the absolute range of the search variables provided in Table 6.2. Note that in this context, the uncertainty set U is a subset of integer values: $\mathrm{U} \subseteq \mathbb{Z}$, since all design variables take integer values[2]. For the probabilistic setting of the uncertainty, i.e., CR, the uncertainty is modeled according to a discrete uniform probability distribution: $\Delta_{\mathbf{x}} \sim \mathcal{U}(a, b)$, where the boundaries a and b are defined similar to the boundaries of the the set U in the deterministic case.

---

[1] Only the Kriging and Polynomials are transformed to "Mixed-Integer Surrogate Models" in this context since current implementations do not allow other modeling techniques to be extended.

[2] It is not difficult to verify that the internal optimization loop of the MMR in this context refers to the complete enumeration over a full factorial design of all unique noise combinations. Hence, the noise levels in this experimental setup are significantly reduced to be 0.5 and 1 % respectively, as opposed to the 5 and 10 % (and sometimes even 20 %), discussed previously in the thesis. Increasing the noise levels to 5 and 10 % makes solving the problem infeasible since the size of the full factorial design increases rapidly with each new level.

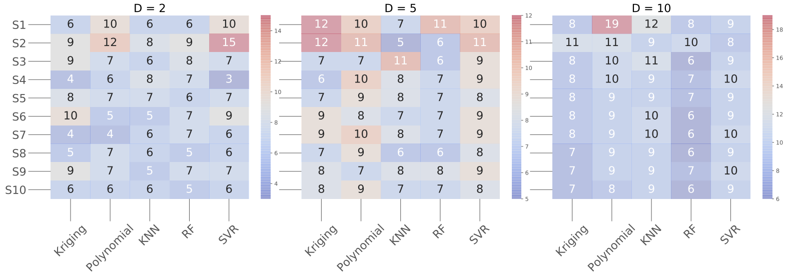
The sample size is set to be $50 \times D$, and the resulting surrogate model each time is directly utilized to find robust solution according to the noise level and the robustness formulation chosen. For the purpose of data generation, we utilize the same procedure applied for modeling accuracy. Moreover, the numerical optimization algorithm employed to find robust solution is L-BFGS-B (Morales and Nocedal, 2011).

6.2.2 Results

Graphs showing the quality of the surrogate models, based on the criterion of RMAE (lower is better), and by varying the training sample size, are presented in Figs. 6.3 – 6.5. Each figure contains three subplots corresponding to three settings of the dimensionality, whereas each subplot indicates the RMAE values for five modeling techniques and ten sample sizes, i.e., S1 – S10. In particular, Fig. 6.3 illustrates the quality in this context for predicting "Mass (kg)". Fig. 6.4 shows the quality regarding "Deformation (mm)", whereas Fig. 6.5 indicates the quality for "Stress (MPa)".

Similar to RMAE, the difference in the quality of the optimal solution, obtained from OSO strategy, to that of the baseline ($\mathcal{DQ}$ cf. Eq. (3.12)), is presented in Figs. 6.6 – 6.8. Similar to modeling accuracy, each figure in this context also contains three subplots corresponding to three settings of the dimensionality, whereas each subplot indicates the $\mathcal{DQ}$ values (lower is better) for five modeling techniques, two noise levels, i.e., NL 1 and 2, and two robustness formulations, i.e., MMR and CR. Note that in this context, $\mathcal{DQ}$ values are computed based on objective function values. The baseline values are computed by solving the corresponding optimization problem – with corresponding settings of task, noise level, robustness formulation, and modeling technique – with a baseline surrogate model, which is trained with all 18 design variables for all car hood designs available. In the following, we report the major findings from our investigation.

- **Sample Size**



D = 2

	Kriging	Polynomial	KNN	RF	SVR
S1	6	10	6	6	10
S2	9	12	8	9	15
S3	9	7	6	8	7
S4	4	6	8	7	3
S5	8	7	7	6	7
S6	10	5	5	7	9
S7	4	4	6	7	6
S8	5	7	6	5	6
S9	9	7	5	7	7
S10	6	6	6	5	6

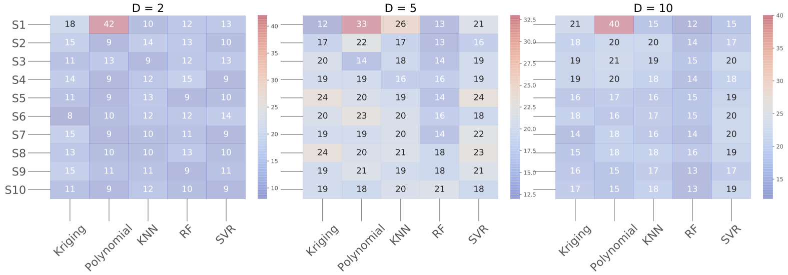
D = 5

	Kriging	Polynomial	KNN	RF	SVR
S1	12	10	7	11	10
S2	12	11	5	6	11
S3	7	7	11	6	9
S4	6	10	8	7	9
S5	7	9	8	7	8
S6	9	8	7	7	9
S7	9	10	8	7	9
S8	7	9	6	6	8
S9	8	7	8	8	9
S10	8	9	7	7	8

D = 10

	Kriging	Polynomial	KNN	RF	SVR
S1	8	19	12	8	9
S2	11	11	9	10	8
S3	8	10	11	6	9
S4	8	10	9	7	10
S5	8	9	9	7	9
S6	8	9	10	6	9
S7	8	9	10	6	10
S8	7	9	9	6	9
S9	7	9	9	7	10
S10	7	8	9	6	9

Figure 6.3: Quality of the surrogate models for all five modeling techniques, and ten sample sizes, i.e., S1 – S10, evaluated based on the criterion of RMAE (lower is better). The surrogate models are trained to predict "Mass (kg)" of the car hood designs.



D = 2

	Kriging	Polynomial	KNN	RF	SVR
S1	18	42	10	12	13
S2	15	9	14	13	10
S3	11	13	9	12	13
S4	14	9	12	15	9
S5	11	9	13	9	10
S6	8	10	12	12	14
S7	15	9	10	11	9
S8	13	10	10	13	10
S9	15	11	11	9	11
S10	11	9	12	10	9

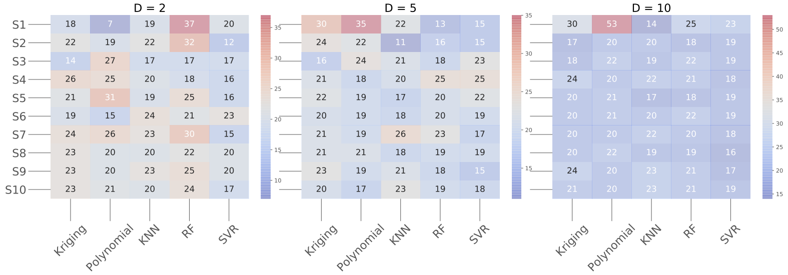
D = 5

	Kriging	Polynomial	KNN	RF	SVR
S1	12	33	26	13	21
S2	17	22	17	13	16
S3	20	14	18	14	19
S4	19	19	16	16	19
S5	24	20	19	14	24
S6	20	23	20	16	18
S7	19	19	20	14	22
S8	24	20	21	18	23
S9	19	21	19	18	21
S10	19	18	20	21	18

D = 10

	Kriging	Polynomial	KNN	RF	SVR
S1	21	40	15	12	15
S2	18	20	20	14	17
S3	19	21	19	15	20
S4	19	20	18	14	18
S5	16	17	16	15	19
S6	18	16	17	15	20
S7	14	18	16	14	20
S8	15	18	18	16	19
S9	16	15	17	13	17
S10	17	15	18	13	19

Figure 6.4: Quality of the surrogate models for all five modeling techniques, and ten sample sizes, i.e., S1 – S10, evaluated based on the criterion of RMAE (lower is better). The surrogate models are trained to predict "Maximum Directional Deformation (mm)" of the car hood designs.



D = 2

	Kriging	Polynomial	KNN	RF	SVR
S1	18	7	19	37	20
S2	22	19	22	32	12
S3	14	27	17	17	17
S4	26	25	20	18	16
S5	21	31	19	25	16
S6	19	15	24	21	23
S7	24	26	23	30	15
S8	23	20	20	22	20
S9	23	20	23	25	20
S10	23	21	20	24	17

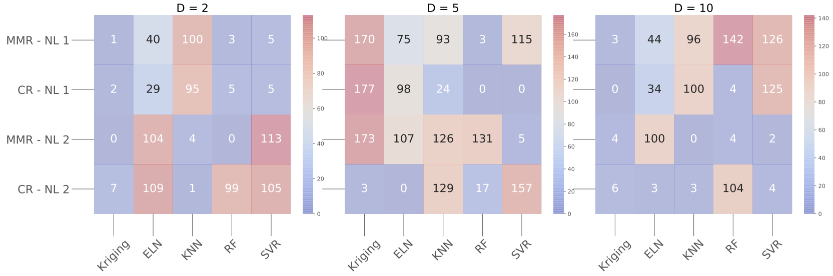
D = 5

	Kriging	Polynomial	KNN	RF	SVR
S1	30	35	22	13	15
S2	24	22	11	16	15
S3	16	24	21	18	23
S4	21	18	20	25	25
S5	22	19	17	20	22
S6	20	19	18	20	19
S7	21	19	26	23	17
S8	21	21	18	19	19
S9	23	19	21	18	15
S10	20	17	23	19	18

D = 10

	Kriging	Polynomial	KNN	RF	SVR
S1	30	53	14	25	23
S2	17	20	20	18	19
S3	18	22	19	22	19
S4	24	20	22	21	18
S5	20	21	17	18	19
S6	20	21	20	22	19
S7	20	20	22	20	18
S8	20	22	19	19	16
S9	24	20	23	21	17
S10	21	20	23	21	19

Figure 6.5: Quality of the surrogate models for all five modeling techniques, and ten sample sizes, i.e., S1 – S10, evaluated based on the criterion of RMAE (lower is better). The surrogate models are trained to predict "Maximum Equivalent Stress (MPa)" for the car hood designs.



D = 2

	Kriging	ELN	KNN	RF	SVR
MMR - NL 1	1	40	100	3	5
CR - NL 1	2	29	95	5	5
MMR - NL 2	0	104	4	0	113
CR - NL 2	7	109	1	99	105

D = 5

	Kriging	ELN	KNN	RF	SVR
MMR - NL 1	170	75	93	3	115
CR - NL 1	177	98	24	0	0
MMR - NL 2	173	107	126	131	5
CR - NL 2	3	0	129	17	157

D = 10

	Kriging	ELN	KNN	RF	SVR
MMR - NL 1	3	44	96	142	126
CR - NL 1	0	34	100	4	125
MMR - NL 2	4	100	0	4	2
CR - NL 2	6	3	3	104	4

Figure 6.6: Quality of the surrogate models for all five modeling techniques, and for all four cases based on the combinations of two noise levels, i.e., NL 1 and 2, and two robustness formulations, i.e., MMR and CR. The quality is measured according to the criterion of $\mathcal{DQ}$ (lower is better), introduced in Chapter 3. The goal of optimization in this context is to minimize the "Mass (kg)" of the geometry.

An analysis of the averaged results in Figs. 6.3 – 6.5 w.r.t. sample size indicates that we can achieve a good approximation quality with reasonable sample size in most test scenarios. Increasing the sample size does not strictly increase the approximation quality of the surrogate model. However, the surrogate models with the highest number of training points, i.e., S10, usually produce one of the best averaged results. In a loosely speaking manner, our observations re-affirm the generally employed heuristic in model-assisted optimization, which states that the initial sample size can be set linearly in terms of dimensionality (Forrester et al., 2008; Jurecka, 2007).

- **Modeling Technique**

  An analysis of the averaged results in Figs. 6.3 – 6.5 w.r.t. modeling techniques indicates that, in general, all five modeling techniques, produce good approximations, for most test scenarios. RF produces the best averaged result in terms of the first and second tasks, i.e., "Mass (kg)" and "Deformation (mm)", whereas KNN performs best in terms of the third task, i.e., "Stress (MPa)". This gives us a new perspective of considering RF and KNN as well, when modeling the real-world complex objective functions.

  In terms of the averaged results in Figs. 6.6 – 6.8 w.r.t. modeling techniques, we find that the optimal solutions obtained from RF achieve the highest quality for the first task, whereas Kriging produces the best solutions in terms of the second task. Optimal solutions obtained from KNN perform

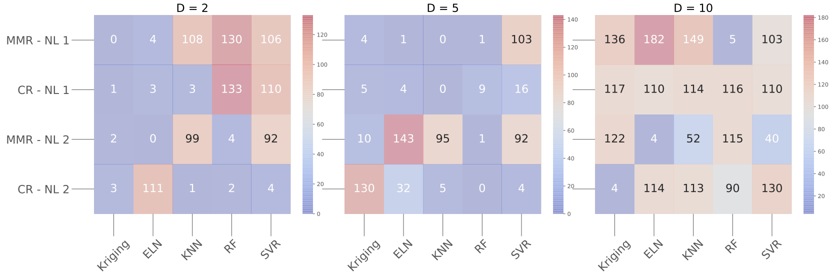


Figure 6.7: Quality of the surrogate models for all five modeling techniques, and for all four cases based on the combinations of two noise levels, i.e., NL 1 and 2, and two robustness formulations, i.e., MMR and CR. The quality is measured according to the criterion of $\mathcal{DQ}$ (lower is better). The goal of optimization in this context is to minimize the "Maximum Directional Deformation (mm)" of the car hood design.

the best in terms of the third task. Overall, in terms of the quality of the optimal solutions, we conclude Kriging produces excellent results in most test scenarios. An important thing to note here is that we do not achieve the higher quality expected from polynomial surrogates.

- **Applicability**

  Based on the overall performance of the surrogate models in terms of modeling accuracy, and quality of the robust optimal solutions, we deem surrogate modeling to be applicable for efficiently solving optimization problems under uncertainty. This is due to the fact that in most cases, the quality of the approximation obtained from Kriging, SVM, RF and KNN is good enough to employ a surrogate to find robust solution. The quality of the optimal solutions in most cases is also satisfactory, since the optimal function value found on the model surface is close to the baseline/ground truth in most cases.

6.3 Bayesian Optimization

We are interested in benchmarking the performance of the Bayesian optimization algorithm (cf. Alg. 1), which is based on the "sequential model-based optimization" framework, to find the optimal solutions in an efficient manner (Jones et al., 1998). In Chapter 4, we extended the Bayesian optimization algorithm to the robust

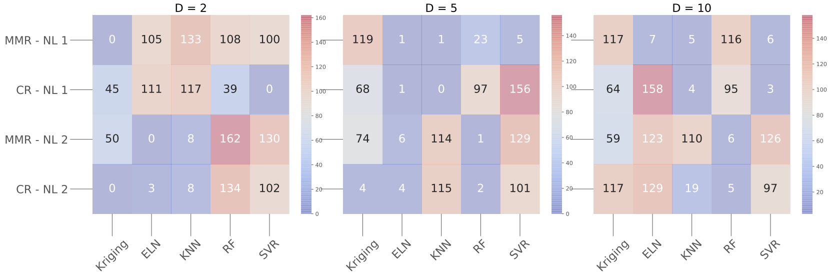


Figure 6.8: Quality of the surrogate models for all five modeling techniques, and for all four cases based on the combinations of two noise levels, i.e., NL 1 and 2, and two robustness formulations, i.e., MMR and CR. The quality is measured according to the criterion of $\mathcal{DQ}$ (lower is better). The goal of optimization in this context is to minimize the "Maximum Equivalent Stress (MPa)" of the car hood design.

scenario, in order to efficiently find the robust solutions (Ullah et al., 2021). We now emphasize on the performance of the Bayesian optimization algorithm, as well as the choice of the sampling infill criterion. To this end, we study three sampling infill criterion: LCB, EIC, and MGFI, which have been introduced earlier (cf. Chapter 4) in the thesis.

6.3.1 Experimental Setup

We start with 9 optimization tasks, introduced earlier in this Chapter. These 9 optimization tasks refer to the minimization of three structural mechanics performance indicators: "Mass (kg)", "Maximum Directional Deformation (mm)", and "Maximum Equivalent Stress (MPa)", each for three settings of the dimensionality as: $\mathscr{D} = \{2, 5, 10\}$. Furthermore, we consider two levels of additive noise as: $\mathscr{L} = \{0.005, 0.01\}$, and two robustness formulations: MMR and CR, respectively. In addition, we consider three sampling infill criteria for the Bayesian optimization algorithm: LCB, EIC, and MGFI, respectively. This gives rise to a total of 108 test cases, owing to the unique combinations of 9 optimization tasks, 2 noise levels, 2 robustness formulations and 3 sampling infill criteria.

In our study, the size of the initial training data is set to be $5 \times D$, where $D \in \mathscr{D}$ denotes the corresponding setting of the dimensionality. Likewise, the maximum number of iterations for Bayesian optimization is set to be $30 \times D$. Note that our Kriging surrogate is based on the "absolute exponential" kernel (Rasmussen and Williams, 2006), and we standardize the function responses: $\mathbf{y} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots,$

$f(\mathbf{x}_N)]^\top$, before constructing the Kriging surrogate $\mathcal{K}_f$. Furthermore, the Kriging surrogate is based on the implementation of (Garrido-Merchán and Hernández-Lobato, 2020), which transforms the model to handle variables that take integer values. The hyper-parameters β and t for LCB and MGFI are set similar to the setup described in Section 4.3.

For the parallel execution of Bayesian optimization for each of the 108 test cases considered, we utilize Das-5 (Bal et al., 2016), where each standard node has a dual 8-core 2.4 GHz (Intel Haswell E5-2630-v3) cpu configuration and 64 GB memory. We implement our experiments in python 3.7.0 with the help of scikit-learn module (Pedregosa et al., 2011). The performance assessment of the robust solutions in our experiments is based on 15 independent runs $\mathscr{R}$ of the Bayesian optimization algorithm for each of the 108 test cases considered. Note that for each trial, i.e., the unique combination of the independent run and the test case, we ensure the same configuration of hardware and software to account for fairness. Furthermore, in each trial, we measure the cpu time for all iterations of the algorithm to measure the efficiency.

After the successful parallel execution of all trials, we evaluate the performance of our robust solutions based on quality difference $\mathcal{DQ}$ from the baseline (cf. Eq. (3.12))). Note that $\mathcal{DQ}$ in this case is based on the space of objective function values[1]. After this, we perform six different analyses to answer the questions outlined earlier. The first two type of analyses are referred to as the fixed cpu time analysis, and the fixed iteration analysis respectively. In fixed cpu time analysis, we fix 50 different settings of the cpu time, and report the best $\mathcal{DQ}$ (the lowest) for each trial. The $\mathcal{DQ}$ in this context is averaged over all 50 settings of the cpu time. For fixed iteration analysis, we fix 30 different settings of the iterations (checkpoints) to report the best $\mathcal{DQ}$ (the lowest) for each trial. The $\mathcal{DQ}$ in this context is also averaged over all 30 checkpoints.

After fixed cpu time and fixed iteration analysis, we perform a fixed target analysis. The fixed target analysis is also based on two different settings: by fixing a target $\mathcal{DQ}$ and reporting the cpu time as well as the number of iterations taken to reach that target. We fix ten different settings for the target in this context, and the corresponding cpu time and iterations are averaged over these target values. Note that each target describes the minimum desirable quality threshold of the robust

---

[1] In this study, we do not divide $\mathcal{DQ}$ with the number of independent runs $\mathscr{R}$ as Eq. (3.12) suggests, but rather report all trials.

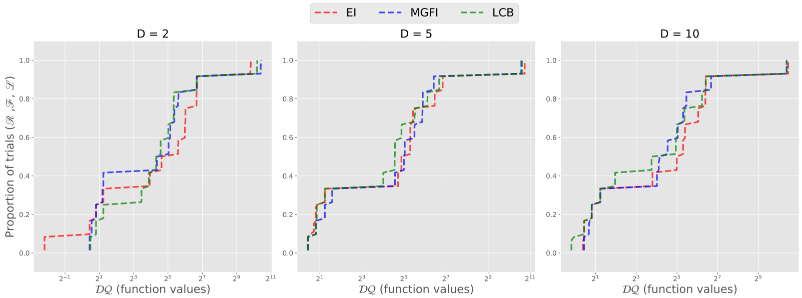


Figure 6.9: Fixed cpu time analysis for car hood design optimization. Each subplot contains 3 Ecdf curves based on 3 infill criteria discussed. Each Ecdf curve is based on 180 data points owing to the combinations of 6 optimization tasks $\mathscr{F}$ (3 optimization scenarios and 2 robustness criteria), 2 noise levels $\mathscr{L}$, and 15 independent runs $\mathscr{R}$.

solution. If such a quality is never achieved, we report the penalized cpu time and penalized number of iterations respectively. The penalized cpu time is set to be $D \times T_{\max}$, whereas penalized number of iterations is set to be $D \times N_{\max}$. Here D is the corresponding setting of the dimensionality, and $N_{\max}$ and $T_{\max}$ indicate the maximum number of iterations of the BO algorithm and the cpu time taken to execute it. After the fixed budget and fixed target analyses, we also report the average cpu time per iteration for the BO algorithm. In addition, we also report $T_{\max}$: the accumulated cpu time at the last iteration of the BO algorithm, for each trial.

6.3.2 Results

The results originating from this are shown in Figs. 6.9 – 6.13. Each of these figures contains the graphs for a particular type of analysis. In particular, Fig. 6.9 shares the results based on a fixed cpu time analysis. The figure contains 3 different subplots corresponding to 3 different settings of the dimensionality. Each subplot shares the empirical cumulative distribution function (ecdf) of DQ for 3 different sampling infill criteria considered.

Likewise, Fig. 6.10 shares the ecdf plots corresponding to fixed iteration analysis, whereas the analysis based on fixed targets is presented in Fig. 6.11. The average cpu time per iteration of the BO algorithm to find robust solutions is presented

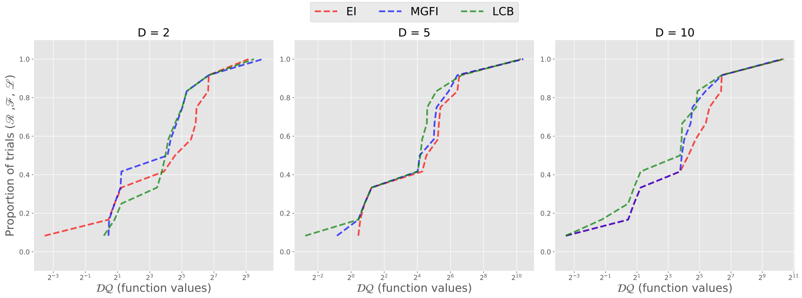


Figure 6.10: Fixed Iteration analysis. Each subplot contains 3 Ecdf curves based on 3 infill criteria discussed. Each Ecdf curve is based on 180 data points owing to the combinations of 6 optimization tasks $\mathscr{F}$ (due to 2 optimization scenarios and 2 robustness criteria), 2 noise levels $\mathscr{L}$, and 15 independent runs $\mathscr{R}$.

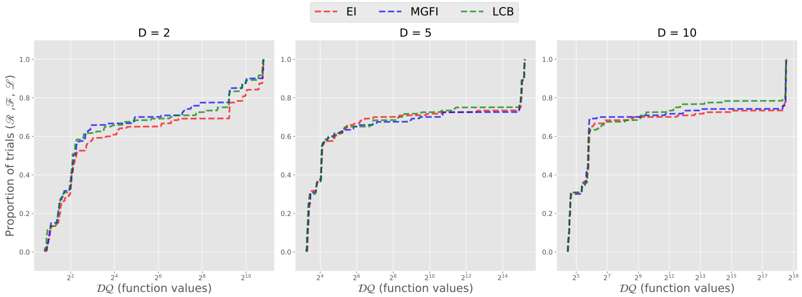


Figure 6.11: Fixed Target analysis. Each subplot contains 3 Ecdf curves based on 3 infill criteria discussed. Each Ecdf curve is based on 180 data points owing to the combinations of 6 optimization tasks $\mathscr{F}$ (3 optimization scenarios and 2 robustness criteria), 2 noise levels $\mathscr{L}$, and 15 independent runs $\mathscr{R}$.

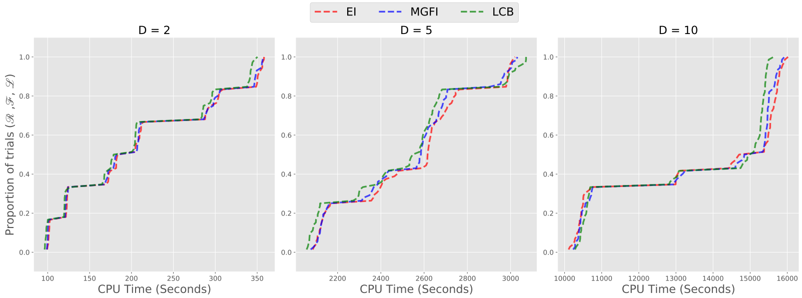


Figure 6.12: Average cpu time per iteration for the BO algorithm. Each subplot contains 3 Ecdf curves based on 3 infill criteria discussed.

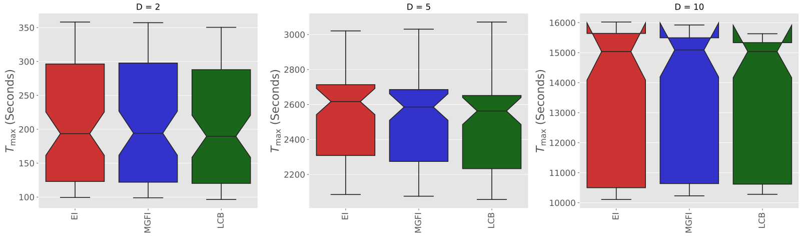


Figure 6.13: Maximum accumulated cpu time: $T_{\max}$ for each trial, for each setting of the dimensionality.

in Fig. 6.12. Lastly, we present the maximum accumulated cpu time: $T_{\max}$, for each trial in the form of box plots in Fig. 6.13.

In the following, we report the major findings of these results.

- **Applicability of the Bayesian Optimization**

  Based on the results presented in Figs. 6.9 – 6.10, we deem Bayesian optimization as a promising heuristic to efficiently find robust solutions in practical scenarios. This is due to the fact that the empirical success rate of the Bayesian optimization algorithm is satisfactory. For instance, if we cut-off the DQ values at 8, the empirical success rate is between 35-40 %.

- **Factors with Significant Influence**

  Based on the results presented in Figs. 6.9 – 6.11, we find that dimensionality affects the quality of the robust solutions. Because of the dimensionality, the computational budget, i.e., whether measured in cpu time or number of iterations, also affects the quality of the robust solutions in a significant manner. For instance, in Fig. 6.9, we see that the empirical success measured at 2^4 seconds (cpu time) is more than 40 % for trials belonging to two-dimensional problems. On the other hand, the empirical success rate drops to under 35 % when dimensionality is increased (for five and ten-dimensional cases).

- **Impact of Infill Criterion**

  In the context of fixed budget analyses, the performance of LCB_{eff} and $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$ is superior to that of the $\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})]$. In the context of fixed target analysis, we we do not observe a significant difference in the performance for most trials. Hence, we cannot find a clear winner in this case.

- **Infill Criterion for Practical Scenarios**

  For choosing a sampling infill criteria for practical scenarios, we emphasize on the average running cpu time per iteration (ARCTPI), as well as the maximum cpu time required for an independent run: $T_{\max}$, in addition to the fixed budget and fixed target analyses. In the context of ARCTPI, i.e., Fig. 6.12, we deem $\mathcal{M}_{\text{eff}}(\mathbf{x};t)$ and LCB_{eff} performing better than $\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})]$. In the context of $T_{\max}$, i.e., we deem LCB_{eff} performing superior to its competitors for two and five-dimensional problems. Combining the performance in the context of fixed budget analyses, fixed target analysis, ARCTPI, and $T_{\max}$, we deem LCB_{eff} and $\mathcal{M}_{\text{eff}}(\mathbf{x};t)$ as suitable sampling infill criteria.

6.4 Summary and Discussion

In this chapter, we benchmarked the applicability of surrogate modeling on a real-world engineering case study. To this end, we considered a benchmark engineering case study based on the design of car hood frames. The associated data set contains over 10,000 3D mesh geometries for variants of card hood frames. This data set was generated through an automated, industry-grade CAD workflow, described in (Ramnath et al., 2019), and further benchmarked in (Wollstadt et al., 2022). The data set provided realistic designs of car hood frames, which were validated by experts with respect to realism, manufacturability, variability, and performance.

Based on this data set, we focused on two goals, which emphasized on benchmarking the performance of "one-shot optimization" strategy (Ta'asan et al., 1992) and the Bayesian optimization algorithm (Jones et al., 1998) for finding robust solutions. Our findings validate the performance of Kriging (Morales and Nocedal, 2011) as one of the most important modeling techniques in surrogate modeling. Furthermore, we observed the promising nature of ensemble methods, i.e., Random Forest, to effectively model the objective function in practical scenarios. We also validated the commonly-employed heuristic of utilizing a linear sample size to construct the model (Jurecka, 2007). Finally, in this context, we were satisfied with the quality of the optimal solutions obtained from surrogate modeling.

In the context of Bayesian optimization (Jones et al., 1998), we validated the impact of dimensionality, and consequently, the computational budget, on the performance of the algorithm. Furthermore, We validated the performance of the

“Moment-Generating Function of the Improvement” as an effective sampling infill criterion in Bayesian optimization, in addition to the “Lower Confidence Bound”. However, we could not validate the highly competitive nature of the “Expected-Improvement” Criterion. We believe this is due to the fact that the our design variables and noise settings take integer (rather than continuous) values[1].

It is important to note that our study has certain limitations. Ideally, we should have constructed the surrogate models (in both cases) from the continuous (latent) variables, derived from the 3D point cloud auto-encoders (Wollstadt et al., 2022), which in turn could have been constructed from the car hood geometries. This, however, would have given rise to further difficulties, since such design variables are generally not interpretable. Furthermore, defining the bounds and the constraints for such latent variables is a difficult task, i.e., again, due to the fact that they are not interpretable in the nominal sense. This, in turn, would also have meant that we cannot specify uncertainty and noise since that requires a precise understanding of the bounds of the design variables. Combining these points, we believe our methodology in the experimental setups makes more sense from a practical point of view, since it validates some of our earlier findings, and offers us a new perspective to learn from.

---

[1] We believe the performance of the ensemble methods is also excellent due to the same reason, i.e., integer values for design and noise variables

CHAPTER 7

Conclusion and Outlook

In this thesis, several important aspects of robust optimization (Ben-Tal et al., 2009) are empirically investigated in depth. **Chapter 1** introduces the fundamental research questions of the thesis. The first three questions are related to each other, in that they deal with the applicability of surrogate modeling to find robust solution in an efficient manner, by taking into account a list of factors such as noise level, problem landscape, dimensionality, and design of experiment. Note that the notion of efficiency in this context is based on the utilization of computational resources.

We made two attempts to answer these questions in a comprehensive manner. The first one is based on "one-shot optimization" and described in detail in **Chapter 3**. The key findings from this investigation indicate the following points.

1. Kriging, Response Surface Models (Polynomials), and Support Vector Machines construct good quality surrogate models with linear sample sizes. These models can then be utilized to estimate robust solution. The robust solutions estimated with these models are very close to the baseline.
2. Dimensionality is a detrimental factor on the quality of the surrogate models, whereas the noise level does not play a significant role in this context.

Due to the significant impact of dimensionality on the quality of surrogate models, we devote the rest of **Chapter 3** to find dimensionality reduction techniques that can be utilized for efficient surrogate modeling. To this end, we empirically compare the performance of Principal Component Analysis, Kernel Principal Component Analysis, Autoencoders, and Variational Autoencoders. Following points summarize the key findings from this study.

1. Based on the criteria of modeling accuracy, Autoencoders are the most promising dimensionality reduction technique.
2. Based on the quality of optimal solutions obtained from low dimensional surrogate models, Principal Component Analysis perform superior to the other competitors.
3. The quality of the optimal solutions obtained after dimensionality reduction can be very low in some cases. Therefore, dimensionality reduction is not always feasible.

In **Chapter 4**, we attempt to answer the first and the third research questions of our thesis with “sequential model-based optimization” framework (Jones et al., 1998). We refer to it as the “Bayesian optimization” framework, since we always employ Kriging (or Gaussian process) as the modeling technique. Here, we also consider the impact of the acquisition function to find robust solution.

Following points summarize the applicability of the Bayesian optimization approach to find robust solution.

1. The Bayesian optimization algorithm is to be extended to account for parametric uncertainty in the search variables. The extended Bayesian optimization algorithm is computationally tractable, and able to find robust solutions efficiently as backed by the empirical investigation.
2. Dimensionality and computational budget play a significant role in the performance of our extended version.
3. Noise level does not directly affect the quality of the robust solution in an adverse manner.
4. “Expected Improvement” criterion and “Moment-Generating Function of the Improvement” prove to be excellent choices for the acquisition function, as opposed to the “Lower Confidence Bound”, which is affected adversely if the dimensionality increases.
5. The evaluation of the “Lower Confidence Bound” is also computationally costlier when compared with the other two sampling infill criteria.

Chapter 5 focuses on the fourth research question of our thesis – What is the impact of robustness formulation/criterion in efficiently solving black-box problems subject to uncertainty and noise, and which robustness formulations are recommended to practitioners with regards to computational efficiency?

An empirical study (Ullah et al., 2022) is conducted to answer this questions. The major findings from this study are as follows.

1. In the situations where the designer cannot afford the computational budget beyond a certain threshold, “mini-max robustness”, “mini-max regret robustness”, “expectation-based robustness”, and “composite robustness” can be utilized to find robust solutions in an efficient manner.
2. On the other hand, if the designer cannot compromise on the quality of the solution, “mini-max robustness” is the most efficient robustness criterion to be employed.
3. The average cpu time per iteration of the Bayesian optimization algorithm is lowest when “mini-max robustness” is employed.

Chapter 6 emphasizes on benchmarking the surrogate modeling approaches, described earlier in the thesis, on a real-world engineering application. To this end, we emphasize on the design optimization of car hood frames, obtained from (Ramnath et al., 2019). Our findings from this case study validate some of our earlier results, and also provide a new perspective in the applicability of surrogate modeling. For instance, we observe that Kriging and Random Forest generally perform excellently in the context of “one-shot optimization”, and the sample size can be set linearly in terms of dimensionality. Furthermore, we note that “Moment-Generating Function of the Improvement” and “Lower Confidence Bound” perform competitively as the sampling infill criteria, and that dimensionality affects the quality of the optimal solutions in an adverse manner.

7.1 Challenges and Opportunities

Robust optimization (Ben-Tal et al., 2009) has received a lot of attention in the last two decades due to the advancements in several field of engineering. For instance, shortening the product-development cycle, reducing the resource consumption during the complete process, and creating more balanced and innovative products has become a desirable outcome in the field of product engineering. To achieve this, designers have to account for uncertainties and noise in an efficient manner. Therefore, a practical approach to robust optimization is necessitated.

When accounting for uncertainties and noise, we believe “environmental variables” (or operating conditions of the product) have been overlooked in the literature. As stated earlier, they can impact the quality of an optimal design in an adverse manner. Therefore, effectively modeling the uncertainties surrounding these “environmental variables” is of huge significance.

In the context of parametric uncertainties in the search variables, the choice of robustness criterion is very important due to three main reasons: “computational cost of robustness”, “price of robustness”, and “problem landscape induced by the robustness criterion”. We believe all three of these aspects have been overlooked in the literature. The first one of these, namely the “computational cost of robustness” has been studied in an empirical fashion in **Chapter 5**, but the findings need to be validated with real-world engineering case studies. Furthermore, the “price of robustness”: the aspect of compromising on the performance/optimality to achieve robustness/stability, also needs to be systematically studied. Lastly, it may be the case that the “problem landscape induced by the robustness criterion” encompasses certain attributes, making the robust counterpart easier or more difficult to solve. We believe this aspect of robustness criterion also needs to be systematically studied.

In practical scenarios, high dimensionality poses a major obstacle in the applicability of surrogate modeling. Albeit we discuss the issue of high dimensionality at great length in **Chapter 3**, further research is necessary to validate our findings for the robust scenario. In particular, answering the following is very important:

“In the face of high dimensionality, what can be done to find robust solutions in an efficient and effective manner via surrogate modeling? Which dimensionality reduction techniques are most suitable in this context? What factors influence the performance of the dimensionality reduction techniques in this context?”

When extending the Bayesian optimization algorithm to the robust scenario, certain practical compromises have to be made, in order to effectively model the true “robust” response of the function. For instance, we assumed in **Chapter 4** that the true “robust” response of the function can also be modeled according to a Gaussian process, similar to the nominal scenario. However, this approach is not entirely rigorous, as we are not estimating the true joint posterior distribution of all search points induced by the uncertainty. Estimating this posterior distribution would be a significant contribution to the literature, as it would enable us to extend the Bayesian optimization algorithm to the robust scenario in a seamless fashion (from the nominal case).

Benchmarking the empirical performance of the Bayesian optimization algorithm in robust optimization also entails an interesting opportunity for researchers. To this end, we made an attempt in **Chapter 4**, which includes the variability in problem landscape, dimensionality, noise level, robustness and sampling infill criteria. Note, however, that, further research is necessary to cover a broad spectrum of test scenarios.

Lastly, observing the synergies between surrogate modeling and machine learning, we note that "robustness" also needs to be incorporated in machine learning. This is due to the fact that learning and mining in the presence of uncertain (industrial) data poses additional challenges for the modeling techniques. Therefore, these modeling techniques need to be extended to care for "robustness", in order to effectively account for the uncertainties present in the training data. A major contribution in this direction is to extend the Support Vector Machines to the robust scenario, based on the conceptual framework proposed in (Ben-Tal et al., 2009). Similarly, the adaptation of the "Variational Recurrent Models" to account for irregular, highly-sporadic, and asynchronous sequential data is also an important contribution (Ullah et al., 2020b) in this direction.

Bibliography

Alarie, S., C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel (2021). Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization 9*, 100011.

Audet, C. and W. Hare (2017). *Derivative-free and blackbox optimization*, Volume 2. Springer.

Audet, C. and M. Kokkolaras (2016). Blackbox and derivative-free optimization: theory, algorithms and applications.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research 3*(Nov), 397–422.

Averbakh, I. and Y.-B. Zhao (2008). Explicit reformulations for robust optimization problems with general uncertainty sets. *SIAM Journal on Optimization 18*(4), 1436–1466.

Bäck, T., D. B. Fogel, and Z. Michalewicz (2018). *Evolutionary computation 1: Basic algorithms and operators.* CRC press.

Bäck, T. and H.-P. Schwefel (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation 1*(1), 1–23.

Bagheri, M., M. Miri, and N. Shabakhty (2016). Fuzzy reliability analysis using a new alpha level set optimization approach based on particle swarm optimization. *Journal of Intelligent & Fuzzy Systems 30*(1), 235–244.

Bal, H., D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff (2016). A medium-scale distributed system for computer science research: Infrastructure for the long term. *Computer 49*(5), 54–63.

Balakrishnan, A. V. (2012). *Introduction to optimization theory in a Hilbert space*, Volume 42. Springer Science & Business Media.

Barthelemy, J.-F. and R. T. Haftka (1993). Approximation concepts for optimum structural designa review. *Structural optimization 5*(3), 129–144.

Beasley, D., D. R. Bull, and R. R. Martin (1993). A sequential niche technique for multimodal function optimization. *Evolutionary computation 1*(2), 101–125.

Belkin, M. and P. Niyogi (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pp. 585–591.

Bellman, R. E. and L. A. Zadeh (1970). Decision-making in a fuzzy environment. *Management science 17*(4), B–141.

Belotti, P., C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan (2013). Mixed-integer nonlinear optimization. *Acta Numerica 22*, 1–131.

Ben-Tal, A., L. El Ghaoui, and A. Nemirovski (2009). *Robust optimization*, Volume 28. Princeton University Press.

Ben-Tal, A., L. E. Ghaoui, and A. Nemirovski (2009). *Robust Optimization*, Volume 28 of *Princeton Series in Applied Mathematics*. Princeton University Press.

Ben-Tal, A., A. Goryashko, E. Guslitzer, and A. Nemirovski (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical programming 99*(2), 351–376.

Bengio, Y., L. Yao, G. Alain, and P. Vincent (2013). Generalized denoising autoencoders as generative models. In *Advances in neural information processing systems*, pp. 899–907.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl (2011). Algorithms for hyperparameter optimization. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pp. 2546–2554.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl (2011). Algorithms for hyperparameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, USA, pp. 2546–2554. Curran Associates Inc.

Bergstra, J., D. Yamins, and D. Cox (2013a). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pp. 115–123. PMLR.

Bergstra, J., D. Yamins, and D. D. Cox (2013b). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML*.

Berlinet, A. and C. Thomas-Agnan (2011). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media.

Bertsimas, D., D. B. Brown, and C. Caramanis (2011). Theory and applications of robust optimization. *SIAM review 53*(3), 464–501.

Bertsimas, D. and N. Koduri (2022). Data-driven optimization: A reproducing kernel hilbert space approach. *Operations Research 70*(1), 454–471.

Bertsimas, D., O. Nohadani, and K. M. Teo (2010). Robust optimization for unconstrained simulation-based problems. *Operations research 58*(1), 161–178.

Beyer, H.-G. and B. Sendhoff (2007). Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering 196*(33-34), 3190–3218.

Bhosekar, A. and M. Ierapetritou (2018). Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering 108*, 250–267.

Bishop, C. M. (2007). *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer.

Boggs, P. T. and J. W. Tolle (1995). Sequential quadratic programming. *Acta numerica 4*, 1–51.

Bossek, J., P. Kerschke, A. Neumann, F. Neumann, and C. Doerr (2019). One-shot decision-making with and without surrogates. *CoRR abs/1912.08956*.

Box, G. E. and N. R. Draper (1987). *Empirical model-building and response surfaces*. John Wiley & Sons.

Bubeck, S., R. Munos, and G. Stoltz (2009). Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, pp. 23–37. Springer.

Chen, S., J. Montgomery, and A. Bolufé-Röhler (2015). Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence 42*(3), 514–526.

Chowdhury, S. and S. Taguchi (2016). *Robust Optimization: World's Best Practices for Developing Winning Vehicles.* John Wiley & Sons.

Coello, C. C. and M. S. Lechuga (2002). Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, Volume 2, pp. 1051–1056. IEEE.

Conn, A. R., K. Scheinberg, and L. N. Vicente (2009). *Introduction to derivative-free optimization.* SIAM.

Cox, G. and W. Cochran (1957). *Experimental designs.* New York.

Cristianini, N. and J. Shawe-Taylor (2004). *Support Vector Machines and other kernel-based learning methods.* Cambridge.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management science 1*(3-4), 197–206.

Das, I. (2000). Robustness optimization for constrained nonlinear programming problems. *Engineering Optimization+ A35 32*(5), 585–618.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation 6*(2), 182–197.

Demartines, P. and J. Hérault (1997). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on neural networks 8*(1), 148–154.

Der Kiureghian, A. and O. Ditlevsen (2009). Aleatory or epistemic? does it matter? *Structural safety 31*(2), 105–112.

El Ghaoui, L. and H. Lebret (1997). Robust solutions to least-squares problems with uncertain data. *SIAM Journal on matrix analysis and applications 18*(4), 1035–1064.

Emmerich, M. (2005). *Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels.* Ph. D. thesis, Dortmund, Univ., Diss., 2005.

Emmerich, M. T. and A. H. Deutz (2018). A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing 17*(3), 585–609.

Fan, J. (2007). Variable screening in high-dimensional feature space. In *Proceedings of the 4th international congress of chinese mathematicians*, Volume 2, pp. 735–747.

Ferreira, S. C., R. Bruns, H. Ferreira, G. Matos, J. David, G. Brandão, E. P. da Silva, L. Portugal, P. Dos Reis, A. Souza, et al. (2007). Box-behnken design: an alternative for the optimization of analytical methods. *Analytica chimica acta 597*(2), 179–186.

Fisher, R. A. (1936). Design of experiments. *Br Med J 1*(3923), 554–554.

Forrester, A., A. Sobester, and A. Keane (2008). *Engineering design via surrogate modelling: a practical guide.* John Wiley & Sons.

Frazier, P. I. (2018). Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pp. 255–278. Informs.

Gabrel, V., C. Murat, and A. Thiele (2014). Recent advances in robust optimization: An overview. *European journal of operational research 235*(3), 471–483.

Garrido-Merchán, E. C. and D. Hernández-Lobato (2020). Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing 380*, 20–35.

Goh, T. (1993). Taguchi methods: some technical, cultural and pedagogical perspectives. *Quality and Reliability Engineering International 9*(3), 185–202.

Golub, G. H. and C. F. Van Loan (2013). *Matrix computations.* JHU press.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep learning.* MIT press.

Gramacy, R. B. (2020). *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences.* Chapman and Hall/CRC.

Gregory, C., K. Darby-Dowman, and G. Mitra (2011). Robust optimization and portfolio selection: The cost of robustness. *European Journal of Operational Research 212*(2), 417–428.

Hansen, N., A. Auger, R. Ros, O. Mersmann, T. Tusar, and D. Brockhoff (2021). COCO: a platform for comparing continuous optimizers in a black-box setting. *Optim. Methods Softw. 36*(1), 114–144.

Hastie, T., R. Tibshirani, J. H. Friedman, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*, Volume 2. Springer.

Herrmann, J. W. (1999). A genetic algorithm for minimax optimization problems. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Volume 2, pp. 1099–1103. IEEE.

Hinton, G. E. and R. S. Zemel (1994). Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pp. 3–10.

Hoffman, M., B. Shahriari, and N. Freitas (2014). On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *Artificial Intelligence and Statistics*, pp. 365–374. PMLR.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011). 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*, Volume 6683 of *Lecture Notes in Computer Science*, pp. 507–523. Springer.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011). Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pp. 507–523. Springer.

Hutter, F., H. H. Hoos, K. Leyton-Brown, and T. Stützle (2009). Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research 36*, 267–306.

Ionescu-Bujor, M. and D. G. Cacuci (2004). A comparative review of sensitivity and uncertainty analysis of large-scale systemsi: Deterministic methods. *Nuclear science and engineering 147*(3), 189–203.

Jiang, R., J. Wang, M. Zhang, and Y. Guan (2013). Two-stage minimax regret robust unit commitment. *IEEE Transactions on Power Systems 28*(3), 2271–2282.

Jin, R., X. Du, and W. Chen (2003). The use of metamodeling techniques for optimization under uncertainty. *Structural and Multidisciplinary Optimization 25*(2), 99–116.

Jin, Y. and J. Branke (2005). Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on evolutionary computation 9*(3), 303–317.

Johnson, D. H. (2006). Signal-to-noise ratio. *Scholarpedia 1*(12), 2088.

Johnson, M. E., L. M. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *Journal of statistical planning and inference 26*(2), 131–148.

Jolliffe, I. T. (1986). Principal components in regression analysis. In *Principal component analysis*, pp. 129–155. Springer.

Jolliffe, I. T. and J. Cadima (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374*(2065), 20150202.

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

Jurecka, F. (2007). *Robust design optimization based on metamodeling techniques.* Ph. D. thesis, Technische Universität München.

Kall, P., S. W. Wallace, and P. Kall (1994). *Stochastic programming.* Springer.

Kalman, D. (1984). The generalized vandermonde matrix. *Mathematics Magazine 57*(1), 15–21.

Keane, A., A. Forrester, and A. Sobester (2008). *Engineering design via surrogate modelling: a practical guide.* American Institute of Aeronautics and Astronautics, Inc.

Kim, M., T. Hiroyasu, M. Miki, and S. Watanabe (2004). SPEA2+: improving the performance of the strength pareto evolutionary algorithm 2. In *Parallel*

Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings, Volume 3242 of *Lecture Notes in Computer Science*, pp. 742–751. Springer.

Kingma, D. P. and M. Welling (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.*

Kingma, D. P., M. Welling, et al. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning 12*(4), 307–392.

Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi (1983). Optimization by simulated annealing. *science 220*(4598), 671–680.

Kleijnen, J. P., W. Van Beers, and I. Van Nieuwenhuyse (2012). Expected improvement in efficient global optimization through bootstrapped kriging. *Journal of global optimization 54*(1), 59–73.

König, O. and M. Wintermantel (2004). Cad-based evolutionary design optimization with catia v5. *Proceedings of 1st Weimar Optimization and Stochastic Days WOST, Weimar*, 1–30.

Korte, B. H., J. Vygen, B. Korte, and J. Vygen (2011). *Combinatorial optimization*, Volume 1. Springer.

Kruisselbrink, J. W. (2012). *Evolution strategies for robust optimization.* Ph. D. thesis, Leiden University.

Lee, K.-H. and G.-J. Park (2001). Robust optimization considering tolerances of design variables. *Computers & Structures 79*(1), 77–86.

Lewis, R. M., V. Torczon, and M. W. Trosset (2000). Direct search methods: then and now. *Journal of computational and Applied Mathematics 124*(1-2), 191–207.

Liu, J., Z. Han, and W. Song (2012). Comparison of infill sampling criteria in kriging-based aerodynamic optimization. In *28th congress of the international council of the aeronautical sciences*, pp. 23–28.

Maaten, L. v. d. and G. Hinton (2008). Visualizing data using t-sne. *Journal of machine learning research 9*(Nov), 2579–2605.

McIlhagga, M., P. Husbands, and R. Ives (1996). A comparison of search techniques on a wing-box optimisation problem. In *International Conference on Parallel Problem Solving from Nature*, pp. 614–623. Springer.

Merkuryeva, G. and V. Bolshakovs (2011). Benchmark fitness landscape analysis. *International Journal of Simulation Systems, Science and Technology 12*(2), 38–45.

Močkus, J. (1975). On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pp. 400–404. Springer.

Močkus, J. (2012). *Bayesian approach to global optimization: theory and applications*, Volume 37. Springer Science & Business Media.

Montgomery, D. C. (2017). *Design and analysis of experiments.* John wiley & sons.

Morales, J. L. and J. Nocedal (2011). Remark on algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathematical Software (TOMS) 38*(1), 1–4.

Mulvey, J. M., R. J. Vanderbei, and S. A. Zenios (1995). Robust optimization of large-scale systems. *Operations research 43*(2), 264–281.

Myers, R. H., D. C. Montgomery, and C. M. Anderson-Cook (2016). *Response surface methodology: process and product optimization using designed experiments.* John Wiley & Sons.

Nissen, V. and J. Propach (1998). Optimization with noisy function evaluations. In *International Conference on Parallel Problem Solving from Nature*, pp. 159–168. Springer.

Olsson, A., G. Sandberg, and O. Dahlblom (2003). On latin hypercube sampling for structural reliability analysis. *Structural safety 25*(1), 47–68.

Orr, M. J. et al. (1996). Introduction to radial basis function networks.

Pareto, V., A. S. Schwier, A. N. Page, et al. (1971). *Manual of political economy.* Macmillan London.

Parr, J., C. M. Holden, A. I. Forrester, and A. J. Keane (2010). Review of efficient surrogate infill sampling criteria with constraint handling. In *2nd International conference on engineering optimization*, pp. 1–10.

Parr, W. C. (1989). Introduction to quality engineering: designing quality into products and processes.

Paté-Cornell, M. E. (1996). Uncertainties in risk analysis: Six levels of treatment. *Reliability Engineering & System Safety 54*(2-3), 95–111.

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

Persson, J. and J. Ölvander (2013). Comparison of different uses of metamodels for robust design optimization. In *51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, pp. 1039.

Pignatiello Jr, J. J. (1988). An overview of the strategy and tactics of taguchi. *IIE transactions 20*(3), 247–254.

Pignatiello Jr, J. J. and J. S. Ramberg (1991). Top ten triumphs and tragedies of genichi taguchi. *Quality Engineering 4*(2), 211–225.

Plackett, R. L. and J. P. Burman (1946). The design of optimum multifactorial experiments. *Biometrika 33*(4), 305–325.

Ponweiser, W., T. Wagner, and M. Vincze (2008). Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In *2008 IEEE congress on evolutionary computation (IEEE World Congress on Computational Intelligence)*, pp. 3515–3522. IEEE.

Pošík, P., W. Huyer, and L. Pál (2012). A comparison of global search algorithms for continuous black box optimization. *Evolutionary computation 20*(4), 509–541.

Queipo, N. V., R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker (2005). Surrogate-based analysis and optimization. *Progress in aerospace sciences 41*(1), 1–28.

Ramnath, S., P. Haghighi, J. H. Kim, D. Detwiler, M. Berry, J. J. Shah, N. Aulig, P. Wollstadt, and S. Menzel (2019). Automatically generating 60,000 cad variants for big data applications. In *International Design Engineering Technical*

Conferences and Computers and Information in Engineering Conference, Volume 59179, pp. V001T02A006. American Society of Mechanical Engineers.

Ramnath, S., J. J. Shah, P. Wollstadt, M. Bujny, S. Menzel, and D. Detwiler (2022). Osu-honda automobile hood dataset (carhoods10k). *[Online].*

Ranzato, M., C. Poultney, S. Chopra, and Y. L. Cun (2007). Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pp. 1137–1144.

Rao, C. R. (1946). Hypercubes of strength'd'leading to confounded designs in factorial experiments. *Bull. Calcutta Math. Soc. 38*, 67–78.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press.

Rehman, S. U. (2016). *Robust optimization for computationally expensive systems: With applications to integrated photonics.* Ph. D. thesis, Technical University, Delft.

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1278–1286.

Rifai, S., P. Vincent, X. Muller, X. Glorot, and Y. Bengio (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 833–840. Omnipress.

Robinson, T., M. Eldred, K. Willcox, and R. Haimes (2008). Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. *Aiaa Journal 46*(11), 2814–2822.

Roux, W., N. Stander, and R. T. Haftka (1998). Response surface approximations for structural optimization. *International journal for numerical methods in engineering 42*(3), 517–534.

Roweis, S. T. (1998). Em algorithms for pca and spca. In *Advances in neural information processing systems*, pp. 626–632.

Roweis, S. T. and L. K. Saul (2000). Nonlinear dimensionality reduction by locally linear embedding. *science 290*(5500), 2323–2326.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and analysis of computer experiments. *Statistical science 4*(4), 409–423.

Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering 28*(6-7), 971–983.

Santner, T. J., B. J. Williams, W. I. Notz, and B. J. Williams (2003). *The design and analysis of computer experiments*, Volume 1. Springer.

Schmit Jr, L. and B. Farshi (1974). Some approximation concepts for structural synthesis. *AIAA journal 12*(5), 692–699.

Schölkopf, B., A. Smola, and K.-R. Müller (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation 10*(5), 1299–1319.

Shahraki, A. F. and R. Noorossana (2014). Reliability-based robust design optimization: a general methodology using genetic algorithm. *Computers & Industrial Engineering 74*, 199–207.

Shan, S. and G. G. Wang (2010). Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and multidisciplinary optimization 41*(2), 219–241.

Sharma, S., S. Sharma, and A. Athaiya (2017). Activation functions in neural networks. *towards data science 6*(12), 310–316.

Shcherbakov, M. V., A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, V. A. Kamaev, et al. (2013). A survey of forecast error measures. *World applied sciences journal 24*(24), 171–176.

Snoek, J., H. Larochelle, and R. P. Adams (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 2960–2968.

Sóbester, A., A. I. Forrester, D. J. Toal, E. Tresidder, and S. Tucker (2014). Engineering design applications of surrogate-assisted optimization techniques. *Optimization and Engineering 15*(1), 243–265.

Sóbester, A., S. J. Leary, and A. J. Keane (2005). On the design of optimization strategies based on global response surface approximation models. *Journal of Global Optimization 33*(1), 31–59.

Srinivas, N., A. Krause, S. M. Kakade, and M. W. Seeger (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 1015–1022. Omnipress.

Stork, J., M. Friese, M. Zaefferer, T. Bartz-Beielstein, A. Fischbach, B. Breiderhoff, B. Naujoks, and T. Tušar (2020). Open issues in surrogate-assisted optimization. In *High-Performance Simulation-Based Optimization*, pp. 225–244. Springer.

Svanberg, K. (1987). The method of moving asymptotesa new method for structural optimization. *International journal for numerical methods in engineering 24*(2), 359–373.

Szabó, B. and I. Babuška (2021). Finite element analysis: Method, verification and validation.

Ta'asan, S., G. Kuruvila, and M. Salas (1992). Aerodynamic design and optimization in one shot. In *30th aerospace sciences meeting and exhibit*, pp. 25.

Taguchi, G. (1995). Quality engineering (taguchi methods) for the development of electronic circuit technology. *IEEE Transactions on Reliability 44*(2), 225–229.

Taguchi, G. and M. S. Phadke (1989). Quality engineering through design optimization. In *Quality Control, Robust Design, and the Taguchi Method*, pp. 77–96. Springer.

Tan, M. H. (2013). Minimax designs for finite design regions. *Technometrics 55*(3), 346–358.

Tanaka, H. and K. Asai (1984). Fuzzy linear programming problems with fuzzy numbers. *Fuzzy sets and systems 13*(1), 1–10.

Tanaka, H., T. Okuda, and K. Asai (1973). Fuzzy mathematical programming. *Transactions of the society of instrument and control engineers 9*(5), 607–613.

Tenenbaum, J. B., V. De Silva, and J. C. Langford (2000). A global geometric framework for nonlinear dimensionality reduction. *science 290*(5500), 2319–2323.

Tipping, M. E. and C. M. Bishop (1999a). Mixtures of probabilistic principal component analyzers. *Neural computation 11*(2), 443–482.

Tipping, M. E. and C. M. Bishop (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 61*(3), 611–622.

Toropov, V. V., A. Filatov, and A. Polynkin (1993). Multiparameter structural optimization using fem and multipoint explicit approximations. *Structural optimization 6*(1), 7–14.

Ullah, S., H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2019). An empirical comparison of meta-modeling techniques for robust design optimization. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 819–828. IEEE.

Ullah, S., H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2021). A new acquisition function for robust bayesian optimization of unconstrained problems. In *2021 Genetic and Evolutionary Conference Companion.*

Ullah, S., H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2022). A systematic approach to analyze the computational cost of robustness in model-assisted robust optimization. In *Parallel Problem Solving from Nature - PPSN VIII, 17th International Conference, Dortmund, Germany, September 10-14, 2022, Proceedings*, Lecture Notes in Computer Science. Springer.

Ullah, S., Z. Xu, H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2020a). Exploring clinical time series forecasting with meta-features in variational recurrent models. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE.

Ullah, S., Z. Xu, H. Wang, S. Menzel, B. Sendhoff, and T. Bäck (2020b). Exploring clinical time series forecasting with meta-features in variational recurrent models. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE.

ur Rehman, S., M. Langelaar, and F. van Keulen (2014). Efficient kriging-based robust optimization of unconstrained problems. *J. Comput. Sci. 5*(6), 872–881.

Van Der Maaten, L., E. Postma, and J. Van den Herik (2009). Dimensionality reduction: a comparative. *J Mach Learn Res 10*(66-71), 13.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks 10*(5), 988–999.

Vyas, G. M., A. Andre, and R. Sala (2020). Toward lightweight smart automotive hood structures for head impact mitigation: Integration of active stiffness control composites. *Journal of Intelligent Material Systems and Structures 31*(1), 71–83.

Wang, H. (2009). Forward regression for ultra-high dimensional variable screening. *Journal of the American Statistical Association 104*(488), 1512–1524.

Wang, H. (2018). *Stochastic and deterministic algorithms for continuous black-box optimization.* Ph. D. thesis, Leiden University.

Wang, H., M. Emmerich, and T. Bäck (2018). Cooling strategies for the moment-generating function in bayesian global optimization. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE.

Wang, H., B. van Stein, M. Emmerich, and T. Back (2017). A new acquisition function for bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 507–512. IEEE.

Wang, H., B. van Stein, M. Emmerich, and T. Bäck (2017). A new acquisition function for bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017, Banff, AB, Canada, October 5-8, 2017*, pp. 507–512. IEEE.

Wets, R. J.-B. (1966). Programming under uncertainty: the equivalent convex program. *SIAM Journal on Applied Mathematics 14*(1), 89–105.

Wollstadt, P., M. Bujny, S. Ramnath, J. J. Shah, D. Detwiler, and S. Menzel (2022). Carhoods10k: An industry-grade data set for representation learning and design optimization in engineering applications. *IEEE Transactions on Evolutionary Computation.*

Woodard, R. (2000). Interpolation of spatial data: Some theory for kriging. *Technometrics 42*(4), 436–437.

Wright, S., J. Nocedal, et al. (1999). Numerical optimization. *Springer Science 35*(67-68), 7.

Yang, P., K. Tang, and X. Yao (2017). Turning high-dimensional optimization into computationally expensive optimization. *IEEE Transactions on Evolutionary Computation 22*(1), 143–156.

Ying, X. (2019). An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, Volume 1168, pp. 022022. IOP Publishing.

Žilinskas, A. (1992). A review of statistical models for global optimization. *Journal of Global Optimization 2*(2), 145–153.

Zimmermann, H.-J. (1975). Description and optimization of fuzzy systems. *International journal of general System 2*(1), 209–215.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology) 67*(2), 301–320.

Index

modeling accuracy, 51

acquisition function, 29, 33, 34, 53, 76, 80, 87
adaptive sampling, 80
additive noise, 18, 49, 50
additive uncertainty, 16, 37, 38
Aleatory and Epistemic uncertainties, 18
Algorithm Configuration, 25
Analysis of Variance, 3
Artificial Neural Networks, 45
Autoencoders, 60, 63, 132
average cpu time per iteration, 107
average running cpu time per iteration, 94, 129

Bayesian Optimization, 4, 79
Bayesian optimization, 7, 80, 95, 97, 112, 113, 117, 133–136
bi-objective optimization problem, 40
black-box, 9, 11
black-box functions, 10
black-box optimization, 4, 19, 29, 76
black-box problems, 1, 79, 80
BOX-BEHNKEN Designs, 41

classical set, 18
composite robustness, 49, 50, 76, 101, 134
computational budget, 5
computational complexity, 5, 36
computational cost of robustness, 5
computational resources, 41
Computer Aided Design, 112
constrained optimization problem, 12
continuity, 10
continuous optimization, 10, 42, 59, 76
controllable parameters, 3
correlation, 25
correlation matrix, 26
cosine similarity, 119
crisp sets, 18
cross-validation, 25, 51
cumulative distribution function, 82, 87
Curse of Dimensionality, 59, 97

data normalization, 44
data pre-processing, 44, 69
data preparation, 32
data set, 113
data-generating process, 44
decision variable, 62

decision variables, 13, 22, 32, 37, 38, 43, 44, 48, 50
deep generative models, 63
derivative-free methods, 22
Design and Analysis of Computer Experiments, 25
Design of Experiments, 3
design optimization, 21, 117
design variables, 112, 114–120
deterministic set, 37, 47
deterministic uncertainty, 38
differentiability, 10
dimensionality reduction, 33, 43, 44, 60–62, 65–67, 69, 76, 77
dimensionality reduction techniques, 44, 48, 60, 135
direct-search methods, 1, 22
Discrete spaces, 10
dispersion-based robustness, 101
Domain, 9

ecdf, 91–96, 104–106
elastic-net penalty, 25
empirical success rate, 92
environmental conditions, 13
environmental variables, 29, 135
Euclidean metric, 42
Evidence Lower Bound, 65
expectation-based robustness, 39, 101, 134
Expected Improvement, 27, 33, 79, 88, 97, 133
Expected Improvement Criterion, 6, 82
experimental design, 40

factor space, 42
feasible solutions, 20
feature space, 62, 66, 67, 69, 71
feed-forward neural networks, 60, 64
finite element analysis, 113
fixed budget analyses, 108
fixed cpu time analysis, 91, 103, 104, 125
fixed iteration analysis, 91, 92, 103, 104, 125
fixed target analysis, 91, 104, 108, 125
flexible programming, 4
fractional factorial design, 41
free parameters, 25
full factorial design, 41
fuzzy programming, 4
fuzzy sets, 19
fuzzy statement, 19

Gaussian process, 88
generalized least square estimator, 26
global minimum, 10, 11
Global Optimization, 25

hard constraints, 17
high dimensional, 59
high dimensionality, 22, 32, 33, 59, 60, 66, 76, 112
Hilbert spaces of functions, 10
Hyper-parameter optimization, 47
hyper-parameter optimization, 37
hyper-parameters, 37, 47, 69, 70

infill criteria, 92–96
infill criterion, 29, 82, 88, 89, 97

K Nearest-Neighbour, 118
K-Nearest Neighbors, 50
kernel function, 25
kernel Hilbert space, 62
Kernel Principal Component Analysis, 62, 132
Kriging, 24, 25, 36, 42, 45, 50, 52, 53, 67, 71, 76, 77, 80, 81, 85, 87, 88,

95, 103, 115, 118, 122, 123, 133
Kriging , 69
Kullback-Leibler divergence, 65

Lasso and Ridge regularization, 25
latent space, 62
latin hype-cube design, 43
latin hyper-cube sampling, 42
least upper bound, 38
linear cooling strategy, 90
Lower Confidence Bound, 88, 97, 133

machine learning, 136
manufacturing tolerances, 13
Matérn 3/2 kernel, 25
mathematical modeling of uncertainty, 34
mathematical programming, 4
maxi-min design, 42
maximum likelihood principle, 24
mean squared error, 26
membership function, 18
meta-model, 23
mini-max design, 42
Mini-max Regret Robustness, 38
mini-max regret robustness, 101, 134
Mini-max robustness, 37
mini-max robustness, 33, 49, 50, 101, 102, 134
Mixed-integer spaces, 10
model-assisted optimization, 122
model-assisted optimization, 42, 45, 52, 59, 61, 66
model-based optimization, 48
modeling accuracy, 53
modeling accuracy, 45, 46, 51, 52, 59, 69–72, 118, 123
modeling technique, 32, 34, 36, 37, 40, 44, 45, 48–50, 52, 67, 69, 76, 120
modeling techniques, 119–124, 136
Moment-Generating Function of the Improvement, 6, 79, 82, 88, 97, 133, 134
Moore-Penrose pseudo-inverse, 25
MOPSO, 11
multi-modal functions, 89
multi-modality, 22
multi-objective optimization, 17

nature inspired heuristics, 11
neural networks, 64
noise level, 2, 34, 50, 76, 89, 119, 120
noise level,, 52
noise levels, 120
noise parameters, 3
non-deterministic, 17
non-deterministic simulator, 12
non-linearity, 69
NSGA-II, 11

objective function, 10, 37–39, 46, 53, 80, 87, 90, 103, 120, 125
one-shot optimization, 7, 30, 48, 112, 117, 118
operations research, 4
optimization under uncertainty, 21, 23
optimizer, 14
orthogonal array, 3
over-fitting, 25, 45

Pareto optimal solutions, 11
PLACKETT-BURMAN Designs, 41
polynomial approximation, 25

Polynomial Regression, 45, 50, 69, 71
polynomial regression models, 24
possibilistic programming, 4
practical goal of optimization, 12
Principal Component Analysis, 44, 60, 61, 132
probability density function, 47, 82, 87
probability density functions, 3, 19
probability distribution, 50, 64
probability distributions, 65
problem description, 34
problem landscape, 2, 32, 34, 44, 49, 53, 67
process mean, 3

quasi-Newton methods, 23

Radial Basis Function Network, 50
Random Forest, 50, 118
real parameter optimization problems, 10
regret, 38
regularization, 25
Relative Mean Absolute Error, 46
reliability-based robustness, 20
Response Surface Approximation, 23
Response Surface Models, 24
robust black-box optimization, 22
robust counterpart, 38, 39, 100, 135
robust counterpart approach, 37
robust design, 3
robust objective function, 39
robust optimization, 1, 7, 20, 29, 100, 112, 132, 136
robust solution, 6, 36, 37, 48, 49, 51, 76, 80, 84
robust solutions, 32, 76, 89, 112
robustness, 37
robustness criteria, 100–102
robustness criterion, 38, 39, 49, 88, 97, 112
robustness formulation, 2, 6, 7, 32, 34, 37, 39, 47, 49, 119, 120
robustness formulations, 120

sample size, 5, 34, 50, 51, 67
samples size, 48
sampling infill criteria, 133, 136
sampling infill criterion, 80, 103
sampling locations, 48, 69, 80
sampling plan, 36
sampling plans, 5, 36
sampling points, 40–42, 48
sampling points,, 42
scale of the uncertainty, 32, 49
scale of uncertainty, 34
screening experiments, 41
search point, 37, 38
search space, 9, 24, 36, 42–44, 46, 47, 61, 77, 82, 87, 119
search variables, 13, 29, 34, 39, 76, 87–89, 102, 103, 119, 135
sensitivity robustness, 20
Sequential Least Square Programming, 51
sequential model-based optimization, 30, 133
severity of the uncertainty, 88, 93, 97
signal-to-noise-ratio, 3
smoothness, 10
space-filling designs, 43
space-filling property, 42
SPEA2, 11
standardizing, 43
statistical learning, 36, 44
stochastic programming, 4

structural mechanics, 113, 114
structure of the uncertainty, 37
Support Vector Machines, 50, 76, 118, 136
Support Vector Regression, 45
surface mesh, 113
surrogate modeling, 123
surrogate model, 32, 36, 37, 40, 43, 45–48, 51, 52, 59, 61, 66, 67, 69, 76, 79, 81, 115, 118, 120, 122
surrogate modeling, 4, 7, 25, 29, 30, 32, 40, 47, 49, 53, 59, 76, 77, 112, 118, 135, 136
surrogate models, 23, 34, 49, 52, 53, 60, 61, 66, 120–124
Surrogate-Assisted Evolutionary Algorithms, 25
Surrogate-Assisted Optimisation, 23
surrogate-assisted optimization, 1, 33
surrogate-assisted robust optimization, 34
system constants, 3

Taylor series expansion, 25
testing data set, 46
training data set, 43
Tree Parzen Estimator algorithm, 47

uncertainty and noise, 18, 32, 53, 59, 79
uncertainty and robustness specification, 34
uncertainty assessment, 34
uncertainty handling, 38
uncertainty level, 89, 97, 103
uncertainty specification, 34
Under-complete Autoencoders, 64
unexpected drifts and changes, 22
uniform probability distribution, 89, 103, 120

Vandermonde matrix, 24
Variational Autoencoders, 132
Variational autoencoders, 64
variational distribution, 64

worst-case, 38, 39
worst-case scenario, 38

English Summary

While solving real-world optimization problems, e.g., in the area of automotive engineering, building construction, and steel production, the issue of uncertainty and noise is frequently-encountered. Common sources of uncertainty and noise include search/decision variables (that describe the system to be optimized), the environmental variables or operating conditions the system is subject to, the evaluation of the (physical) system (or model of the system), and the preference in objectives and vagueness in constraints when modeling the (physical) system. It is therefore intuitive that uncertainty and noise surround the system in most practical scenarios of continuous optimization, and can significantly compromise the applicability of the optimization algorithms and the (nominal) optimal solutions obtained from these algorithms. In the ECOLE (Experience-based COmputation: Learning to optimisE) project, the focus of this thesis is on the parametric uncertainties in the search/decision variables that are assumed to be structurally symmetric, additive in nature, and can be modeled in a deterministic or a probabilistic fashion. Accounting for these uncertainties and noise leads us to robust optimization, which emphasizes on the solutions that are still optimal and useful in the face of such uncertainties and noise.

Some of the most important performance indicators in the area of product engineering include shortening the product-development cycle, reducing the resource consumption during the complete process, and creating more balanced and innovative products. These practical aspects necessitate solving the robust optimization problem in an efficient manner. Since it is very costly to assess candidate solutions, we substitute the expensive function evaluations with a statistical model, which is referred to as the “surrogate model”, or the “meta model”. In this way, the model predicts the function response, and the optimization algorithm can query the function response instead of actually running the real production process.

Solving robust optimization problems with surrogate models, we answer some of the most important research questions in Chapter 3. This chapter implements surrogate modeling with the help of a “one-shot optimization” strategy, and discusses the practical applicability of surrogate modeling to find robust solutions, and the related difficulties thereof. In this chapter, it is found that we can construct surrogate models with Kriging, Polynomials, and Support Vector Machines, with a reasonable (linear) sample size. The resulting surrogate model can find the robust solution in most situations, which is very close to the true baseline robust solution.

Since in practical scenarios, high dimensionality can affect the performance of surrogate modeling, we devote the second half of Chapter 3 to focus on dimensionality reduction techniques. The dimensionality reduction techniques discussed include Principal Component Analysis, Kernel Principal Component Analysis, Autoencoders, and Variational Autoencoders. An empirical performance assessment indicates the suitability of Autoencoders and Principal Component Analysis to help construct a low dimensional surrogate model.

A major manifestation of surrogate modeling, which is referred to as the “Bayesian optimization” algorithm, is discussed in Chapter 4. The major research results in this chapter include adapting the Bayesian optimization algorithm to find robust solutions in an efficient manner, as well as benchmarking its performance. To this end, it is found that the “Expected Improvement” criterion, and the “Moment-Generating Function of the Improvement” are good choices of sampling infill criteria to be utilized in the Bayesian optimization algorithm. Furthermore, the performance of the Bayesian optimization algorithm is deemed satisfactory in the light of a fixed budget and a fixed target analysis.

A major point of concern in the context of robust optimization is the choice of a robustness criterion, which can have tremendous implications for the designers in the area of product engineering. This is due to the fact that the choice of robustness criterion can dictate the computational budget and quality of the optimal solution to a large degree. In Chapter 5, we focus on the computational aspect concerning the choice of the robustness criterion. Based on a broad spectrum of test cases, we assess and rank commonly employed robustness criteria with respect to a fixed budget and a fixed target analysis, in addition to the analysis on the average running time per iteration.

The major findings from these analyses provide a novel perspective on the choice of the robustness criteria. For instance, it is found that the robustness criterion based on the "worst-case scenario" is also the most suitable criterion in terms of computational cost. Furthermore, the probabilistic robustness criteria have a higher variance in terms of quality of the solution, but lower variance in terms of utilization of the computational resources. Lastly, it is found that the probabilistic robustness criteria scale well with the dimensionality, whereas the deterministic criteria become inapplicable as the dimensionality increases.

Some of our findings, reported above, are validated, when benchmarking our approaches on a real-world design optimization scenario based on car hood frames. These findings include the promising nature of Kriging as the modeling technique, as well as the heuristics commonly employed for determining the initial sample size. Furthermore, the "Moment-Generating Function of the Improvement" is corroborated as an effective sampling infill criterion for the Bayesian optimization algorithm.

Nederlandse Samenvatting

Bij het oplossen van optimalisatieproblemen in de echte wereld, zoals in de automobieltechniek, bouwconstructie en staalproductie, wordt vaak geconfronteerd met het probleem van onzekerheid en lawaai. Veelvoorkomende bronnen van onzekerheid en ruis zijn onder meer zoek-/beslissingsvariabelen (die het te optimaliseren systeem beschrijven), de omgevingsvariabelen of bedrijfsomstandigheden waaraan het systeem is onderworpen, de evaluatie van het (fysieke) systeem (of model van het systeem), en de voorkeur in doelstellingen en vaagheid in beperkingen bij het modelleren van het (fysieke) systeem. Het is daarom intuïtief dat onzekerheid en ruis het systeem omringen in de meeste praktische scenario's van continue optimalisatie, en de toepasbaarheid van de optimalisatie-algoritmen en de (nominaal) optimale oplossingen die uit deze algoritmen worden verkregen. In het ECOLE-project (Experience-based COMputation: Learning to optimisE) focussen we op de parametrische onzekerheden in de zoek-/beslissingsvariabelen waarvan wordt aangenomen dat ze structureel symmetrisch, additief van aard zijn en op een deterministische of probabilistische manier kunnen worden gemodelleerd. Rekening houdend met deze onzekerheden en ruis leidt ons tot robuuste optimalisatie, die de nadruk legt op de oplossingen die nog steeds optimaal en bruikbaar zijn in het licht van dergelijke onzekerheden en ruis.

Enkele van de belangrijkste aspecten op het gebied van productengineering zijn het verkorten van de productontwikkelingscyclus, het verminderen van het verbruik van hulpbronnen tijdens het volledige proces en het creëren van meer evenwichtige en innovatieve producten. Deze praktische aspecten maken het noodzakelijk om het robuuste optimalisatieprobleem op een efficiënte manier op te lossen. Omdat het erg kostbaar is om kandidaat-oplossingen te beoordelen, vervangen we de dure functie-evaluaties door: een statistisch model, dat het "surrogaatmodel" of het “metamodel” wordt genoemd. Op deze manier voorspelt het model de functierespons en kan het optimalisatiealgoritme de functierespons opvragen in plaats van daadwerkelijk het echte productieproces leiden.

Door robuuste optimalisatieproblemen met surrogaatmodellen op te lossen, proberen we een beantwoord enkele van de belangrijkste onderzoeksvragen in hoofdstuk 3. Dit hoofdstuk surrogaatmodellering implementeren met behulp van een "one-shot-optimalisatie"-strategie egy, en bespreekt de praktische toepasbaarheid van surrogaatmodellering om robuust te vinden oplossingen en de daarmee samenhangende moeilijkheden. In dit hoofdstuk blijkt dat we kan surrogaatmodellen bouwen met Kriging, Polynomials en Support Vector Machines, met een redelijke steekproefomvang. Het resulterende surrogaatmodel, kan vinden de robuuste oplossing in de meeste situaties, die zeer dicht bij de baseline ligt.

Omdat in praktische scenario’s een hoge dimensionaliteit de prestaties van surrogaatmodellering, besteden we de tweede helft van dit hoofdstuk aan de mensionaliteit reductie technieken. De dimensionaliteitsreductietechnieken die in dit hoofdstuk worden behandeld, zijn onder meer Analyse van hoofdcomponenten, Kernel-principal Componentanalyse, auto-encoders en variabele auto-encoders. een empirische prestatiebeoordeling geeft de geschiktheid van Autoencoders en Principal aan Componentenanalyse om een laagdimensionaal surrogaatmodel te construeren.

Een belangrijke manifestatie van surrogaatmodellering, waarnaar wordt verwezen als de "Bayesiaanse" optimalisatie-algoritme, wordt uitgebreid besproken in hoofdstuk 4. De belangrijkste punten van overweging in dit hoofdstuk zijn onder meer het aanpassen van de Bayesiaanse optimalisatie-algoritme ritme om op een efficiënte manier robuuste oplossingen te vinden en de uitvoering. Hiertoe blijkt dat het criterium "Verwachte verbetering", en de "Momentgenererende functie van de verbetering" zijn goede keuzes van bemonsteringsinfill-criteria die moeten worden gebruikt in het Bayesiaanse optimalisatie-algoritme. Verder wordt de prestatie van het Bayesiaanse optimalisatie-algoritme geacht bevredigend in het licht van een vast budget en een analyse van een vast doel.

Een belangrijk aandachtspunt in het kader van robuuste optimalisatie is de keuze voor: robuustheidscriterium, dat enorme gevolgen kan hebben voor de ontwerpers in het gebied van productengineering. Dit komt doordat de keuze voor robuuste ness-criterium kan het rekenbudget en de kwaliteit van de optimale oplossing voor een groot deel. In dit proefschrift richten we ons op het computationele aspect over de keuze van het robuustheidscriterium. Gebaseerd op een breed spectrum van testgevallen beoordelen en rangschikken we veelgebruikte robuustheidscriteria met respect naar een vast budget en een analyse van een vast doel, naast de analyse op de gemiddelde looptijd per iteratie.

De belangrijkste bevindingen van deze analyses bieden een nieuw perspectief op de keuze van de robuustheidscriteria. Zo blijkt dat het robuustheidscriterium op basis van het worst case scenario is ook qua termen het meest geschikte criterium van rekenkosten. Verder hebben de probabilistische robuustheidscriteria een hogere variantie in termen van kwaliteit van de oplossing, maar lagere variantie in termen van gebruik van de rekenhulpmiddelen. Ten slotte blijkt dat de probabilistische robuustheidscriteria passen goed bij de dimensionaliteit, terwijl de deterministische criteria worden niet meer van toepassing naarmate de dimensionaliteit toeneemt.

Sommige van onze bevindingen, hierboven gerapporteerd, zijn gevalideerd bij het benchmarken van onze benaderingen van een realistisch ontwerpoptimalisatiescenario op basis van autokapframes. Deze bevindingen omvatten de veelbelovende aard van Kriging als modelleringstechniek, evenals de heuristieken die gewoonlijk worden gebruikt voor het bepalen van de initiële steekproef maat. Bovendien is de Moment-Generating Function of the Improvement: bevestigd als een effectief bemonsteringsinvulcriterium voor de Bayesiaanse optimalisatie algoritme.

Acknowledgements

I would like to express my deepest gratitude to my supervisors Thomas Bäck, Bernhard Sendhoff, and Hao Wang for their invaluable guidance and support throughout my PhD journey. Their constant encouragement, insightful feedback, and constructive criticism have been instrumental in shaping the direction of my research and improving the quality of my work. Their expertise and unwavering commitment to excellence have been an inspiration to me and have helped me to become a better researcher. I am truly grateful for the privilege of having such remarkable mentors.

I would also like to extend my heartfelt appreciation to my family for their unwavering love, support, and encouragement throughout my academic pursuits. A special thanks to my parents Qazi Naimat Ullah and Farah Deeba for their constant presence, understanding, and encouragement during challenging times. Their unwavering belief in me and my abilities has been the cornerstone of my success, and I cannot thank them enough for their sacrifices and contributions to my achievements. My Ph.D. would not have been possible without their love, support, and encouragement, and I am eternally grateful for having them. Last but not least, I would like to express my love towards my wife Tehreem and my son Daniyal. I love you both very much and find it privileged to have you by my side.

About the Author

Sibghat Ullah was born in 1995 in Dera Ghazi Khan, Pakistan. In 2011, he started his Bachelor in computer science at National University of Computer and Emerging Sciences, Islamabad. In 2016, he moved to Europe to study data science and machine learning at Sapienza University of Rome. After completing his Master in data science, he started as a PhD candidate in the Natural Computing group of Prof. Thomas Bäck in November 2018. During his time as a PhD candidate, he traveled to Germany for research collaborations at NEC Labs Europe GmbH and Honda Research Institute Europe GmbH. His research interests include optimization under uncertainty, model-assisted optimization, explainable artificial intelligence, and learning and mining in the presence of uncertain (industrial) data.