Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

A Systematic Approach to Analyze the Computational Cost of Robustness in Model-Assisted Robust Optimization

Sibghat Ullah, Hao Wang, Stefan Menzel, Bernhard Sendhoff, Thomas Bäck

2022

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in International Conference on Parallel Problem Solving from Nature (PPSN). The final authenticated version is available online at:
https://doi.org/10.1007/978-3-031-14714-2_5

A Systematic Approach to Analyze the Computational Cost of Robustness in Model-Assisted Robust Optimization*

Sibghat Ullah[1], Hao Wang[1], Stefan Menzel[2], Bernhard Sendhoff[2], and Thomas Bäck[1]

[1] Leiden Institute of Advanced Computer Science (LIACS), Leiden University, The Netherlands
{s.ullah,h.wang,t.h.w.baeck}@liacs.leidenuniv.nl
[2] Honda Research Institute Europe GmbH (HRI-EU), Offenbach/Main, Germany
{stefan.menzel,bernhard.sendhoff}@honda-ri.de

The article has been accepted for publication in Parallel Problem Solving from Nature (PPSN 2022).

DOI: 10.1007/978-3-031-14714-2_5

A Systematic Approach to Analyze the Computational Cost of Robustness in Model-Assisted Robust Optimization*

Sibghat Ullah[1], Hao Wang[1], Stefan Menzel[2], Bernhard Sendhoff[2], and Thomas Bäck[1]

[1] Leiden Institute of Advanced Computer Science (LIACS), Leiden University, The Netherlands
{s.ullah,h.wang,t.h.w.baeck}@liacs.leidenuniv.nl
[2] Honda Research Institute Europe GmbH (HRI-EU), Offenbach/Main, Germany
{stefan.menzel,bernhard.sendhoff}@honda-ri.de

Abstract. Real-world optimization scenarios under uncertainty and noise are typically handled with robust optimization techniques, which reformulate the original optimization problem into a robust counterpart, e.g., by taking an average of the function values over different perturbations to a specific input. Solving the robust counterpart instead of the original problem can significantly increase the associated computational cost, which is often overlooked in the literature to the best of our knowledge. Such an extra cost brought by robust optimization might depend on the problem landscape, the dimensionality, the severity of the uncertainty, and the formulation of the robust counterpart.
This paper targets an empirical approach that evaluates and compares the computational cost brought by different robustness formulations in Kriging-based optimization on a wide combination (300 test cases) of problems, uncertainty levels, and dimensions. We mainly focus on the CPU time taken to find the robust solutions, and choose five commonly-applied robustness formulations: "mini-max robustness", "mini-max regret robustness", "expectation-based robustness", "dispersion-based robustness", and "composite robustness" respectively. We assess the empirical performance of these robustness formulations in terms of a fixed budget and a fixed target analysis, from which we find that "mini-max robustness" is the most practical formulation w.r.t. the associated computational cost.

Keywords: optimization under uncertainty · robust optimization · surrogate-assisted optimization · Kriging

1 Introduction

Solving a real-world optimization problem entails dealing with uncertainties and noise within the system [2, 9, 16]. Due to various reasons, various types of uncertainties and noise can emerge in optimization problems, e.g., uncertainty and

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

noise in the output of the system if the system is non-deterministic in nature. Hence, for practical scenarios, optimization methods are needed which can deal with these uncertainties, and solutions have to be found which take into account the impact of the unexpected drifts and changes in the optimization setup. The practice of optimization that accounts for uncertainties and noise is often referred to as Robust Optimization (RO) [2, 13, 14].

Despite its significance, achieving robustness in modern engineering applications is quite challenging due to several reasons [4,8]. One of the major reasons is the computational cost involved to find the robust solution. The computational cost mainly depends on the problem landscape, high dimensionality [15], the type and structure of the uncertainty [3], and the robustness formulation (RF) or criterion among others.

While the impact of high dimensionality and problem landscape in RO is understood to some extent [3,4,9,15], the impact of the chosen RF, e.g., "mini-max robustness", on the computational cost, has been overlooked in the literature. For expensive-to-evaluate black-box problems, the chosen robustness criterion can have a significant impact on the computational cost. This is due to the fact that obtaining a robust solution requires additional computational resources as opposed to a nominal one, since the optimizer has to take into account the impact of uncertainty and noise as well. This need for additional computational resources is based on the robustness criterion[§§] chosen, e.g., RO based on the "mini-max" principle requires an internal optimization loop to compute the worst impact of the uncertainty, whereas RO based on the "expectation" of a noisy function requires computing an integral [4, 16].

Since the Computational Cost of Robustness (CCoR) – the need for additional computational resources to find the robust instead of a nominal optimal solution – depends on the robustness criterion chosen, it is desirable to evaluate and compare commonly-employed robustness criteria with regards to computational cost, where the computational cost is based on the CPU time taken to find the robust solution. By evaluating and comparing different robustness criteria based on computational cost and quality of the solution, a novel perspective is provided to practitioners for choosing the suitable robustness criterion for practical scenarios. To the best of our knowledge, there are no systematic studies dealing with this issue so far.

Our contribution in this paper is as following. First, we generalize the Kriging-based RO proposed in [11] (for the "mini-max robustness") to other RFs. Second, we evaluate and compare the empirical performance of Kriging-based RO based on five of the most common RFs, namely "mini-max robustness", "mini-max regret robustness", "expectation-based robustness", "dispersion-based robustness", and "composite robustness" respectively. Note that the performance assessment

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

is based on 300 test cases, owing to the combinations of ten well-known benchmark problems from the continuous optimization domain, two noise levels, three different settings of dimensionality, and five RFs reported. Additionally, the performance in this context is characterized by a fixed budget and a fixed target analysis, as well as the analysis on the average and maximum CPU time. Based on the findings of our investigation, we provide a novel perspective on the computational aspects of these RFs, which is useful when employing these RFs in practical scenarios.

The remainder of this paper is organized as follows. Section 2 describes the basic working mechanism of Kriging-based optimization, and introduces the RFs mentioned above. Section 3 extends the nominal Kriging-based optimization to the robust scenario to account for parametric uncertainties in the search variables. Section 4 describes the experimental setup of our study. This is followed by our experimental results in section 5. The discussion on these results is presented in section 6. Finally, we discuss the conclusions of the paper along side the potential future research in section 7.

2 Background

In this paper, we aim to minimize an unconstrained numerical black-box optimization problem, i.e., $f{:}\mathcal{S} \subseteq \mathbb{R}^D \rightarrow \mathbb{R}$, using Kriging-based optimization (KBO) [7, 11]. KBO works on the principle of adaptive sampling, whereby the Kriging model is sequentially updated according to a sampling infill criterion, such as the "expected improvement" (EI) criterion. The sampling infill criterion tries to balance the search behavior – exploration and exploitation – to find a globally optimal solution on the model surface.

KBO starts by generating an initial data set $\mathcal{D} = (\mathrm{X}, \mathbf{y})$ on the objective function f. The locations: $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, can be determined by the Design of Experiment (DoE) methodologies, such as the Latin Hyper-cube Sampling (LHS) scheme [12]. After this, function responses: $\mathbf{y} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^\top$, are computed on these locations. The next step involves constructing the Kriging model $\mathcal{K}_f$ based on the available data set $\mathcal{D}$. Following this, the next query point $\mathbf{x}_{\text{new}}$ (to sample the function) is determined with the help of a sampling infill criterion, such as the EI criterion. The function response $f(\mathbf{x}_{\text{new}})$ is computed at this location, and the data set is extended. Finally, the Kriging model $\mathcal{K}_f$ is updated based on the extended data set. This process is repeated until either a satisfactory solution is obtained, or a predetermined computational budget or other termination criterion is reached.

When optimizing the function f in real-world applications, we note that it is surrounded by the parametric uncertainties in the decision variables [13, 14]. These uncertainties, commonly denoted as $\boldsymbol{\Delta}_{\mathbf{x}}$, are assumed to be structurally symmetric, additive in nature, and can be modeled in a deterministic or a probabilistic fashion [9]. For the objective function f, the notion of robustness refers to the quality of the solution with respect to these uncertainties. In the following, we define robustness with respect to the five RFs discussed in our paper.

2.1 Robustness Formulations

We start with the so-called "mini-max robustness" (MMR), which deals with deterministic uncertainty [2, 9, 16]. Given a real-parameter objective function: $f(\mathbf{x})$, and the additive uncertainty in the decision variables: $\Delta_{\mathbf{x}}$, the "mini-max" treatment considers minimizing the worst-case scenario of each search point $\mathbf{x}$, where the worst-case is defined as to take 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 $\mathbf{x}$). Effectively, this is to minimize the following objective function

$$f_{\text{eff}}(\mathbf{x}) = \max_{\Delta_{\mathbf{x}} \in \mathrm{U}} \ f(\mathbf{x} + \Delta_{\mathbf{x}}). \tag{1}$$

Note that the radius of the compact set U is based on the anticipated scale of the uncertainty, i.e., based on the maximum anticipated deviation of the decision variables from their nominal values. The worst-case scenario refers to the fact that we consider the maximal f-value under additive uncertainty at each search point, and try to minimize that [16].

As opposed to MMR, the "mini-max Regret Robustness" (MMRR) [8] 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

$$f^*(\Delta_{\mathbf{x}}) = \min_{\mathbf{x} \in \mathcal{S}} \ f(\mathbf{x} + \Delta_{\mathbf{x}}), \tag{2}$$

and the effective (robust) objective function can be defined as

$$f_{\text{eff}}(\mathbf{x}) = \max_{\Delta_{\mathbf{x}} \in \mathrm{U}} \ (f(\mathbf{x} + \Delta_{\mathbf{x}}) - f^*(\Delta_{\mathbf{x}})). \tag{3}$$

Minimizing Eq. (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. (2). The benefit of employing MMRR is that even in the worst-case scenario, we do not compromise significantly in terms of optimality. The biggest challenge, however, is the prohibitively high computational demand. Note that solving Eq. (3) inside an iterative optimization framework, e.g., Kriging-based optimization, implies a quadrupled nested loop, which is computationally infeasible even for a modest setting of dimensionality.

Different from the first two RFs, the expected output of a noisy function can also serve as a robustness criterion [4, 8, 9]. 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 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 counterpart of the original function based on the "expectation-based robustness" (EBR) is defined as

$$f_{\text{eff}}(\mathbf{x}) = \mathbb{E}_{\Delta_\mathbf{x} \sim \mathcal{U}(a,b)}[f(\mathbf{x} + \Delta_\mathbf{x})], \tag{4}$$

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

As opposed to EBR, the "dispersion-based robustness" (DBR) emphasizes on minimizing the performance variance under variation of the uncertain search variables [8, 9]). 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

$$f_{\text{eff}}(\mathbf{x}) = \sqrt{\text{Var}_{\Delta_\mathbf{x} \sim \mathcal{U}(\text{a,b})}[f(\mathbf{x} + \Delta_\mathbf{x})]}. \tag{5}$$

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

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), similar to [16]. 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 scalar output. The optimization goal thus becomes to find a point $\mathbf{x}^*$ in $\mathcal{S}$, which minimizes this scalar

$$f_{\text{eff}}(\mathbf{x}) := \mathbb{E}_{\Delta_\mathbf{x} \sim \mathcal{U}(a,b)}[f(\mathbf{x} + \Delta_\mathbf{x})] + \sqrt{\text{Var}_{\Delta_\mathbf{x} \sim \mathcal{U}(a,b)}[f(\mathbf{x} + \Delta_\mathbf{x})]}. \tag{6}$$

3 Kriging-based Robust Optimization

When aiming to find a solution based on the RFs discussed above, we note that the standard KBO approach as described in section 2 cannot be employed. There are mainly two reasons for that. Firstly, the potential "improvement" which is defined in the nominal scenario renders inapplicable. 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). Secondly, the Kriging surrogate $\mathcal{K}_f$ in the nominal scenario does not model the robust/effective response of the function[††], which is desirable when aiming for a robust solution. Therefore, the standard KBO approach must be extended to the robust scenario, which is henceforth referred to as Kriging-based Robust Optimization (KB-RO) in this paper.

[††]The robust or effective function response has already been defined in section 2 for five of the most common RFs.

Following the approach in [11], the adaptation of the KBO algorithm to KB-RO is done in the following manner. Firstly, one must 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: $\hat{f}^*(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{S}} \ \hat{f}_{\text{eff}}(\mathbf{x})$. 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. (1)- (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. (4), $\hat{f}_{\text{eff}}(\mathbf{x})$ is approximated with monte-carlo samples based on the Kriging prediction at each search point $\mathbf{x} + \Delta_{\mathbf{x}}$.

We model the robust Kriging response of the function using a normally distributed random variable: $Y_{\text{eff}}(\mathbf{x})$, with mean $\hat{f}_{\text{eff}}(\mathbf{x})$ and variance $s^2_{\text{eff}}(\mathbf{x})$, i.e., $Y_{\text{eff}}(\mathbf{x}) \sim \mathcal{N}(\hat{f}_{\text{eff}}(\mathbf{x}), s^2_{\text{eff}}(\mathbf{x}))$. Note that the assumption that $Y_{\text{eff}}(\mathbf{x})$ is normally distributed is not entirely rigorous, but rather a practical compromise [11]. Ideally, we should have attempted to estimate the actual 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 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 RF can never be described with certainty in KB-RO.

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 mean squared error at point $\mathbf{x} + \Delta^*_{\mathbf{x}}$, where $\Delta^*_{\mathbf{x}}$ indicates the worst setting of the uncertainty – which maximizes Eq. (1)or (3) as the case may be. In the context of EBR, $s^2_{\text{eff}}(\mathbf{x})$ has a closed form expression as: $s^2_{\text{eff}} = \frac{1}{J^2}\sum_{i,j}^{J} \mathcal{C}$, 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: $\mathbf{x}^{'}_j = \mathbf{x} + \Delta^j_{\mathbf{x}}$, where $\Delta^j_{\mathbf{x}}$ 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 and its expectation in the robust scenario as

$$\mathcal{I}_{\text{eff}}(\mathbf{x}) = \max\{0, \hat{f}^*(\mathbf{x}) - Y_{\text{eff}}(\mathbf{x})\}, \tag{7}$$

Algorithm 1: Kriging-based Robust Optimization

1: **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
2: Generate the initial data set $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$ on the objective function.
3: Construct the Kriging model $\mathcal{K}_f$ on $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$.
4: **while** the stop criteria are not fulfilled **do**
5: Find robust optimum on the Kriging surface $\mathcal{K}_f$
$$\hat{f}^*(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{S}} \quad \hat{f}_{\text{eff}}(\mathbf{x}).$$
6: Choose a new sample $\mathbf{x}_{\text{new}}$ by maximizing the robust (effective) acquisition function
$$\mathbf{x}_{\text{new}} \leftarrow \operatorname{argmax}_{\mathbf{x}\in\mathcal{S}} \quad \mathscr{A}_{\text{eff}}(\mathbf{x}).$$
7: Compute function response $f(\mathbf{x}_{\text{new}})$.
8: Extend the data set $\mathcal{D}$ by appending the pair $(\mathbf{x}_{\text{new}}, f(\mathbf{x}_{\text{new}}))$ to $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$.
9: Reconstruct the Kriging model $\mathcal{K}_f$ on $\mathcal{D} = \{\mathrm{X}, \mathbf{y}\}$.
10: **end while**
11: **end procedure**

and

$$\mathbb{E}[\mathcal{I}_{\text{eff}}(\mathbf{x})] := (\hat{f}^*(\mathbf{x}) - \hat{f}_{\text{eff}}(\mathbf{x}))\Phi\left(\frac{\hat{f}^*(\mathbf{x}) - \hat{f}_{\text{eff}}(\mathbf{x})}{s_{\text{eff}}(\mathbf{x})}\right) + s_{\text{eff}}(\mathbf{x})\phi\left(\frac{\hat{f}^*(\mathbf{x}) - \hat{f}_{\text{eff}}(\mathbf{x})}{s_{\text{eff}}(\mathbf{x})}\right), \tag{8}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ in Eq. (8) represent the cumulative distribution function and probability density function of the standard normal random variable respectively. An important thing to note here is that the point $\mathbf{x} + \Delta_{\mathbf{x}}$ can become infeasible with respect to the original search space $\mathcal{S}$ if $\mathbf{x}$ is already close to the boundary of $\mathcal{S}$. In this case, we simply clip the infeasible point with the boundary it breaks similar to the approach in [11]. The working mechanism of KB-RO is summarized in Algorithm 1, where the only significant difference to the nominal KBO is the evaluation of steps 5 and 6, which emphasize on robustness.

4 Experimental Setup

Our aim in this paper is to understand the impact of RF in KB-RO with regards to computational efficiency. Intuitively, RF can have a significant impact on the performance of KB-RO since steps 5 and 6 in Algorithm 1 require much more computational resources as opposed to the nominal KBO [7]. This need for additional computational resources is based on the chosen RF. Through our experimental setup**, we aim to better understand this impact for each of the five RFs discussed in the paper. To make our setup comprehensive, we take into account the variability in external factors such as problem landscape, dimensionality, and the scale/severity of the uncertainty.

**The source code to reproduce the experimental setup and results is available at: https://github.com/SibghatUllah13/UllahPPSN2022.

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) [6]. 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 subject to minimization, and is evaluated on three different settings of dimensionality as: $\mathscr{D} = \{2, 5, 10\}$. 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: $\boldsymbol{\Delta}_{\mathbf{x}} \sim \mathcal{U}(a, b)$, 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 KB-RO is set to be $50 \times D$. Note that our Kriging surrogate is based on the popular Matérn 3/2 kernel [5], 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 robust EI in Eq. (8) as the sampling infill criterion for our experiments.

For the parallel execution of KB-RO for each of the 300 test cases considered, we utilize the Distributed ASCI Supercomputer 5 (DAS-5) [1], 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 [10]. The performance assessment of the solutions in our experiments is based on 15 independent runs $\mathscr{R}$ of the KB-RO for each of the 300 test cases considered. Note that for each trial, i.e., the unique combination of the independent run and the test case, we ensure the same configuration of hardware and software to account for fairness. Furthermore, in each trial, we measure the CPU time for all iterations of KB-RO. After the successful parallel execution of all trials, we assess the quality of the optimal solutions based on Normalized Mean Absolute Error (NMAE) as: $\mathrm{NMAE} = \left(\frac{|f^* - \hat{f}^*|}{|f^*|}\right)$, where f^* denotes the true robust optimal function value obtained from solving the original function under uncertainty, also referred to as the ground truth for the particular choice of the test case, and $\hat{f}^*$ serves as the robust optimal func-

tion value obtained from KB-RO (step 5 in Algorithm 1). As noted in [16], the benefit of utilizing NMAE is that the quality of the optimal solution is always determined relative to the corresponding ground truth, and the performance of KB-RO across different RFs can be easily compared.

Having obtained the quality of the optimal solution and CPU time for all iterations of the KB-RO for each trial, we perform a fixed budget and a fixed target analysis. For the fixed budget analysis, we consider two possibilities. Firstly, we perform the analysis with respect to the running CPU time by fixing 50 different settings of the CPU time. For each such setting, we report the best quality solution (measured in terms of NMAE) obtained from KB-RO. The performance in this context is averaged over all 50 settings of the CPU time. Secondly, we perform the fixed budget analysis also with respect to the number of iterations. In this context, we identify 30 different settings of the number of iterations (checkpoints) to analyze the performance similar to the previous case.

Contrary to this, in fixed target analysis, we identify 10 distinct target values for the NMAE – a set of thresholds describing the minimum desirable quality of the solution. As soon as a particular target is achieved, we report the accumulated CPU time taken by KB-RO to reach that target. If such a target is never achieved, we report the penalized CPU time which is set to be $D \times T_{\max}$, where $D \in \mathscr{D}$ is the corresponding setting of the dimensionality, and $T_{\max}$ is the accumulated CPU time at the last iteration of that trial. In addition to the fixed budget and fixed target analysis, we also report the average CPU time per iteration (ACTPI), and $T_{\max}$ for each trial.

5 Results

We share the results originating from our experiments in Figs. 1 and 2. In particular, Fig. 1 focuses on four distinct analyses, which include fixed CPU time analysis, fixed iterations analysis, fixed target analysis, and the analysis on the ACTPI. On the other hand, Fig. 2 reports the accumulated CPU time at the last iteration of KB-RO: $T_{\max}$, which is averaged over 15 independent runs $\mathscr{R}$, and grouped by the RFs. Note that the results in Fig. 1 are presented in the form of empirical cumulative distribution function (ECDF) for each RF and for each type of analysis. The first row of plots in Fig. 1 illustrates the results on fixed CPU time and fixed iteration analyses respectively (from left to right). In a similar fashion, the second row of plots illustrates the performance with respect to the fixed target analysis and the analysis on the ACTPI. Note that each curve in these analyses is based on 900 data points (trials) due to 15 independent runs $\mathscr{R}$ of KB-RO, 10 test functions in $\mathscr{F}$, 3 settings of dimensionality in $\mathscr{D}$, and 2 noise levels in $\mathscr{L}$. The results in Fig. 2 are presented in the form of box plots, where each box inside a subplot presents $T_{\max}$ values for the test cases corresponding to the particular setting of the dimensionality and RF. The reported $T_{\max}$ in this context is averaged over 15 independent runs $\mathscr{R}$ of KB-RO.

In terms of performance comparison with respect to the fixed CPU time analysis, we note the promising nature of all RFs except DBR, which performs poorly compared to its competitors in most trials. Furthermore, we also note the

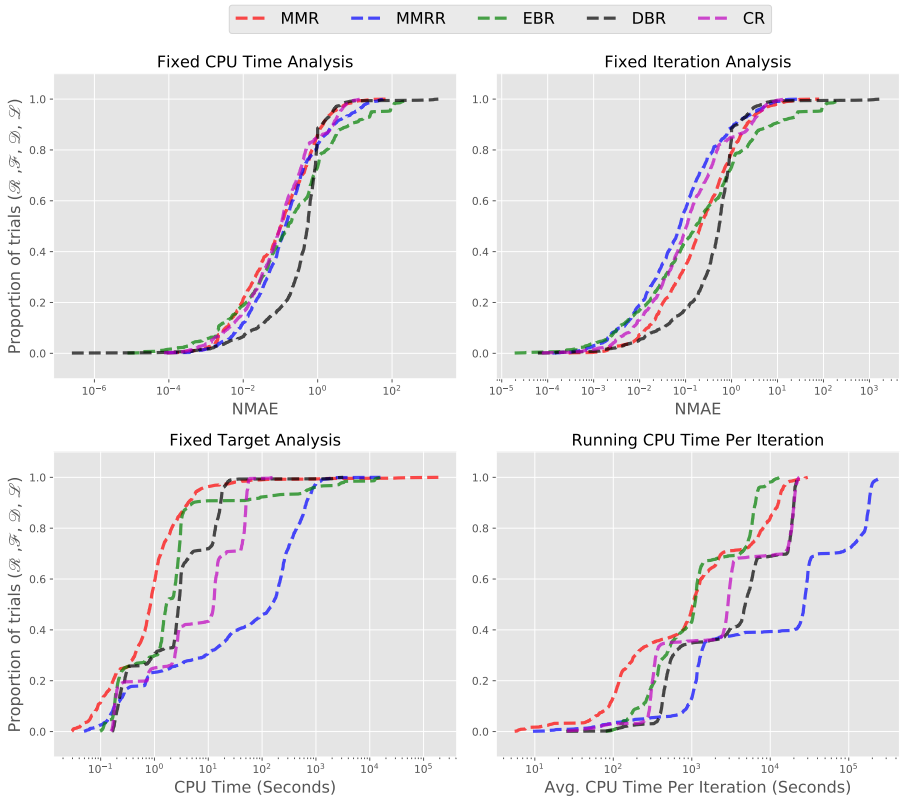

Fig. 1. Upper left: Fixed CPU time analysis, Upper right: Fixed iteration analysis, Lower left: Fixed target analysis, Lower right: Average running CPU time per iteration. For each analysis, the empirical cumulative distribution function (ECDF) for all five RFs is presented. Each ECDF curve is based on 900 data points (trials) owing to 15 independent runs $\mathscr{R}$, 10 test functions in $\mathscr{F}$, 3 settings of dimensionality in $\mathscr{D}$, and 2 noise levels in $\mathscr{L}$.

highest variation in quality (NMAE) for DBR. Although no RF is deemed a clear winner for this analysis, we note that MMR, MMRR and CR have high empirical success rates. Likewise, we note the highest variation in quality for DBR also in the context of fixed iteration analysis. In this case, MMRR and CR perform superior to the other RFs as we observe a high empirical success rate for both. For the performance measure with respect to the fixed target analysis, we observe that MMR outperforms the competitors, albeit the variation in the running CPU time for MMR is also deemed higher. Here, we note a clear distinction in the empirical success rate between MMRR and other RFs. For instance, if we cut-off the running CPU time at 100 seconds, we observe that MMRR has an empirical success rate of 45 %, whereas MMR, DBR, and CR achieve a success rate of more than 99 %. We also note that MMRR has the highest variance of ACTPI, and the lowest empirical success rate of all RFs. In this case, none of the MMR

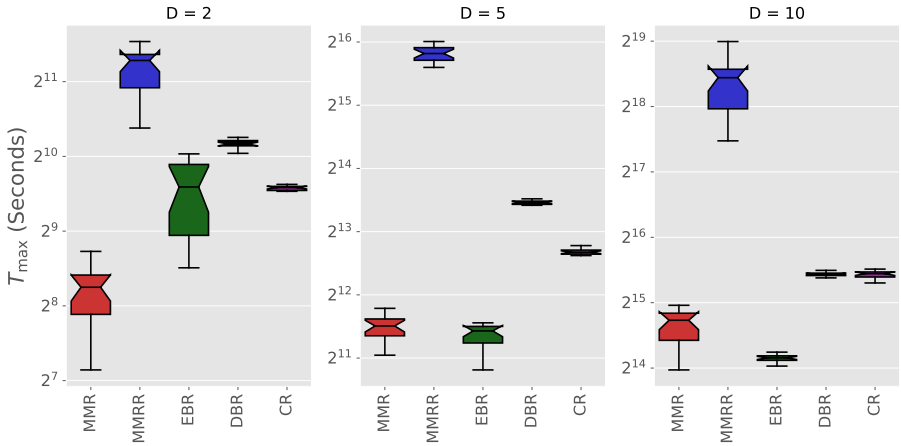

Fig. 2. Left: $2D$ test cases, Middle: $5D$ test cases, Right: $10D$ test cases. Each box plot shares the maximum CPU time spent to run KB-RO: $T_{\max}$, averaged over 15 independent runs in $\mathscr{R}$ and grouped by the RFs.

and EBR can be deemed a clear winner, although both perform superior to other RFs in most trials.

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

6 Discussion

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

In terms of performance, we find that MMRR poses an interesting situation as it performs competitively in the context of fixed budget analyses. However, its performance is significantly worse to other RFs in the context of fixed target

analysis, the ACTPI, and the maximum CPU time $T_{\max}$ for running KB-RO. We believe this is aligned with the intuition of MMRR (as discussed in section 2), since within an iterative optimization framework, we have to employ a quadrupled nested loop to find the robust solution based on MMRR, which in turn exponentially increases the computational cost per iteration. The MMRR, therefore, has the highest CCoR, and takes much more CPU time to reach the same target value as opposed to other RFs.

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

7 Conclusion and Outlook

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

A fixed budget analysis on our experimental results suggests the applicability of "mini-max robustness", "mini-max regret robustness", "expectation-based robustness", and "composite robustness" in practical scenarios where the designer cannot afford the computational budget beyond a certain threshold. On the other hand, a fixed target analysis deems the 'mini-max robustness" as the most efficient robustness criterion in the scenario where the designer cannot compromise the quality of the optimal solution below a certain threshold. The analysis on the ACTPI and $T_{\max}$ also determines "mini-max robustness" as one of the most efficient robustness criteria. Overall, "mini-max robustness" is understood to be the most suitable robustness criterion with regards to the associated computational cost.

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

References

1. Bal, H., Epema, D., de Laat, C., van Nieuwpoort, R., Romein, J., Seinstra, F., Snoek, C., Wijshoff, H.: A medium-scale distributed system for computer science research: Infrastructure for the long term. Computer **49**(5), 54–63 (2016)
2. Ben-Tal, A., Ghaoui, L.E., Nemirovski, A.: Robust Optimization, Princeton Series in Applied Mathematics, vol. 28. Princeton University Press (2009)
3. Beyer, H.G.: Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. Computer methods in applied mechanics and engineering **186**(2-4), 239–267 (2000)
4. Beyer, H.G., Sendhoff, B.: Robust optimization–a comprehensive survey. Computer methods in applied mechanics and engineering **196**(33-34), 3190–3218 (2007)
5. Genton, M.G.: Classes of kernels for machine learning: A statistics perspective. J. Mach. Learn. Res. **2**, 299–312 (2001)
6. Hansen, N., Auger, A., Mersmann, O., Tusar, T., Brockhoff, D.: COCO: A platform for comparing continuous optimizers in a black-box setting. CoRR **abs/1603.08785** (2016)
7. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)
8. Jurecka, F.: Robust design optimization based on metamodeling techniques. Ph.D. thesis, Technische Universität München (2007)
9. Kruisselbrink, J.W.: Evolution strategies for robust optimization. Leiden Institute of Advanced Computer Science (LIACS), Faculty of Science (2012)
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
11. ur Rehman, S., Langelaar, M., van Keulen, F.: Efficient kriging-based robust optimization of unconstrained problems. J. Comput. Sci. **5**(6), 872–881 (2014)
12. Stein, M.: Large sample properties of simulations using latin hypercube sampling. Technometrics **29**(2), 143–151 (1987)
13. Taguchi, G., Konishi, S.: Taguchi Methods: Orthogonal Arrays and Linear Graphs. Tools for Quality Engineering. American Supplier Institute Dearborn, MI (1987)
14. Taguchi, G., Phadke, M.S.: Quality engineering through design optimization. In: Quality Control, Robust Design, and the Taguchi Method, pp. 77–96. Springer (1989)
15. Ullah, S., Nguyen, D.A., Wang, H., Menzel, S., Sendhoff, B., Bäck, T.: Exploring dimensionality reduction techniques for efficient surrogate-assisted optimization. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 2965–2974. IEEE (2020)
16. Ullah, S., Wang, H., Menzel, S., Sendhoff, B., Back, T.: An empirical comparison of meta-modeling techniques for robust design optimization. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 819–828. IEEE (2019)