Model-assisted robust optimization for continuous black-box problems

Ullah, S.

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

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

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.

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

[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.

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.

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:

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

where $\hat{f}^{'} = \hat{f} - s^2t$, 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 $\mathbf{x} + \boldsymbol{\Delta}_{\mathbf{x}}$.

- 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 $\mathbf{x} + \boldsymbol{\Delta}^*_{\mathbf{x}}$, where $\boldsymbol{\Delta}^*_{\mathbf{x}}$

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 $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$ on the objective function.

  Construct the Kriging model $\mathcal{K}_f$ on $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$.

  while the stop criteria are not fulfilled **do**

    Find robust optimum on the Kriging surface $\mathcal{K}_f$ as:

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

    Choose a new sample $\mathbf{x}_{\text{new}}$ by maximizing the robust (effective) acquisition function:

$$\mathbf{x}_{\text{new}} \leftarrow \operatorname*{argmax}_{\mathbf{x} \in \mathcal{S}} \quad \mathscr{A}_{\text{eff}}(\mathbf{x}).$$

    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 $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$.

    Reconstruct the Kriging model $\mathcal{K}_f$ on $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$.

  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

[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.

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 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.

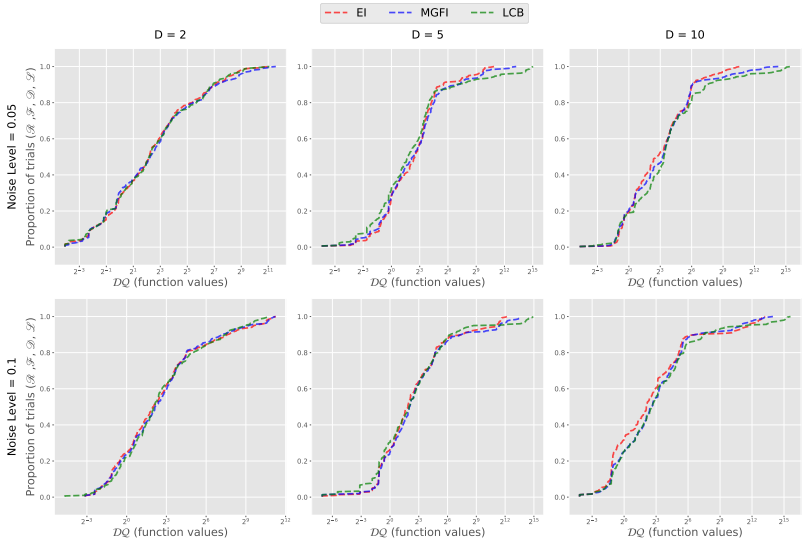

Figure 4.1: Fixed cpu time analysis. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

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 $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$,

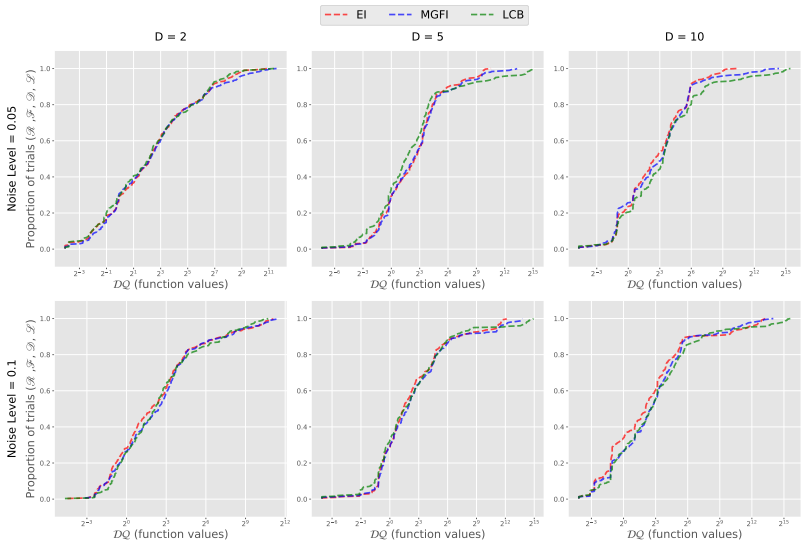

Figure 4.2: Fixed iteration analysis. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

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

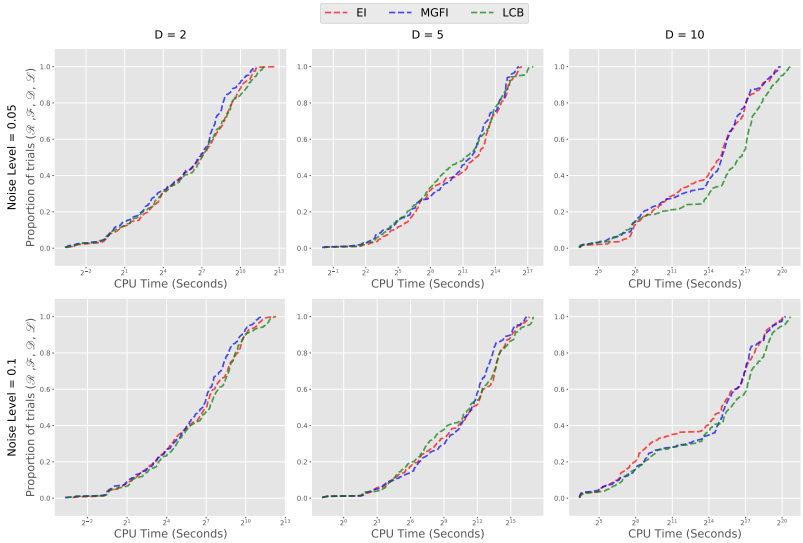

Figure 4.3: Fixed target analysis based on cpu time. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

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 LCB_{eff} and $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$. 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 $\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})]$ 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 $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$ as clearly superior to its competitors in most trials. Likewise, in the context of $T_{\max}$, i.e., we find $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$ as clearly superior to its competitors. Combining the performance for all type of analyses, we find $\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})]$ and $\mathcal{M}_{\text{eff}}(\mathbf{x}; t)$ as suitable AF to be employed in the BO algorithm to find robust solutions.

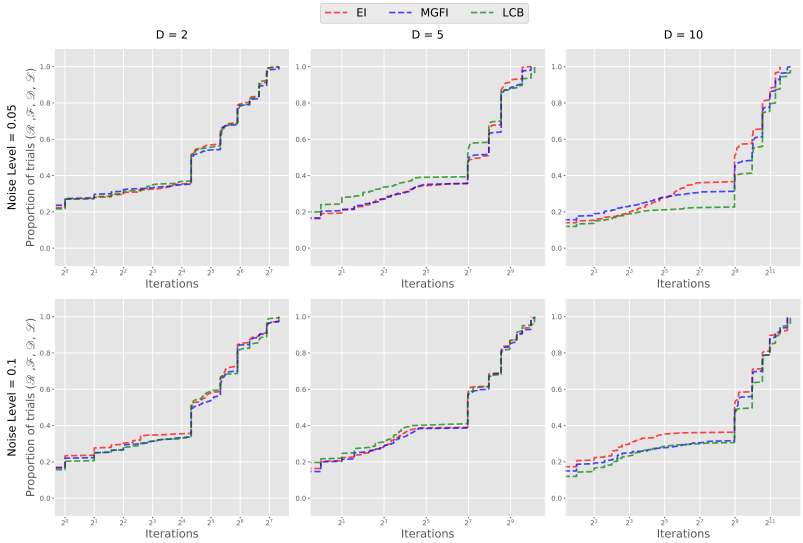

Figure 4.4: Fixed target analysis based on number of iterations. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

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-

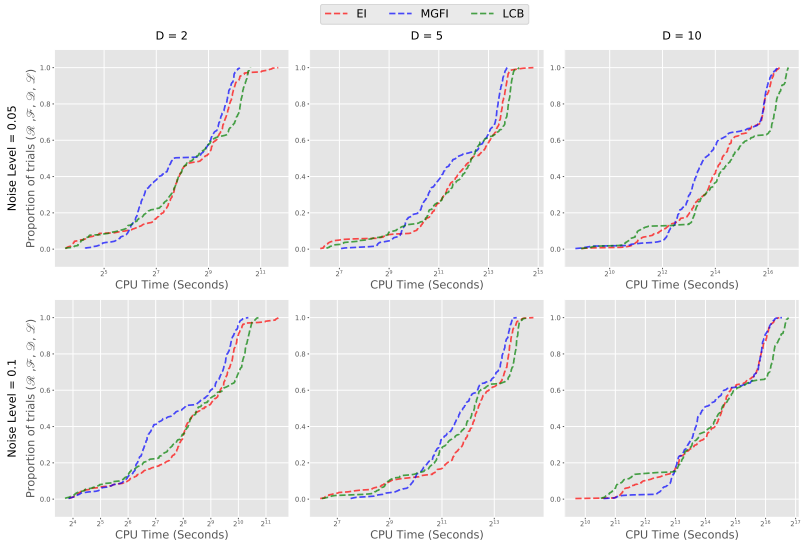

Figure 4.5: Average cpu time per iteration for the BO algorithm. Rows distinguish between noise levels, whereas columns identify different settings of dimensionality. Each plot contains three ECDF curves based on three infill criteria discussed. Each ECDF curve is based on 300 data points.

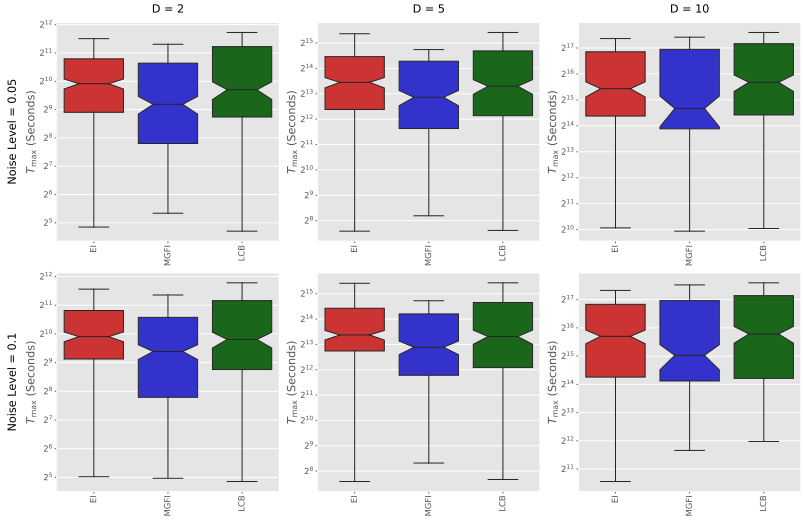

Figure 4.6: Maximum accumulated cpu time: $T_{\max}$ for each trial. Rows distinguish between noise levels, whereas columns help identify different settings of dimensionality.

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.