ABSTRACT

Title of Dissertation:	SINGLE- AND MULTI-OBJECTIVE FEASIBILITY ROBUST OPTIMIZATION UNDER INTERVAL UNCERTAINTY WITH SURROGATE MODELING
	Randall J. Kania, Doctor of Philosophy 2022
Dissertation directed by:	Professor Shapour Azarm, Mechanical Engineering

This dissertation presents new methods for solving single- and multi-objective optimization problems when there are uncertain parameter values. The uncertainty in these problems is considered to come from sources with no known or assumed probability distribution, bounded only by an interval. The goal is to obtain a single solution (for single-objective optimization problems) or multiple solutions (for multi-objective optimization problems) that are optimal and "feasibly robust". A feasibly robust solution is one that remains feasible for all values of uncertain parameters within the uncertainty interval. Obtaining such a solution can become computationally costly and require many function calls. To reduce the computational cost, the presented methods use surrogate modeling to approximate the functions in the optimization problem.

This dissertation aims at addressing several key research questions. The first Research Question (RQ1) is: How can the computational cost for solving single-objective robust optimization problems be reduced with surrogate modelling when compared to previous work?

RQ2 is: How can the computational cost of solving bi-objective robust optimization problems be improved by using surrogates in concert with a Bayesian optimization technique when compared to previous work? And RQ3 is: How can surrogate modeling be leveraged to make multi-objective robust optimization computationally less expensive when compared to previous work?

In addressing RQ1, a new single-objective robust optimization method has been developed with improvements over an existing method from the literature. This method uses a deterministic, local solver, paired with a surrogate modelling technique for finding worst-case scenario of parameter configurations. Using this single-objective robust optimization method, improved large-scale performance and robust feasibility were demonstrated. The second method presented solves bi-objective robust optimization problems under interval uncertainty by introducing a relaxation technique to facilitate combining iterative robust optimization and Bayesian optimization techniques. This method showed improved feasibility robustness and performance at larger problem sizes over existing methods. The third method presented in this dissertation extends the current literature by considering multiple (beyond two) competing objectives for surrogate robust optimization. Increasing the number of objectives adds more dimensions and complexity to the search for solutions and can greatly increase the computational costs. In the third method, the robust optimization strategy from the bi-objective second method was combined with a new Monte Carlo approximated method.

The key contributions in this dissertation are 1) a new single-objective robust optimization method combining a local optimization solver and surrogate modelling for robustness, 2) a bi-objective robust optimization method that employs iterative Bayesian optimization technique in tandem with iterative robust optimization techniques, and 3) a new acquisition function for robust optimization in problems of more than two objectives.

SINGLE- AND MULTI-OBJECTIVE FEASIBILITY ROBUST OPTIMIZATION UNDER INTERVAL UNCERTAINTY WITH SURROGATE MODELING

by

Randall Joseph Kania

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Shapour Azarm, Chair
Professor Balakumar Balachandran
Professor Jin-Oh Hahn
Professor Jeffrey Herrmann
Professor P.K. Kannan, Dean's Representative

Dedication

This work is dedicated to my beloved family, dear friends, and patient pets.

Acknowledgements

First off, I would like to thank my advisor, Dr. Shapour Azarm for the years of support and patience. It is largely through his efforts that I am able to present this dissertation.

I would also like to thank the members of my committee, Dr. Balakumar Balachandran, Dr. Jin-Oh Hahn, Dr. Jeffrey Herrmann, and Dr. P. K. Kannan for agreeing to sit on my committee, and for the time and effort they put into providing constructive feedback.

Additionally, I would like to gratefully acknowledge funding received, in partial support of this work, from the Air Force Office of Scientific Research (AFOSR) through Grant FA95501510134. This work was in part supported by a generous gift from Dr. Alex Mehr (PhD ’03) through the Design Decision Support Laboratory Research and Education Fund. Finally, I would like to acknowledge the University of Maryland High Performance Computing resources (http://hpcc.umd.edu) made available for expediting some of the research results reported in this dissertation.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AA-MORO	Approximation Assisted Multi-Objective Robust Optimization
CMC	Centroidal Monte Carlo
CTP	Constrained Test Problem
DTLZ	Deb, Thiele, Laumanns, and Zitzler
EC	Exact Centroidal
EHVI	Expected Hypervolume Improvement
GP	Gaussian Process
HD	Hypervolume Difference
MORO-KS	Multi-Objective Robust Optimization with Kriging and Support vector machines
OS	Overall Spread
SFTC-RO	Surrogate Feasibility Testing-Cutting Robust Optimization
SGLRO	Scenario Generation with Local Robust Optimization
SP	Spacing
SRO	Scenario Robust Optimization
TNK	Tanaka
WFG	Walking Fish Group
ZDT	Zitzler, Deb, and Thiele

Nomenclature

a	constant
b	constant
c	constant
d	dimension of design vector
E	expectation operator
f_i	objective function i
$\bar{f}_i$	centroid of region for objective function i
$\hat{f}_i$	surrogate model of objective function i
$\boldsymbol{F}(\boldsymbol{G})$	total set of objective (constraint) values for training models
f_i^{new}	component of new candidate solution for the i^{th}objective
f_i^{ND}	component of current non-dominated solutions for the i^{th} objective
$f_i^{\mathrm{ND},c}$	component of non-dominated solution closest to candidate point for the i^{th} objective
$\boldsymbol{F}_{nd}(\boldsymbol{G}_{nd})$	set of points with objective (constraint) values excluding infeasible points that dominate feasible points
$\boldsymbol{F}_{nda}(\boldsymbol{G}_{nda})$	points from $F_{nd}(G_{nd})$ except those predicted to be infeasible
f_{min}	current best sample for single objective Bayesian optimization
$\boldsymbol{F}_p$	current candidate non-dominated samples
g_j	constraint function j
$\hat{g}_j$	surrogate model of constraint function j
$\boldsymbol{G}_{wc}$	set of worst constraint values
$i(I)$	index of (total number of) objectives
Imp	improvement over current solution
$j(J)$	index of (total number of) constraints
k	index of non-dominated optimal points
K	covariance function
m	index of design vector dimension
n	total number of design variables and uncertain parameters
N_c	number of iterations in a loop (for adding multiple points per iteration)
P	probability operator
$r^{\mathrm{ND},c}$	distance to the closest non-dominated solution from candidate point
$s(S)$	Index of (total number of) scenarios

$\hat{s}$	standard deviation of Gaussian process
$\boldsymbol{u}$	vector of uncertain parameters
$\boldsymbol{u}^*$	optimizing worst-case scenario
$\boldsymbol{u}_0$	nominal uncertain scenario
$\boldsymbol{u}_l(\boldsymbol{u}_u)$	lower (upper) bounds of uncertain parameters
$\boldsymbol{u}_{wc}$	worst-case scenario of an iteration
$\boldsymbol{U}_{wc}$	set of worst-case scenarios
$\boldsymbol{x}$	vector of design variables
$\boldsymbol{x}^*$	optimizing design vector
$\boldsymbol{x}'$	vector of design variables at previously sampled location
$\boldsymbol{x}_l(\boldsymbol{x}_u)$	lower (upper) bounds of design variables
$\boldsymbol{Y}$	sample normalized by surrogate prediction and predicted error
μ	mean function of Gaussian process
σ_f, σ_m	covariance function hyperparameters
ϕ	Gaussian probability distribution function
Φ	Gaussian cumulative distribution function
$\leftarrow$	'assign' operator
$\leftarrow^+$	'append' operator

Chapter 1: Introduction

This dissertation considers solution methods for a class of nonlinear engineering design optimization problems in which there is uncertainty in some parameter values. The uncertainty is assumed to be a value within a fixed interval. The solution to these problems can require a large number of function calls, and as such can be computationally costly to obtain. To reduce the computational cost, surrogate modelling can be used.

The work in this dissertation ties together two major fields of study namely: robust optimization and surrogate modelling. These components are utilized for the three main research directions in this work, namely, single-objective robust optimization under interval uncertainty with surrogate modelling (Chapter 2), bi-objective robust optimization under interval uncertainty with surrogate modelling (Chapter 3), and multi-objective (beyond two) robust optimization under interval uncertainty with surrogate modelling (Chapter 4).

The subsections below will motivate and introduce these components, while reviewing the current state of the literature, research gaps and questions for this dissertation. In the following, first, background information on robust optimization is provided in Section 1.1, second, background information on surrogate modelling and surrogate optimization is provided in Section 1.2. Next, current techniques in single-objective robust optimization are given in Section 1.3. Then, the current techniques in multi-objective robust optimization are presented in Section 1.4. Research gaps and

research questions to be answered in this dissertation are given in Section 1.5. Finally, the organization of the rest of this dissertation is outlined in Section 1.6.

1.1 Background: Robust Optimization

One way to account for uncertainty in engineering design optimization problems is by using robust optimization. Specifically, feasibility robust optimization is the process through which an optimum solution is found such that it remains feasible despite uncertainty in the values of parameters. In the context here, an optimum solution is composed of decision variables that are the entities that the decision maker has, for the most part, control over, i.e., they are controllable, such as planned dimensions of a part to manufacture. Parameters are those that the decision maker does not have control over, such as weather conditions. A robust optimal solution may also include a solution which limits deterioration in the objective function value when subjected to uncertainty. However, as this can be accounted for by introducing constraints on the objective function's variation range [1], this dissertation will focus on feasibility robust optimization and so only includes uncertainty in the constraint functions.

Uncertainty in an optimization problem can be considered in one of several ways. It can be assumed that there is complete distributional knowledge about the uncertainty [2], partial distributional knowledge [3], or interval uncertainty knowledge [4], [5]. In the methods presented in this dissertation it is assumed that there is interval uncertainty, or interval knowledge of uncertain parameters. Generally speaking, interval uncertainty can be handled by semi-infinite programming [5],

uniform discretization [4], or by searching for the worst-case scenario [6]–[9].

Any of the above-mentioned techniques for handling interval uncertainty adds substantially to the computational cost of the optimization process. Iterative methods (often termed "sequential" in the literature) for searching for the worst-case scenario have shown improved results in terms of numbers of function evaluations [1], [10], [11]. Although the iterative method is more computationally efficient than other such scenario searching methods, further cost savings can be achieved by combining it with approximation methods. To this end, surrogate modelling with the worst-case search is used in the proposed methods of this dissertation.

1.2 Background: Surrogate Modelling and Optimization

If the objective or constraint functions are computationally intensive, then optimizing can become prohibitively computationally expensive. A useful method for improving computational cost is surrogate modelling, a general form of which is seen in Eq. (1.1). Surrogate models or metamodels are models that approximate a computationally expensive function, f, by fitting an approximate (surrogate) function, $\hat{f}$, using a relatively small number of samples from the expensive function. Computational expense is thus measured as the number of function evaluations. Sample pairs of inputs, x, and outputs, $f(x)$, are used to determine the weight parameters, w, to make the approximation as accurate as possible.

$$\hat{f}(x,w) \cong f(x) \tag{1.1}$$

Some of the most common surrogate models in use are Support Vector

Regression [12], Radial Basis Functions [12], [13], and Gaussian Processes (GP) or Kriging models [12], [14]. The three models are listed in order of generality, i.e., Gaussian Processes (or Kriging models) are a specific case of a Radial Basis Functions, and Radial Basis Functions are a specific case of Support Vector Regression [12]. In this dissertation, all surrogates are GPs with the automatic relevance determination squared exponential kernel allowing for a separate hyperparameter to be fit for each design variable [15]. Readers interested in an in depth discussion of surrogate optimization are directed to [12], or for GPs to [14].

All of the methods discussed in this dissertation are concerned with objective and constraint functions with high computational cost, e.g., from evaluation of computational fluid dynamics simulations. In order to test a variety of examples in a reasonable time frame and with multiple independent trials, less computationally intensive problems are substituted. To enable comparison between the cheaper examples and the costlier real-world problems, computational expense is measured by the number of function evaluations used to find a solution. The implicit assumption being that the costly function evaluations would dwarf the optimization and approximations computations in terms of resources or time. Additionally, it is assumed in this dissertation that every objective and constraint function is expensive and so is approximated by a surrogate model whenever possible. Realistically, problems often contain some expensive and some cheaper functions. The methods in this dissertation could be modified to only approximate the expensive functions and allow the cheaper ones to be evaluated directly without approximation.

1.3 Single-Objective Robust Optimization

For many applications, restricting problems to be convex is an apt assumption [6]. Especially for the applications considering real-time decision making such as in control problems [7], [8], [16]–[18]. For engineering design optimization, convexity cannot always be assumed, and more general methods must be employed [10], [19]. Considering nonconvexities will naturally lead to solving more difficult and computationally intensive optimization problems. To bring the computational cost within a more affordable range, algorithms such as the modified Bender's decomposition [4] can be used, or the problem can be approximated by surrogates [12]. The modified Bender's decomposition [4] assumes a discretized range for the uncertain parameters. This can lead to infeasibilities in the solution if any of the constraints is violated at a value of the uncertain parameters not included in the discretization. Semi-infinite programming and worst-case search are not as susceptible to this but can be computationally costly.

To find a robust optimum solution via worst-case search, the task is to consider and add all necessary scenarios and their corresponding constraints to pare down the feasible domain while avoiding adding extra scenarios and constraints. Performing a worst-case search can either be done in a nested [11], [20] or iterative manner [10], [21]. Because random sampling and uniform discretization run the risk of producing infeasible solutions, and semi-infinite programming and nested worst-case search can have prohibitive computational costs, the iterative worst-case search is utilized throughout this dissertation.

From the previous literature, one notable iterative technique is the Scenario Generation with Local Robust Optimization (SGLRO) method which used as its bottom-level problem a random sampling technique combined with sequential quadratic programming [10]. This method uses a sequential quadratic programming solver in the top-level for finding a robust optimal solution as was done by [11], and only added constraints violated in the bottom-level to the top-level problem. SGLRO has the best reported performance in the literature for the class of problems of interest in this dissertation, so the proposed method will be compared against SGLRO to gage performance.

1.4 Multi-Objective Robust Optimization

Engineering design optimization problems usually have two or more competing objective functions. When the correct balance between the competing objectives or their utility to the decision maker cannot be known *a priori*, a set of Pareto (or non-dominated) optimal solutions can be obtained. For the set of Pareto solutions, none of the points is better than any other with respect to all the objectives, while they are the "best" solutions that can be obtained. Techniques for finding these solutions fall under the topic of multi-objective design optimization.

In addition, for these problems, there is almost always a need to account for uncertainty. Pareto optimal design solutions obtained under deterministic conditions will often fall short when introduced to uncertain conditions, sometimes with catastrophic results. To safeguard against such an outcome, the concept of "robustness" can be introduced. By robustness, it is meant that the solutions obtained

are rather insensitive to the uncertainty. Robustness can be added to the objective functions of the optimization problem to limit performance degradation, and/or to the constraints to ensure feasibility of an optimized design outside of expected conditions. These are respectively termed objective and feasibility robustness [1].

Much of the studies of robust optimization are focused on single-objective design optimization problems [6], [7], [9], [10], [22], [23]. Linear [9] and conic problems have been well studied [6], as have convex robust optimization problems [7], [22]. More general nonconvex problems have been considered as well while attempting to lower their high computational costs without surrogate methods [10], [23]. Surrogate modelling of these expensive and nonlinear functions has become an area of increasing interest [24]–[29], showing great potential for lowering the numbers of function evaluations. Surrogates have also started to be employed to offset the increased computational costs while considering robustness [2], [30]–[32]. However, this is largely for distributional uncertainty, where a probability distribution is assumed to be known for the uncertain parameters [2], [30], [32], with only recent work [31] addressing interval uncertainty.

Uncertainty has also begun to be considered for multiple objective design optimization problems which include expensive functions to calculate, increasing the computational burden [20], [21], [33]–[35]. As such, surrogates have been utilized to lower the computational cost of multi-objective robust optimization problems under interval uncertainty [21], [36]–[38] and distributional uncertainty (or noise) [34], [35]. Focusing on interval uncertainty, this can be done in an offline sense, where the

sampling and model training are performed all-at-once in advance, and the surrogates are then directly substituted for the expensive functions in the optimization scheme [36], [37], [39]. It has also been done online, where the surrogates are updated iteratively, with the actual functions directly substituted with the surrogate's mean predictions [21], [38], [40], [41]. Two of these methods, Approximation Assisted Multi-objective Robust Optimization (AA-MORO) [21] and Multi-objective Robust Optimization with Kriging and Support vector machines (MORO-KS) [38] are compared against in the Examples section of Chapter 3. Those methods do not utilize their function and uncertainty predictions together, instead they use the uncertainty predictions secondarily to the function predictions.

For the case with multiple objective functions, the Approximation Assisted Multi-Objective Robust Optimization (AA-MORO) method developed by [21] treated both decision variables and uncertain parameters as inputs to the surrogate model to train over. In this way, the top- and bottom-level problems are solved using the same constraint function surrogate model. Although AA-MORO uses surrogates to lessen the computational burden, it does not incorporate other techniques like those used by [10]. No other method known to the authors using surrogate modelling has been shown to solve black-box, single-objective robust optimization problems using a worst-case search. Also, the technique in AA-MORO has never been applied to single-objective optimization problems in the literature. In the work presented here on single-objective robust optimization, it is shown that formulating a surrogate on the bottom level can lead to computational cost savings, compared to when it is not used.

Within the surrogate optimization literature, there are techniques that combine the function mean prediction and its error together to balance "exploitation" and "exploration" [24]. This enables more efficient use of the original costly function, decreasing the amount of function calls it needs to be queried. The process of building a global model based on observations and an assumed prior distribution and repeatedly sampling and updating the posterior distribution until arriving at an optimum is called Bayesian optimization [29]. In Bayesian optimization, the function used to select the next point or location to sample based on the current samples and the prior distribution is known as the acquisition function [25]. Acquisition functions can also allow for constraint handling as a standard penalty [28], [42], modified by the probability that the constraint is expected to be violated [26], [27], or directly augmenting with the Lagrangian [43], among other approaches. Although outside of the context of Bayesian optimization, another approach of interest is that of [44], where constraints are relaxed in proportion to the estimated error in the constraint model.

One of the most widely utilized acquisition functions in bi-objective (or single-objective) Bayesian optimization is the expected improvement function [12], [24], [43], [45]. Bayesian optimization combines the model prediction and the standard deviation of the prediction to estimate the amount that a new sample is expected to improve over existing samples. One of the more popular bi-objective formulations, sometimes called the centroid method, does not seek to create a dense Pareto frontier but instead only tries to improve upon existing points leaving the number of solutions fairly constant [12]. Another expected improvement approach

uses the hypervolume improvement metric to improve over existing points and improve solution density [46]. The two methods are complementary, with the centroidal approach using the nondominated area and the hypervolume approach using the dominated area. However, the centroidal approach relies on the approximated distribution of the surrogate model, and the hypervolume approach requires a user supplied reference point dominated by all Pareto solutions. The methods in this dissertation focus on the centroid method. Of some interest is another analogous method for the lower confidence bound acquisition function with a modified hyper volume difference quality metric [47].

For many applications, formulating a design problem as single- or bi-objective is sufficient to capture a decision maker's goals. However, for problems with greater complexity or more stakeholders it may be desired to relax this limit and consider even more competing objectives. The restriction to two or fewer objectives is largely due to a lack of intuition and ease of visualization at higher dimensions. Multi-objective surrogate optimization inherits many techniques from deterministic multi-objective optimization. Most commonly, improvement of the whole solution set is often described by a single metric which is then optimized using a single-objective solver [12], [45]–[51]. Alternatively, some methods utilize multi-objective solvers, generally multi-objective genetic algorithms, and use improvement measures in the surrogate for each objective as the objective functions for the solver [52]. Some of the most popular of these methods are ParEGO [45], Expected Hyper-Volume Improvement [34], [46], [53], and the centroidal (or Euclid) method [12], [48], [49]. A comparison of these methods for unconstrained optimization has been performed in

[48]. Fast calculation of the Expected Hyper-Volume Improvement is an area of active research [34], [54]–[57]. As of yet, none of these methods for multi-objective optimization have been employed in finding solutions robust to uncertainty. Some of the current multi-objective surrogate robust optimization solvers instead use the surrogate predictions directly [21], [37], [38], [40]. Because of the direct substitution of the surrogate for the expensive functions, these methods are largely applicable to problems with greater than two objective functions, even though they are almost entirely applied to bi-objective problems. The drawback is that this direct substitution creates an exploitation focused approach which may not perform as well in higher dimensions.

1.5 Research Gaps and Questions

In this section, an overview of the research gaps in the existing literature is provided. The research gaps are used to construct a few key research questions that this dissertation aims to investigate and shed light on.

1.5.1 Single-Objective Robust Optimization: Research Gap and Question

Within the current literature for single-objective robust optimization under interval uncertainty, there is a lack of investigation into the use of surrogates to solve worst-case scenario search subproblems. Interval uncertainty is generally handled either through discretization of the parameter space [4], random sampling [22], or some search of the parameter space, usually for the worst case [10]. However, there has not yet been introduced a combined approach that uses a deterministic, local solver for the scenario robust optimization problem at the top level and a surrogate for

the global worst-case search on the bottom level for single-objective robust optimization problems under interval uncertainty. Thus, the first Research Question (RQ1) is as follows:

RQ1: How can surrogate modelling be introduced to single-objective robust optimization under interval uncertainty to reduce the computation burden of the worst-case scenario search?

1.5.2 Bi-Objective Robust Optimization: Research Gap and Question

Current literature on multiobjective robust optimization using surrogate modelling is largely restricted to two objective functions [21], [36]–[38], [40]. Of these methods, some use only offline sampling and training of the surrogates [36], [37], [40], while others use an exploitation focused approach for the surrogate model optimization [21], [38]. Bayesian optimization approaches that balance exploitation and exploration have not been previously introduced to the field of multi-objective surrogate robust optimization. Thus, the second research question is as follows:

RQ2: How can Bayesian optimization techniques be included to improve over current surrogate modelling methods for bi-objective robust optimization under interval uncertainty?

1.5.3 Multi-Objective Robust Optimization: Research Gap and Question

As stated above, there are few known approaches to multi-objective robust optimization using surrogate modelling that have been shown to handle problems with greater than two objective functions. The currently used techniques all multi-objective genetic algorithms as their primary solver, and so can theoretically be

adapted to problems of three or more objective functions [21], [36]–[38], [40]. However, use of these methods on problems with more than two objective functions has not been shown. Additionally, the methods mentioned are focused on exploitation of the surrogates with some performing exploration secondarily [21], [38] and some not at all [36], [37], [40]. The third research question to be addressed in this dissertation is then:

RQ3: How can Bayesian optimization techniques be included in robust optimization under interval uncertainty for problems with more than two objectives?

1.6 Organization

The remainder of the dissertation is outlined in this section.

1.6.1 Chapter 2: Single-Objective Robust Optimization

The first research push of this dissertation is the application of surrogate optimization techniques to single-objective robust optimization. Uncertainty is handled using a two-level approach with the robust optimization problem on the top level, and a worst-case scenario search on the bottom level. Several numerical and engineering examples from the literature of single-objective problems with interval uncertainty are tested for comparison with an existing method. Computational savings are shown for only replacing the bottom level problem with the surrogate optimization. Trends in performance for problems with increasing numbers of design variables and uncertain parameters is also investigated empirically. Finally, a verification study is presented to examine the robustness of the optimal solutions.

1.6.2 Chapter 3: Bi-Objective Robust Optimization

The second research push of this dissertation covered in Chapter 3 is the introduction of Bayesian optimization to robust optimization problems of two objectives. Surrogate models were constructed and utilized for both top and bottom levels of a new iterative robust optimization scheme. The natures of the iterative robust optimization scheme and the Bayesian optimization are at odds with each other, so a new relaxation method was developed to foster compatibility. Examples from the literature of bi-objective problems with interval uncertainty are tested and compared with two methods from the literature.

1.6.3 Chapter 4: Multi-Objective Robust Optimization

The third research push of this dissertation covered in Chapter 4 is the extension to problems with more than two objective functions. An acquisition function for Bayesian optimization of two objectives was extended to be applicable for higher numbers of objectives. This was compared with another acquisition function to see which was better suited for use in robust optimization. Additionally, analytical and numerical forms of the two acquisition functions were compared. Methods are compared on two numerical examples from the literature with variable numbers of objective functions and an engineering example modified to have three objective functions.

1.6.4 Chapter 5: Conclusions

In the last chapter, some concluding remarks are made about the work in this dissertation. The contributions of Chapters 2-4 are then detailed. Finally, directions

for future work to build on this dissertation are provided.

In the next chapter, a surrogate-based methodology for single-objective robust optimization under interval uncertainty is presented to address RQ1.

Chapter 2: Single-Objective Robust Optimization with Surrogates[1]

The proposed method in this chapter, Surrogate Feasibility Testing-Cutting Robust Optimization (SFTC-RO), is implemented by iteratively solving two optimization subproblems. In the first (top-level) subproblem, a Scenario Robust Optimization (SRO) problem is solved using a local optimization technique. In the second (bottom-level) subproblem, wherein constraints are maximized to determine the worst-case feasibility scenario (or uncertain parameter value) generation and constraint cuts. The two subproblems are iteratively solved until convergence when the solution obtained can no longer be affected by additional constraint cuts.

The rest of this chapter is organized as follows. Section 2.1 describes the problem definition. Section 2.2 gives an overview of previous methods. Section 2.3 provides a description of the proposed method. Section 2.4 presents the results for five numerical and engineering examples. Finally, Section 2.5 gives some concluding remarks.

2.1 Problem Definition

A single-objective feasibility robust optimization problem under interval uncertainty can be formulated as in Eq. (2.1)

[1] A version of this chapter was published as a journal article in: Engineering Optimization [31]

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &s.t. \quad g_j(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \quad j \in 1, \dots, J \\ &\qquad \boldsymbol{x} \in [\boldsymbol{x}_l, \boldsymbol{x}_u] \\ &\qquad \forall \boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u] \end{aligned} \tag{2.1}$$

where the constraints are to be satisfied for every combination of uncertain parameter values in the interval $\boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u]$. This interval forms a set which is assumed to be compact, i.e., closed and bounded. The interval is often replaced with a finite number of parameter value combinations (also called scenarios) that are assumed to approximate the interval [22]. For robust optimization, this means that the scenarios should contain the worst-case scenarios. The resulting SRO problem is as follows,

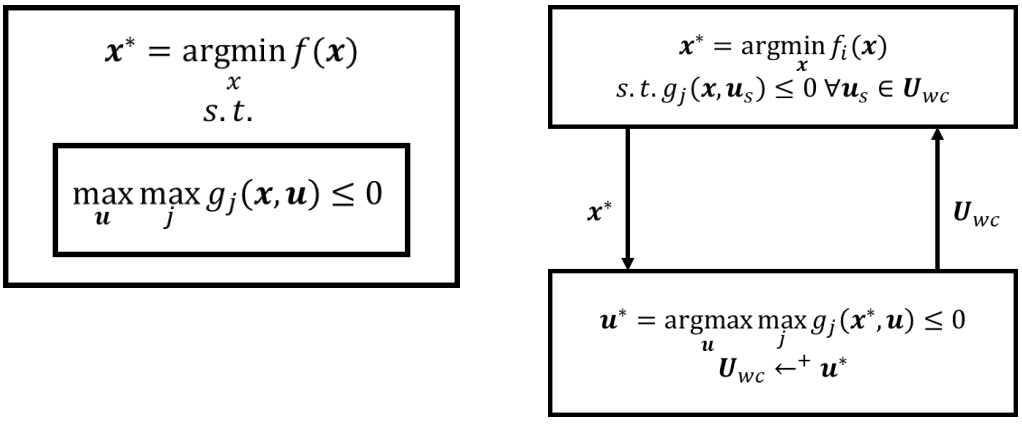
$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &s.t. \quad g_{j,s}(\boldsymbol{x}, \boldsymbol{u}_s) \leq 0, \quad j \in 1, \dots, J, \quad s \in 1, \dots, S \\ &\qquad \boldsymbol{x} \in [\boldsymbol{x}_l, \boldsymbol{x}_u] \\ &\qquad \forall \boldsymbol{u}_s \in \boldsymbol{U} \end{aligned} \tag{2.2}$$

where $\boldsymbol{u}_s$ is a discrete scenario in the finite scenario set $\boldsymbol{U}$. Each of the J constraints is reproduced for each scenario in the set $\boldsymbol{U}$ for a total of $S \times J$ constraints with $S = |\boldsymbol{U}|$. Most scenarios will not create active constraints that affect the solution at the robust optimum. An active constraint is one that if removed from the problem, the location of the optimum will change. An active constraint forms a constraint cut for the feasible domain and thus facilitates the search for worst-case scenarios. Adding inactive constraints to the SRO problem may increase the computational cost without adding a constraint cut to the feasible domain.

Others have approached the problem of finding worst-case scenarios, as discussed in the next section.

2.2 ***Previous Methods: An Overview***

To find a robust optimum solution, the task is to consider and add all necessary scenarios and their corresponding constraints to pare down the feasible domain while avoiding adding any extra scenarios and constraints. This is generally accomplished through random sampling [7] and worst-case search [11], [58]. Pure random sampling can require numerous scenarios to obtain a robust optimum, which can drive the computational cost up. Performing a worst-case search can either be done in a nested [20] or iterative manner [10], [21]. The nested approach, Figure 2.1(a), quickly becomes computationally intractable, but the iterative approach, Figure 2.1(b) can be done in a more efficient manner [10], [21]. Here and elsewhere in this dissertation, the notation "$\leftarrow^{+}$" is used to indicate appending the argument on the right to the argument on the left.



(a) (b)

Figure 2.1: Methods for worst-case search in robust optimization: (a) nested approach, (b) iterative approach

One version of the iterative approach (Algorithm 1) can be represented by the pseudocode shown in Table 2.1 which is detailed in the next section.

Table 2.1: Algorithm 2.1, iterative robust optimization

1: $\boldsymbol{U}_{wc} \leftarrow \boldsymbol{u}_0$
2: $Feasible \leftarrow False$
3: $\boldsymbol{WHILE}\ (Feasible = False)$
4: $\quad \boldsymbol{x}^* \leftarrow Solve\ Scenario\ Robust\ Optimization, \forall \boldsymbol{u} \in \boldsymbol{U}_{wc}$
5: $\quad \boldsymbol{u}^* \leftarrow Maximize\ Constraint\ Violations$
6: $\quad \boldsymbol{IF} \max_j \left(g_j(\boldsymbol{x}^*, \boldsymbol{u}^*) \right) > 0$
7: $\quad\quad \boldsymbol{U}_{wc} \leftarrow^+ \boldsymbol{u}^*$
8: $\quad \boldsymbol{ELSE}$
9: $\quad\quad Feasible \leftarrow True$
10: **RETURN** $\boldsymbol{x}^*$

2.2.1 Iterative Robust Optimization

Following Table 2.1, the steps in an iterative approach for robust optimization are presented in the following.

2.2.1.1 Initialization (Lines 1-3)

The iterative approach for robust optimization is initialized by including the nominal scenario $\boldsymbol{u}_0$ in the scenario set $\boldsymbol{U}_{wc}$. If nominal values are not inherent in the problem (e.g., average ambient temperature or zero manufacturing variability) $\boldsymbol{u}_0$ is taken as the average of the lower and upper bounds $\boldsymbol{u}_l$ and $\boldsymbol{u}_u$. The feasibility indicator is set to $False$. The main loop is begun and will close when the feasibility indicator is no longer $False$.

2.2.1.2 Scenario Robust Optimization (Line 4)

The SRO problem from Eq. (2.2) is solved for every scenario contained in $\boldsymbol{U}_{wc}$ and the robust optimal solution $\boldsymbol{x}^*$ is returned.

2.2.1.3 Constraint Violation Maximization (Line 5)

The constraints are then maximized with respect to the uncertain parameters,

holding solution $\boldsymbol{x}^*$ constant as in Eq. (2.3).

$$\begin{aligned} &\max_{\boldsymbol{u}} \max_{j} g_j(\boldsymbol{x}^*, \boldsymbol{u}) \\ &j \in \{1, \dots, J\} \\ &\boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u] \end{aligned} \tag{2.3}$$

2.2.1.4 Feasibility Testing (Lines 6-10)

If a scenario $\boldsymbol{u}^*$ can be found that makes any of the constraints infeasible at $\boldsymbol{x}^*$, then that scenario is added to $\boldsymbol{U}_{wc}$, adding constraint cuts to the SRO, and the process returns to Line 4. If, instead, the maximized constraints are all feasible, then the current solution is a robust optimum, so the feasibility indicator is set to $True$ and the robust optimizer $\boldsymbol{x}^*$ is returned.

2.3 Proposed Method: SFTC-RO

The SFTC-RO approach is detailed in Table 2.2 as a pseudocode. Each step of SFTC-RO, as shown in Table 2.2, is detailed in the following.

2.3.1 Initialization (Lines 1-5)

SFTC-RO begins with initializing the SRO problem of Eq. (2.2) by selecting a nominal scenario $\boldsymbol{u}_0$ (Line 1). With no prior knowledge, this is taken to be the mean parameter values over the interval $\boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u]$. Next, the constraint set $\boldsymbol{R}$ is initialized. $\boldsymbol{R}$ can be viewed as a matrix with rows corresponding to each scenario in $\boldsymbol{U}_{wc}$ and each column corresponding to one of the J constraints. For the initial scenario, $\boldsymbol{R}$ contains all J constraints. For numerical stability, all decision variables and uncertain parameters are scaled between 0 and 1. Before the main loop is opened, the feasibility indicator is set to $False$. The main loop will close when the indicator is

instead set to $True$.

2.3.2 Top-Level: SRO (Line 6)

The SRO problem in Eq. (2.2) is solved in this step via MATLAB's fmincon with the sequential quadratic programming solver. Here, every scenario $\boldsymbol{u}_s$ that is added to $\boldsymbol{U}_{wc}$ has a corresponding set of constraints which can be violated during the scenario search that returned that $\boldsymbol{u}_s$. Indices for these violated constraints were stored in row s of $\boldsymbol{R}$ in the previous step. During this step, only the violated constraints, $R_{s,j} = 1$, are evaluated while solving the SRO problem. For the first iteration, $\boldsymbol{u}_0$ is evaluated for all constraints, and for subsequent iterations it and the corresponding constraints are all removed. This can be done because $\boldsymbol{u}_0$ is chosen arbitrarily and thus is not expected to affect the robust optimal solution. If needed, $\boldsymbol{u}_0$ will be re-added to $\boldsymbol{U}$ in a subsequent iteration. This helps to reduce the size of the SRO problem.

Table 2.2: Algorithm 2.2, surrogate feasibility testing-cutting robust optimization (SFTC-RO)

1: $\boldsymbol{U}_{wc} \leftarrow \boldsymbol{u}_0$
2: $k \leftarrow 1$
3: $R_{k,j} \leftarrow g_j(\boldsymbol{x}, \boldsymbol{u}_0), \quad j = \{1, \dots, J\}$
4: $Feasible \leftarrow False$
5: $\boldsymbol{WHILE}\ (Feasible = False)$
6: $\quad \boldsymbol{x}^* \leftarrow Solve\ Scenario\ Robust\ Optimization, \forall \boldsymbol{u_s} \in \boldsymbol{U}_{wc},$
$s.t. \{ g_j(\boldsymbol{x}, \boldsymbol{u}_s) \leq 0 | R_{s,j} = 1 \}$
7: $\quad \boldsymbol{U}_t \leftarrow N_{uf}\ random\ scenarios$
8: $\quad \textbf{FOR}\ (n \in \{1, \dots, N_{if}\})$
9: $\quad\quad \hat{g}(\boldsymbol{u}) \leftarrow Construct\ Surrogate$
$with \left\{ \boldsymbol{U}_t, \max_j g_j(\boldsymbol{x}^*, \boldsymbol{U}_t) \right\}$
10: $\quad\quad \boldsymbol{u}^* \leftarrow argmax(\hat{g}(\boldsymbol{u}))$
11: $\quad\quad \boldsymbol{U}_t \leftarrow^+ \boldsymbol{u}^*$
12: $\quad \boldsymbol{U}_{wc} \leftarrow^+ \boldsymbol{u}^*$
13: $\quad \textbf{IF}\ (k{=}1)$
14: $\quad\quad Remove\ \boldsymbol{u}_0, R_{1,j}$
15: $\quad \textbf{FOR}\ (j \in \{1, \dots, J\})$
16: $\quad\quad \textbf{IF}\ (g_j(\boldsymbol{x}^*, \boldsymbol{u}^*) > 0)$
17: $\quad\quad\quad R_{k+1,j} \leftarrow^+ g_j(\boldsymbol{x}, \boldsymbol{u}^*)$
18: $\quad \textbf{IF}\ \left(all\left(g_j(\boldsymbol{x}^*, \boldsymbol{u}^*) \leq 0 \right) \right)$
19: $\quad\quad Feasible \leftarrow True$
20: $\quad \textbf{ELSE}$
21: $\quad\quad k \leftarrow k + 1$
22: $\textbf{RETURN}\ \boldsymbol{x}^*$

2.3.3 Bottom-Level: Surrogate Scenario Testing (Lines 7-11)

The optimal solution from the top-level problem $\boldsymbol{x}^*$ is then brought to the bottom-level for worst-case scenario searching. A surrogate is trained based on N_{uf} samples per constraint. The surrogate models used in this dissertation are all Gaussian Process (GP) models. Several empirical tests were performed in Appendix B suggesting that GPs were best suited to the problems of interest. The GPs are constructed using MATLAB's "fitrgp" function[59] with the automatic relevance

determination (ARD) squared exponential kernel[15]. All of the parameters for fitrgp were left as their default configurations. The surrogate is iteratively used to search for a new scenario at which point the constraints are evaluated. The surrogate is then retrained, including the new scenario, until the infill budget of N_{if} samples per constraint is depleted.

2.3.3.1 Perform Design of Experiments (Line 7)

The initial budget of function evaluations, N_{uf} points $\boldsymbol{u}$ are randomly sampled from a uniform distribution. These are stored in the training set $\boldsymbol{U}_t$.

2.3.3.2 Evaluate and Train (Lines 8-9)

A loop is opened to perform the following lines for the budgeted infill function evaluations, N_{if}. All J constraints are evaluated at any new scenarios $\boldsymbol{u} \in \boldsymbol{U}_t$. The greatest of the J constraints for each $\boldsymbol{u} \in \boldsymbol{U}_t$ is taken together with $\boldsymbol{U}_t$ and used to train a GP surrogate model $\hat{g}(\boldsymbol{u})$.

2.3.3.3 Perform Scenario Optimization (Line 10)

The surrogate function $\hat{g}(\boldsymbol{u})$ is then maximized with respect to the uncertain parameters $\boldsymbol{u}$.

2.3.3.4 Add Scenario and Return (Line 11)

The maximizing parameter values are added to the training set $\boldsymbol{U}_t$. The updated training set is then returned to Line 9 where the new scenario is evaluated. This continues until all N_{if} evaluations are exhausted.

2.3.4 Return Scenario and Constraint Cuts (Lines 12-22)

After the N_{if} evaluations have been used, the worst-case scenario $\boldsymbol{u}^*$ is added to the scenario set $\boldsymbol{U}_{wc}$ (Line 12). Of the J constraints $g_j(\boldsymbol{x}, \boldsymbol{u}^*)$ only those that are

violated at $g_j(\boldsymbol{x}^*, \boldsymbol{u}^*)$ are added to set $\boldsymbol{R}$ as constraint cuts in the top-level SRO problem (Lines 16-17). Following the first execution only, the initial scenario $\boldsymbol{u}_0$ is removed from $\boldsymbol{U}_{wc}$ along with the corresponding constraints (Lines 13-14). The initial scenario $\boldsymbol{u}_0$ and some of its corresponding constraints may be added back to $\boldsymbol{U}_{wc}$ in future iterations, but they are unlikely to become active at the robust optimal solution unless the nominal scenario was determined in advanced to be important for the robust optimal solution. If none of the constraints are violated when maximized with respect to $\boldsymbol{u}$ in the bottom-level problem, then $\boldsymbol{x}^*$ is returned as an optimal solution. Otherwise, the iteration counter is increased, and the process is returned to the top-level SRO problem in Line 6 (Lines 18-22).

2.4 Examples

To evaluate the performance of the proposed method, five (5) numerical and engineering examples are considered. These include the circle problem (Example 2.1), a modified version of the welded beam problem (Example 2.2) [10], [60], the speed reducer problem (Example 2.3) [10], [61], the design of an angle grinder (Example 2.4) [62], [63], and a variable scaling problem (Example 2.5) with M which is varied from 10 to 250 [10], [64]. The characteristics of the example problems are summarized in Table 2.3.

Table 2.3: Example problems characteristics

Example	No. of Variables	No. of Uncertain Parameters	No. of Constraints
1	2	2	1
2	4	8	6
3	7	7	13
4	9	9	13
5*	M	M	$M/2$

*M was varied from 10 to 250 in increments of 10

For each example, the performance of the proposed SFTC-RO method and a method from the recent literature [10], SGLRO, are compared in terms of the final objective function value, number of objective function evaluations, number of constraint function evaluations, and a verification of the feasibility of the solutions when robustness is considered. Solution results for the SGLRO method were previously published for all examples except Example 2.4: Design of an Angle Grinder. The published SGLRO program, see the link in [10], was used to implement this problem, following the provided examples. For consistency in all examples except the speed reducer, the initial point is taken to be the lower bound of the variables, $\boldsymbol{x}_0 = \boldsymbol{x}_l$. For the speed reducer, a different initial point, provided in Section 2.4.3, is used for consistency with the literature. The difference in the results for the two methods were tested for statistical significance by a two-sample z-test with a significance level of 0.05. The differences between both methods in this chapter were all found to be statistically significant.

It is to be noted that verification studies were performed for each example to examine how robust the solutions were. These studies were undertaken by evaluating all constraint functions at the robust optimal solution and with 10,000 random

scenarios unless otherwise noted. For each solution, the maximum constraint value over all random scenarios and all constraints is evaluated. Each problem was replicated multiple (100) times for both methods being tested to capture any effects of random sampling. The reported values are the mean and standard deviation of the maximum constraint values over all 100 replicates.

2.4.1 Example 2.1: 2D Demonstration Example

To illustrate the methodology step-by-step, the basic circle problem [10] is solved in detail here. The problem setup is as follows:

$$\begin{aligned} &\min_{x_1,x_2} f(x_1,x_2) = -x_1^2 - x_2^2 \\ &s.t.\ g(x_1,x_2,u_1,u_2) = (x_1-u_1)^2 + (x_2-u_2)^2 - 5 \le 0 \\ &\quad \forall u_i \in [-1,1] \\ &\quad -5 \le x_i \le 5 \end{aligned} \tag{2.4}$$

where the objective function is maximizing Euclidean distance-squared from the origin in the x_1, x_2 plane. The constraint is a bounding circle of radius $r = \sqrt{5}$ with uncertain center coordinates. The uncertainty is bounded within the interval $[-1,1]$ for both dimensions u_1 and u_2. This bounded region can be seen in Figure 2.2 as the light shaded square in the centre. The white surrounding region is the area which may be feasible under some scenario. The outer dark grey region is the domain that is infeasible. As scenarios and constraints are added, constraint cuts are made to the feasible domain, causing the infeasible (feasible) region to expand (contract). The computational budget for each iteration of the lower problem (second subproblem) is set at an initial budget of $N_{uf} = 10$ and an infill budget of $N_{if} = 10$.

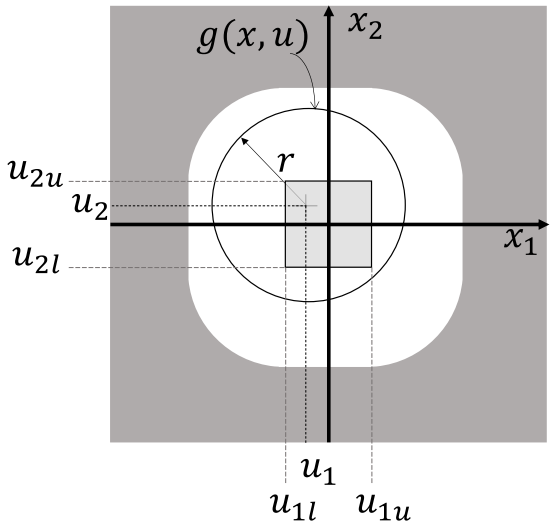


Figure 2.2: Diagram of Example 2.1

2.4.1.1 Initialization

First, an initial scenario $\boldsymbol{u}_0$ is assigned. Here, this nominal scenario is taken to be $\boldsymbol{u}_0 = (0, 0)$ as that is the middle of the uncertain range. Since there is only one constraint, it is added to the constraint set $\boldsymbol{R}$.

2.4.1.2 Iteration 1: Top Level

The SRO problem in Eq. (2.2) is solved for the single scenario $\boldsymbol{u}_0$ using sequential quadratic programming, as shown in Eq. (2.5):

$$\begin{aligned} &\min_{x_1,x_2} f(x_1,x_2) = -x_1^2 - x_2^2 \\ &s.t.\, g(x_1,x_2,u_1,u_2) = \left(x_1 - u_{0,1}\right)^2 + \left(x_2 - u_{0,2}\right)^2 - 5 \le 0 \\ &\quad U = u_0 = (0,0) \\ &\quad -5 \le x_i \le 5 \end{aligned} \tag{2.5}$$

The single constraint is the bounding circle, centered at the origin. Since all points on the boundary of this (circle) constraint are equidistant from the origin, there are infinite minima with a value of $f^* = -5$. Since, the starting point is the lower bound, $\boldsymbol{x}_0 = \boldsymbol{x}_l = (-5, -5)$ the solution is at $\boldsymbol{x}^* = (-1.58, -1.58)$.

2.4.1.3 Iteration 1: Bottom Level

N_{uf} random values of u are sampled from a uniform distribution and the constraint is evaluated at all points. A GP model $\hat{g}(\boldsymbol{u})$ is then fit to the sample points. This function is maximized returning a candidate $\boldsymbol{u}^*$ value. The constraint is evaluated at this new value of $\boldsymbol{u}$. The model $\hat{g}(\boldsymbol{u})$ is trained and maximized again until the N_{if} function evaluations are exhausted. Here, the total budget was 10 function calls. A contour plot of $\hat{g}(\boldsymbol{u})$ over the range $[\boldsymbol{u}_l, \boldsymbol{u}_u]$ is shown in Figure 2.3(a). It can be seen that the greatest constraint value occurs in the top-right of the range. The final $\boldsymbol{u}^*$ value is $\boldsymbol{u}^* = (1,1)$ with a constraint violation of 8.325.

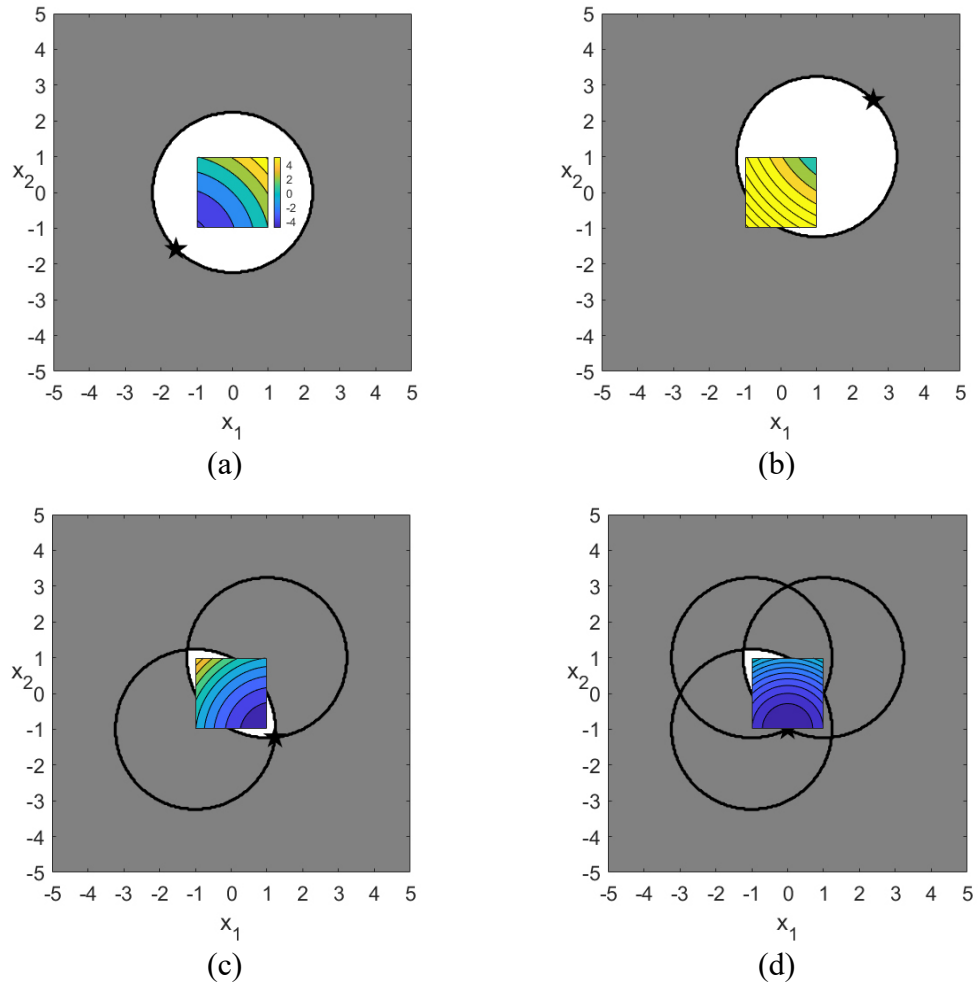


Figure 2.3: Plots of top-level constraint cuts and surrogate predicted location for next cut for (a) iteration 1, (b) iteration 2, (c) iteration 3, and (d) iteration 4

2.4.1.4 Iteration 2: Top Level

Before the scenario robust optimization problem is solved again, the initial scenario and corresponding constraints are removed, leaving only $\boldsymbol{u} = (1,1)$. The new optimum is $f^* = -13.325$ at $\boldsymbol{x}^* = (2.581,2.581)$, Figure 2.3(b). As scenarios are added to the set $\boldsymbol{U}_{wc}$, it should be expected that the optimum will increase with each iteration. However, because of the removal of $\boldsymbol{u}_0$, the optimum may decrease at this step, as can be seen in Figure 2.4.

2.4.1.5 Iteration 2: Bottom Level

The bottom level proceeds as described above, resulting in $\boldsymbol{u} = (-1, -1)$ and a constraint violation of 20.65. Note that this is greater than the constraint violation in the previous iteration.

2.4.1.6 Iteration 3: Top Level

Starting this iteration, no scenarios are removed. Since there was a constraint violation in the bottom level of the previous iteration, the violated constraint is added to $\boldsymbol{R}$. These two constraints can be seen in Figure 2.3(c) as the two black circles, with the white intersection in the middle being the feasible domain. Upon solving, the solution is $f^* = -3$ at $\boldsymbol{x}^* = (1.225, -1.225)$.

2.4.1.7 Iteration 3: Bottom Level

After generating a new set of N_{uf} points, and proceeding as above, the maximum constraint violation is found to be 4.89 at $\boldsymbol{u} = (-1,1)$.

2.4.1.8 Iteration 4: Top Level

With the set of scenarios $\boldsymbol{U}_{wc} = \{(1,1), (-1, -1), (-1,1)\}$, the optimal solution is $f^* = -1$ at $\boldsymbol{x}^* = (0, -1)$.

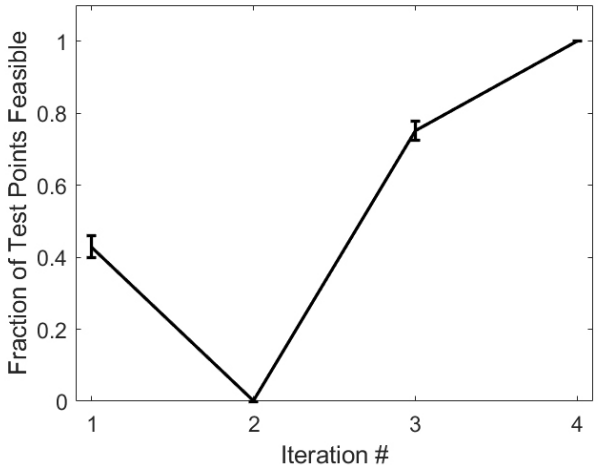


Figure 2.4: Progress of robust solution for Example 2.1 as a fraction of 1000 test points found to be feasible

2.4.1.9 Iteration 4: Bottom Level

Again, the contour plot in Figure 2.3(d) shows the predicted constraint values. These occur in the top left and right corners. However, the maximum constraint value found in this iteration is 0, meaning the constraints are satisfied for any scenario, and the solution is robust. Thus, $\boldsymbol{x}^* = (0, -1)$ is returned as the optimal solution. As mentioned by [10], to fully determine the feasible region of this problem, 4 scenarios are needed. However, the local solution found here is not affected by the fourth scenario.

To look at the progression of the feasibility of the intermediate solutions, a verification study was performed. The solutions produced in the top level of the four iterations were taken and the constraint was evaluated for this solution at 1000 randomly generated scenarios. The progression is shown in Figure 2.4. Between iterations 1 and 2, there is a noticeable drop in the percent of scenarios that were feasible. This is the result of removing $\boldsymbol{u}_0$ following the first iteration. However,

Figure 2.3(d) shows that the constraint from $\boldsymbol{u}_0$ is inactive and Figure 2.4 indicates that this solution was feasible for all of the verification points. Removing this constraint, therefore, did not negatively affect the final solution, but it did reduce the number of constraints evaluated in each top-level SRO problem after the first iteration.

The numerical results from 100 independent runs of Example 2.1 using the proposed SFTC-RO and a method from the literature are presented in Table 2.4. The results are reported by the mean value with the standard deviation in parentheses. Although the function calls for the objective function were greater for SFTC-RO than for SGLRO, the function calls for the constraints was less. Since the function calls of the constraints are an order of magnitude larger, there is an overall improvement in the function calls for SFTC-RO over SGLRO. In the fourth column, the robust feasibility results are reported as the greatest constraint violation. Each of the optimal solutions from the 100 runs were compared against 10,000 randomly generated scenarios. The greatest constraint violation over all scenarios was averaged over the 100 optimal solutions and reported along with the standard deviation. Here, SGLRO appears to be slightly more conservative, finding a robust feasible solution in 98% of runs compared to SFTC-RO with 95% robust feasible solution. Lastly, in the fifth column, the final objective function value from the optimization is provided. Both methods appear to have essentially the same performance with little variation.

Table 2.4: Results from Example 2.1

Approach	Total Obj. Function Calls Mean (std. dev.)	Total Constraint Function Calls Mean (std. dev.)	Largest Constraint Violation Mean (std. dev.)	Final Obj. Value Mean (std. dev.)
SFTC-RO	$68.63(50.66)$	$122.8(109.3)$	$-0.072(0.043)$	$-0.76(0.14)$
SGLRO [10]	$56.9(22.2)$	$314.6(115.8)$	$-0.051(0.026)$	$-1(-1.65 \times 10^{-15})$

2.4.2 Example 2.2: Welded Beam

The second example is the well-known problem of designing a welded beam originally formulated by [60] and modified to include uncertainty [10]. The uncertain form of the problem has 4 design variables, 8 uncertain parameters, and 6 constraints. The formulation is provided in Eq. (A.2) in Appendix A, section A.2. The results from Example 2.2, provided in Table 2.5 show the proposed SFTC-RO approach again having a greater number of objective function evaluations, but a lower number of constraint evaluations for an overall smaller number of function evaluations. Unlike Example 2.1, here the proposed SFTC-RO method appears more conservative in terms of constraint violation with 94% robust feasible solutions compared to SGLRO's 9% robust feasible solutions.

Table 2.5: Results from Example 2.2

Approach	Total Obj. Function Calls Mean (std. dev.)	Total Constraint Function Calls Mean (std. dev.)	Largest Constraint Violation Mean (std. dev.)	Final Obj. Value Mean (std. dev.)
SFTC-RO	$298(60)$	$2{,}313(598)$	$-0.0045(0.0029)$	$2.782(0.008)$
SGLRO [10]	$278.2(19.4)$	$5{,}349.4(452)$	$0.015(0.011)$	$2.786(1.97 \times 10^{-8})$

2.4.3 Example 2.3: Enhanced Speed Reducer

The third example is the design of a speed reducer, originally formulated by [65], had uncertainty introduced by [62], and was further modified by [10]. This engineering example has 7 design variables each considered to have uncertainty and

13 constraints. The design variables with uncertainty are all treated as the sum of a deterministic design variable and an uncertain parameter. For consistency with the literature, instead of starting at the lower bound of the variable range, optimization started with the initial condition $\boldsymbol{x}_0 = [3.58, 0.71, 18, 8, 8, 3.5, 5.3]$. The formulation used here can be found in Eq. (A.3) in Appendix A, section Example 2.3: Enhanced Speed Reducer. The results using SFTC-RO as well as SGLRO are shown in Table 2.6. For this example, the two methods performed similarly, but SFTC-RO had slightly fewer function calls. However, SFTC-RO produced some infeasible solutions as seen in column 4. In this example SFTC-RO found a robust feasible solution in 12% of the runs compared to SGLRO which found a robust feasible solution in 100% of runs

Table 2.6: Results from Example 2.3

Approach	Total Obj. Function Calls Mean (std. dev.)	Total Constraint Function Calls Mean (std. dev.)	Largest Constraint Violation Mean (std. dev.)	Final Obj. Value Mean (std. dev.)
SFTC-RO	954.5(135.5)	10,472.3(1,937.3)	0.013(0.011)	1.88(0.019)
SGLRO [10]	731.1(86)	11,310(1,564)	-7.55×10^{-5} (2.64×10^{-5})	1.886 (3.94×10^{-6})

2.4.4 Example 2.4: Design of an Angle Grinder

This angle grinder design problem was formulated by [63]. A multiobjective collaborative formulation with uncertainty was developed by [58]. A single objective feasibility robust optimization formulation was formed by [62] by taking the original objective from [63] and the constraints with uncertainty from [58]. The formulation from [62] is used here and is provided in detail in Eq. (A.4) in Appendix A, section A.4. There are 9 variables pertaining to the mechanical and electrical properties of the angle grinder design and 9 uncertain parameters, one for each of the design variables.

There are 13 constraints. The results are provided in Table 2.7. In this example, the number of function calls from SFTC-RO are substantially lower than those from SGLRO. The verification study for the angle grinder design problem was performed with 1 million random scenarios instead of 10,000 due to the larger number of uncertain parameters. As with the other examples, the constraint violation is larger in the proposed approach which found robust feasible solutions in 21% of the runs compared to SGLRO which found a robust feasible solution in 42% of the runs. It should also be reiterated that the results for SGLRO were not previously published but were generated using the published program from [10].

Table 2.7: Results from Example 4

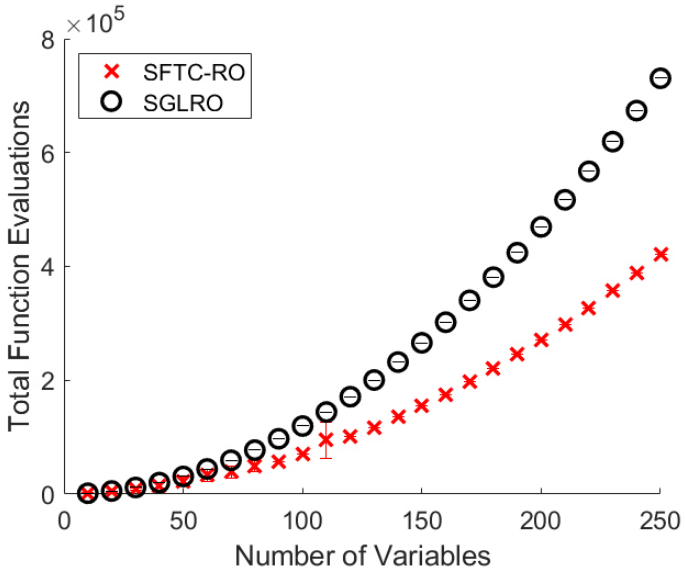
Approach	Total Obj. Function Calls Mean (std. dev.)	Total Constraint Function Calls Mean (std. dev.)	Largest Constraint Violation Mean (std. dev.)	Final Obj. Value Mean (std. dev.)
SFTC-RO	3,297(914)	45,447 (17,914)	0.031 (0.038)	15.08 (0.94)
SGLRO [10]	22,258.7 (18,251)	6,078,968 (6,869,654)	2.23×10^{-8} (1.13×10^{-7})	16.2 (0.22)

2.4.5 Example 2.5: DTLZ9 – Variable Scaling

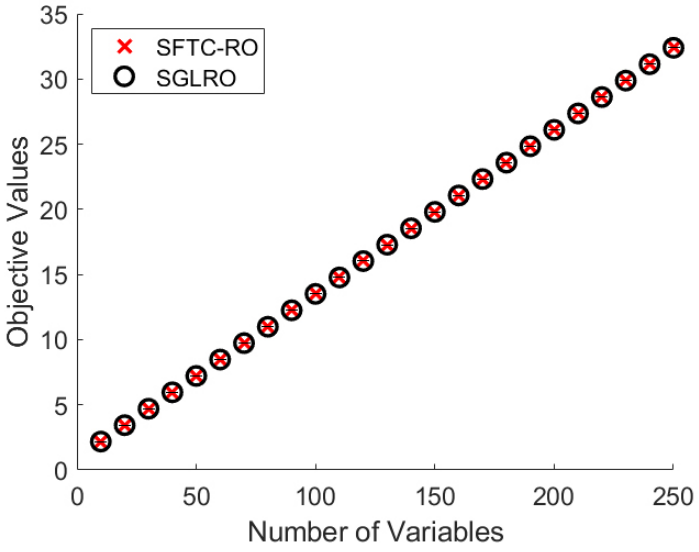
The fifth example is a numerical example that can be scaled to different sizes. This example was solved for different sizes: from 10 decision variables up to 250 decision variables in increments of 10. There are as many uncertain parameters as decision variables and half as many constraints as decision variables (recall Table 2.3). The formulation is provided in Eq. (A.17) in Appendix A, section A.11. This test problem was originally developed by [64] for multi-objective optimization and was modified for single objective robust optimization by [10].

In Figure 2.5, there is a comparison of the proposed SFTC-RO and a method

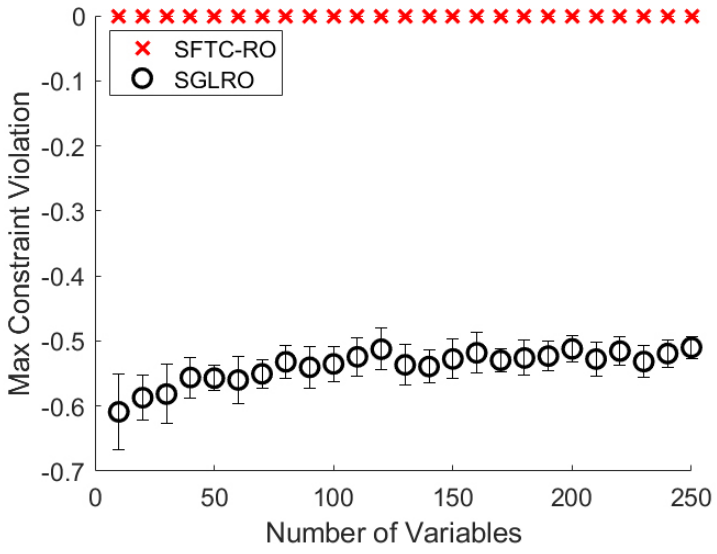
from the literature: Scenario Generation with Local Robust Optimization (SGLRO) [10]. The results in Figure 2.5 for SGLRO were generated using tools published by [10]. For both methods, the results plotted were the average values of 10 independent runs plotted with error bars representing the standard deviation. In Figure 2.5(a), the total function evaluations are shown, and it can be seen that as the problem size scales up, SFTC-RO performs much better than SGLRO. The standard deviations are small and can only be easily seen for the case with 110 variables. The objective function values are shown in Figure 2.5(b). Both methods had the same objective function values indicating that there was not loss of optimality caused by the introduction of the surrogate. Lastly, the maximum constraint values are plotted in Figure 2.5(c). Although SFTC-RO has greater constraint values than SGLRO, they are all below zero. So, the introduction of the surrogate did not impact the robustness of the solution. Even when the problem size is increased to 1000 variables and 500 uncertain parameters, there is no observable decrease in the performance in SFTC-RO. This suggests that computer memory would be the limiting factor for maximum problem size.


×10^5
8
6
4
2
0
SFTC-RO
SGLRO
Total Function Evaluations
0
50
100
150
200
250
Number of Variables

(a)


35
30
25
20
15
10
5
0
SFTC-RO
SGLRO
Objective Values
0
50
100
150
200
250
Number of Variables

(b)



©

Figure 2.5: Variable sized problem DTLZ9 comparison against existing method for (a) total function evaluations, (b) objective function values, and (c) maximum constraint values

2.4.6 Discussion of Results

The results of the verification study for the first four examples discussed above are shown in Figure 2.6. The worst-case constraint values over 10,000 random scenarios were found and averaged for both methods for all problems except for the design of an angle grinder. For the angle grinder design which used 1 million random scenarios. The average worst-case constraint values are shown with 95% confidence intervals. It can be seen that in all cases SGLRO had a much smaller confidence interval. Again, this is to be expected since SFTC-RO is an approximation method. Roughly speaking, the problems increase in size from left to right and it appears that constraint violations get somewhat worse for SFTC-RO with increasing problem size. Taking note of the scales for all four plots, both methods experience this increasing

trend in worst-case constraint values although it is larger for SFTC-RO. For the third and fourth examples, SFTC-RO has some infeasibility which is less than 3% as the constraints are normalized.

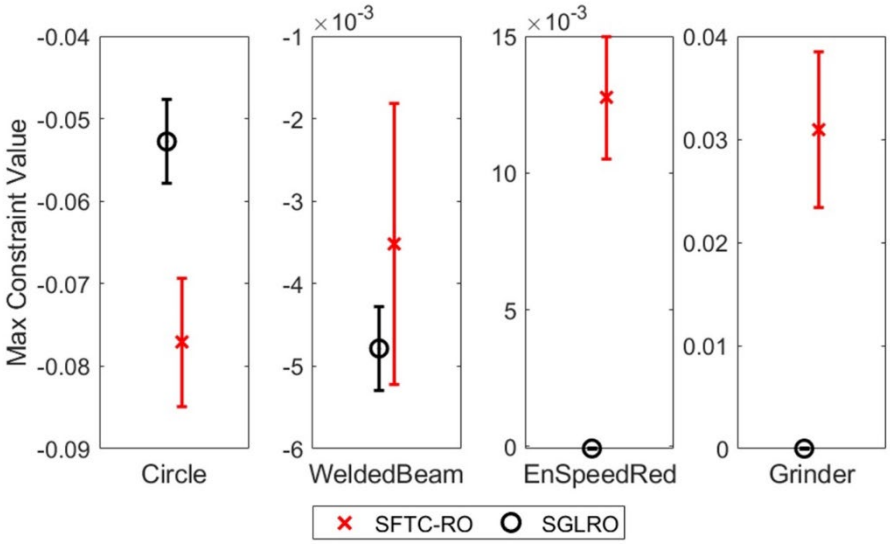


Figure 2.6: Feasibility 95% confidence intervals for Examples 2.1-2.4

2.5 Conclusions

In this chapter, a novel approach for finding robustly optimum feasible solutions by way of surrogate modelling is presented. While the use of surrogate modelling of both the decision variables at the top-level subproblem and uncertain parameters at the bottom-level subproblem have been reported before, it has not been shown advantageous by isolating and using it for solving the bottom-level (or 'feasibility') subproblem. Therefore, in this chapter, previous work was expanded upon by applying a surrogate optimization technique to the bottom-level subproblem

of a bi-level iterative robust optimization method. The contributions demonstrated in this chapter are as follows: First, compared to an existing method, a computational saving was shown in the first 4 of the example problems with little or no loss of objective function value or constraint feasibility. Second, the fifth example problem was performed at problem sizes ranging from 10 to 250 variables (and uncertain parameters) with 5 to 125 constraints. At all problem sizes, the proposed SFTC-RO method had a less computational cost (or function calls) compared with an existing method from the literature. As the number of variables, uncertain parameters, and constraint functions increased, the improvement of SFTC-RO over other method from the literature increased. The last contribution is the confidence-based verification study comparing this method and an existing method from the literature. This verification study showed that for the larger engineering examples, SFTC-RO solutions were, on average, less robust than the solutions from SGLRO.

In the next chapter, a method using surrogate modelling and Bayesian optimization for robust optimization problems under interval uncertainty with two objectives is presented. The developed method seeks to address RQ2.

Chapter 3: Bi-Objective Robust Optimization with Surrogates[2]

This chapter proposes a new method for using the expected improvement Bayesian optimization for constrained problems with two objectives under interval uncertainty.

The remainder of the chapter is laid out as follows. In Section 3.1, the multi-objective robust optimization problem is formulated followed by an introduction of bi-objective Bayesian optimization. In Section 3.2, the methodology for the proposed approach is provided. In Section 3.3, the example problems are detailed, and the results compared for the proposed approach and AA-MORO and MORO-KS from the literature. Lastly, in Section 3.4, some conclusions about the chapter are stated.

3.1 Problem Statement

In this section, first, a formulation for the multi-objective feasibility robust optimization problem is provided in Subsection 3.1.1. This is followed by Subsection 3.1.2 wherein a description of the surrogate formulation used in this chapter is presented.

3.1.1 Problem Formulation

A formulation for multi-objective feasibility robust optimization is provided in Eq. (3.1).

[2] A version of this chapter is to appear as an article in the Journal of Mechanical Design (ASME Transactions) [66]

$$\begin{aligned} &\min_{\boldsymbol{x}} f_i(\boldsymbol{x}) && i \in 1,2 \\ &s.t.\ \ g_j(\boldsymbol{x},\boldsymbol{u}) \le 0 && j \in 1,\dots,J \\ &\quad \forall \boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u] \\ &\quad \boldsymbol{x} \in [\boldsymbol{x}_l, \boldsymbol{x}_u] \end{aligned} \tag{3.1}$$

In Eq. (3.1), the vector of uncontrollable (uncertain) parameters $\boldsymbol{u}$ has been added, bounded below by $\boldsymbol{u}_l$ and above by $\boldsymbol{u}_u$. The formulation in Eq. (3.1) thus gives a bounded uncertainty. To solve a problem with bounded uncertainty exactly would require semi-infinite programming which can be computationally intensive [67]. A common method of circumventing this cost is to replace Eq. (3.1) with Eq. (3.2).

$$\begin{aligned} &\min_{\boldsymbol{x}} f_i(\boldsymbol{x}) && i \in 1,2 \\ &s.t.\ \ g_j(\boldsymbol{x},\boldsymbol{u}) \le 0 && j \in 1,\dots,J \\ &\quad \forall \boldsymbol{u} \in \boldsymbol{U}_{wc} \\ &\quad \boldsymbol{x} \in [\boldsymbol{x}_l, \boldsymbol{x}_u] \end{aligned} \tag{3.2}$$

In Eq. (3.2), instead of considering a continuous range of values for $\boldsymbol{u}$, only a finite set of values or scenarios $\boldsymbol{U}_{wc}$ needs to be considered. The scenarios are commonly found iteratively by doing a worst-case scenario search as shown in Eq. (3.3).

$$\begin{aligned} &\max_{\boldsymbol{u}} \max_{j} g_j(\boldsymbol{x}^*,\boldsymbol{u}) \\ &\boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u] \end{aligned} \tag{3.3}$$

The optimal design output by Eq. (3.2), $\boldsymbol{x}^*$, is input to Eq. (3.3) where the constraints are maximized for obtaining a worst-case (or maximizing) scenario. The worst-case scenario is then added to $\boldsymbol{U}_{wc}$. By iterating between Eq. (3.2) and Eq. (3.3)

in this way, until the maximized constraints are feasible, a robust feasible solution can be found [10], [20], [21], [38], [40].

The method for selecting which scenarios to include in $\boldsymbol{U}_{wc}$ will be discussed in more detail below. Although less computationally expensive than the continuous uncertainty range in Eq. (3.1), Eq. (3.2) could still require substantial numbers of function calls. Thus, a surrogate formulation is introduced next.

3.1.2 Surrogate Formulation

The surrogate formulation used in this chapter is one of the more widely employed, Gaussian Processes (GPs) modelling. Many works describe the derivation behind the GPs modelling, and, in the interest of saving space, the reader is directed to [28], and [31] in particular for further details. GPs assume that the function to be modelled is a realization of a Gaussian distributed stochastic process composed of a mean function and a random process with a mean of zero [68]. A surrogate built with GPs makes predictions based on the expectation of the stochastic process, given the known samples [69]. Practically, this prediction takes the form of Eq. (3.4).

$$\hat{f}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + K(\boldsymbol{x}, \boldsymbol{x}') \tag{3.4}$$

In Eq. (4), $\hat{f}$ is the approximating (or predicted) function or surrogate, μ is the mean function, K is the covariance function, $\boldsymbol{x}$ is a vector representing a new design point being evaluated, and $\boldsymbol{x}'$ is a previously evaluated design point. The mean function μ is generally a constant, linear, or quadratic function which describes the overall trend of the approximated function f. A number of different covariance functions is used with GPs, but the one used here is the Kriging kernel, also called the

"ARD squared exponential" kernel [15]. It has the form of Eq. (3.5).

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \, exp\left(-\frac{1}{2}\sum_{m=1}^{d} \frac{(x_m - x_m')^2}{\sigma_m^2}\right) \tag{3.5}$$

Here, σ_f and σ_m are hyperparameters to be fit based on the data. One of the greatest strengths of these models is that the posterior distribution after training can be used to predict the original function as well as an estimate of the confidence of that prediction [69]. As mentioned earlier, the expected improvement criterion is a widely used example of Bayesian optimization, balancing the exploration and exploitation of the unknown functions. The single objective formulation is given in Eq. (3.6)

$$E[Imp(\boldsymbol{x})] = \left(f_{min} - \hat{f}(\boldsymbol{x})\right)\Phi(\boldsymbol{x}) + \hat{s}\phi(\boldsymbol{x}) \tag{3.6}$$

The expected improvement function has a "ratcheting" effect on the objective function. Consideration for sampling is only given to points that will potentially improve the objective function. The general approach for solving feasibility robust optimization problems under interval uncertainty is to iteratively optimize with all known worst-case scenarios for feasibility and find new worst-case scenarios through maximizing of the constraints as is done in [10], [21], [38]. This can cause the current optimal solution to increase with respect to the objective function with each iteration. Clearly there is an incompatibility between this iterative robust optimization approach and the expected improvement criterion that will need to be resolved.

In the following paragraphs, the formulation of bi-objective expected improvement adapted from [12] is discussed. Instead of integrating in one dimension, the first moment of the joint probability distribution needs to be integrated over the

entire region in the objective function space that dominates the non-dominated frontier. This can be extended beyond two-objective functions by integrating over the dominating volume or hypervolume. These analytical integrals become increasingly cumbersome and so can be replaced with numerical methods such as Monte Carlo integration for higher numbers of objective functions [53]. The present chapter focuses on the analytical two-objective case.

As a background illustration: consider Figure 3.1 where the black dots represent the current non-dominated solutions in a bi-objective minimization problem. The light grey region, including the white rectangle, shows where a new solution could be added that would dominate one or more of the current solutions or extend the range of the non-dominated frontier. The dark grey rectangles between the dots illustrate the regions where a new point could be added that would be non-dominated. Note that the region to the left of all of the non-dominated points and the region below all of the non-dominated points are shown in light grey (along with the white rectangle). In the formulation considered here, these are included in the expected improvement search because they would extend the non-dominated frontier. The probability that a new randomly drawn point would dominate point B but not point A would be: $P[f_1^{new}(\boldsymbol{x}) \leq f_1^B \cap f_2^{new}(\boldsymbol{x}) \leq f_2^B \cap f_2^{new} > f_2^A]$. All possible f^{new} values that would satisfy this are visualized in Figure 3.1 as the white rectangle. Note, that it extends off to $-\infty$ in the f_2 direction. For notational compactness, allow the substitution of Eq. (3.7) for a current solution normalized by the predicted value and standard deviation of a new point, where i indexes the objective functions and k indexes one of the points on the non-dominated frontier.

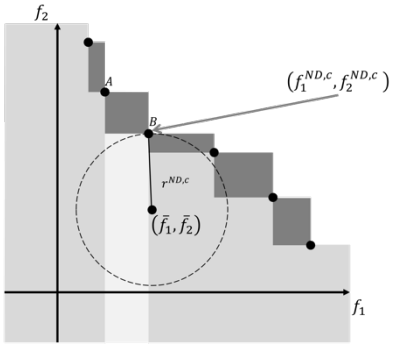


Figure 3.1: Illustration of a candidate robust non-dominated frontier, with color coded dominated/dominating regions

$$Y_{i,k}(\boldsymbol{x}) = \frac{f_i^k - \hat{f}_i(\boldsymbol{x})}{\hat{s}_i(\boldsymbol{x})} \tag{3.7}$$

Now, integrating the probability over this rectangular domain established by points A and B can be expressed as in Eq. (3.8).

$$P[f_1^{new}(\boldsymbol{x}) \leq f_1^B \cap f_2^{new}(\boldsymbol{x}) \leq f_2^B \cap f_1^{new} > f_1^A] = \left[\Phi\left(Y_{1,B}\right) - \Phi\left(Y_{1,A}\right)\right]\Phi\left(Y_{2,B}\right) \tag{3.8}$$

Summing these probabilities for every non-dominated solution and adding terms for the regions to the left of the first point and to the right of the last gives the total probability that a new sample will dominate a current point or extend the non-dominated frontier. The expression for this total probability over the n points on the non-dominated frontier is given in Eq. (3.9).

$$\begin{aligned} &P[Imp(\boldsymbol{x})] = \\ &P[f_1^{new}(\boldsymbol{x}) \leq f_1^{ND} \cap f_2^{new}(\boldsymbol{x}) \leq f_2^{ND}] \; = \Phi\left(Y_{1,1}\right) + \\ &\sum_{k=1}^{n-1}\left[\Phi\left(Y_{1,k+1}\right) - \Phi\left(Y_{1,k}\right)\right]\Phi\left(Y_{2,k+1}\right) + \left[1 - \Phi\left(Y_{1,n}\right)\right]\Phi\left(Y_{2,n}\right) \end{aligned} \tag{3.9}$$

The expected improvement is similarly defined but must be calculated for each objective function i separately first and with respect to a particular non-

dominated solution k, as in Eq. (3.10).

$$E\left[Imp_{i,k}(\boldsymbol{x})\right] = \hat{f}_i(\boldsymbol{x})\Phi\left(Y_{i,k}\right) - \hat{s}_i(\boldsymbol{x})\phi\left(Y_{i,k}\right) \tag{3.10}$$

For non-infinitely wide regions, this is multiplied by the cumulative distribution function. Summed together, these give the total expectation of improving the non-dominated frontier for objective function i, $E[Imp_i(\boldsymbol{x})]$, as in Eq. (3.11).

$$\begin{aligned} E[Imp_1(\boldsymbol{x})] = \\ E\left[Imp_{1,1}(\boldsymbol{x})\right] + \sum_{k=1}^{n-1}\left[E\left[Imp_{1,k+1}(\boldsymbol{x})\right] - E\left[Imp_{1,k}(\boldsymbol{x})\right]\right]\Phi\left(Y_{2,k+1}\right) \\ + E\left[Imp_{1,n}(\boldsymbol{x})\right]\Phi\left(Y_{2,n}\right) \end{aligned} \tag{3.11}$$

The centroid of the distributions $\bar{f}_i$ for each objective i can then be obtained as $\bar{f}_i = \frac{E[Imp_i(\boldsymbol{x})]}{P[Imp(\boldsymbol{x})]}$. And finally, the total expected improvement over both objective functions $E[Imp(\boldsymbol{x})]$ can be found from Eq. (3.12), with $f_i^{\mathrm{ND},c}$ being the closest of the non-dominated solutions (this would be point B in Figure 3.1).

$$E[Imp(\boldsymbol{x})] = P[Imp(\boldsymbol{x})]\sqrt{\sum_i \left(\bar{f}_i - f_i^{ND,c}\right)^2} \tag{3.12}$$

Constraints can be handled in a similar way, taking the product of the probability that each constraint is satisfied and multiplying the expected improvement by it. However, this is known to give conservative results [70]. Another approach is to relax all of the constraints in proportion to their uncertainty as was done in [44]. An expression for this is given in Eq. (3.13).

$$\hat{g}(\boldsymbol{x},\boldsymbol{u}) - a\hat{s}(\boldsymbol{x},\boldsymbol{u}) \leq 0 \tag{3.13}$$

In Eq.(3.13), $\hat{g}$ is the approximated constraint function (mean function), $\hat{s}$ is the expected error in the model of the constraint, and a is the proportionality of the relaxation. Taken together, Eq. (3.12) and Eq. (3.13) yield the principal problem of interest in this chapter, termed "Scenario Robust Expected Non-Dominated Set Improvement Problem" which is provided in Eq. (3.14).

$$\begin{aligned} &\max_{\boldsymbol{x}} E[Imp(\boldsymbol{x})] + br^{ND,c} \\ &s.t.\ \ \hat{g}_j(\boldsymbol{x},\boldsymbol{u}) - a\hat{s}_j(\boldsymbol{x},\boldsymbol{u}) \leq 0 \\ &\qquad \forall \boldsymbol{u} \in \boldsymbol{U}_{wc}, j = 1, \dots, J \end{aligned} \tag{3.14}$$

An additional term has been added to Eq. (3.14) to lightly penalize clustering of solutions. Here, $r^{\mathrm{ND},c}$ is the Euclidean distance of the centroid to the closest point, as shown in Figure 3.1, and b is the weight. In this work, a weight of $b = 0.1$ was sufficient. As with other Bayesian optimization approaches, this is an iterative method. In every iteration Eq. (3.14) will be solved, producing a new candidate design which will be evaluated using the actual functions. Because the surrogates used in this work are functions of both $\boldsymbol{x}$ and $\boldsymbol{u}$, a corresponding $\boldsymbol{u}$ value needs to be found before the models can be retrained. This is done using Eq. (3.15).

$$\begin{aligned} &\max_{\boldsymbol{u}} \max_{j} \hat{g}_j(\boldsymbol{x}^*, \boldsymbol{u}) \\ &\boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u] \end{aligned} \tag{3.15}$$

The scenario chosen for evaluation is that which produces the greatest constraint value. The next task at hand is how to select scenarios to add to $\boldsymbol{U}_{wc}$. For each of the predicted non-dominated robust solutions, Eq. (3.15) is applied. The scenario that produces the greatest constraint violation for any of these candidate solutions is added to the set $\boldsymbol{U}_{wc}$.

As scenarios are added to $\boldsymbol{U}_{wc}$ and the size of the feasible region shrinks, there may at times be no predicted robust feasible samples. In that event, it is useful to find one before proceeding. To accomplish this, another subproblem is defined, the "Robust Feasible Sample Search" which is shown in Eq. (3.16), which maximizes the probability that a new sample will be feasible under all current scenarios.

$$\begin{aligned} &\max_{\boldsymbol{x}} P\left[\max_{j} \hat{g}_j(\boldsymbol{x}, \boldsymbol{u}) \leq 0\right] \\ &\forall \boldsymbol{u} \in \boldsymbol{U}_{wc} \end{aligned} \tag{3.16}$$

Switching objective functions and adding new scenarios to $\boldsymbol{U}_{wc}$ will cause the bi-objective robust solutions at times to move backwards. The ratcheting effect of the expected improvement functions, however, does not allow this. Another objective of this chapter is then to relax this aspect of the expected improvement acquisition function to enable iterative robust optimization.

3.2 Methodology

In this section, first, the relaxation method for the candidate solution relaxation is given in Subsection 3.2.1, followed by a modified expected improvement in Subsection 3.2.2, and then the details of the proposed approach in Subsection3.2.3.

3.2.1 Candidate Solution Relaxation

As stated in the previous section, the "ratcheting" effect of the expected improvement and non-dominated set expected improvement criteria are at odds with the iterative improving and worsening of the iterative robust optimization scheme.

Without adjustment, therefore, the expected improvement criteria will fail to allow the robust optimization solver to find more conservative solutions with improved robustness. It is straightforward enough to remove the infeasible samples (a design and a scenario at which the objective and constraint functions have been evaluated). There is more ambiguity, however, for feasible samples. An infeasible sample will never be robust, but a feasible sample may or may not be. When a new scenario is added to the set $\boldsymbol{U}_{wc}$ and the surrogates are updated, feasible samples can be revisited even when they have been labeled as infeasible by a surrogate in one iteration. Then, in the next iteration, with further refinement to the surrogates, it is possible for the sample to be predicted as feasible again. Therefore, as can be seen in Figure 3.4, feasible samples that are predicted as infeasible are removed for that iteration, but not entirely discarded. On subsequent iterations they are re-evaluated. This allows for them to be removed to relax the candidate robust non-dominated frontier but avoids being overly conservative.

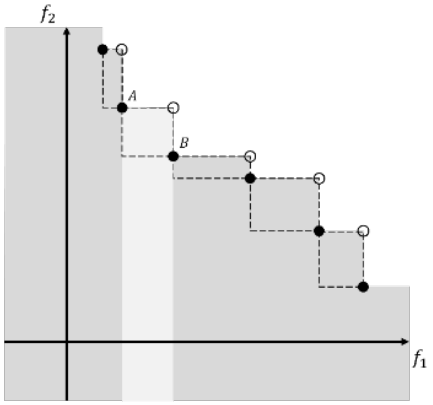


Figure 3.2: Modifications to expected improvement integration

3.2.2 Modified Expected Improvement

The bi-objective expected improvement formulation of Eq. (3.12) is designed

to find new solutions that will dominate existing solutions. It does not seek to identify more solutions in the non-dominated areas between existing points on the non-dominated frontier [12]. To improve the solution density of the non-dominated frontiers, a modification was made to the expected improvement. The set of points to improve upon $\boldsymbol{f}^{\mathrm{ND}}$ was augmented for the calculations with pseudo-points at the peaks of the non-dominated regions between the points. In Figure 3.2, the real non-dominated solutions are shown as solid black circles, and the pseudo-points are shown as empty circles.

The regions that contain possible solutions under this formulation that were excluded in the original are indicated in Figure 3.2 by the dashed lines. This should promote a more densely populated solution set.

3.2.3 Proposed Method

The proposed method is provided as a pseudo-code in Table 3.1. A flowchart is also presented in Figure 3.4 which outlines the method with focus on the sets of samples and how they are used. The rest of this section is a detailed description of the steps of the method.

Table 3.1: Algorithm 3.1

1:	$[\boldsymbol{X}, \boldsymbol{U}] \leftarrow DOE$
2:	$\boldsymbol{F} \leftarrow f_i(\boldsymbol{X})$
3:	$\boldsymbol{G} \leftarrow g_j(\boldsymbol{X}, \boldsymbol{U})$
4:	$\hat{f}_i, \hat{g}_j \leftarrow train\ surrogates(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{F}, \boldsymbol{G})$
5:	$\boldsymbol{U}_{wc} \leftarrow \boldsymbol{u}_0$
6:	$[\boldsymbol{F}_{nd}, \boldsymbol{G}_{nd}] \leftarrow [\boldsymbol{F}, \boldsymbol{G}]$
7:	$Remove\ infeasible\ samples\ that\ dominate\ a\ feasible\ sample\ from\ \boldsymbol{F}_{nd}$
8:	$[\boldsymbol{F}_{nda}, \boldsymbol{G}_{nda}] \leftarrow [\boldsymbol{F}_{nd}, \boldsymbol{G}_{nd}]$
9:	$\boldsymbol{F}_p \leftarrow \boldsymbol{F}_{nda}$
10:	$\boldsymbol{FOR}(n \in 1, \dots, N_c)$
11:	$Remove\ dominated\ samples\ from\ \boldsymbol{F}_p$
12:	$\boldsymbol{x}_n^* \leftarrow solve\ scenario\ robust\ expected\ non-dominated\ set\ improvement\ problem\ for\ \hat{f}_i, \hat{g}_j, \boldsymbol{F}_p, \boldsymbol{U}_{wc}$
13:	$\boldsymbol{u}_n^* \leftarrow solve\ predicted\ constraint\ maximization$
14:	$[\boldsymbol{F}, \boldsymbol{G}] \leftarrow^+ evaluate\ f_i(\boldsymbol{x}_n^*)\ and\ g_j(\boldsymbol{x}_n^*, \boldsymbol{u}_n^*)$
15:	$\boldsymbol{IF}(evaluated\ samples\ feasible)$
16:	$[\boldsymbol{F}_{nd}, \boldsymbol{G}_{nd}] \leftarrow^+ [f_i(x_n^*), g_j(\boldsymbol{x}_n^*, \boldsymbol{u}_n^*)]$
17:	$\boldsymbol{F}_p \leftarrow^+ [f_i(\boldsymbol{x}_n^*), g_j(\boldsymbol{x}_n^*, \boldsymbol{u}_n^*)]$
18:	$[\hat{f}_i, \hat{g}_j] \leftarrow retrain\ surrogates(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{F}, \boldsymbol{G})$
19:	$\boldsymbol{G}_{wc} \leftarrow \emptyset$
20:	$[\boldsymbol{F}_{nda}, \boldsymbol{G}_{nda}] \leftarrow [\boldsymbol{F}_{nd}, \boldsymbol{G}_{nd}]$
21:	$\boldsymbol{WHILE}(\boldsymbol{F}_{nda}\ not\ empty\ or\ feasible\ not-dominated\ frontier\ is\ found)$
22:	$\boldsymbol{F}_p \leftarrow nondominated\ samples\ of\ \boldsymbol{F}_{nda}$
23:	$\boldsymbol{FOR}(k \in 1, \dots, \lvert\boldsymbol{F}_p\rvert)$
24:	$(u_k^*, g_k^*) \leftarrow worst\ case\ prediction$
25:	$\boldsymbol{IF}(g_k^*\ predicted\ infeasible)$
26:	$Remove\ \boldsymbol{F}_{p,k}\ from\ \boldsymbol{F}_{nda}$
27:	$\boldsymbol{G}_{wc} \leftarrow^+ g_k^*$
28:	$\boldsymbol{IF}(\boldsymbol{G}_{wc}\ empty)$
29:	$\boldsymbol{RETURN}\ \boldsymbol{F}_p$
30:	$\boldsymbol{ELSE}$
31:	$\boldsymbol{U}_{wc} \leftarrow^+ \boldsymbol{u}\ corresponding\ to\ max\ of\ \boldsymbol{G}_{wc}$
32:	$\boldsymbol{GO\ TO}\ 8$

3.2.3.1 Initialization (Lines 1-8)

- To begin with, several storage structures need to be initialized. First, the list "$\boldsymbol{XU}$" (for $[\boldsymbol{X}, \boldsymbol{U}]$) needs to be populated with a design of experiments (DOE). List $\boldsymbol{XU}$ is the complete list of design variables $\boldsymbol{x}$ and uncertain parameters $\boldsymbol{u}$ that will have their objectives and constraints evaluated. The DOE used here is a Latin hypercube sampling [Line 1].
- Each entry in $\boldsymbol{XU}$ is then evaluated for the objectives and stored in $\boldsymbol{F}$ [Line 2] and the constraints' values are stored in $\boldsymbol{G}$ [Line 3].
- These three lists are then used to train the surrogate models $\hat{f}_i(\boldsymbol{x})$ and $\hat{g}_j(\boldsymbol{x}, \boldsymbol{u})$ [Line 4].
- Next, the worst-case scenario set $\boldsymbol{U}_{wc}$ is initialized with a nominal value $\boldsymbol{u}_0$, which for the examples is taken to be the mean between the upper and lower bounds on $\boldsymbol{u}$ [Line 5].
- The values stored in $\boldsymbol{F}$ and $\boldsymbol{G}$ are duplicated into $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$, respectively [Line 6].

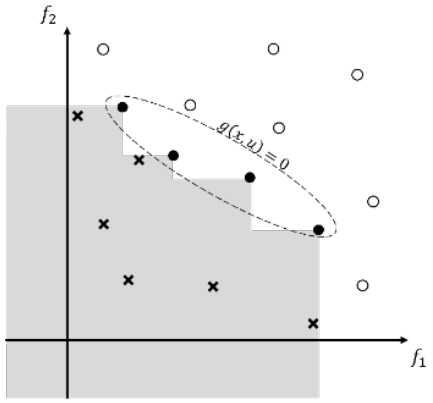


Figure 3.3: Removal of Dominating infeasible points

- Any evaluated samples contained in $\boldsymbol{G}_{nd}$ that are infeasible and their corresponding values in $\boldsymbol{F}_{nd}$ dominate at least one feasible sample are

removed from both lists. This is repeated until the feasible solutions are no longer dominated by infeasible points. This is illustrated in Figure 3.3 where the solid circles are feasible, the x's are infeasible points that will be removed, and the empty circles are infeasible solutions that will not be removed [Line 7].

- Lastly, the lists $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$ are copied into $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$ [Line 8].

The initialization is also shown in the top portion of Figure 3.4. Note that the left half of Figure 3.4 shows the operations performed on the actual functions, and the right half shows what is done with the surrogates.

3.2.3.2 Sample Acquisition (Lines 9-18)

- If there are feasible points contained in $\boldsymbol{G}_{nda}$, the following process will be repeated N_c times where N_c is a parameter specified by the user. This allows multiple candidate points to be found during each iteration before evaluating the original functions in batches [Line 9].
- The non-dominated values of $\boldsymbol{F}_{nda}$ are stored in $\boldsymbol{F}_p$ [Line 11].
- A new candidate design $\boldsymbol{x}_n^*$ is then found by using a genetic algorithm to solve the scenario robust expected non-dominated set improvement problem from Eq. (3.14), improving over the values in $\boldsymbol{F}_P$ and trying to satisfy all constraints created by scenarios contained in $\boldsymbol{U}_{wc}$ [Line 12]. This step is shown in the "optimize" portion of Figure 3.4.
- The predicted constraint values for the candidate design are then maximized using a genetic algorithm to find the corresponding worst-case scenario $\boldsymbol{u}_n^*$ [Line 13] ("Robustness check" in Figure 3.4).

- The pair can then be added to $\boldsymbol{XU}$ (not shown) and the objective and constraint functions can be evaluated, and the results added to $\boldsymbol{F}$ and $\boldsymbol{G}$ [Line 14].
- If the constraint values are feasible, then the sample is also added to $\boldsymbol{F}_{nd}$, $\boldsymbol{G}_{nd}$, and $\boldsymbol{F}_{p}$ [Lines 15-17].

These steps are "Evaluate and retrain" in Figure 3.4, and it is performed in the left column because this operation is done on the actual functions. If there are no feasible samples, then Line 12 is replaced with maximizing the probability of feasibility as shown in Eq. (3.16) and setting N_c to 1 for the current iteration. The change in N_c is due to only needing a single feasible point to resume the search for non-dominated robust solutions.

3.2.3.3 Initialize for Non-Dominated Set Relaxation (Lines 19-20)

- The three lists $\boldsymbol{XU}$, $\boldsymbol{F}$, and $\boldsymbol{G}$ which were augmented in the previous steps are used to retrain the surrogate models [Line 18].
- The list $\boldsymbol{G}_{wc}$ is emptied of any values from previous iterations [Line 19].
- Next, $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$ are set to take on the values of $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$ (the feasible evaluated samples) [Line 20].

3.2.3.4 Constraint Cuts and Non-Dominated Set Relaxation (Lines 21-31)

- To prepare for the next round of optimization, the constraint cuts need to be added to the feasible domain. The increased constraints will make it unlikely that the acquisition function will find new feasible solutions. To facilitate the finding of new solutions, the non-dominated frontier needs to be relaxed. This

process continues while $\boldsymbol{F}_{nda}$ is not empty or a feasible non-dominated frontier is found [Line 21].

- The non-dominated samples in $\boldsymbol{F}_{nda}$ are stored in $\boldsymbol{F}_{p}$ [Line 22].
- Each element in $\boldsymbol{F}_{p}$ has its corresponding constraints maximized with respect to the uncertain parameters [Lines 23-24].
- If the worst-case constraint value is predicted to be infeasible, $\boldsymbol{F}_{p,k}$ is removed from $\boldsymbol{F}_{nda}$ and $\boldsymbol{F}_{p}$ relaxing the non-dominated frontier [Lines 25-26].
- The maximum constraint value is then added to $\boldsymbol{G}_{wc}$ [Line 27].
- If any points have been removed from $\boldsymbol{F}_{nda}$ and $\boldsymbol{F}_{nda}$ is not empty, then return to Line 21. After the while loop has terminated, if $\boldsymbol{G}_{wc}$ is empty, then no scenarios have been discovered that are predicted to make any of the current non-dominated points infeasible. Therefore, the current non-dominated frontier can be returned as robust [Lines 28-29].
- Otherwise, the worst-case constraint violation contained in $\boldsymbol{G}_{wc}$ is found, and the corresponding scenario is added to $\boldsymbol{U}_{wc}$, creating a constraint cut [Lines 30-31].

In Figure 3.4, it can be seen that after the initialization, $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$ are only used in the left side of the figure and elements are never removed. $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$ on the other hand are only on the right side of the figure and only have points removed. $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$ are also reset each iteration to the current values of $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$. The significance is that points can only be candidates for robust optima if they are feasible when sampled, but they can be excluded as candidates based on the current surrogate

approximation of them. As the surrogate is retrained each iteration, the estimate of points' feasibility is reset so long as the initial sampling was feasible. The surrogate is always trained on the full set of evaluated samples. The removal of points predicted to be infeasible only affects the possible candidate solutions and the reference points that the expected improvement is calculated against.

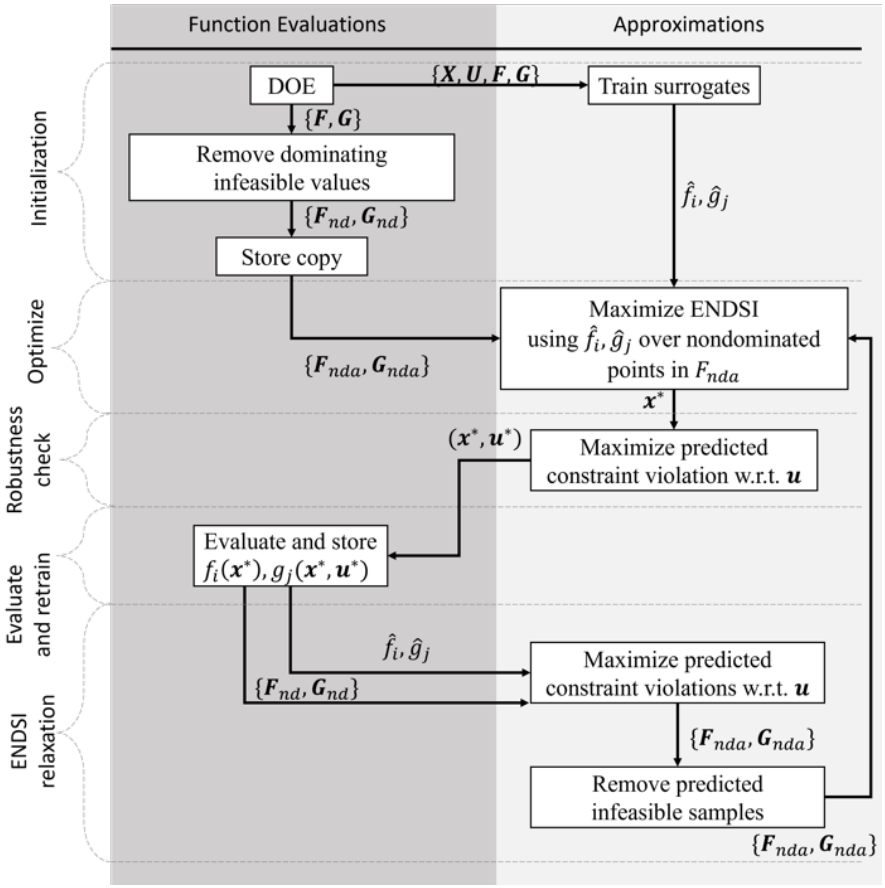


Figure 3.4: Flowchart of proposed method

In earlier experiments, it was seen that rejecting points that were predicted infeasible for all future iterations generally removed too many feasible points in early iterations. This is due to the models being less refined in early stages. The models improve in accuracy with each iteration, so re-evaluating only increases the accuracy

of the feasibility predictions.

3.3 Examples

Five examples are provided in this section. The first three are well known numerical examples: TNK [21], CTP [21], and a third that is unnamed but widely used [38], the optimization of a three-bar truss as an engineering example [38], and a problem which is scalable, or can be varied in size, ZDT3 [21]. The first three numerical examples were chosen because results found using AA-MORO (a surrogate robust optimization solver against which the proposed method is compared) [21] or its predecessor, MORO, [71] have been published. In all examples the proposed approach was solved using MATLAB 2021b's fmincon solver with 10 random starts and default settings [59]. AA-MORO was solved using MATLAB 2021b's multi-objective genetic algorithm solver with all default settings except a population size of 200 and a maximum number of generations of 500 [59]. The surrogate models used in the proposed approach were all GPs implemented using MATLAB 2021b's fitrgp function [59] with the ARD squared exponential kernel function [15]. The function parameters were set at the default values except for the parameters corresponding to signal noise, "Sigma" and "SigmaLowerBound" which were set to 0.1 and 0.0001 respectively. These settings were used in this method and that of the following chapter.

The results of the first four example problems are presented after their descriptions in Figure 3.7 and Figure 3.8. Each example was solved using the proposed approach and AA-MORO [21]. The two solvers were each performed for 20

independent runs and the results of these runs are summarized by bounding attainment surfaces (staircase or zigzagged lines, as shown in Figure 3.7 for the best and worst points. In the legend of each figure, next to the name of the method is the mean (standard deviation) of the number of function evaluations. Function evaluations are counted per sample, meaning an evaluation of both objectives and all J constraints is counted as a single function evaluation. Here, and throughout this dissertation, the number of function evaluations reported includes initial evaluations for training as well as all online evaluations. The number of function evaluations used for each problem is a function of the number of variables and uncertain parameters. Numbers of allotted function evaluations were determined empirically based on the minimum input requirements of the surrogate models and the amount of time needed to perform the test. Additional quantitative results are shown in Table 3.2, including the published results for Examples 3.3 and 3.4 from another method, MORO-KS [38]. The fifth example is similarly presented in Figure 3.9, and feasibility verification results for all examples are presented in Table 3.3.

3.3.1 Demonstrative Example – TNK

The TNK example is used to demonstrate the proposed approach. This problem was first formulated in [72], and a robust adaptation presented in [21]. The specifics of the problem are given below followed by the first several steps of the proposed approach. Finally, the results for the problem are presented. The TNK problem formulation is given in [73]. There are two objective functions, two constraint functions, two design variables, and two uncertain parameters.

3.3.1.1 Demonstration of Proposed Approach

In this section, the steps of the proposed approach will be described in detail for the first several iterations of solving TNK.

Initialization

The problem is initialized with the set of worst-case scenarios $\boldsymbol{U}_{wc}$ only containing the nominal scenario $\boldsymbol{u}_0 = (0,0)$. A Latin hypercube sampling is performed with 30 samples for the four input arguments (x_1, x_2, u_1, u_2) and is stored in the set $\boldsymbol{XU}$. These are all evaluated at the two objective functions and the two constraint functions and stored in $\boldsymbol{F}$ and $\boldsymbol{G}$. These

samples are plotted in Figure 3.5(a) as black dots. The feasible samples are shown in the figure as squares. Four surrogates are trained $\hat{f}_1, \hat{f}_2, \hat{g}_1, \hat{g}_2$. In the figure, a black line (crescent shaped region) indicates the feasible region as approximated by $\hat{g}_1$ and $\hat{g}_2$. Notice, that one of the samples, located at $(0.08,1.02)$, lies in this region but was not labeled as feasible. This is because the feasible region shown is only calculated based on the constraint values evaluated at the scenarios in $\boldsymbol{U}_{wc}$. On the other hand, a sample is feasible if, when it is evaluated at some $\boldsymbol{x}$ and $\boldsymbol{u}$ the constraints are satisfied.

Next, the sets $\boldsymbol{F}$ and $\boldsymbol{G}$ are duplicated into $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$. In non-robust approaches, all infeasible samples would be removed from these two sets, however, they will be needed in future steps. Instead, only the infeasible samples which dominate the feasible samples are removed. These are labeled as x's in the plot.

For illustrative purposes, the negative of the expected improvement function

was also plotted in the shaded area. It is clear that the greatest improvement is expected on the left tip of the feasible region.

Iteration 1 – Expected Improvement

In Figure 3.5(b), the non-dominated solutions (both feasible and infeasible) are shown as stars. The scenario robust expected non-dominated set improvement problem (Eq. (3.2)) is solved with these solutions as the frontier to be improved. Following Eq. (3.3), the constraints are then maximized at the new candidate design point to find what scenario is expected to be the worst-case for that sample. The scenario that is found, and the design point are taken together as a new sample, added to $\boldsymbol{XU}$, and evaluated with the actual functions, storing the outputs in $\boldsymbol{F}$ and $\boldsymbol{G}$. This is repeated nine times, adding any non-dominated samples to the non-dominated frontier, to produce ten total candidate designs. All ten are shown in Figure 3.5(b) as diamonds. For the most part, they are evenly spread across the constraint boundary with some clustering on the right and a single candidate solution in the interior of the feasible region.

Iteration 1 – Feasibility Check

The next step is to predict the robust feasibility of the feasible samples. In Figure 3.5(c), the newly added samples that were infeasible have been labeled with a downward arrow, and the feasible samples have been labeled with an upward arrow. The approximated feasible region has also been updated. The samples near the constraint boundary are infeasible indicating that the robust feasible region will exclude those. All of the samples that have been feasible need to be checked with the

updated surrogate models. Any sample that was infeasible does not need to be checked because they cannot be robust solutions. So, at each of the feasible samples, the constraint surrogates are maximized to see if a scenario can be found which will make the

design at that sample infeasible. In Figure 3.5(c), samples that have been made infeasible in this way are labeled with a black x, and those predicted feasible have been labeled with a black square.

At this point, $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$ are copied into $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$. All of the feasible samples that have been predicted to be infeasible are

removed from $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$. This process needs to be done each iteration because the updated surrogate models may find that a sample predicted to be infeasible one iteration will be feasible the next. The samples that were made infeasible have their maximum constraint violations compared. The sample with the greatest constraint violation has the corresponding maximizing u added to the set $\boldsymbol{U}_{wc}$.

Iteration 2 – Expected Improvement

Again, the scenario robust expected non-dominated set improvement problem, Eq. (3.2), is solved. The set of scenarios now contains an additional scenario which reduces the feasible region. Removing the predicted infeasible points from $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$ has relaxed the non-dominated frontier to allow for new solutions in the shrinking feasible region.

Iteration 2 – Feasibility Check

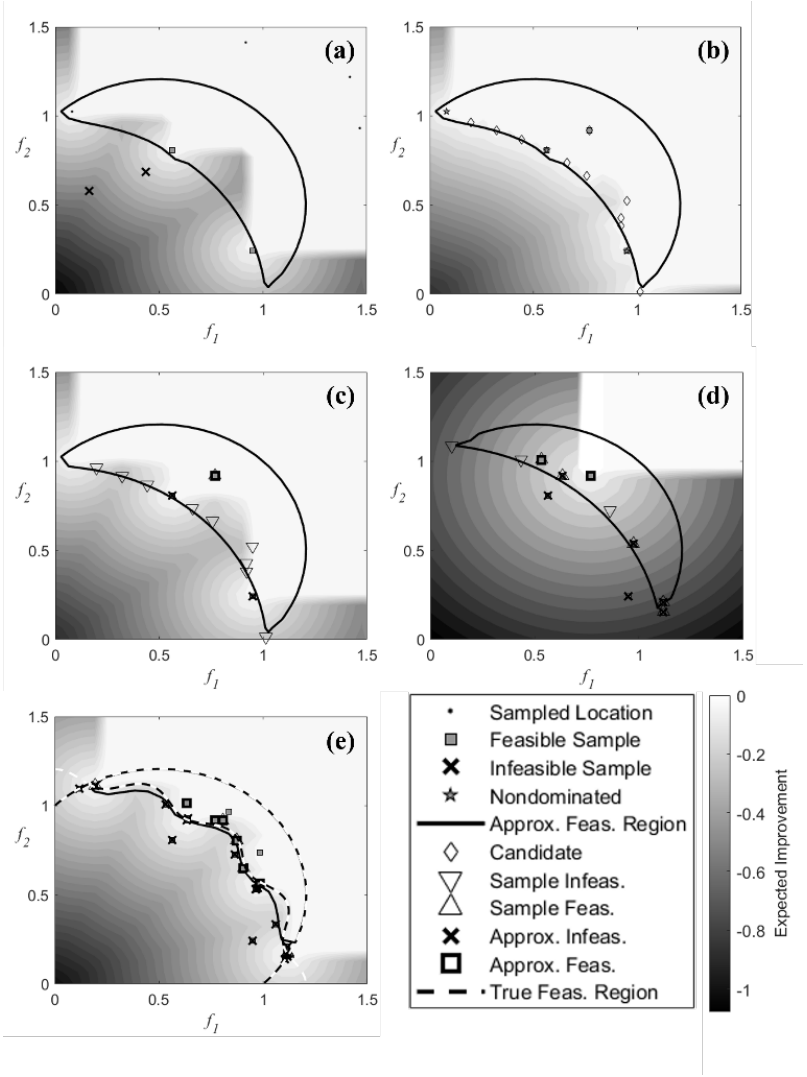


Figure 3.5: Demonstrations of proposed approach on Example 2.1 for a) initialization, b) Iteration 1 – Improvement c) Iteration 1 – feasibility d) Iteration 2 – feasibility and e) iteration 5 – feasibility

The evaluated candidates can be seen in Figure 3.5. Samples were found at the two extreme corners of the feasible domain, but more were also filled in along the boundary. At least five samples are visible that were sampled as feasible, but four of them are predicted to become infeasible with the updated model. So, when $\boldsymbol{F}_{nd}$ and $\boldsymbol{G}_{nd}$ are copied into $\boldsymbol{F}_{nda}$ and $\boldsymbol{G}_{nda}$ this time, only points that are predicted to be

feasible, the two squares, will remain.

Iteration 5 – Feasibility Check

Skipping ahead, the status at the fifth iteration is shown in Figure 3.5©. In addition, the true robust feasible region has been overlayed on the plot in a dashed line. The top constraint boundary (without uncertainty) has been captured perfectly. The bottom constraint has not yet been fully captured, but the waviness is apparent in the approximation, and it is approaching the true constraint. There are five robust feasible solutions after only five iterations. The robust solutions are somewhat clustered toward the center of the non-dominated frontier, but it can be seen that there is sampling occurring broadly that should yield more robust solutions as the robust feasible region's approximation converges.

3.3.2 Numerical Example 2.2 – CTP

The second numerical example was originally posed without uncertainty in [74] and then adapted for robust optimization in [21]. There are 10 design variables, two uncertain parameters, two objective functions, and one constraint. The formulation is presented in [73].

3.3.3 Numerical Example 3.3

The third numerical example has been presented in [71] and [38]. This problem has two objectives, two constraints, two variables, and one uncertain parameter. Additionally, there are restrictions on the variability of the objective functions under uncertainty. The last two restrictions have been formulated as constraint functions. The formulation of this problem is presented in [73]. This is the

simplest of the example problems.

3.3.4 Engineering Example 3.4: Three-bar Truss

As a more practical example, the proposed method is applied to the three-bar truss problem from [37]. It was first formulated by [75], uncertainty was added to the problem by [61], before being adapted for multiple objectives in [37]. A diagram of the problem configuration is given in Figure 3.6.

A diagram of the three-bar truss problem is provided in Figure 3.6. It consists of three bars pinned to the same horizontal surface $l = 1\ m$ apart and meeting at a single joint distance $l = 1\ m$ below it. A constant load $P = 100\ kN$ is applied at the joint. Under the fixed load, the objectives are to minimize the horizontal displacement u of the loading point and the total volume of members 1 and 3. The design variables are the angle θ at which the load is applied and the cross-sectional area A_1 of members 1 and 3 which is the same for both. There is uncertainty added to both design variables. Constraints in place are that the internal stress σ in the members should not exceed the yield strength σ_{yield}, and the vertical displacement v should not exceed $0.5\ cm$. Also, the uncertainty added to the design variables should not cause the first objective to change by more than $0.07\ cm$ and the second objective by $400\ cm^3$. The exact formulation for the problem is provided in [73].

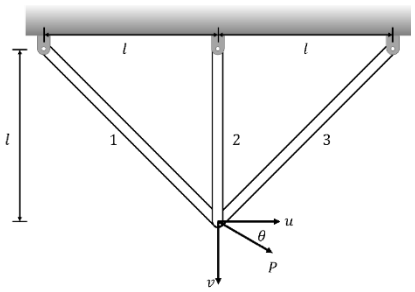


Figure 3.6: Three-bar truss example

3.3.5 Results: Examples 3.1 – 3.4

The results for the first four examples are provided in this section. Qualitative analysis is performed using summary plots shown in Figure 3.7, depicting bounds on the best and worst points found over twenty independent runs. In the bounding plots, external corners indicate a solution. These are shown for both AA-MORO and the proposed approach. Afterwards, a verification study was done, testing each design against 10,000 random scenarios. Designs found infeasible for any scenario are deemed not robust. The results after removing non-robust points are shown in Figure 3.8. Quantitative comparisons on the robust solutions are made in Table 3.2, which contains the function evaluations and three quality metrics the Hypervolume Difference (HD) [76], Overall Spread (OS) [76], and Spacing (SP) [77], as well as the total number of non-dominated solutions found. For HD, smaller values indicate better convergence, for OS, larger values indicate wider range, and for SP, smaller values indicate more uniform spacing. AA-MORO found more than twice as many non-dominated solutions than the proposed approach for all examples except for the first example which was closer to 1.6 times as many solutions as seen in column 7 of Table 3.2. Differences in the results for the proposed approach and AA-MORO were

tested for statistical significance at a significance level of 0.05 using a two-sample z-test. Differences for all problems and all metrics were found to be statistically significant except for the HD metric for examples 1, 3, and 4. This indicates that for those three examples, the values of HD for the proposed approach and AA-MORO could effectively have the same mean value. MORO-KS was not included in these calculations because the standard deviations were not provided in the published results.

3.3.5.1 Example 3.1

The best and worst results for Example 3.1 are summarized in Figure 3.7(a). The robust feasible results are similarly shown in Figure 3.8(a). A number of the solutions from AA-MORO are removed indicating that they were not robust. The most likely reason for this is an assumption made in the AA-MORO algorithm. If the predicted error of a possible solution is below a threshold, then that point is not sampled. A more conservative threshold could prevent so many points from being non-robust. The proposed approach has a wider band, or difference in performance between its best and worst solutions whereas AA-MORO is more consistent. The larger step-like shapes visible for the proposed method's solution indicate that fewer optimal solutions were found, and they were somewhat further apart from each other. However, the SP metric from Table 3.2 indicates more uniform spacing for the proposed approach over AA-MORO. After removing the non-robust points, the proposed approach has better HD and OS values as well.

3.3.5.2 Example 3.2

The second example's plot is shown in Figure 3.7(b) and Figure

3.8(b). Both surrogate methods have substantial separation between their best and worst runs, although AA-MORO has a wider range. Before removing the non-robust solutions, the best solutions from AA-MORO outperform the best of the proposed method. In Figure 3.8(b), however, the best solutions of the proposed approach seem better than those of AA-MORO at almost all locations. It should be noted that both methods had some non-robust points that were removed. Referring to Table 3.2, the proposed approach has better HD and OS values, but AA-MORO has the better SP value.

3.3.5.3 Example 3.3

Results from the third example are shown in Figure 3.7(c) and Figure 3.8(c). Both sets of solutions are seemingly unaffected by the removal of the infeasible points. For this example, AA-MORO was able to find the robust feasible solution nearly each of the twenty runs. The proposed approach had wider variation in its solutions. Of the examples provided, this is the worst performance of the proposed approach compared to AA-MORO, except in terms of OS. As stated before, AA-MORO has a larger emphasis on exploitation of the surrogate models than exploration. For a simple case like this one, that exploitation is rewarded since there is relatively little need to explore. Comparing the results in Table 3.2, MORO-KS used over twenty times more function calls but had worse HD and OS than the proposed approach. The number of non-dominated solutions found was not reported for MORO-KS in [38].

3.3.5.4 Example 3.4

The results for the three-bar truss example are shown in Figure 3.7(d) and

Figure 3.8(d). Removal of non-robust points had little impact, except at the far-right tail. AA-MORO has a consistent band throughout the non-dominated frontier, but the proposed approach extends further to the right. Both surrogate methods appear to have found the robust optimal solution at least once of the twenty runs. In Table 3.2, the quality metrics for the proposed approach, AA-MORO, and the published results for MORO-KS are shown. MORO-KS used 17 times more function calls than the other two resulting in a better value for HD. For OS, both the proposed approach and AA-MORO performed better than MORO-KS. For SP, the proposed approach is again the weakest. This consistent shortcoming of SP is due to the proposed approach finding new sample points individually while the other two search in parallel. Both AA-MORO and MORO-KS return unevaluated points that are expected to be feasible, allowing them to create denser solution sets.

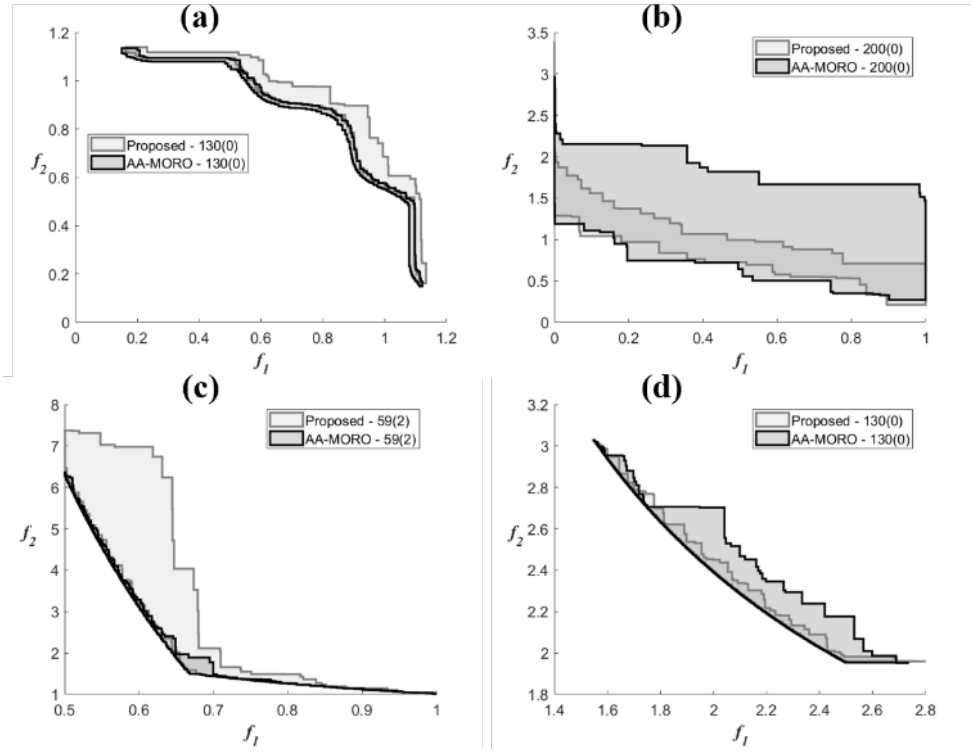


Figure 3.7: Summary results from a) TNK b) CTP c) numerical example 3.3, and d) Three-Bar Truss

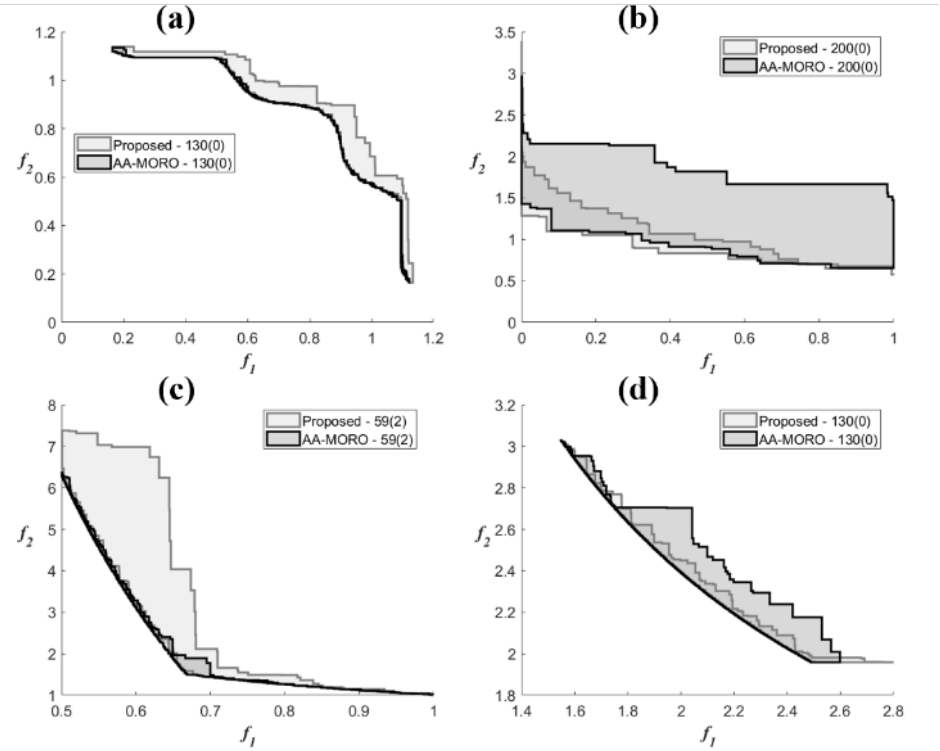


Figure 3.8: Summary results with infeasible points removed from a) TNK b) CTP c) numerical example 3.3, and d) Three-Bar Truss

Table 3.2: Quality metrics of non-dominated solutions: **mean**(standard deviation)

Ex	Approach	Function Calls	HD	OS	SP	Number
1	Proposed	130	**0.618** (0.044)	**0.402** (0.118)	**0.043** (0.016)	**15.6** (3.35)
	AA-MORO [21]	130	**0.634** (0.224)	**0.176** (0.176)	**0.122** (0.197)	**59.1** (6.43)
2	Proposed	200	**0.345** (0.014)	**0.186** (0.120)	**0.181** (0.090)	**16.4** (2.74)
	AA-MORO [21]	200	**0.571** (0.268)	**0.122** (0.103)	**0.029** (0.016)	**67.5** (4.00)
3	Proposed	59(2)	**0.205** (0.028)	**0.836** (0.067)	**0.198** (0.044)	**19.5** (2.82)
	AA-MORO [21]	59(2)	**0.190** (0.038)	**0.341** (0.200)	**0.033** (0.008)	**69.4** (2.62)
	MORO-KS [38]	1260	**0.300**	**0.234**	**0.033**	**-**
4	Proposed	130	**0.491** (0.003)	**0.719** (0.071)	**14.375** (2.917)	**41.2** (10.18)
	AA-MORO [21]	130	**0.496** (0.015)	**0.506** (0.086)	**5.776** (0.991)	**68.2** (2.73)
	MORO-KS [38]	2240	**0.358**	**0.304**	**6.826**	**-**

3.3.6 Example 3.5: Scalable Problem ZDT3

To investigate the changes in performance of the proposed method with increasing problem size, a scalable example problem ZDT3 is presented here with a varying number of decision variables. This problem was first formulated in [78] as a scalable test problem with a discontinuous non-dominated frontier. It later had uncertainty added in [21].

3.3.6.1 Example 3.5 – Setup

The ZDT3 problem with uncertainty as presented in [21] has two objective functions with uncertainty, one constraint on objective variation, and ten variables,

two of which have added uncertainty. Converting to a feasibility robustness formulation, the uncertainty in the objectives is removed and so the uncertainty only remains in the objective variation constraint. The number of variables is also relaxed to allow n variables as in [78] and solved for $n = 5, 10, 20,$ and 50. The formulation is provided in Eq. (3.17). Each of these cases was tested over five independent runs.

3.3.6.2 Example 5 – Results

$$
\begin{aligned}
&\min_{\boldsymbol{x}} f_1(\boldsymbol{x}, \boldsymbol{u}_0), \qquad \min_{\boldsymbol{x}} f_2(\boldsymbol{x}, \boldsymbol{u}_0) \\
&s.t. \\
&\quad g = |f_2(\boldsymbol{x}, \boldsymbol{u}) - f_2(\boldsymbol{x}, \boldsymbol{u}_0)| \leq \Delta f_2 \\
&where \\
&\quad f_1(\boldsymbol{x}, \boldsymbol{u}) = x_1 + \Delta x_1 \\
&\quad f_2(\boldsymbol{x}, \boldsymbol{u}) = h(\boldsymbol{x}, \boldsymbol{u}_0) p(\boldsymbol{x}, \boldsymbol{u}_0) \\
&\quad p(\boldsymbol{x}, \boldsymbol{u}) = 1 + \frac{9}{n+1}\left(\sum_{i=2}^{n} x_i + \Delta x_1 + \Delta x_2\right) \\
&\quad h(\boldsymbol{x}, \boldsymbol{u}) = 1 - \sqrt{\frac{f_1}{p}} - \left(\frac{f_1}{p}\right) sin(10\pi f_1) \\
&\quad \forall \Delta x_1 \in [0, 0.1]; \Delta x_2 \in [-0.2, 0.2] \\
&\quad 0 \leq x_i \leq 1, i = 1, \ldots, n \\
&\quad \Delta f_2 = 0.5
\end{aligned}
\tag{3.17}
$$

Results for the four cases are shown in Figure 3.9. For the 5-variable case, shown in Figure 3.9(a), the proposed approach consistently converged to a solution seemingly tight to the feasible boundary. AA-MORO, had better range, finding solutions over the full range of 0 to 1, but was somewhat inconsistent. The second case, with 10 variables, shown in Figure 3.9(b) has AA-MORO converging to a better solution consistently, but the proposed approach has the better range. Starting with the 20-variable case in Figure 3.9(c) the proposed approach starts to outperform AA-MORO in terms of solution convergence and range. The best solutions of the proposed approach are better than the best solutions of AA-MORO, and in all but one region, the worst solutions of the proposed approach are better than the worst

solutions of AA-MORO. This trend continues in the 50-variable case. In Figure 3.9(d), the region where the proposed approach's worst case is worse than AA-MORO's worst case is smaller than before. Except for AA-MORO's consistency, the proposed approach for the 50-variable case seems to outperform AA-MORO. This improved relative performance with problem size is most likely due to the different balancing of exploring and exploiting the surrogate models. AA-MORO decomposes the two into a primary exploit step, followed by an exploring step. As seen in the 5-variable case, this works well for smaller problems. Increasing problem sizes require more emphasis on exploring the design space. The expected improvement criterion that the proposed approach is based on balances the two in its search for new solutions. To confirm the results, the feasibility of these solutions needs to also be verified.

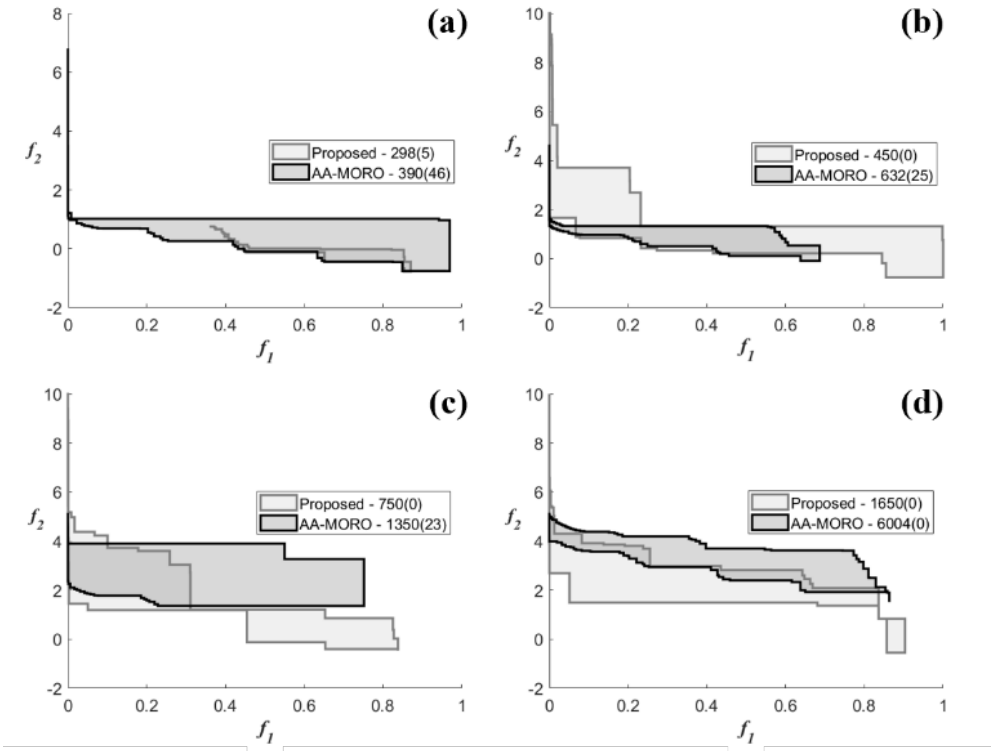


Figure 3.9: Results for scalable example ZDT3 with number of variables: a) 5, b) 10, c) 20, and d) 50

3.3.7 Feasibility Verification of Solutions

Feasibility verification of solutions was performed for the five examples. Each design point in each robust non-dominated frontier was tested against 10,000 random scenarios. The fraction of these that were feasible was averaged across all independent runs and a 95% confidence interval was generated. For Examples 3.1-3.4 20 runs were used, and 5 runs were used for Example 3.5. The results of this verification are provided in Table 3.3. Examples 3.3, and 3.4 had nearly equal feasibility results for the two methods. Both found robustly feasible solutions. Example 3.1 and the larger example, Example 3.2, had more differentiation. For Example 3.1, the proposed approach was feasible for more scenarios and had a tighter

confidence bound. However, for Example 3.2, the proposed approach had a significantly smaller fraction of robust feasible solutions which seems at odds with the results shown in Figure 3.7(b) and Figure 3.8(b). In the figures, it appears that AA-MORO is more affected by the removal of the non-robust points. More of the best solutions for AA-MORO were removed, but a greater fraction of all of the solutions was removed from the proposed approach. For the scalable example, Example 3.5, both methods were feasible for almost all cases. Only AA-MORO had some infeasibility for the 5-variable case.

Table 3.3: Fraction of feasible test scenarios a 95% confidence interval

Example	Proposed	AA-MORO [21]
3.1	1[1,1]	0.96[0.95,0.97]
3.2	0.93[0.92,95]	0.96[0.94,0.98]
3.3	1[1,1]	1[1,1]
3.4	1[1,1]	1[1,1]
3.5: $n = 5$	1[1,1]	0.95[0.93,0.97]
3.5: $n = 10$	1[1,1]	1[1,1]
3.5: $n = 20$	1[1,1]	1[1,1]
3.5: $n = 50$	1[1,1]	1[1,1]

3.4 Conclusions

In this chapter, a new method for bi-objective feasibility robust optimization is proposed. The formulation utilized the surrogate model and its predicted standard deviation using the expected improvement criterion to optimize sampling to improve a set of candidate robustly feasible non-dominated optimal points. This was staggered

with a worst-case scenario search and a relaxation of the frontier to allow for increasingly robust and conservative solutions to be found. The proposed method was then compared against existing methods in the literature, AA-MORO and MORO-KS.

The proposed method appears promising, especially from the point of view of finding robust feasible designs for very low numbers of function evaluations. In most of the examples, AA-MORO provided a more optimistic set of non-dominated solutions. However, when the non-robust solutions were removed the proposed approach had better HD in three of the four examples and better OS in all four. The primary shortcoming of the proposed approach is the lower number of solutions. AA-MORO provided greater numbers of solutions which could be useful to a decision maker with better SP values in three of the four examples. This is an aspect that could be improved in the proposed method. As seen in the fifth example, larger problem sizes caused AA-MORO to produce increasingly conservative results, but the proposed approach was less susceptible to this.

The next chapter seeks to expand the techniques developed in this chapter to optimization problems of greater than two objective functions, addressing RQ3.

Chapter 4: Multi-Objective Robust Optimization with Surrogates

This chapter proposes a method for robust optimization under interval uncertainty for multi-objective problems, particularly for those with more than two objective functions.

The organization of this chapter is as follows. First, the problem is defined in Section 4.1, with emphasis on the differences with bi-objective robust optimization. Next, in Section 4.2 the proposed method is presented. In Section 4.3, several example problems are detailed along with its results. Finally, in Section 4.4, the concluding remarks on the efficacy of the proposed method are provided.

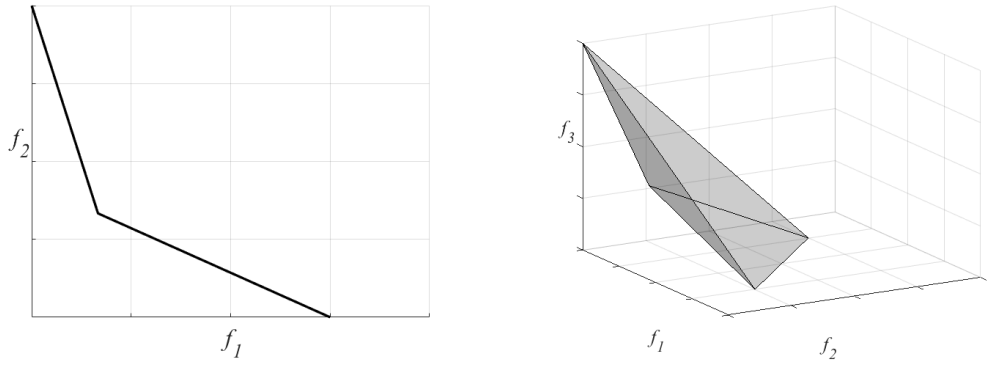
4.1 Problem Statement

The multi-objective robust optimization problem under interval uncertainty has the same form as Eq. (3.1), but the number of objective functions I is no longer limited to two as seen in Eq. (4.1).

$$\begin{aligned} &\min_{\boldsymbol{x}} f_i(\boldsymbol{x}) && i \in 1, \dots, I \\ &s.t. \;\; g_j(\boldsymbol{x}, \boldsymbol{u}) \leq 0 && j \in 1, \dots, J \\ &\quad \forall \boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u] \\ &\quad \boldsymbol{x} \in [\boldsymbol{x}_l, \boldsymbol{x}_u] \end{aligned} \tag{4.1}$$

The two main effects that this has on multi-objective solvers are on the size of the problem and the relationship in the ordering in solutions with respect to each objective.

The first challenge for solving the problem in Eq. (4.1) is that as the number of objective functions increases from two to three and beyond, the geometry of the non-dominated frontier (the solution set for the multi-objective problem) increases from a curve (or piecewise linear) to a surface and hyper-surface. In Figure 4.1(a), an example of a solution set to a bi-objective optimization problem is shown. The optimal, non-dominated solutions in this case are all along the black piecewise line. Moving to a three-objective optimization problem in Figure 4.1(b), the solution set is represented by the grey surface. As such, the numbers of samples needed to approximate these geometries goes up exponentially as the number of objective functions is increases. Keeping the number of function evaluations low for such problems becomes even more imperative.



(a) (b)

Figure 4.1: (a) A set of optimal solutions along (a) a curve in a two-objective space and (b) a surface in a three-objective space.

The second challenge that is faced by problems with more than two objective functions is the lack of ordering or arrangement of the solutions. In a bi-objective optimization problem, sorting the non-dominated solutions by one objective

automatically sorts them by the second objective in reverse order. Beyond two objectives, this property no longer holds and integration of acquisition functions over the non-dominated space requires a more sophisticated programmatic approach.

The next section will discuss two current acquisition functions for finding the next location to sample in a Bayesian optimization scheme.

4.1.1 Multi-Objective Centroidal Acquisition Functions

The framework of Algorithm 3.1, as presented in Chapter 3, is applicable to robust optimization problems of any number of objective functions. However, the acquisition function used in Chapter 3, the centroidal method, adapted from [12] only applies to problems with two objectives. Variations of the centroidal method and other acquisition functions are thus sought.

Expansion of the centroidal method to higher numbers of objectives has been discussed before [46] and shown [54] but the formulations are not readily available. These existing methods require a partitioning of the non-dominated space, as seen in Figure 4.2, which can become intractable at higher dimensions. The most straightforward method partitions the space into $(\nu + 1)^I$ cells over which the improvement metrics are calculated. Here, ν is the number of non-dominated solutions, and I is the number of objectives [79]. This expression for the number of cells comes from dividing the non-dominated space by every combination of objective function values from the current non-dominated set. A demonstration in a two-objective space is shown in Figure 4.2 for a minimization problem. Here, there are six non-dominated solutions shown as black dots and two reference points B and G shown as empty circles. These are the "bad" point and the "good" point

respectively. The bad point should be chosen to be dominated by every solution, and the good point should dominate every solution. These are generally taken to be $(\infty, \dots, \infty)$ and $(-\infty, \dots, -\infty)$. Each non-dominated solution defines the lower bound for a region in each dimension and the upper bound for a region in each dimension. The number of upper bounds on partitioned regions is then equal to the number of non-dominated solutions plus one for the bad point or $\nu + 1$. This is true for each objective function. The partitioned regions (or cells) are then the intersections of the regions bounded by the non-dominated solutions and the good and bad point. For m objective functions this gives us $(\nu + 1)^m$. For the six non-dominated solutions in Figure 4.2, with two objective functions, there are $(6 + 1)^2 = 49$ regions. Note that this process partitions the entire objective space. It must then be established which of the cells are in the dominated and non-dominated regions. Faster partitioning methods have been developed to reduce the number of partitions needed [54], [55], [57], [80] as well as an approximation that ignores less impactful partitions [79].

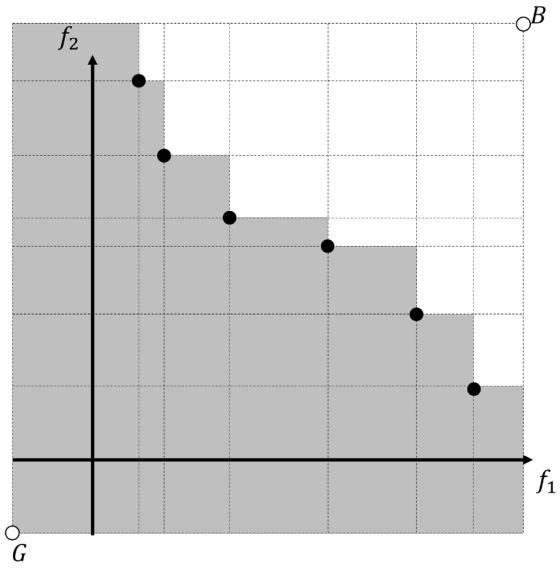


Figure 4.2: An example of a partitioning of the objective space based on the current non-dominated solutions and a "good" point G and a "bad" point B. The number of partitions is $(\nu + 1)^I = (6 + 1)^2 = 49$

One method to avoid dependency on the partitioning is with a Monte Carlo sampling. A method for doing so is discussed in Section 4.2.

4.1.2 Multi-Objective Expected Hypervolume Improvement

Another acquisition function which has received much attention is the expected hypervolume improvement. The Walking Fish Group (WFG) method was originally developed as a means of efficiently calculating the hypervolume difference [57], but has also been adapted for partitioning the non-dominated region [54]. It utilizes a branch-and-bound procedure based on the principle that cells which dominate a non-dominated cell will also be non-dominated. This allows for cells to be grouped to form fewer, larger cells without explore each one.

4.2 Methodology

The proposed methodology adopts the robust optimization scheme with non-dominated relaxation from Section 3.2.3, shown in Figure 3.4 and Table 3.1. Algorithm 3.1 is combined with a general Monte Carlo version of the centroidal expected improvement acquisition function, termed Centroidal Monte Carlo (CMC), is presented here for multiple objectives. Instead of integrating the probability distribution function over the entire non-dominated domain with exponentially increasing numbers of cells, a finite set of samples can be taken using CMC as described here.

- Given training sets of designs $\boldsymbol{X}$ and objective function values $\boldsymbol{F}_i$, a GP model is trained for each of the objective functions.
- The optimizer then generates a candidate design $\boldsymbol{x}_c$ to be evaluated.
- Samples are drawn from the multi-variate normal distribution described by the predicted output of the surrogate models of the objective functions at point $\boldsymbol{x}_c$ as in Eq. (4.2).

$$\boldsymbol{f}_{candidate} \sim N\left(\boldsymbol{\mu}(\boldsymbol{x}_c), \boldsymbol{\sigma}(\boldsymbol{x}_c)\right) \tag{4.2}$$

  Here, $\boldsymbol{\mu}(\boldsymbol{x})$ and $\boldsymbol{\sigma}(\boldsymbol{x})$ are I dimensional vectors of the mean prediction and error prediction of the surrogates for the I objective functions at a given configuration of designs $\boldsymbol{x}$. These random samples are demonstrated in Figure 4.3 as the small points (both black and red).

- These samples are filtered according to their relationship to the existing non-dominated frontier as either “non-dominated” or “dominated”. In Figure 4.3, the non-dominated random samples are shown in red.
- Integrating the positions of the non-dominated samples gives the centroid of the improvement region. This the indicated by the black square labeled $(\bar{f}_1, \bar{f}_2)$ in Figure 4.3.
- The probability of improvement (probability that the predicted mean value will be on the non-dominated frontier) is the fraction of samples that are non-dominated.
- The expected improvement is calculated as in Eq. (3.12) as the product of Euclidean distance between the centroid and the closest existing non-dominated solution, $(f_1^{ND,c}, f_2^{ND,c})$ in Figure 4.3, and the probability of improvement.

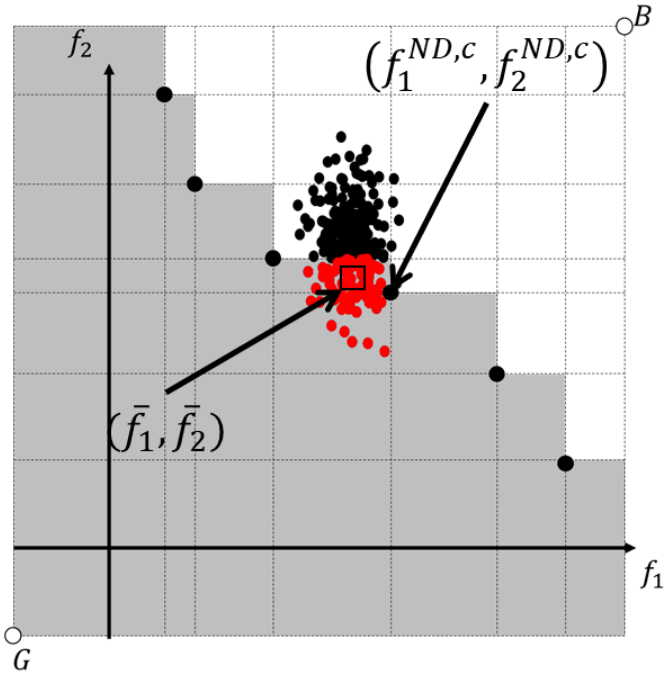


Figure 4.3: Demonstration of the centroidal Monte Carlo method for multi-objective expected improvement

This can then be combined with Algorithm 3.1, Table 3.1, to find robust multi-objective solutions by replacing the previous expected improvement calculations. Recall, Algorithm 3.1 iteratively trains surrogate models, solves the robust optimization problem for the known scenarios, updates models and estimates of feasible points, then repeats. The resulting method uses Gaussian process surrogates, trained on the previously gathered data, to approximate the objective and constraint function values at a candidate design. Constraints are handled using the strategy in Eq. (3.14), relaxing the mean constraint value in proportion to the prediction uncertainty of the GP. Optimizing the CMC acquisition function, a point which will be feasible for all known scenarios and is expected to improve the non-dominated set the most is sampled. The approximated worst-case constraint violation is then solved for producing a design and scenario pair. The models are retrained with the new sample point (the design and scenario pair) and the scenario is added to the list of worst-case scenarios. The updated models are then used to estimate which of the current feasible points are not robust and expected to dominate a robust point. These are removed for the current iteration. The expected centroidal improvement over the remaining non-dominated points can then be calculated, starting the procedure over.

Several examples and their results for the above method and some from the literature are provided in the next section.

4.3 Examples

In this section several examples are provided for optimization problems with multiple objectives and parameters with interval uncertainty in the constraint functions. The first two problems in Subsections 4.3.1 and 4.3.2 are examples from [64], which can be tested at varying numbers of decision variables and/or objectives and constraints. Here, both problems are taken to have three objective functions. The third problem in Subsection 4.3.3 is a modification of the engineering example from Subsection 3.3.4, the optimization of a three-bar truss. The methods being compared are the robust optimization loop described in Algorithm 3.1 with the proposed CMC acquisition function, as proposed in this chapter, an exact calculation of the centroidal method (EC)[12], [49] for multiple objective functions [54], and the expected hypervolume improvement (EHVI) using the Walking Fish Group (WFG) algorithm [57]. Additionally, AA-MORO [21] and the deterministic MORO [21] are used to solve the robust optimization problems.

Solutions of problems with three or more objectives are difficult to compare by visual inspection, so instead several quality metrics are used to describe the results. The metrics used here are the same as those described in Section 3.3, namely HD to measure convergence based on the size of the non-dominated region [76], OS to measure spread as the hypervolume contained between the furthest end points [76], and SP to measure the regularity of the spacing as the standard deviation of the distances between subsequent non-dominated solutions [77]. Additionally, the number of non-dominated robust solutions is reported for each method. All values are

reported with their mean and standard deviation over the five independent runs of each method. The mean and standard deviation were then used to perform a two-sample z-test with a significance level of 0.05 to test if the differences in performance between the metrics for the proposed CMC and other tested methods were statistically significant. Except where otherwise stated, the differences were statistically significant.

The deterministic MORO [21] method used MATLAB's 'gamultiobj' [59] function with all default values to solve the robust optimization problem. All four surrogate methods used an initial sample size equal to: $\frac{(2n+1)(2n+2)}{2} + 1$, where n is the total number of design variables and uncertain parameters. Each method was run for five iterations, adding five new sample points each iteration. Verification was also performed for all results by testing the constraints for feasibility against 10,000 random scenarios. Results from the verification study are omitted because all solutions for all methods tested were found to be fully feasibly robust.

4.3.1 Example 4.1: DTLZ8

The DTLZ8 problem is the first of two constrained variable-size test problems introduced in [64] which can be solved for different numbers of objective functions and design variables. There are I objective functions, I constraints, n variables, and n uncertain parameters. It has been modified here to have uncertainty in each of the design variables. The formulation is provided in Eq. (4.3). The constant c is the uncertainty in the design variable. Here the value of c is 0.1.

$$
\begin{aligned}
&\min_{x} f_i(\boldsymbol{x}) = \frac{1}{\left\lfloor \frac{d}{I} \right\rfloor} \sum_{n=\left\lfloor (i-1)\frac{d}{I} \right\rfloor}^{\left\lfloor i\frac{d}{I} \right\rfloor} x_i, \qquad i \in 1, \dots, I \\
&s.t.\, g_i(\boldsymbol{x}, \boldsymbol{u}) = 1 - f_I(\bar{\boldsymbol{x}}) - 4f_i(\bar{\boldsymbol{x}}) \leq 0, \quad i \in 1, \dots, (I-1) \\
&\quad g_I(\boldsymbol{x}, \boldsymbol{u}) = 1 - 2f_I(\bar{\boldsymbol{x}}) - \min_{\substack{n \neq i \\ n,i \in [1,I-1]}} [f_n(\bar{\boldsymbol{x}}) + f_i(\bar{\boldsymbol{x}})] \leq 0 \\
&\quad u_i \in [-c, c], x_i \in [0 + c, 1 - c] \\
&\quad \bar{x}_i = x_i + u_i
\end{aligned}
\tag{4.3}
$$

The DTLZ8 problem was solved for three objective functions and three constraints. Tests were performed for the case with five variables and five uncertain parameters (uncertain variables) and the case with ten variables and ten uncertain parameters (uncertain variables). The results are given in Table 4.1 and Table 4.2 respectively.

Table 4.1: Quantitative results for Example 4.1 as **mean** (standard deviation) for 5 variables

Approach	# Solutions	HD	OS	SP
CMC	**52** (3.35)	**0.73** (0.01)	**2.39** (0.10)	**0.09** (0.01)
EHVI [57]	**34.4** (4.45)	**0.69** (0.03)	**2.05** (0.36)	**0.14** (0.01)
AA-MORO [21]	**68.6** (2.3)	**0.70** (0.01)	**2.56** (0.13)	**0.08** (0.00)
MORO [21]	**67.8** (1.72)	**0.65** (0.01)	**2.41** (0.15)	**0.07** (0.01)

Recalling that for HD lower values are better, for OS higher values are better, for SP lower values are better, and more solutions are better, the proposed CMC approach combined with Algorithm 3.1 had the worst HD value and second worst values for the other quality metrics. The second method tested, EHVI combined with Algorithm 3.1 performed similarly but with better HD. The two methods entirely from the literature, AA-MORO and MORO, generally performed the best.

Table 4.2: Quantitative results for Example 4.1 as **mean** (standard deviation) for 10 variables

Approach	# Solutions	HD	OS	SP
CMC	**74.8** (7.93)	**0.61** (0.01)	**0.56** (0.04)	**0.05** (0.01)
EHVI [57]	**44** (6.4)	**0.61** (0.01)	**0.47** (0.05)	**0.06** (0.01)
AA-MORO [21]	**68.8** (1.85)	**0.58** (0.03)	**0.76** (0.12)	**0.05** (0.01)
MORO [21]	**69.6** (0.8)	**0.57** (0.01)	**0.83** (0.01)	**0.05** (0.00)

For the ten-variable case, we see two general trends. First, differences in performance for all methods decreased for all parameters except OS. The second general trend is that OS values for all of the methods are an order of magnitude smaller, suggesting a clustering of the solutions for all methods. Other than the number of solutions and the spacing metric, the proposed CMC method again performed worse in comparison. One possible explanation of this performance could be due to DTLZ8 being the least complex of the provided examples. The emphasis on balancing exploration and exploitation in Algorithm 3.1 may not be paying off in a region that is easier to exploit.

4.3.2 Example 4.2: DTLZ9

The DTLZ9 problem is the second of the constrained, variable-sized test problems from [64]. A version of this problem was previously modified to include uncertainty in [10], but it was used in a single-objective optimization setting. The form used here has I objectives, $I - 1$ constraints, n design variables, and n uncertain parameters. The uncertainty is added to the design variables. The amount of uncertainty c is set to 0.1. Results for the three objective, five variable case are shown in Table 4.3, for the three objective, ten variable case are shown in Table 4.4.

$$
\begin{aligned}
&\min_{\boldsymbol{x}} f_i(\boldsymbol{x}) = \sum_{n=\left\lfloor (i-1)\frac{d}{I} \right\rfloor}^{\left\lfloor i\frac{d}{I} \right\rfloor} x_i^{0.1}, \qquad i \in 1, \dots, I \\
&s.t.\, g_i(\boldsymbol{x}, \boldsymbol{u}) = 1 - f_I^2(\bar{\boldsymbol{x}}) - f_i^2(\bar{\boldsymbol{x}}) \leq 0, \quad i \in 1, \dots, (I-1) \\
&\quad u_i \in [-c, c], x_i \in [0 + c, 1 - c] \\
&\quad \bar{x}_i = x_i + u_i
\end{aligned} \tag{4.4}
$$

Table 4.3: Quantitative results for Example 4.2 as **mean** (standard deviation) for 5 variables

Approach	# Solutions	HD	OS	SP
CMC	**26.4** (3.35)	**0.92** (0.01)	**3.37** (0.36)	**0.03** (0.01)
EC	**18.4** (5.00)	**0.93** (0.00)	**3.86** (0.04)	**0.06** (0.02)
EHVI	**20** (1.67)	**0.91** (0.00)	**3.57** (0.07)	**0.04** (0.01)
AA-MORO	**70** (0)	**0.93** (0.01)	**3.78** (0.08)	**0.02** (0.01)
MORO	**70** (0)	**0.93** (0.00)	**3.22** (0.39)	**0.01** (0.00)

Compared to DTLZ8, the DTLZ9 problem is more complicated due to its nonconvex constraints. In this case, the HD, OS, and SP metrics were all very close. However, the three acquisition functions used with Algorithm 3.1 all produced low numbers of solutions compared to MORO and AA-MORO.

Table 4.4: Quantitative results for Example 4.2 as **mean** (standard deviation) for 10 variables

Approach	# Solutions	HD	OS	SP
CMC	**41.67** (11.79)	**0.81** (0.00)	**2.24** (0.03)	**0.03** (0.01)
EC	**15.5** (4.03)	**0.77** (0.01)	**1.9** (0.01)	**0.14** (0.03)
EHVI	**25.8** (5.9)	**0.81** (0.00)	**2.22** (0.03)	**0.06** (0.01)
AA-MORO	**70** (0)	**0.79** (0.01)	**2.00** (0.04)	**0.02** (0.01)
MORO	**70** (0)	**0.79** (0.00)	**1.97** (0.02)	**0.01** (0.00)

Moving to the ten-variable case, the EC outperformed the others in terms of HD. The proposed approach shows an advantage in OS but the improvement over EHVI is not statistically significant. MORO had the best SP value while EC had the worst. The number of solutions is again low for the methods used with Algorithm 3.1, but the CMC method performed better than the other two such methods.

Next, the DTLZ9 problem is expanded to four objectives and 10 variables and the results are shown in Table 4.5. The HD values show CMC on par with AA-MORO and MORO performing slightly better. The proposed CMC performed the best for OS and only trailed behind AA-MORO at a level that was not statistically significant. Compared to the previous, smaller, cases the relative performance of CMC is noticeably improved. Similar to the bi-objective case, this starts to hint at advantages in problems of increased size or complexity and needs to be investigated with more problems of larger sizes.

Table 4.5: Quantitative results for Example 4.2 as **mean** (standard deviation) for 4 objectives and 10 variables

Approach	# Solutions	HD	OS	SP
CMC	**112.5** (17.89)	**0.89** (0.00)	**0.98** (0.01)	**0.04** (0.01)
AA-MORO	**71.4** (2.8)	**0.89** (0.00)	**0.96** (0.02)	**0.04** (0.00)
MORO	**70** (0)	**0.88** (0.01)	**0.91** (0.01)	**0.05** (0.01)

4.3.3 Example 4.3: Three Bar Truss

The three-bar truss problem is detailed in Subsection 3.3.4 with the formulation given in [73]. It has been modified here to allow for a third objective to be optimized. In addition to minimizing the horizontal tip deflection and the volume of members 1 and 3, the stress in the first member should also be minimized. The corresponding constraint on the stress in the first member is also removed. The new formulation is provided in Eq. (4.5) with three objective, four constraints, two design variables, and two uncertain parameters.

$$
\begin{aligned}
&\min_{A_1,\theta} f_1 = u(A_1,\theta)\\
&\min_{A_1,\theta} f_2 = V(A_1,\theta)\\
&\min_{A_1,\theta} f_3 = \sigma_1(A_1,\theta)\\
&s.t.\ g_1 = \sigma_2 \le \sigma_{yield}\\
&\quad g_2 = \nu \le \nu_{max}\\
&\quad g_3 = |u(A_1,\theta) - u(A_1+\Delta A_1,\theta+\Delta\theta)| \le \Delta f_1\\
&\quad g_4 = |V(A_1,\theta) - V(A_1+\Delta A_1,\theta+\Delta\theta)| \le \Delta f_2\\
&\quad 2 \le A_1 \le 10\ cm^2,\ \ 30 \le \theta \le 40°\\
&\quad -0.1 \le \Delta A_1 \le 0.1\ cm^2, -5° \le \Delta\theta \le 5°
\end{aligned}
\tag{4.5}
$$

The definitions of all calculated values are given in Eq. (4.6).

$$
\begin{aligned}
u &= \frac{\sqrt{2}lPcos(\theta)}{A_1E}\\
\nu &= \frac{\sqrt{2}lPsin(\theta)}{\left(A_1+\sqrt{2}A_2\right)E}\\
\sigma_1 &= \frac{P}{\sqrt{2}}\left[\frac{cos(\theta)}{A_1} + \frac{sin(\theta)}{A_1+\sqrt{2}A_2}\right]\\
\sigma_2 &= \frac{\sqrt{2}Psin(\theta)}{\left(A_1+\sqrt{2}A_2\right)}\\
V &= 2\sqrt{2}A_1l + A_2l
\end{aligned}
\tag{4.6}
$$

The results for the three objective three-bar truss problem are provided in Table 4.6. The CMC and EHVI methods produced the greatest number of solutions and had the greatest OS values. The EHVI acquisition function combined with Algorithm 3.1 also had the best average HD value. Other than more irregular spacing of the solutions, EHVI appears to have generally performed the best for this example.

Table 4.6: Quantitative results for Example 4.3 as **mean** (standard deviation)

Approach	# Solutions	HD	OS	SP
CMC	**77.6** (1.85)	**0.11** (0.01)	**2.66** (0.01)	**0.05** (0.01)
EHVI [57]	**74.5** (2.5)	**0.004** (0.03)	**2.66** (0.01)	**0.09** (0.02)
AA-MORO [21]	**70** (0)	**0.11** (0.03)	**2.45** (0.04)	**0.05** (0.01)
MORO [21]	**70** (0)	**0.10** (0.02)	**2.61** (0.04)	**0.05** (0.00)

4.4 Conclusions

The results in the previous section show that there is little difference in performance between the CMC, EC, and EHVI methods for most cases. For the simplest of the examples, DTLZ8, the methods from the literature, AA-MORO and MORO, performed the best. For DTLZ9 and the three-bar truss engineering problem, the three methods based on Algorithm 3.1 started to show relative improvement over MORO and AA-MORO, particularly with respect to the HD metric. Based on that trend, and the results of Chapter 3, it seems likely that the proposed methods would show improved performance for problems of greater dimension and complexity.

It is worth noting that all of the quality metrics can only give insight into the specific values that they measure. Different decision makers and applications may value some metrics over others. Although this dissertation generally considers computational cost in terms of function calls, it should be mentioned that the CMC method took substantially less wall-clock time to complete than the EC and EHVI methods employed.

In the next concluding chapter, this dissertation is summarized, the contributions listed, and some future directions for the research are given.

Chapter 5: Conclusions

In this dissertation, methods were sought for efficiently solving robust optimization problems under interval uncertainty for single-, bi-, and multi-objective problems. Surrogate modelling techniques were employed to decrease computational costs in the form of numbers of function calls.

In Chapter 2, a method for solving single-objective problems was introduced, using surrogate modelling to search for worst-case scenarios while alternating with a deterministic solver. In Chapter 3, a method for solving bi-objective optimization problems Bayesian optimization surrogate techniques was presented and compared with existing techniques from the literature. Lastly, in Chapter 4, the robust optimization procedure previously developed for bi-objective problems was extended to multi-objective optimization problems using several different acquisition functions for comparison.

In the next section, some highlights from these chapters are given.

5.1 Summaries

5.1.1 Chapter 2: Single-Objective Robust Optimization with Surrogates

A robustly feasible optimum solution, as shown in this chapter for several engineering problems, can be obtained at a lower computational cost than an existing method [10]. This cost savings is accomplished through the introduction of a surrogate model for the constraints during the solution of a subproblem wherein the constraints are maximized to determine the worst-case feasibility. Using this

subproblem to evaluate feasibility robustness is a popular technique in optimization problems with interval uncertainty [58], [81], [82]. Within the area of multi-objective robust optimization, this has been done using surrogate models of both the decision variables at the top-level subproblem and uncertain parameters at the bottom-level [58], but it has not been shown advantageous before by isolating the second (or 'feasibility') subproblem (bottom-level). It is shown here that iterative solving of the approximated feasibility subproblem and discarding of old surrogate models can more efficiently produce feasibly robust optimum solutions compared to an existing method. A further computational cost savings was accomplished by adding a nominal scenario to initialize the feasible region search as is done in methods such as that by [58] but removing it after the first iteration to reduce the number of scenarios closer to that of [10].

Many approaches used to solve robust optimization problems quickly become prohibitively more computationally expensive with increasing problem size [1], [61]. For small problem sizes, some of these methods can perform well, but that performance drops off as the number of variables, constraints, and/or scenarios increases. A variable-sized example was used to demonstrate that the use of surrogates in the proposed SFTC-RO enables better performance than an existing method when the number of variables in and uncertain parameters in the problem increases.

In addition to exploring the computational advantages of the proposed SFTC-RO, a thorough random scenario-based verification study was performed on the

solutions produced by SFTC-RO and by another method in the literature. Comparing the robust optimal solutions against randomly generated scenarios enables a comparison of trade-offs in computational savings from the use of a surrogate and any resulting loss of robustness.

5.1.2 Chapter 3: Bi-Objective Robust Optimization with Surrogates

Firstly, this chapter presented a new bi-objective robust optimization algorithm which utilizes the expected improvement criteria in the context of Bayesian methods. Outside of the robust optimization literature, multi-objective optimization problems with computationally expensive-to-evaluate functions have been presented before using Bayesian methods [12], [45]. Bayesian methods, unlike the direct substitution methods mentioned earlier, utilize not only the surrogate prediction, but also some statistical properties of the surrogate models to optimize the selection of a new point to sample. The proposed method here seeks to incorporate the expected improvement Bayesian optimization criterion into a new bi-objective robust optimization algorithm with constraint cuts.

A modified formulation of the expected improvement was proposed that augments the non-dominated set to improve the diversity of the solution set. This overcomes one of the major drawbacks of the bi-objective expected improvement acquisition function, which is small solution set size. Using this formulation, the search space is expanded to include nondominated regions between points on the frontier. This encourages the solver to find more non-dominated optimal points providing the designer with more robust options to choose from.

As part of the proposed method, a new technique for facilitating robust optimization with surrogates, termed “non-dominated relaxation”, was introduced. This replaces the usual filtering out of all infeasible points, enabling more efficient use of the expected improvement acquisition function in the two-level robust optimization loop.

Finally, conditions under which the proposed approach performs better than previous methods from the literature on AA-MORO and MORO-KS were demonstrated. As can be seen in the examples, the variable-sized example in particular, the proposed approach maintains its performance for larger problems better than the other existing methods considered in this chapter.

5.1.3 Chapter 4: Multi-Objective Robust Optimization

In this chapter a Monte Carlo method for calculating the centroidal improvement acquisition function was presented. The Monte Carlo approach does not rely on a partitioning of the non-dominated domain. This is beneficial because the partitioning alone can be computationally expensive.

Secondly, a comparison of multiple different acquisition functions in the context of robust multi-objective optimization was performed. The acquisition functions considered are the proposed centroidal Monte Carlo, an exact centroidal method, and the expected hypervolume improvement. Each of these was inserted into Algorithm 3.1 and applied to several example problems. These methods were also compared with two methods from the literature, AA-MORO and MORO. Insight is thus gained into their applicability in the robust optimization setting.

5.2 Contributions

5.2.1 Chapter 2 Contribution

Addressing Research Question 1: How can surrogate modelling be introduced to single-objective robust optimization under interval uncertainty to reduce the computation burden of the worst-case scenario search?

A new method for surrogate modelling was used to perform the worst-case scenario search in a bi-level feasibility-based robust optimization scheme under interval uncertainty. Based on the test results, it was observed that one of the key features of the proposed approach was its improved performance over an existing method as the number of variables and uncertain parameters is increased in the optimization problem.

5.2.2 Chapter 3 Contribution

Addressing Research Question 2: How can Bayesian optimization techniques be included to improve over current surrogate modelling methods for bi-objective robust optimization under interval uncertainty?

An improvement over the current methods from the literature was made in the form of a robust optimization scheme with a non-dominated solution relaxation. This enabled the integration of an iterative Bayesian optimization technique and iterative robust optimization technique. For smaller problems, it was seen that the proposed method with a balanced exploration and exploitation did not perform as well as exploitation focused methods from the literature. However, the test results showed

improved performance for problems with larger numbers of design variables and robust feasibility over existing techniques.

5.2.3 Chapter 4 Contribution

Addressing Research Question 3: How can Bayesian optimization techniques be included in robust optimization under interval uncertainty for problems with more than two objectives?

A new method of approximating improvement in a Bayesian optimization scheme was presented for problems of more than two objectives that improved the computational overhead of partitioning the objective space. This new approximation-based acquisition function was combined with a robust optimization strategy for the first study examining robust optimization in multiple objective problems. The proposed approach performed worse than the other methods in most cases. However, as was seen with the bi-objective method, there appears to be an improving trend as the number of variables, uncertain parameters, and objective functions increases. In the largest example tested in Chapter 4, the proposed approach appeared on par with the other methods being compared. More study is required, but it suggests that problems above this size will favor the proposed approach.

5.3 Directions for Future Work

The problems in this dissertation pertain to the increasing dimensions of robust optimization problems of multiple objectives. One area of future study would be further increasing of the problem sizes and as such improving performance of the

proposed methods for more competing objectives, constraints, variables, and uncertain parameters.

A second area for future study would be the consideration of different types of uncertainty. Interval uncertainty was considered for all of the methods in this dissertation because it makes no assumption on the distribution. If there is available distributional information, then interval uncertainty could be overly conservative. Allowing for some parameters to be interval and some distributional (or mixed uncertainty), could expand the applicability of the presented methods.

A third direction for future work would be exploration of different acquisition functions. The more common acquisition functions were utilized in the presented methods, but they all have strengths and weaknesses. Other acquisition functions, or combinations of them, could be better suited for the class of problems used in this work.

A final direction for future work would be the consideration of time as a parameter. This would allow for the developed methodologies to be used for optimal planning. The goal would be to create an optimal trajectory that is feasible for all times within a range while limiting the evaluations of a time varying simulation.

Appendix A: Example Problem Formulations

A.1 Example 2.1: Circle Problem

The circle problem is a simple numerical problem with a single objective, one constraint, two variables, and two uncertain parameters. It was formulated in [10] with uncertainty to demonstrate the progression of constraint cuts to the feasible region. The position of a point, described by the design variables, is maximized away from the center, but constrained by a circle. The circular constraint has a fixed radius size of $\sqrt{5}$, and the origin is centered at an unknown position described by the two uncertain parameters. This formulation is provided in Eq. (A.1)

$$\begin{aligned} &\min_{x_1,x_2} -x_1^2 - x_2^2 \\ &s.t. \\ &g(x,u) = (x_1 - u_1)^2 + (x_2 - u_2)^2 - 5 \geq 0 \\ &x_1, x_2 \in [-5,5] \\ &\forall u_1, u_2 \in [-1,1] \end{aligned} \tag{A.1}$$

A.2 Example 2.2: Welded Beam Design with Uncertainty

The design of a welded beam problem originated by [60] where it was described as choosing the thickness and breadth of a bar for welding along with the weld thickness and length. The cost of the beam is minimized, and the design must satisfy constraints on the dimensions, normal and shear stresses, buckling load, and maximum tip deflection. The number of design variables is 4, uncertain parameters is 8, and constraints is 6. The problem has been modified to include uncertainty in the design variables, applied load, distance between the load and support, maximum allowed shear stress, and maximum allowed normal stress [10]. The formulation is given below in Eq. (A.2).

$$\min_{x_1,x_2}(1+c_3)x_1^2x_2 + c_4x_3x_4(L+x_2) \tag{A.2}$$

$s.t.$

$$\frac{\sqrt{\tau'^2 + \frac{2\tau'\tau''(x_2+u_2)}{2R} + \tau''^2}}{(\tau_d+u_7)} - 1 \le 0$$

$$\frac{6(F+u_5)(L+u_6)}{(x_4+u_4)(x_3+u_3)^2(\sigma_d+u_8)} - 1 \le 0$$

$$(F+u_5)/(\frac{4.014\sqrt{EI\alpha}}{(L+u_6)^2} \times \frac{1-x_3-u_3}{2(L+u_6)}\sqrt{\frac{EI}{\alpha}} - 1 \le 0$$

$$\frac{16(F+u_5)(L+u_6)^3}{E(x_3+u_3)^3(x_4+u_4)} - 1 \le 0$$

$$\frac{x_1+u_1}{x_4+u_4} - 1 \le 0$$

$$\frac{1}{8(x_1+u_1)} - 1 \le 0$$

$where$:

$$\tau' = \frac{F+u_5}{\sqrt{2}(x_1+u_1)(x_2+u_2)}$$

$$\tau'' = \frac{(F+u_5)\left(L+u_6+\frac{x_2+u_2}{2}\right)R}{2\left(\sqrt{\frac{1}{2}}(x_1+u_1)(x_2+u_2)\left(\frac{(x_2+u_2)^2}{12}+\frac{x_3+u_3+x_1+u_1}{2}\right)^2\right)}$$

$$R = \sqrt{\frac{1}{4}(x_2+u_2)^2 + \left(\frac{x_3+u_3+x_1+u_1}{2}\right)^2}$$

$$I = \frac{1}{12}(x_3+u_3)(x_4+u_4)^3$$

$$\alpha = \frac{1}{3}G(x_3+u_3)(x_4+u_4)^3$$

$0.1 \le x_1, x_4 \le 2$

$0.1 \le x_2, x_3 \le 10$

$\forall u_1, u_2, u_3, u_4 \in [-0.05, 0.05]$

$\forall u_5 \in [-500, 500]$

$\forall u_6 \in [-0.2, 0.2]$

$\forall u_7, u_8 \in [-1000, 1000]$

$F = 6000, L = 14, c_3 = 0.1047, c_4 = 0.0481,$

$\tau_d = 13600, \sigma_d = 30000, E = 30 \times 10^6,$

$G = 12 \times 10^6$

A.3 Example 2.3: Enhanced Speed Reducer

The speed reducer problem from [65] was modified to include uncertainty by [11] and further modified by [10]. The seven design variables include face width of the gears, the teeth module, the number of teeth on the pinion, and the length and diameter of the two shafts. The objective function to minimize is the weight of the speed reducer. In addition to constraints on the dimensions of the gears and shafts, the bending stress of the gear teeth and both shafts, the compressive stress in the gear teeth, and the deflection of the two shafts. The uncertainty introduced is uncertainty in the design variables. The functions $m_2(\boldsymbol{x}, \boldsymbol{u})$ and $m_3(\boldsymbol{x}, \boldsymbol{u})$ appear in both the objective and constraint functions. However, since this work focuses on feasibility robustness, uncertainty is assumed to only exist in the constraints. Therefore, when $m_2(\boldsymbol{x}, \boldsymbol{u})$ and $m_3(\boldsymbol{x}, \boldsymbol{u})$ are evaluated in the objective function it is assumed the uncertainty in the design variables is zero. The formulation is provided below in Eq. (A.3).

$$\min_{x} \frac{m_2(\boldsymbol{x},\boldsymbol{0}) + m_3(\boldsymbol{x},\boldsymbol{0})}{1000} \tag{A.3}$$
$$s.t.\ g_j(\boldsymbol{x},\boldsymbol{u}) \le 0, \qquad j = 1,\dots,13;\ \ \forall \boldsymbol{u} \in \boldsymbol{U}_{wc}$$

where

$$g_1(\boldsymbol{x},\boldsymbol{u}) = \left((x_1+u_1)(x_2+u_2)^2(x_3+u_3)^2\right)^{-1} - 397^{-1}$$
$$g_2(\boldsymbol{x},\boldsymbol{u}) = \left((x_1+u_1)(x_2+u_2)^2(x_3+u_3)\right)^{-1} - 27^{-1}$$
$$g_3(\boldsymbol{x},\boldsymbol{u}) = (x_4+u_4)^3\left((x_2+u_2)(x_3+u_3)(x_6+u_6)^4\right)^{-1} - 1.93^{-1}$$
$$g_4(\boldsymbol{x},\boldsymbol{u}) = (x_5+u_5)^3\left((x_2+u_2)(x_3+u_3)(x_7+u_7)^4\right)^{-1} - 1.93^{-1}$$
$$g_5(\boldsymbol{x},\boldsymbol{u}) = -(x_4+u_4) + (x_6+u_6) - 0.3$$
$$g_6(\boldsymbol{x},\boldsymbol{u}) = -(x_5+u_5) + (x_7+u_7) - 0.3$$
$$g_7(\boldsymbol{x},\boldsymbol{u}) = 5 - (x_1+u_1)(x_2+u_2)^{-1}$$
$$g_8(\boldsymbol{x},\boldsymbol{u}) = (x_1+u_1)(x_2+u_2)^{-1} - 12$$
$$g_9(\boldsymbol{x},\boldsymbol{u}) = (x_2+u_2)(x_3+u_3) - 400$$
$$g_{10}(\boldsymbol{x},\boldsymbol{u}) = m_1(\boldsymbol{x},\boldsymbol{u}) - 1400$$
$$g_{11}(\boldsymbol{x},\boldsymbol{u}) = m_2(\boldsymbol{x},\boldsymbol{u}) - 1800$$
$$g_{12}(\boldsymbol{x},\boldsymbol{u}) = m_3(\boldsymbol{x},\boldsymbol{u}) - 1100$$
$$g_{13}(\boldsymbol{x},\boldsymbol{u}) = \left((x_4+u_4) - (x_6+u_6)\right)^2 + \left((x_5+u_5) - (x_7+u_7)\right)^2 - 0.3^2$$

$$m_1(\boldsymbol{x},\boldsymbol{u}) = 0.7854(x_1+u_1)(x_2+u_2)^2\left(\frac{10(x_3+u_3)^2}{3} + 14.933(x_3+u_3) - 43.0934\right) - 1.508(x_1+u_1)((x_6+u_6)^2+(x_7+u_7)^2) + 7.477((x_6+u_6)^3+(x_7+u_7)^3) + 0.7854\left((x_4+u_4)(x_6+u_6)^2+(x_5+u_5)(x_7+u_7)^2\right)$$

$$m_2(\boldsymbol{x},\boldsymbol{u}) = \frac{\sqrt{\left(\frac{754(x_4+u_4)}{(x_2+u_2)(x_3+u_3)}\right)^2 + 1.69\times 10^7}}{0.1(x_6+u_6)^3}$$

$$m_3(\boldsymbol{x},\boldsymbol{u}) = \frac{\sqrt{\left(\frac{754(x_5+u_5)}{(x_2+u_2)(x_3+u_3)}\right)^2 + 1.575\times 10^7}}{0.1(x_7+u_7)^3}$$

$$1 \le x_1 \le 15, 0.1 \le x_2 \le 1.5, 8 \le x_3 \le 28, 0.3 \le x_4,$$
$$x_5 \le 12.3, 1 \le x_6 \le 8, 1 \le x_7 \le 8$$

$$-0.01 \le u_1, u_2 \le 0.01, -1 \le u_3 \le 1,$$
$$-0.1 \le u_4, u_5, u_6 \le 0.01, -0.05 \le u_7 \le 0.05$$

A.4 Example 2.4: Angle Grinder

The angle grinder problem originated in [63]. The original objective function is an expression for the cost derived from the current consumption, power output, and weight of 20 existing angle grinder products. The full formulation from [63] is reproduced here. The single objective optimization problem is provided in Eq. (A.4).

$$
\begin{aligned}
&\min_{\boldsymbol{x}} \beta_0 + \beta_1 I + \beta_2 \left(\frac{P}{W}\right) + e \\
&s.t. \\
&g_1(\boldsymbol{x}) = B_r \leq 1.5\ T \\
&g_2(\boldsymbol{x}) = B_s \leq 1.5\ T \\
&g_3(\boldsymbol{x}) = B_g \leq 1.5\ T \\
&g_4(\boldsymbol{x}) = K_r \leq 10{,}000 \\
&g_5(\boldsymbol{x}) = K_s \leq 10{,}000 \\
&g_6(\boldsymbol{x}) = \frac{L}{G} \leq 5 \\
&g_7(\boldsymbol{x}) = N_c = \text{integer} \\
&g_8(\boldsymbol{x}) = N_s = \text{integer} \\
&g_9(\boldsymbol{x}) = N_{out} \leq 10{,}000 \\
&g_{10}(\boldsymbol{x}) = \sigma_b \leq 145\ MPa \\
&g_{11}(\boldsymbol{x}) = \sigma_f \leq 720\ MPa \\
&g_{12}(\boldsymbol{x}) = v_a \leq 3658\ m/s
\end{aligned}
\tag{A.4}
$$

where:

$$
B_r = \frac{\phi}{A_a} \qquad N_{out} = \frac{N}{r}
$$

$$
B_s = \frac{\phi}{2A_s} \qquad \sigma_b = \frac{K_a K_m F_i}{mJ}
$$

$$
B_g = \frac{\phi}{A_g} \qquad \sigma_f = Z_H Z_e \sqrt{\frac{K_a K_m F_i (d_v + D_v)}{d_v D_v}}
$$

$$
K_s = \frac{N_c I}{\pi} \qquad v_a = \pi N l_r
$$

The design variables are defined in Eq. (A.5). Values used here are calculated using motor design calculations in Eq. (A.6) and gear calculations in Eq. (A.7).

Pinion pitch diameter $D_p\ (m)$	$D_p \in [0.009,0.03]$	(A.5)
Current $I\ (A)$	$I \in [6,12]$	
Gap thickness $l_{gap}\ (m)$	$l_{gap} \in [0.0005,0.07]$	
Stack length $L\ (m)$	$L \in [0.01,0.02]$	
Armature turns N_c (# of turns)	$N_c \in [20,300]$	
Stator turns N_s (# of turns)	$N_s \in [10,200]$	
Gear ratio r	$r \in [0.2,4]$	
Stator outer radius R_o (m)	$R_o \in [0.01,0.1]$	
Stator thickness $t\ (m)$	$t \in [0.0001,0.1]$	

Armature diameter $l_r(m)$	$l_r = 2(R_o - t - l_{gap})$	(A.6)
Armature Section $A_r\ (m^2)$	$A_r = (\pi l_r^2)/4$	
Wrap length $l_{rw}\ (m^2)$	$l_{rw} = 2L_r + 2L$	
Wire resistivity $\rho\ (\Omega m)$ 20 awg	$\rho = 0.036\ \Omega\, m$	
Wire area $A_w\ (mm^2)$ 20awg	$A_w = 0.504\ mm^2$	
Armature resistance $R_a(\Omega)$	$R_a = (\rho N_c l_{rw})/A_w$	
Stator resistance $R_s\ (\Omega)$	$R_s = 2(\rho N l_{rw})/A_w$	
Resistance losses $P_{copper}\ (W)$	$P_{copper} = I^2(R_s + R_a)$	
Brush coefficient $\alpha\ (V)$	$\alpha = 2$	
Brush losses $P_{brush}\ (W)$	$P_{brush} = \alpha I$	
Voltage $V\ (V)$	$V = 120\ V$	
Power in $P_{in}\ (W)$	$P_{in} = IV$	
Motor output $P_{out}\ (W)$	$P_{out} = P_{in} - P_{brush} - P_{copper}$	
Density steel $\rho_s\ (kg/m^3)$	$\rho_s = 8000\ kg/m^3$	
Density copper $\rho_{copper}\ (kg/m\text{^}3)$	$\rho_{copper} = 8900\ kg/m^3$	
Stator mass $M_s\ (kg)$	$M_s = (R_o^2 - (R_o - t)^2)\pi L \rho_s$	
Armature mass $M_a\ (kg)$	$M_a = A_r L \rho_s$	
Windings mass $M_w\ (kg)$	$M_w = l_{rw}(N_c + 2N_s)A_w \rho_{copper}$	
Motor mass $M_m\ (kg)$	$M_m = M_s + M_a + M_w$	
Motor constant K	$K = N_c/\pi$	
Magnetomotive force $\Im$	$\Im = N_s I$	
Mean stator path $l_c\ (m)$	$l_c = \pi(2R_o + t)/2$	
Stator cross section $A_s\ (m^2)$	$A_s = Lt$	
Armature section $A_a\ (m^2)$	$A_a = Ll_r$	

Gap cross section $A_g\ (m^2)$	$A_g = Ll_r$
Permeability of steel μ_{steel}	$\mu_{steel} = 1000$
Permeability, free space μ_0	$\mu_0 = 4\pi \times 10^{-7}$
Stator reluctance $\Re_s$	$\Re_s = l_c/2(\mu_{steel}\mu_0 A_s)$
Armature reluctance $\Re_r$	$\Re_r = l_r/(\mu_{steel}\mu_0 A_a)$
Air gap reluctance $\Re_g$	$\Re_g = l_{gap}/(\mu_0 A_g)$
Total reluctance $\Re_{tot}$	$\Re_{tot} = \Re_s + \Re_a + 2\Re_g$
Flux ϕ	$\phi = \Im/\Re_{tot}$
Torque $T\ (N\ m)$	$T = K\phi/I$
Revolutions per minute N	$N = 9.549\ P_{out}(kw)/T(N\ m)$

Pinion torque (load rpm) $T_p\ (N\ m)$	$T_p = 9.459P_{out}/6500$	(A.7)
Gear torque (load rpm) $T_g\ (N\ m)$	$T_g = T_p r$	
Pressure angle ϕ_p	$\phi_p = 20°$	
Cone distance $C\ (m)$	$C = D_p/2\sin(\phi_p)$	
Face width $b\ (m)$	$b = 0.008\ m$	
Gear pitch diameter $D_g\ (m)$	$D_g = D_p r$	
Tooth loading intensity $F_i\ (N)$	$F_i = 2T_p C/(D_p b(C-b))$	
Elasticity factor (carbon steel) Z_e	$Z_e = 189$	
Zone factor Z_H	$Z_H = 4/\sin^2(2\phi_p)$	
Pinion pitch angle θ_g	$\theta_p = Asin(D_p/C)$	
Shaft angle γ	$\gamma = 90°$	
Gear pitch angle θ_g	$\theta_g = \gamma - \theta_p$	
Pinion cone depth $d_v\ (m)$	$d_v = D_p \sec(\theta_p)$	
Gear cone depth $D_v\ (m)$	$D_v = D_p \sec(\theta_g)$	
Amplification (light/med. shock) K_a	$K_a = 1.35$	
Load distribution (precision gears) K_m	$K_m = 1.2$	
Geometry factor J	$J = 0.25$	
Number of pinion teeth N_t	$N_t = 11$	
Module (pinion) m	$m = D_p/N_t$	
Pinion mass $M_p\ (kg)$	$M_p = (\pi D_p^2 b\rho_{steel})/4$	
Gear mass $M_g\ (kg)$	$M_g = (\pi D_g^2 b\rho_{steel}(/4$	
Bevel gears mass $M_{bg}\ (kg)$	$M_{bg} = M_p + M_g$	

The angle grinder problem was modified for multi-objective optimization with uncertainty in [58]. The objectives there are minimization of the total mass and maximization of the output power. The integer constraints on the number of armature turns N_c and stator turns N_s was removed, allowing them to be positive real numbers. Uncertainty appears as a $\pm 10\%$ variability in several variables and parameters, namely, the voltage V, stator outer radius R_o, stack length L, brush coefficient α, wire area A_w, gear face width b, the fixed assembly mass M_f, wire resistivity ρ, and the density of the steel ρ_{steel}. These uncertain parameters were also used in the single objective problem in [62]

A.5 Example 2.5/4.2: DTLZ9

DTLZ9 is a numerical test problem originally for multiobjective solvers [64]. A single objective version was created with added uncertainty by [10]. The single objective version takes the objective to be the sum of the multiple objective functions from the original multiobjective problem. For a selected value of n, the problem has n variables, n uncertain parameters. The other parameter M is set to be $M = \frac{n}{2}$ and $there\ are\ M - 1$ constraints. The uncertain parameters are uncertainty added to the decision variables. The formulation of DTLZ9 is provided here in Eq. (A.8). The multi-objective formulation shown in Eq. (A.9) minimizes the $f_j(x)$ functions and does not summate them. Additionally, in the multi-objective formulation the parameters n and M are set separately by the user. The fraction of the decision variables that is uncertain is expressed as r. In [10], this value is set to $r = 0.1$, and the summed values of the variables and parameters are restricted to the interval [0.01,0.99] instead of [0,1].

$$
\begin{aligned}
&\min_{x_i} \sum_{j\in\{1,\dots,M\}} f_j \\
&where \quad f_j(x) = \sum_{i=\left\lfloor (j-1)\left(\frac{n}{M}\right)\right\rfloor}^{\left\lfloor j\left(\frac{n}{M}\right)\right\rfloor} x_i^{0.1} \\
&s.t.\, g_k(x,u) = f_M^2(x+u) + f_k^2(x+u) - 1 \geq 0 \\
&\forall k \in \{1,\dots,M-1\} \\
&x_i \in [r, 1-r] \\
&\forall u_i \in [-r, r]
\end{aligned}
\tag{A.8}
$$

$$\min_{x_i} f_j \qquad j = 1, \dots, M$$

$$where \quad f_j(x) = \sum_{i=\left\lfloor (j-1)\left(\frac{n}{M}\right)\right\rfloor}^{\left\lfloor j\left(\frac{n}{M}\right)\right\rfloor} x_i^{0.1} \tag{A.9}$$

$$s.t.\, g_k(x,u) = f_M^2(x+u) + f_k^2(x+u) - 1 \geq 0$$

$$\forall k \in \{1, \dots, M-1\}$$

$$x_i \in [r, 1-r]$$

$$\forall u_i \in [-r, r]$$

A.6 Example 3.1: TNK

The TNK example was first formulated in [72] and is named for the author Tanaka. A robust adaptation was presented in [21] and is provided in Eq. (A.10). There are two variables, two objectives, two constraints, and two uncertain parameters.

$$
\begin{aligned}
&\min_{x} x_1 \\
&\min_{x} x_2 \\
&s.t. \\
&-x_1^2 - x_2^2 + 1 + 0.1\cos\left(16\,atan\frac{x_1}{x_2}\right) + 0.35\,sin(u_1)\,cos(u_2) \leq 0 \\
&2(x_1 - 0.5)^2 + 2(x_2 - 0.5)^2 - 1 \leq 0 \\
&10^{-3} \leq x_1, x_2 \leq 3.14 \\
&-2 \leq u_1 \leq 2, \qquad -1 \leq u_2 \leq 1
\end{aligned}
\tag{A.10}
$$

A.7 Example 3.2: CTP

The problem Constrained Test Problem (CTP) was formulated in [74] originally without uncertainty. It was modified to include uncertainty in [21]. The formulation is provided below in Eq. (A.11), with 10 design variables, two objectives, one constraint, and two uncertain parameters.

$$
\begin{aligned}
&\min_x f_1 = x_1 \\
&\min_x f_2 = h\left(1 - \sqrt{\frac{x_1}{h}}\right) \\
&s.t. \\
&a|sin(b\pi(sin\,(\theta)(f_2(x) - e) + cos(\theta)\, f_1(x))^c)|^d \le 0 \\
&\theta = -0.2\pi + \Delta\theta, a = 0.2, b = 10, \\
&c = 1, d = 6 + \Delta d, e = 1 \\
&h = \left|1 + \left(\sum_{k=2}^{10} x_k\right)^{0.25}\right| \\
&-0.2 \le \Delta\theta \le 0.2, -2 \le \Delta d \le 2 \\
&0 \le x_1 \le 1, -5 \le x_i \le 5, i = 2, \dots, 10
\end{aligned}
\tag{A.11}
$$

A.8 Example 3.3

This example is used in multiple locations, including [38] and [71]. It was originally formulated with two objectives, two constraints, two variables, and one uncertain parameter. In that formulation there was uncertainty in the constraints and objectives. For feasibility robustness, variability of the objective functions was constrained to be below 0.3 and 0.6 respectively. The objective constrained formulation is provided in Eq. (A.12).

$$
\begin{aligned}
&\min_{x_1} f_1 = p x_1 \\
&\min_{x_1, x_2} f_2 = \frac{p + x_2}{x_1} \\
&s.t. \\
&g_1 = 6 - 9x_1 - (p + \Delta p)x_2 \leq 0 \\
&g_2 = 1 - 9x_1 + (p + \Delta p)x_2 \\
&g_3 = |p x_1 - (p + \Delta p)x_1| - \Delta f_1 \leq 0 \\
&g_4 = \left| \frac{p + x_2}{x_1} - \frac{(p + \Delta p)x_2}{x_1} \right| - \Delta f_2 \leq 0 \\
&0.1 \leq x_1 \leq 1, \qquad 0 \leq x_2 \leq 5 \\
&-0.3 \leq \Delta p \leq 0.3 \\
&p = 1, \Delta f_1 = 0.3, \Delta f_2 = 0.6
\end{aligned}
\tag{A.12}
$$

A.9 Example 3.4/4.2: Three-bar Truss

The three-bar truss problem is an engineering design example originating from [65]. The single objective formulation was modified to include uncertainty in [61]. This was then modified into a bi-objective problem in [38]. The bi-objective formulation is provided in Eq. (A.13) with values defined in Eq. (A.14) . There are two objectives, minimizing the horizontal displacement of the truss joint, and the volume of the members. The diagram in Figure 3.6 is reproduced here in Figure A.1.

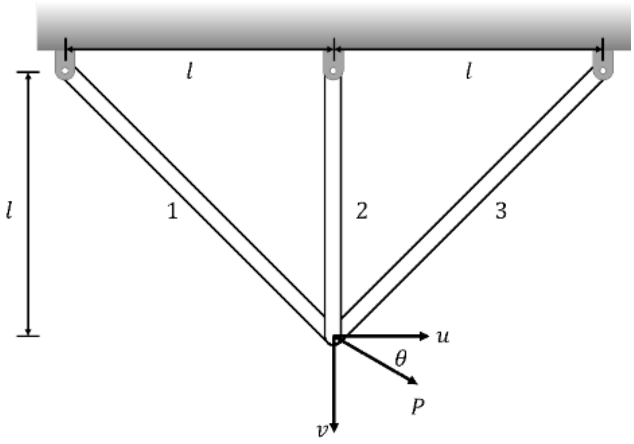


Figure A.1: Three-bar truss example

$$
\begin{aligned}
&\min_{A_1,\theta} f_1 = u(A_1,\theta)\\
&\min_{A_1,\theta} f_2 = V(A_1,\theta)\\
&s.t.\\
&g_1 = \sigma_1 \le \sigma_{yield}\\
&g_2 = \sigma_2 \le \sigma_{yield}\\
&g_3 = \nu \le \nu_{max}\\
&g_4 = |u(A_1,\theta) - u(A_1+\Delta A_1,\theta+\Delta\theta)| \le \Delta f_1\\
&g_5 = |V(A_1,\theta) - V(A_1+\Delta A_1,\theta+\Delta\theta)| \le \Delta f_2\\
&2 \le A_1 \le 10\ cm^2,\ \ 30 \le \theta \le 40°\\
&-0.1 \le \Delta A_1 \le 0.1\ cm^2, -5° \le \Delta\theta \le 5°
\end{aligned}
\tag{A.13}
$$

$$
\begin{aligned}
&u = \frac{\sqrt{2}lPcos(\theta)}{A_1E}, \nu = \frac{\sqrt{2}lPsin(\theta)}{\left(A_1+\sqrt{2}A_2\right)E}\\
&\sigma_1 = \frac{P}{\sqrt{2}}\left[\frac{cos(\theta)}{A_1} + \frac{sin(\theta)}{(A_1+\sqrt{2}A_2}\right],\\
&\sigma_2 = \frac{\sqrt{2}Psin(\theta)}{\left(A_1+\sqrt{2}A_2\right)},\\
&V = 2\sqrt{2}A_1l + A_2l
\end{aligned}
\tag{A.14}
$$

A three-objective variation of the problem is also presented in Eq. (A.15). The stress in the first member is also minimized, and the corresponding constraint is removed.

$$
\begin{aligned}
&\min_{A_1,\theta} f_1 = u(A_1,\theta)\\
&\min_{A_1,\theta} f_2 = V(A_1,\theta)\\
&\min_{A_1,\theta} f_3 = \sigma_1(A_1,\theta)\\
&s.t.\ \ g_1 = \sigma_2 \le \sigma_{yield}\\
&\qquad g_2 = \nu \le \nu_{max}\\
&\qquad g_3 = |u(A_1,\theta) - u(A_1+\Delta A_1,\theta+\Delta\theta)| \le \Delta f_1\\
&\qquad g_4 = |V(A_1,\theta) - V(A_1+\Delta A_1,\theta+\Delta\theta)| \le \Delta f_2\\
&\qquad 2 \le A_1 \le 10\ cm^2,\ \ 30 \le \theta \le 40°\\
&\qquad -0.1 \le \Delta A_1 \le 0.1\ cm^2, -5° \le \Delta\theta \le 5°
\end{aligned}
\tag{A.15}
$$

A.10 Example 3.5: ZDT3

This is a scalable test problem, allowing varying numbers of design variables, produced originally in [78] and is named after the authors, Zitzler, Deb, and Thiele. Uncertainty was added in [58] and the formulation is provided in Eq. (A.16). There are two objectives, n variables, and two uncertain parameters. A single constraint is added as a variability constraint on the second objective function. The nominal scenario $\boldsymbol{u}_0$ used in the objective functions is the mean of the two uncertain parameter ranges, $\boldsymbol{u}_0 = [0.05,0]$.

$$
\begin{aligned}
&\min_{\boldsymbol{x}} f_1(\boldsymbol{x}, \boldsymbol{u}_0), \qquad \min_{\boldsymbol{x}} f_2(\boldsymbol{x}, \boldsymbol{u}_0) \\
&s.t. \\
&\quad g = |f_2(\boldsymbol{x}, \boldsymbol{u}) - f_2(\boldsymbol{x}, \boldsymbol{u}_0)| \leq \Delta f_2 \\
&where \\
&\quad f_1(\boldsymbol{x}, \boldsymbol{u}) = x_1 + \Delta x_1 \\
&\quad f_2(\boldsymbol{x}, \boldsymbol{u}) = h(\boldsymbol{x}, \boldsymbol{u}_0) p(\boldsymbol{x}, \boldsymbol{u}_0) \\
&\quad p(\boldsymbol{x}, \boldsymbol{u}) = 1 + \frac{9}{n+1}\left(\sum_{i=2}^{n} x_i + \Delta x_1 + \Delta x_2\right) \\
&\quad h(\boldsymbol{x}, \boldsymbol{u}) = 1 - \sqrt{\frac{f_1}{p}} - \left(\frac{f_1}{p}\right) sin(10\pi f_1) \\
&\quad \forall \Delta x_1 \in [0,0.1]; \Delta x_2 \in [-0.2,0.2] \\
&\quad 0 \leq x_i \leq 1, i = 1, \dots, n \\
&\quad \Delta f_2 = 0.5
\end{aligned}
\tag{A.16}
$$

A.11 Example 4.1: DTLZ8

The DTLZ8 problem is similar to DTLZ9 and originated in the same work [64]. Uncertainty is added to the decision variables. The fraction of the decision variables that is uncertain is described by a constant value c. The formulation is provided in Eq. (A.17). There are d variables, d uncertain parameters, I objective functions, and I constraints.

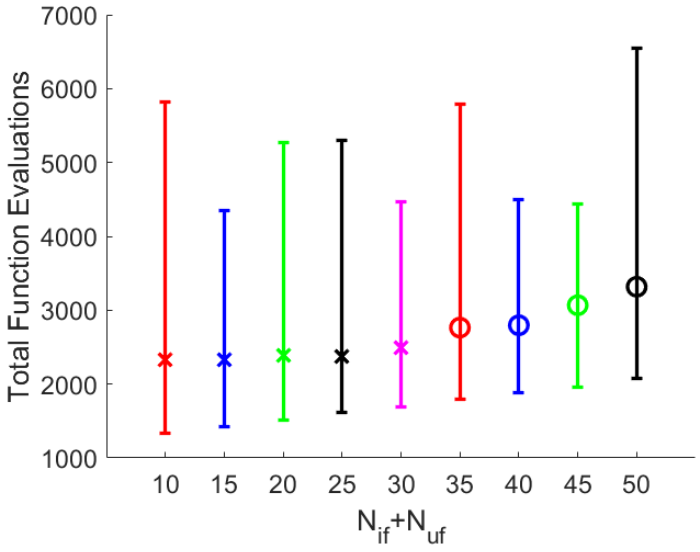
$$
\begin{aligned}
&\min_{\boldsymbol{x}} f_i(\boldsymbol{x}) = \frac{1}{\left\lfloor \frac{d}{I} \right\rfloor} \sum_{n=\left\lfloor (i-1)\frac{d}{I} \right\rfloor}^{\left\lfloor i\frac{d}{I} \right\rfloor} x_i, \qquad i \in 1, \dots, I \\
&s.t.\ g_i(\boldsymbol{x}, \boldsymbol{u}) = 1 - f_I(\bar{\boldsymbol{x}}) - 4f_i(\bar{\boldsymbol{x}}) \leq 0, \quad i \in 1, \dots, (I-1) \\
&\quad g_I(\boldsymbol{x}, \boldsymbol{u}) = 1 - 2f_I(\bar{\boldsymbol{x}}) - \min_{\substack{n \neq i \\ n,i \in [1,I-1]}} [f_n(\bar{\boldsymbol{x}}) + f_i(\bar{\boldsymbol{x}})] \leq 0 \\
&\quad u_i \in [-c, c], x_i \in [0 + c, 1 - c] \\
&\quad \bar{x}_i = x_i + u_i
\end{aligned}
\tag{A.17}
$$

Appendix B: Surrogate Model Selection

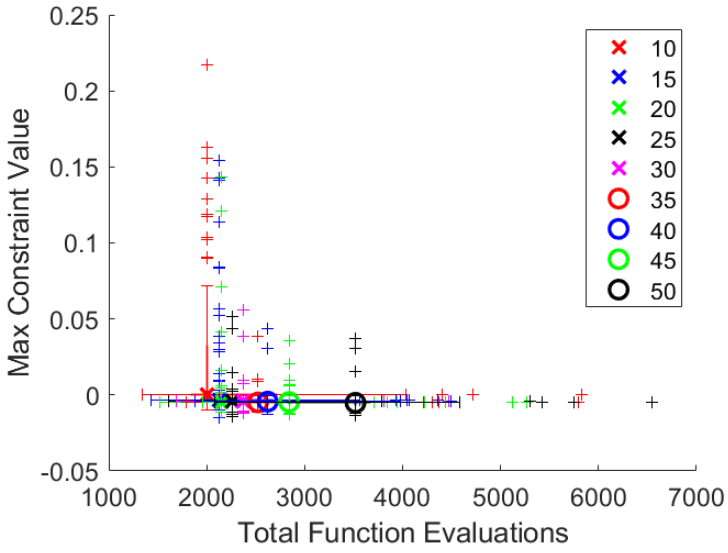
Some of the decisions behind the proposed SFTC-RO were informed through empirical testing. Namely, the budget parameters, N_{uf} and N_{if}, the number of initial samples and the number of infill samples respectively, and the choice of surrogate model were selected based on the results of a comparison study among multiple alternatives which will be discussed in this section. First, the budget parameters, N_{uf} and N_{if}, were determined by comparing the results of applying ten different values for the parameters and selecting the best performing value. Testing was done for pairs of $(N_{uf}, N_{if}) =$ $(5,5), (7,8), (10,10), (12,13), (15,15), (17,18), (20,20), (22,23), (25,25)$. These parameter pairs were evaluated by solving Example 2 and comparing the total function evaluations for 10 independent runs, and the results are shown in Figure B.1(a). It can be seen from this figure that the total number of function evaluations is largely insensitive to these parameters. However, there is a slight jump in function evaluations after $N_{uf} + N_{if} = 30$. To see the impact this has on the solution, Figure B.1(b) shows the maximum constraint values for the verification. All of the results except for $N_{uf} + N_{if} = 10$ demonstrate low constraint values. The first experiment to have all four quartiles below zero (omitting outliers) is $N_{uf} + N_{if} = 30$. Because there is little improvement but increasing cost beyond this one, $N_{uf} + N_{if} = 30$ was selected for the examples. Based on the above results, it seems that this was a reasonable budget for all of the examples.

The choice of surrogate model was similarly determined through

experimentation. Four methods were examined. Two of the methods utilized MATLAB's integrated surrogate optimization functions surrogateopt and bayesopt. The other two used MATLAB surrogate functions fitrsvm and fitrgp to build SVR and GP models, respectively. These models were then optimized as was described in the method proposed in this article. Examples 1-4 were all performed 10 times using each of these methods and the results are shown in Figure B.2. SVR performed well in the first example, Figure B.2(a), poorly in the second example, Figure B.2(b), and could not be used in the third and fourth examples due to insufficient data at the selected function evaluation budget. This suggests that it is not the appropriate technique for problems of this type, where the goal is to limit the function evaluations. Of the other techniques, the GPs performed the best in terms of both constraint values and function evaluations for Examples 1,2, and 4, Figure B.1(a)(b)(d). Although for Example 3, the GPs had higher function calls on average, it performed significantly better in terms of constraint values, Figure B.2(c). Thus, the GP surrogate model was selected for the proposed SFTC-RO methodology.

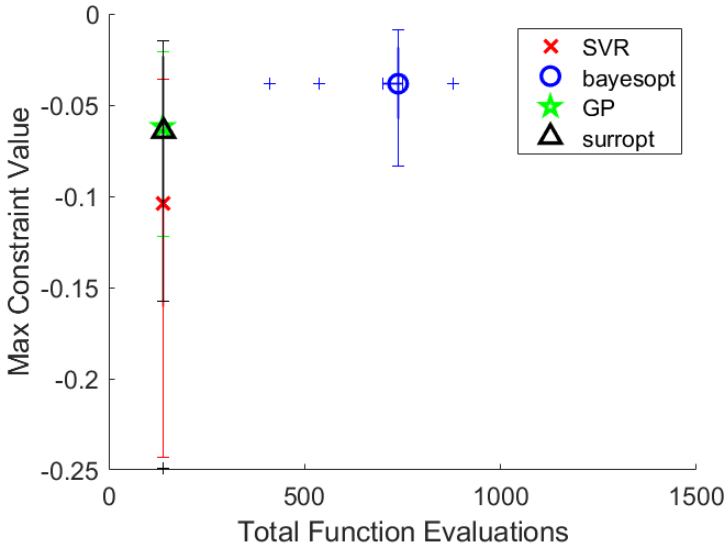


(a)

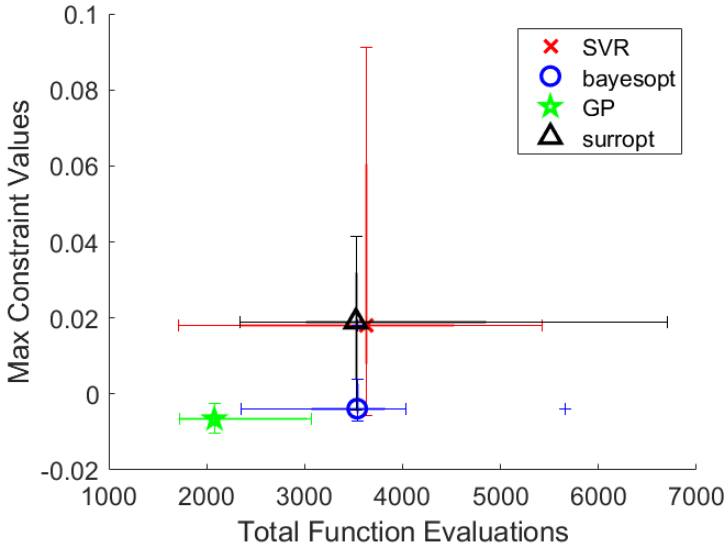


(b)

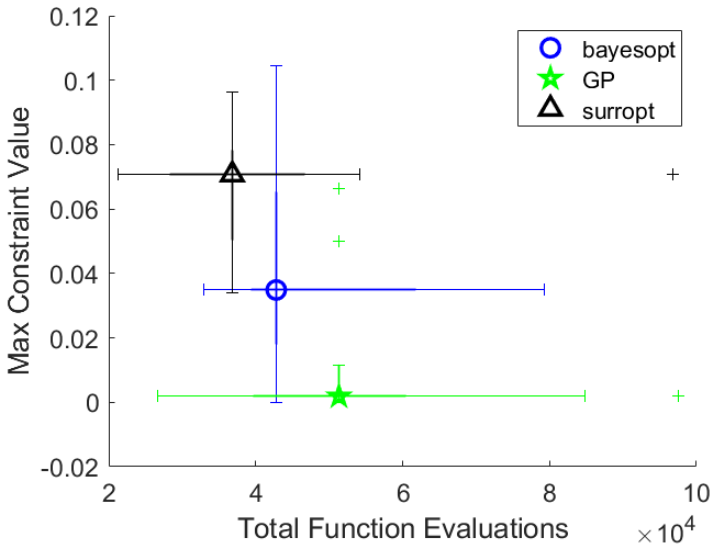
Figure B.1: Results of sensitivity analysis performed on Example 2, changing combined value of N_{if} and N_{uf} parameters with fixed 1: 1 ratio showing (a) the function evaluations for each parameter value, and (b) the maximum constraint value plotted against the total function evaluations


SVR
bayesopt
GP
surropt
Max Constraint Value
Total Function Evaluations
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
500
1000
1500

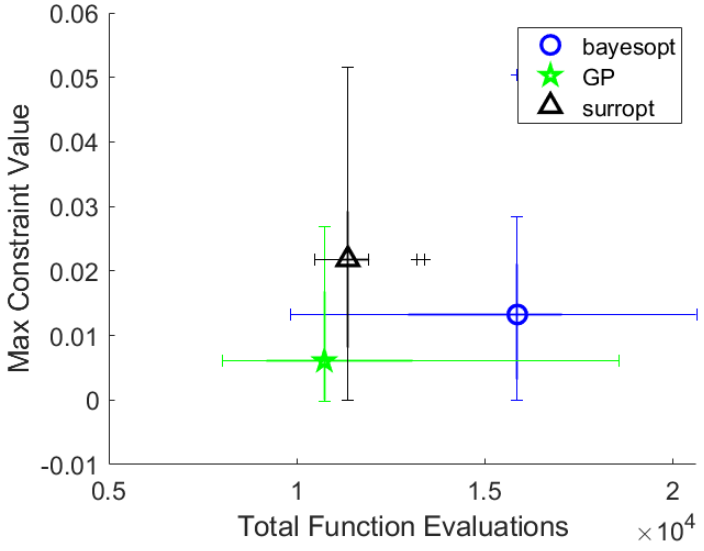
(a)


SVR
bayesopt
GP
surropt
Max Constraint Values
Total Function Evaluations
0.1
0.08
0.06
0.04
0.02
0
-0.02
1000
2000
3000
4000
5000
6000
7000

(b)



(c)



(d)

Figure B.2: Comparison of results for SVR, bayesopt, GP, and surrogateopt for (a) Example 2.1, (b) Example 2.2, (c) Example 2.3, and (d) Example 2.4

Bibliography

[1] M. Li, S. Azarm, and A. Boyars, "A New Deterministic Approach Using Sensitivity Region Measures for Multi-Objective Robust and Feasibility Robust Design Optimization," *Journal of Mechanical Design*, vol. 128, no. 4, p. 874, 2006, doi: 10.1115/1.2202884.

[2] T. Chatterjee, S. Chakraborty, and R. Chowdhury, "A Critical Review of Surrogate Assisted Robust Design Optimization," *Arch Computat Methods Eng*, vol. 26, no. 1, pp. 245–274, Jan. 2019, doi: 10.1007/s11831-017-9240-5.

[3] H. Rahimian and S. Mehrotra, "Distributionally Robust Optimization: A Review," *arXiv:1908.05659 [cs, math, stat]*, Aug. 2019, Accessed: Apr. 02, 2021. [Online]. Available: http://arxiv.org/abs/1908.05659

[4] S. Siddiqui, S. Azarm, and S. Gabriel, "A Modified Benders Decomposition Method for Efficient Robust Optimization under Interval Uncertainty," *Structural and Multidisciplinary Optimization*, vol. 44, no. 2, pp. 259–275, Aug. 2011, doi: 10.1007/s00158-011-0631-1.

[5] E. R. Panier and A. L. Tits, "A Globally Convergent Algorithm with Adaptively Refined Discretization for Semi-Infinite Optimization Problems Arising in Engineering Design," *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 903–908, Aug. 1989, doi: 10.1109/9.29441.

[6] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, "Robust Optimization." Princeton University Press, Princeton, NJ, 2009. doi: 10.1515/9781400831050.

[7] G. C. Calafiore, "Random Convex Programs," *SIAM Journal on Optimization*, vol. 20, no. 6, pp. 3427–3464, Jan. 2010, doi: 10.1137/090773490.

[8] M. C. Campi and S. Garatti, "The Exact Feasibility of Randomized Solutions of Uncertain Convex Programs," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1211–1230, Jan. 2008, doi: 10.1137/07069821X.

[9] D. Bertsimas and M. Sim, "The Price of Robustness," *Operations Research*, vol. 52, no. 1, pp. 35–53, Feb. 2004, doi: 10.1287/opre.1030.0065.

[10] E. Rudnick-Cohen, J. W. Herrmann, and S. Azarm, "Non-Convex Feasibility Robust Optimization Via Scenario Generation and Local Refinement," *Journal of Mechanical Design*, vol. 142, no. 5, p. 051703, May 2020, doi: 10.1115/1.4044918.

[11] J. Zhou, S. Cheng, and M. Li, "Sequential Quadratic Programming for Robust Optimization with Interval Uncertainty," *Journal of Mechanical Design*, vol. 134, no. 10, p. 100913, 2012, doi: 10.1115/1.4007392.

[12] A. Forrester, A. Sobester, and A. Keane, *Engineering Design Via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, 2008.

[13] Y. Liu and M. D. Collette, Eds., "Surrogate-assisted Robust Design Optimization Considering Interval-type Uncertainty," in *Maritime Technology*

and Engineering, 0 ed., CRC Press, 2014, pp. 287–293. doi: 10.1201/b17494-40.
[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Mass: MIT Press, 2006.
[15] R. M. Neal, "Bayesian Learning for Neural Networks," Springer New York, New York, NY, 1996. doi: 10.1007/978-1-4612-0745-0.
[16] D. Bernardini and A. Bemporad, "Scenario-Based Model Predictive Control of Stochastic Constrained Linear Systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Shanghai, Dec. 2009, pp. 6333–6338. doi: 10.1109/CDC.2009.5399917.
[17] G. C. Calafiore and M. C. Campi, "Uncertain Convex Programs: Randomized Solutions and Confidence Levels," *Mathematical Programming*, vol. 102, no. 1, pp. 25–46, Jan. 2005, doi: 10.1007/s10107-003-0499-y.
[18] M. Chamanbaz, F. Dabbene, R. Tempo, V. Venkataramanan, and Q.-G. Wang, "Sequential Randomized Algorithms for Convex Optimization in the Presence of Uncertainty," *IEEE Transactions on Automatic Control*, vol. 61, no. 9, pp. 2565–2571, Sep. 2016, doi: 10.1109/TAC.2015.2494875.
[19] X. Du and W. Chen, "Towards a Better Understanding of Modeling Feasibility Robustness in Engineering Design," *Journal of Mechanical Design*, vol. 122, no. 4, pp. 385–394, Jun. 1999, doi: 10.1115/1.1290247.
[20] M. Li, "Robust Optimization and Sensitivity Analysis with Multi-Objective Genetic Algorithms: Single-and Multi-Disciplinary Applications," Dissertation, University of Maryland, College Park, MD, 2007. Accessed: Apr. 07, 2016. [Online]. Available: https://www.proquest.com/pqdtglobal/docview/304845941/8AB7A64477F140B2PQ/1?accountid=14696
[21] W. Hu, M. Li, S. Azarm, and A. Almansoori, "Multi-Objective Robust Optimization under Interval Uncertainty Using Online Approximation and Constraint Cuts," *J. Mech. Des*, vol. 133, no. 6, pp. 061002–061002, Jun. 2011, doi: 10.1115/1.4003918.
[22] F. A. Ramponi, "Consistency of the Scenario Approach," *SIAM Journal on Optimization*, vol. 28, no. 1, pp. 135–162, Jan. 2018, doi: 10.1137/16M109819X.
[23] D. Bertsimas, O. Nohadani, and K. M. Teo, "Nonconvex Robust Optimization for Problems with Constraints," *INFORMS Journal on Computing*, vol. 22, no. 1, pp. 44–58, Feb. 2010, doi: 10.1287/ijoc.1090.0319.
[24] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998, doi: 10.1023/A:1008306431147.
[25] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian Optimization with Unknown Constraints," *arXiv:1403.5607 [cs, stat]*, Mar. 2014, Accessed: Dec. 18, 2018. [Online]. Available: http://arxiv.org/abs/1403.5607
[26] M. Schonlau, "Computer Experiments and Global Optimization," Dissertation, University of Waterloo, 1997.

[27] J. R. Gardner, M. J. Kusner, Z. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian Optimization with Inequality Constraints," in *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, Jun. 2014, vol. 32, pp. 937–945.
[28] M. Björkman and K. Holmström, "Global Optimization of Costly Nonconvex Functions Using Radial Basis Functions," *Optimization and Engineering*, vol. 1, no. 4, pp. 373–397, 2000.
[29] J. Mockus, V. Tiesis, and A. Zilinskas, "The Application of Bayesian Methods for Seeking the Extremum," *Towards Global Optimization*, vol. 2, pp. 117–129, 1978.
[30] T. Ray and W. Smith, "A Surrogate Assisted Parallel Multiobjective Evolutionary Algorithm for Robust Engineering Design," *Engineering Optimization*, vol. 38, no. 8, pp. 997–1011, Dec. 2006, doi: 10.1080/03052150600882538.
[31] R. J. Kania and S. Azarm, "Surrogate Feasibility Testing–Cutting for Single-Objective Robust Optimization under Interval Uncertainty," *Engineering Optimization*, pp. 1–17, Apr. 2022, doi: 10.1080/0305215X.2022.2052287.
[32] M. Li, R. Silva, and D. Lowther, "Global and Local Meta-Models for the Robust Design of Electrical Machines," *JAE*, vol. 51, no. s1, pp. S89–S95, Apr. 2016, doi: 10.3233/JAE-2013.
[33] K. Deb and H. Gupta, "Introducing Robustness in Multi-Objective Optimization," *Evolutionary Computation*, vol. 14, no. 4, pp. 463–494, Dec. 2006, doi: 10.1162/evco.2006.14.4.463.
[34] S. Daulton, S. Cakmak, M. Balandat, M. A. Osborne, E. Zhou, and E. Bakshy, "Robust Multi-Objective Bayesian Optimization under Input Noise," presented at the International Conference on Machine Learning, Baltimore, Maryland, USA, Jul. 2022. doi: https://doi.org/10.48550/arXiv.2202.07549.
[35] J. Zhang and A. A. Taflanidis, "Evolutionary Multi-Objective Optimization under Uncertainty through Adaptive Kriging in Augmented Input Space," *Journal of Mechanical Design*, vol. 142, no. 1, p. 011404, Jan. 2020, doi: 10.1115/1.4044005.
[36] M. H. S. Mendes, G. L. Soares, J.-L. Coulomb, and J. A. Vasconcelos, "A Surrogate Genetic Programming Based Model to Facilitate Robust Multi-Objective Optimization: A Case Study in Magnetostatics," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 2065–2068, May 2013, doi: 10.1109/TMAG.2013.2238615.
[37] B. Xia, Z. Ren, and C.-S. Koh, "Utilizing Kriging Surrogate Models for Multi-Objective Robust Optimization of Electromagnetic Devices," *IEEE Transactions on Magnetics*, vol. 50, no. 2, pp. 693–696, Feb. 2014, doi: 10.1109/TMAG.2013.2284925.
[38] T. Xie, P. Jiang, Q. Zhou, L. Shu, Y. Zhang, X. Meng, and H. Wei, "Advanced Multi-Objective Robust Optimization under Interval Uncertainty Using Kriging Model and Support Vector Machine," *Journal of Computing and Information Science in Engineering*, vol. 18, no. 4, p. 041012, Dec. 2018, doi: 10.1115/1.4040710.
[39] H. Xia, P. Jia, and L. Ma, "Robust Multi-Objective Optimization for Gas Turbine Operation Based on Kriging Surrogate Model," in *2021 40th Chinese*

Control Conference (CCC), Shanghai, China, Jul. 2021, pp. 6704–6704. doi: 10.23919/CCC52363.2021.9550112.
[40] Q. Zhou, P. Jiang, X. Huang, F. Zhang, and T. Zhou, "A Multi-Objective Robust Optimization Approach Based on Gaussian Process Model," *Struct Multidisc Optim*, vol. 57, no. 1, pp. 213–233, Jan. 2018, doi: 10.1007/s00158-017-1746-9.
[41] J. Wang, X. Shi, and M. Qu, "Evolutionary Optimization of Robust Multi-objective Expensive Problems Assisted by Incremental Kriging Models," in *2021 China Automation Congress (CAC)*, Beijing, China, Oct. 2021, pp. 4536–4541. doi: 10.1109/CAC53003.2021.9728656.
[42] J. N. Siddall, *Optimal Engineering Design: Principles and Applications*. CRC Press, 1982.
[43] B. J. Bichon, M. S. Eldred, S. Mahadevan, and J. M. McFarland, "Efficient Global Surrogate Modeling for Reliability-Based Design Optimization," *Journal of Mechanical Design*, vol. 135, no. 1, p. 011009, Jan. 2013, doi: 10.1115/1.4022999.
[44] M. Li, G. Li, and S. Azarm, "A Kriging Metamodel Assisted Multi-Objective Genetic Algorithm for Design Optimization," *Journal of Mechanical Design*, vol. 130, no. 3, p. 031401, 2008, doi: 10.1115/1.2829879.
[45] J. Knowles, "ParEGO A Hybrid Algorithm with Online Landscape Approximation for Expensive Multiobjective Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.
[46] M. T. M. Emmerich, A. H. Deutz, and J. W. Klinkenberg, "Hypervolume-Based Expected Improvement: Monotonicity Properties and Exact Computation," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, USA, Jun. 2011, pp. 2147–2154. doi: 10.1109/CEC.2011.5949880.
[47] L. Shu, P. Jiang, X. Shao, and Y. Wang, "A New Multi-Objective Bayesian Optimization Formulation with the Acquisition Function for Convergence and Diversity," *Journal of Mechanical Design*, vol. 142, no. 9, p. 091703, Sep. 2020, doi: 10.1115/1.4046508.
[48] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser, "On Expected-Improvement Criteria for Model-based Multi-objective Optimization," in *Parallel Problem Solving from Nature, PPSN XI*, R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 718–727. doi: 10.1007/978-3-642-15844-5_72.
[49] A. J. Keane, "Statistical Improvement Criteria for Use in Multiobjective Design Optimization," *AIAA Journal*, vol. 44, no. 4, pp. 879–891, Apr. 2006, doi: 10.2514/1.16875.
[50] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective Optimization on a Limited Budget of Evaluations Using Model-Assisted S-Metric Selection," p. 11.
[51] W. Liu, Q. Zhang, E. Tsang, C. Liu, and B. Virginas, "On the Performance of Metamodel Assisted MOEA/D," in *Advances in Computation and Intelligence*, vol. 4683, L. Kang, Y. Liu, and S. Zeng, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 547–557. doi: 10.1007/978-3-540-74581-5_60.
[52] Shinkyu Jeong and S. Obayashi, "Efficient Global Optimization (EGO) for Multi-Objective Problem and Data Mining," in *2005 IEEE Congress on*

Evolutionary Computation, Edinburgh, Scotland, UK, 2005, vol. 3, pp. 2138–2145. doi: 10.1109/CEC.2005.1554959.
[53] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BOTORCH: A Framework for Efficient Monte-Carlo Bayesian Optimization," Vancouver, Canada, Dec. 2020, p. 15.
[54] I. Couckuyt, D. Deschrijver, and T. Dhaene, "Fast Calculation of Multiobjective Probability of Improvement and Expected Improvement Criteria for Pareto Optimization," *J Glob Optim*, vol. 60, no. 3, pp. 575–594, Nov. 2014, doi: 10.1007/s10898-013-0118-2.
[55] R. Lacour, K. Klamroth, and C. M. Fonseca, "A Box Decomposition Algorithm to Compute the Hypervolume Indicator," *Computers & Operations Research*, vol. 79, pp. 347–360, Mar. 2017, doi: 10.1016/j.cor.2016.06.021.
[56] K. Yang, M. Emmerich, A. Deutz, and T. Bäck, "Efficient Computation of Expected Hypervolume Improvement Using Box Decomposition Algorithms," *J Glob Optim*, vol. 75, no. 1, pp. 3–34, Sep. 2019, doi: 10.1007/s10898-019-00798-7.
[57] L. While, L. Bradstreet, and L. Barone, "A Fast Way of Calculating Exact Hypervolumes," *IEEE Trans. Evol. Computat.*, vol. 16, no. 1, pp. 86–95, Feb. 2012, doi: 10.1109/TEVC.2010.2077298.
[58] W. Hu, "Approximation Assisted Multiobjective and Collaborative Robust Optimization under Interval Uncertainty," PhD Dissertation, University of Maryland, College Park, MD, 2012. Accessed: Apr. 07, 2016. [Online]. Available: http://drum.lib.umd.edu/handle/1903/13057
[59] "MATLAB (R2021b)." The Mathworks Inc., Natick, Massachusetts, 2022.
[60] K. M. Ragsdell and D. T. Phillips, "Optimal Design of a Class of Welded Structures Using Geometric Programming," *Journal of Engineering for Industry*, vol. 98, no. 3, pp. 1021–1025, Aug. 1976, doi: 10.1115/1.3438995.
[61] S. Gunawan, "Parameter Sensitivity Measures for Single Objective, Multi-Objective, and Feasibility Robust Design Optimization," Dissertation, University of Maryland, College Park, MD, 2004. Accessed: Aug. 18, 2015. [Online]. Available: http://drum.lib.umd.edu/handle/1903/1542
[62] E. Rudnick-Cohen, J. W. Herrmann, and S. Azarm, "Feasibility Robust Optimization Via Scenario Generation and Reduction," 2018.
[63] N. Williams, S. Azarm, and P. K. Kannan, "Engineering Product Design Optimization for Retail Channel Acceptance," *Journal of Mechanical Design*, vol. 130, no. 6, p. 061402, 2008, doi: 10.1115/1.2898874.
[64] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," in *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, A. Abraham, L. C. Jain, and R. Goldberg, Eds. New York: Springer, 2005, pp. 105–145.
[65] J. Golinski, "Optimal Synthesis Problems Solved by Means of Nonlinear Programming and Random Methods," *Journal of Mechanisms*, vol. 5, no. 3, pp. 287–309, Sep. 1970, doi: 10.1016/0022-2569(70)90064-9.
[66] R. Kania and S. Azarm, "Bi-Objective Surrogate Feasibility Robust Design Optimization Utilizing Expected Non-Dominated Improvement with

Relaxation," *Journal of Mechanical Design*, pp. 1–14, Sep. 2022, doi: 10.1115/1.4055738.
[67] F. Guerra Vázquez, J.-J. Rückmann, O. Stein, and G. Still, "Generalized Semi-Infinite Programming: A Tutorial," *Journal of Computational and Applied Mathematics*, vol. 217, no. 2, pp. 394–419, Aug. 2008, doi: 10.1016/j.cam.2007.02.012.
[68] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Mass: MIT Press, 2006.
[69] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and Analysis of Computer Experiments," *Statist. Sci.*, vol. 4, no. 4, pp. 409–423, Nov. 1989, doi: 10.1214/ss/1177012413.
[70] M. J. Sasena, "Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations," Dissertation, University of Michigan, 2002.
[71] W. Hu, M. Li, S. Azarm, S. Al Hashimi, A. Almansoori, and N. Al-Qasas, "Improving Multi-Objective Robust Optimization Under Interval Uncertainty Using Worst Possible Point Constraint Cuts," in *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2009, pp. 1193–1203. Accessed: Apr. 07, 2016. [Online]. Available: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1650056
[72] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino, "GA-Based Decision Support System for Multicriteria Optimization," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, Vancouver, BC, Canada, Oct. 1995, vol. 2, pp. 1556–1561. doi: 10.1109/ICSMC.1995.537993.
[73] R. J. Kania and S. Azarm, "Bi-Objective Surrogate Feasibility Robust Design Optimization Utilizing Expected Non-Dominated Improvement with Relaxation," presented at the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, St. Louis, MO, Aug. 2022.
[74] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, England: John Wiley & Sons, 2001.
[75] L. A. Schmit, "Structural Design by Systematic Synthesis," presented at the ASCE, 1960.
[76] J. Wu and S. Azarm, "Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set," *Journal of Mechanical Design*, vol. 123, no. 1, p. 18, 2001, doi: 10.1115/1.1329875.
[77] J. R. Schott, "Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization," Dissertation, Massachusetts Institute of Technology, Cambridge, Mass, 1995.
[78] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, Jun. 2000, doi: 10.1162/106365600568202.

[79] I. Couckuyt, D. Deschrijver, and T. Dhaene, “Towards Efficient Multiobjective Optimization: Multiobjective statistical criterions,” in *2012 IEEE Congress on Evolutionary Computation*, Brisbane, Australia, Jun. 2012, pp. 1–8. doi: 10.1109/CEC.2012.6256586.
[80] K. Dächert, K. Klamroth, R. Lacour, and D. Vanderpooten, “Efficient Computation of the Search Region in Multi-Objective Optimization,” *European Journal of Operational Research*, vol. 260, no. 3, pp. 841–855, Aug. 2017, doi: 10.1016/j.ejor.2016.05.029.
[81] G. Emch and A. Parkinson, “Robust Optimal Design for Worst-Case Tolerances,” *Journal of Mechanical Design*, vol. 116, no. 4, pp. 1019–1025, Dec. 1994, doi: 10.1115/1.2919482.
[82] I. Elishakoff, R. T. Haftka, and J. Fang, “Structural Design under Bounded Uncertainty—Optimization with Anti-Optimization,” *Computers & Structures*, vol. 53, no. 6, pp. 1401–1405, Dec. 1994, doi: 10.1016/0045-7949(94)90405-7.