Model-assisted robust optimization for continuous black-box problems

Ullah, S.

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.

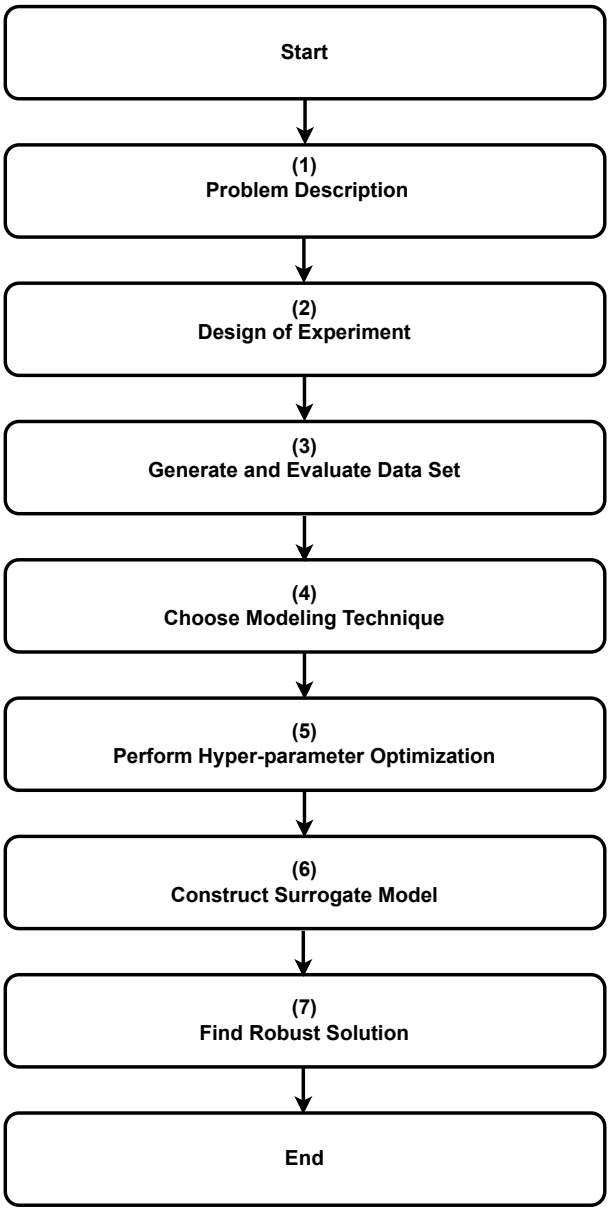

Figure 3.1: Flowchart shows the proposed framework for surrogate-assisted robust optimization in this chapter.

- **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:

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

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:

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

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

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

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: 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

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

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).

- **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.

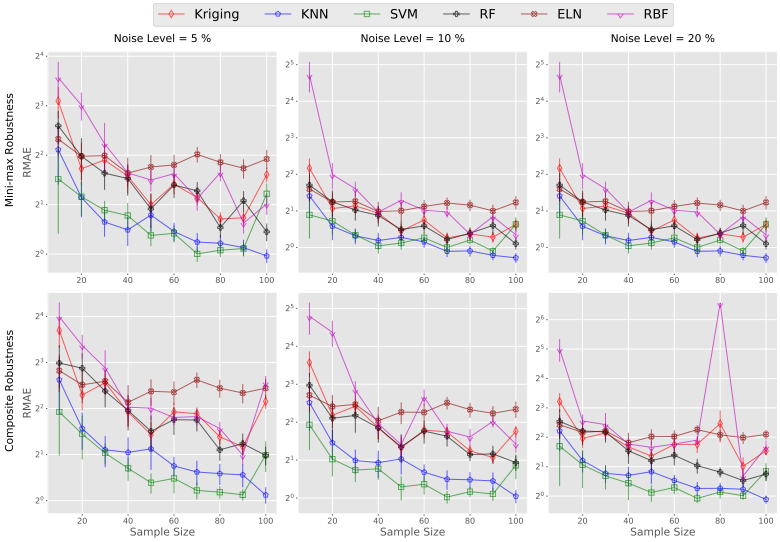

Figure 3.2: Modeling accuracy of the surrogates for the two dimensional Ackley function. The modeling accuracy is evaluated on six test scenarios, owing to the combination of three noise levels, and two robustness criteria.

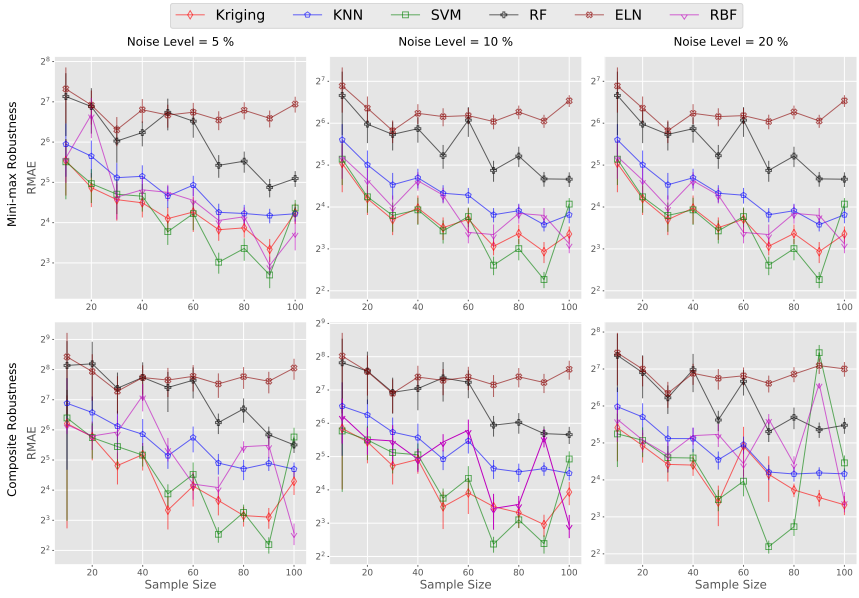

Figure 3.3: Modeling accuracy of the surrogates for the two dimensional Branin function.

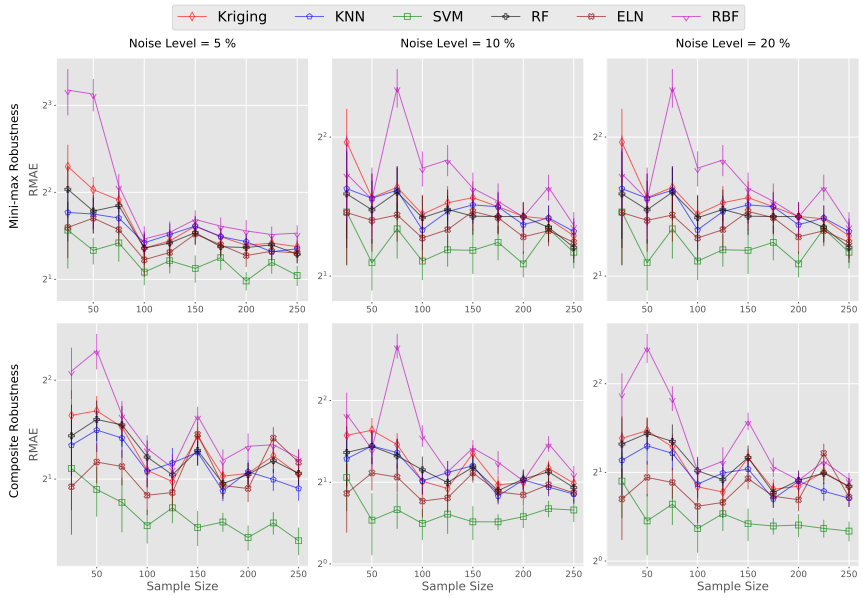

Figure 3.4: Modeling accuracy of the surrogates for the five dimensional Ackley function.

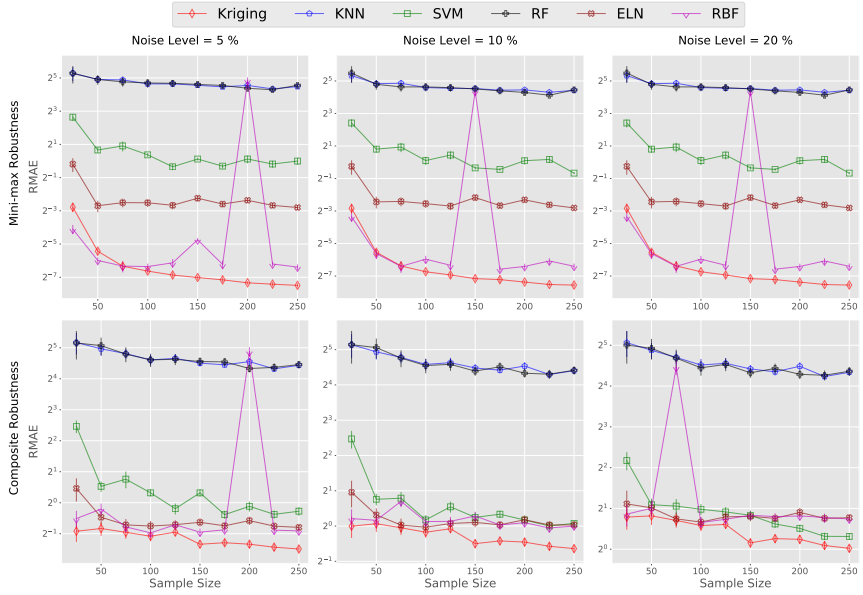

Figure 3.5: Modeling accuracy of the surrogates for the five dimensional Sphere function.

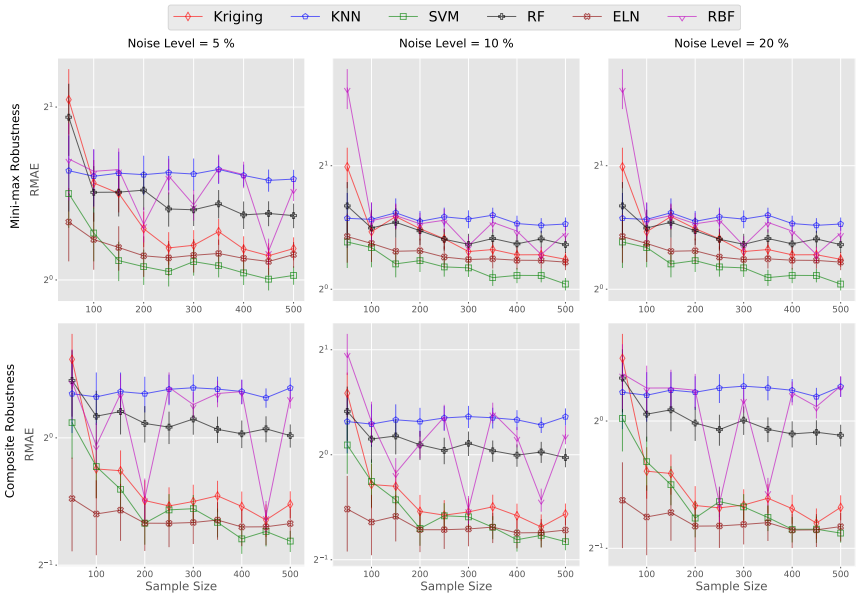

Figure 3.6: Modeling accuracy of the surrogates for the ten dimensional Ackley function.

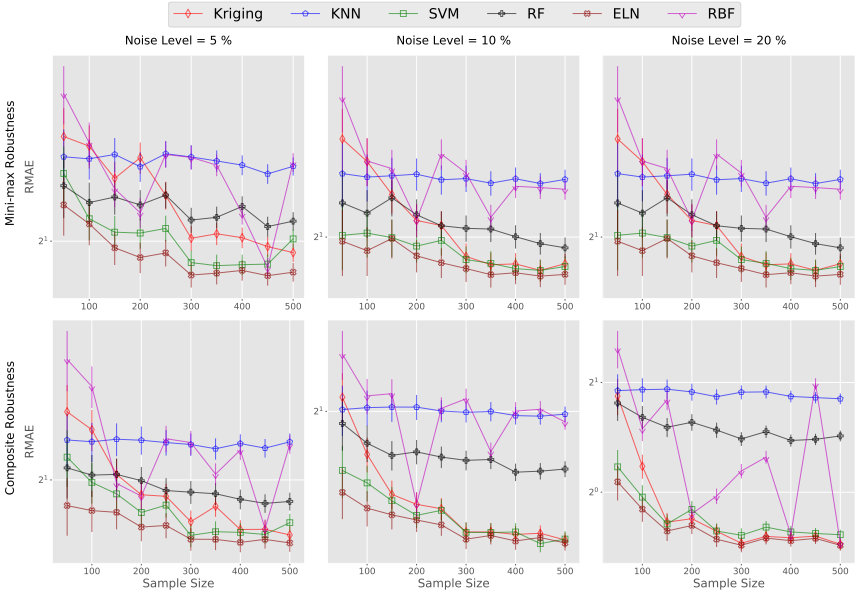

Figure 3.7: Modeling accuracy of the surrogates for the ten dimensional Rastrigin function.

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.

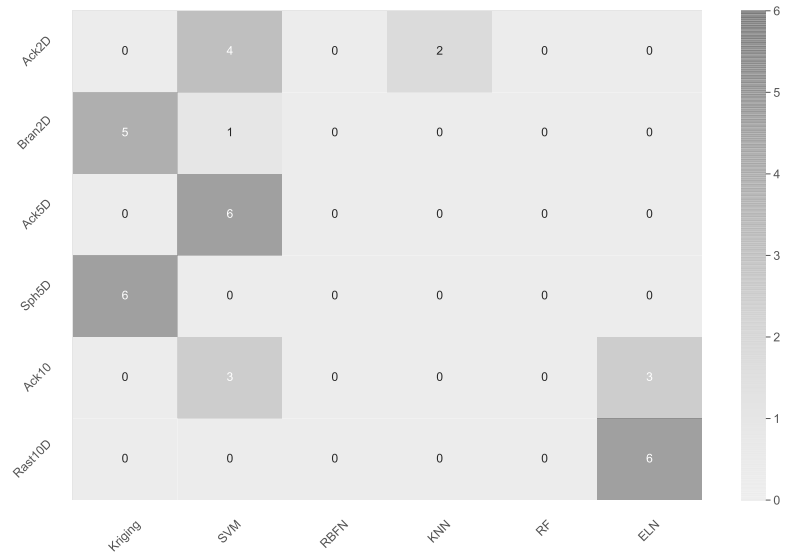

Figure 3.9: Number of occurrences of the modeling techniques, producing the best quality surrogate models. Note that the quality in this context is based on the RMAE values, averaged over all ten settings of the sample size N.

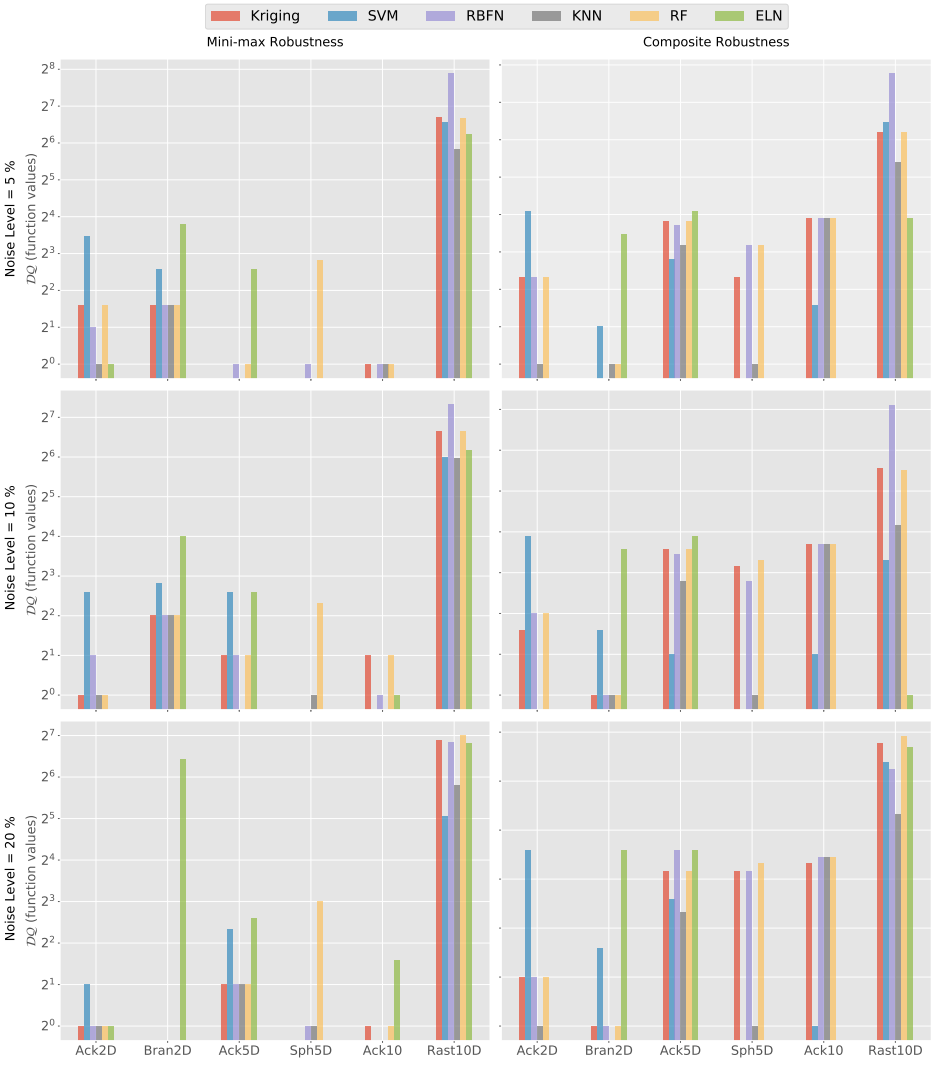

Figure 3.10: The difference in the quality $\mathcal{DQ}$ of the robust solutions (obtained from surrogate modeling), for all thirty six test scenarios. Note that the quality in this context is based on the objective function values, measured according to Eq. (3.12). Furthermore, the number of independent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

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

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

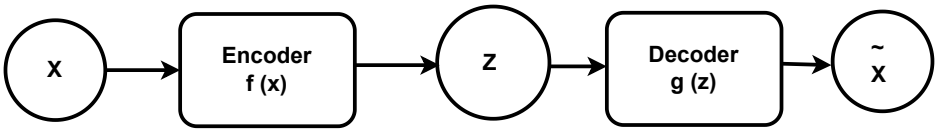

Figure 3.11: A schematic diagram of an under-complete Autoencoder. The Autoencoder receives some data $\mathbf{x}$, which is encoded to produce a code $\mathbf{z}$. The decoder takes this code to reproduce the original data, albeit with some deterioration.

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:

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

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: $\mathbf{HKH}$ ($\mathbf{H} = \mathbf{I}_{N \times N} - N^{-1}\mathbf{1}_N$). 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 $\mathbf{HKH}$.

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

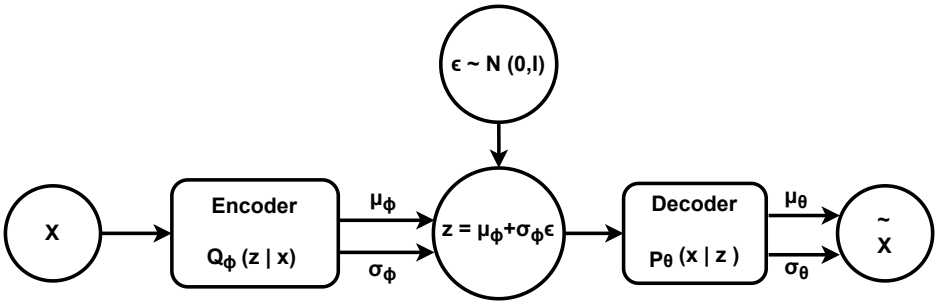

Figure 3.12: The schematic diagram of a vanilla Variational Autoencoder, where the code $\mathbf{z}$ is produced with the help of the reparameterization trick.

Bound (ELBO) as:

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

where $\mathrm{KL}(Q||P)$ 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 $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$, 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:

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

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

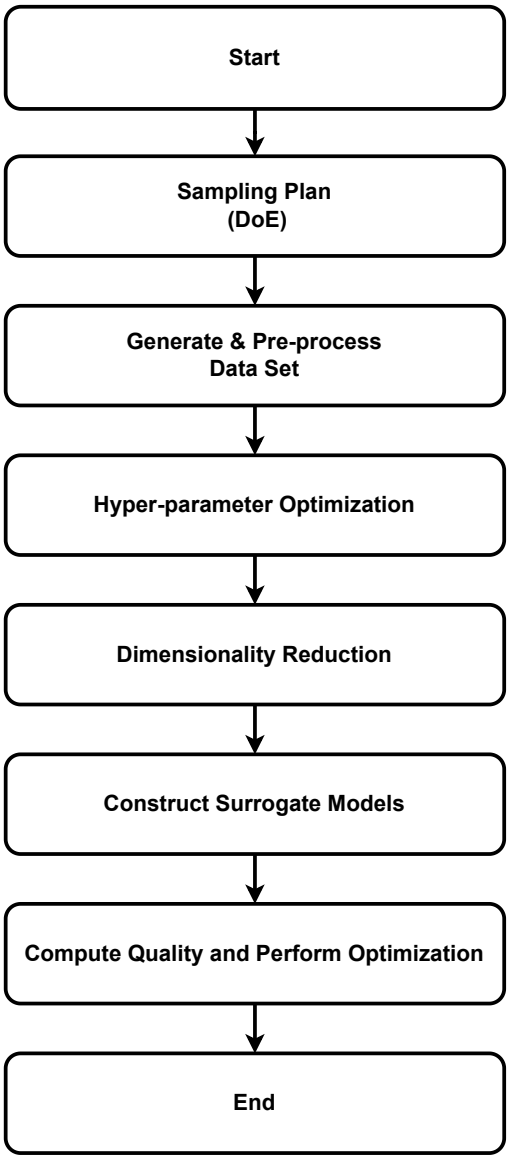

Figure 3.13: A schematic diagram of the experimental setup. Each step of the process is shown in grey rectangles. The central rectangle indicates the HPO loop based on the modeling accuracy of the surrogates.

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.

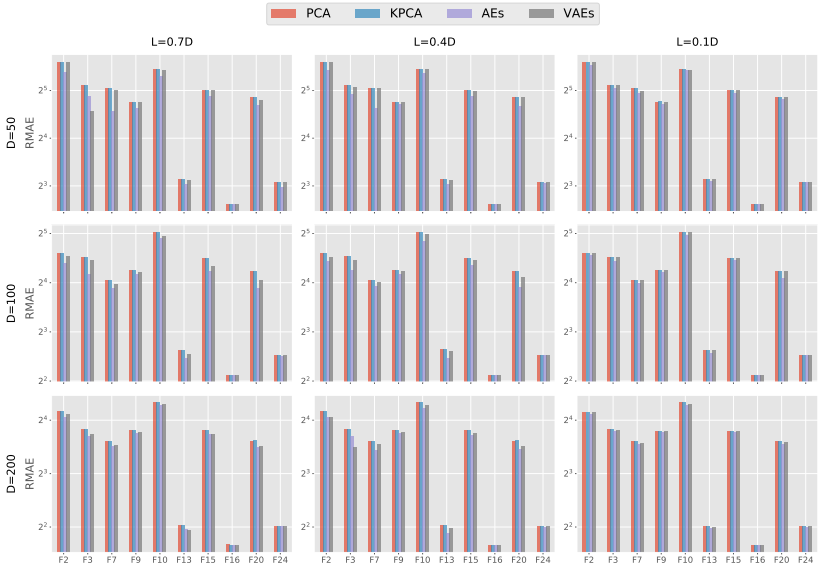

Figure 3.14: Modeling accuracy of the low dimensional Kriging surrogates. The accuracy is measured with Relative Mean Absolute Error (lower is better).

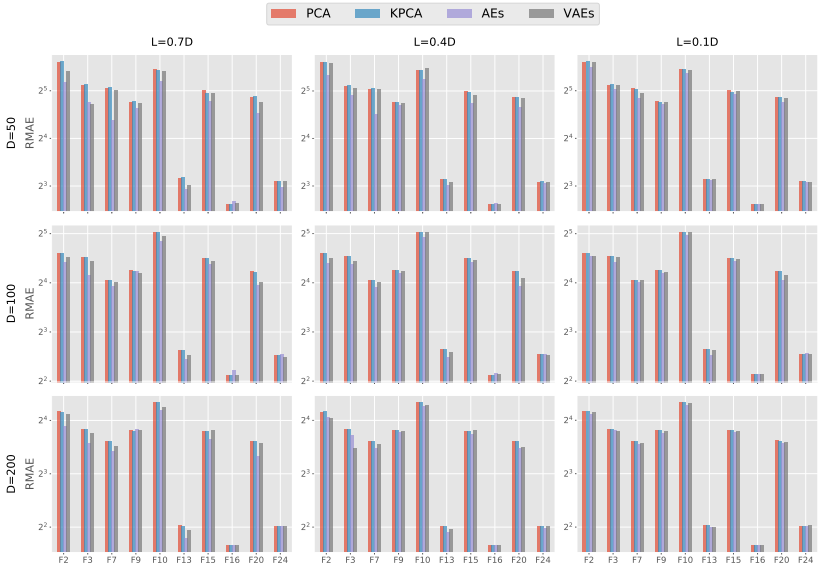

Figure 3.15: Modeling accuracy of the low dimensional Polynomial surrogates. The accuracy is measured with Relative Mean Absolute Error (lower is better).

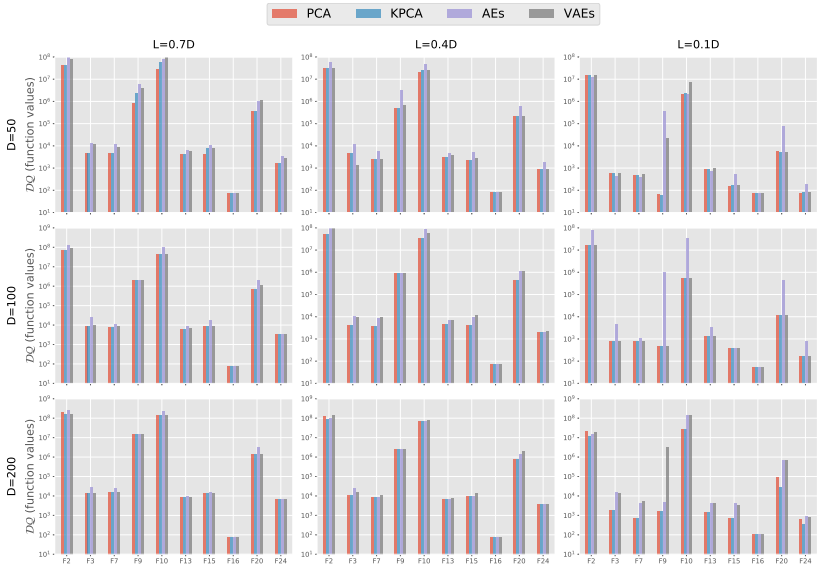

Figure 3.16: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Kriging surrogates, and the corresponding baseline. The difference in the quality is based on the objective function values.

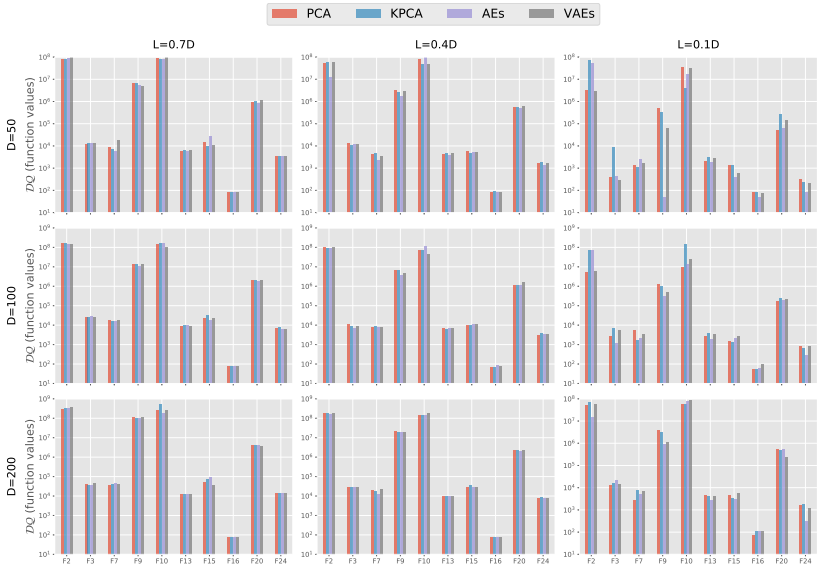

Figure 3.17: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Polynomial surrogates, and the corresponding baseline. The difference in the quality is based on the objective function values.

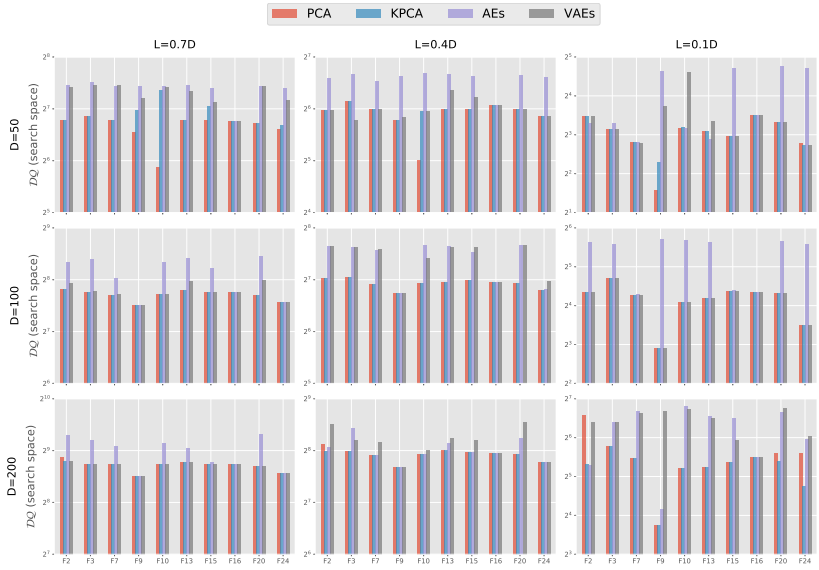

Figure 3.18: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Kriging surrogates, and the corresponding baseline. The difference in the quality is based on the distance in the search space.

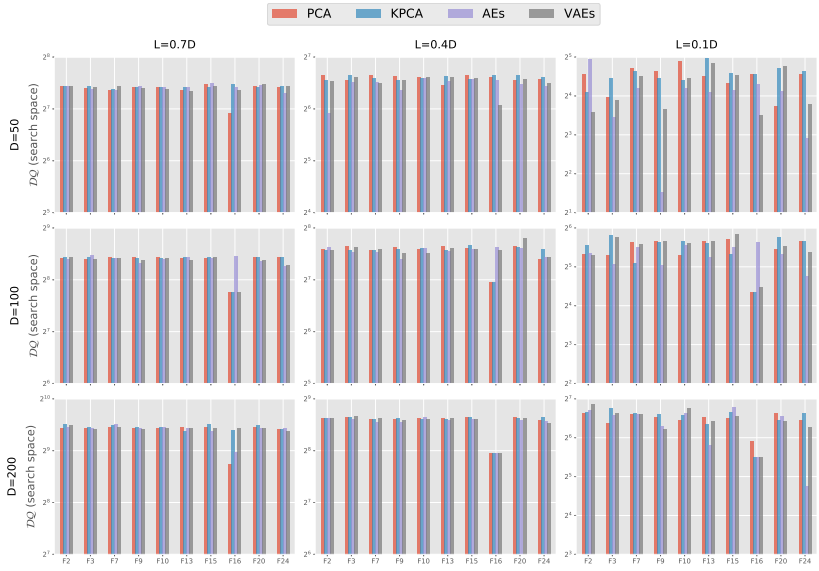

Figure 3.19: The difference in the quality: $\mathcal{DQ}$, between the optimal solutions obtained from the low dimensional Polynomial surrogates, and the corresponding baseline. The difference in the quality is based on the distance in the search space.

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.