

ABSTRACT

Title of Dissertation: **SCENARIO GENERATION-BASED
SINGLE- AND MULTI-OBJECTIVE
ROBUST OPTIMIZATION
UNDER FIXED AND VARIABLE
INTERVAL UNCERTAINTY**

Elizabeth Anne Jordan
Doctor of Philosophy, 2025

Dissertation Directed by: **Professor Shapour Azarm**
Department of Mechanical Engineering

Multi-objective robust optimization is a prominent field of optimization that deals with problems that have multiple, at least partially conflicting objectives, are subject to some constraints, and contain uncertainty. The main goal of these problems is to obtain solutions that are optimal and robust - i.e. relatively insensitive (have uncertainty robustness) to any realization of the uncertain parameters.

This dissertation presents three new methods for solving single- and multi-objective robust optimization (SORO and MORO) problems under fixed and variable uncertainty. The proposed methods share an underlying single-level iterative framework and handle the uncertainty robustness via a scenario generation technique.

The first method is the original sampling-based approach (OSB-MORO) which uses a multi-objective genetic algorithm to solve the optimization problem, and then handles the ro-

business aspect by sampling the fixed uncertainty space to find the worst-case realization and iteratively shrink the feasible domain to find the robust solution. This approach is very general in nature and can be applied to a variety of problems that have fixed interval uncertainty. However, sometimes the computational burden of solving MORO problems can be infeasible, or analyzing the trade-offs between the objectives can be challenging. To address this issue, MORO problems can be reformulated as single-objective problems - namely, via utility functions. The utility-based sampling-based MORO approach (USB-MORO) is an extension of the OSB-MORO approach, where the multiple objectives are converted to a single scalar via a multi-attribute utility function. This makes it possible to apply efficient single-objective optimization techniques, rather than the expensive multi-objective genetic algorithm solver as used by the OSB-MORO. Although the utility approach can help solve the MORO problem more efficiently, it does not solve the issue of dealing with expensive functions - i.e. functions that are based on computationally costly simulations. The second method (SSB-MORO) aims to improve the computational cost of the OSB-MORO approach by integrating an online surrogate model. A Kriging model is developed for all the functions of the problem, and is updated as the MORO algorithm progresses. The previous methods mentioned can only handle fixed interval uncertainties - i.e. the uncertainty bounds are known a priori and do not change. However, in many real-world applications the uncertainty may be dependent on the decision variables, hence the uncertainty bounds become variable. The third method (SB-SORO-DDU) aims to address this issue by extending the OSB-MORO approach to account for decision-dependent uncertainty for single-objective problems.

The key contributions in this dissertation are:

- 1) a single-level, sequential sampling-based single- and multi-objective robust optimization method for problems with fixed interval uncertainty (OSB-MORO);

- 2) a sampling-based multi-objective robust optimization method with integrated online surrogates for problems with fixed interval uncertainty (SSB-MORO); and
- 3) a sampling-based single-objective robust optimization method for problems with decision-dependent (or variable) interval uncertainty (SB-SORO-DDU).

SCENARIO GENERATION-BASED
SINGLE- AND MULTI-OBJECTIVE ROBUST OPTIMIZATION
UNDER FIXED AND VARIABLE INTERVAL UNCERTAINTY

by

Elizabeth Anne Jordan

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
2025

Advisory Committee:

Professor Shapour Azarm, Chair
Professor Vikrant Aute
Professor Yancy Diaz-Mercado
Professor Katrina Groth
Professor Johan Larsson
Professor P.K. Kannan, Dean's Representative

© Copyright by
Elizabeth Anne Jordan
2025

Foreword

This dissertation includes several works published by the author, Elizabeth Jordan, namely [1–3]. The attached letter certifies that the author has made a substantial contribution to the previously published works and that the inclusion of the works in this dissertation has been approved.



2181 Glenn L. Martin Hall
College Park, MD 20742-3035
TEL: 301-405-2410
<http://www.enme.umd.edu>

7/21/2025

Re: Previously Published Materials appearing in Thesis or Dissertation

Dean of the Graduate School,

Elizabeth Jordan ([REDACTED]) has:

☐ NOT INCLUDED any previously published works within their thesis or dissertation.

☒ INCLUDED one or more previously published works within their thesis or dissertation. This letter certifies that the examining committee for the student has determined that the student made a substantial contribution to the previously published work. The inclusion of the previously published work has the approval of the thesis or dissertation advisor and the Graduate Director.

Sincerely,

A handwritten signature in black ink, appearing to read "P. Sandborn".

Peter Sandborn
Director of Graduate Studies
Department of Mechanical Engineering
University of Maryland

Sincerely,

A handwritten signature in black ink, appearing to read "S. Azarm".

Shapour Azarm
Advisor for Elizabeth Jordan

Acknowledgments

First and foremost I would like to thank my advisor, Dr. Shapour Azarm, for his patience, guidance and support throughout my graduate studies. His commitment to doing methodical research and his attention to details have helped me become a better researcher. I would also like to thank my committee members, Dr. Vikrant Aute, Dr. Yancy Diaz-Mercado, Dr. Katrina Groth, Dr. Johan Larsson and Dr. P.K. Kannan, for agreeing to serve on my committee and for their time, effort and insights to improve this dissertation. I would like to acknowledge my fellow graduate students in the Design Decision Support Laboratory, for creating a supportive environment in the lab, for the insightful research discussions and friendly conversations. Finally, I would like to thank my family for encouraging me to pursue a higher level of education and for supporting me throughout my studies.

The work of this dissertation was supported in part by a generous gift from Dr. Alex Mehr (M.S., Ph.D. '03), Co-Founder and CEO of Retail ECommerce Ventures, famous.ai and deal.ai, through the "Design Decision Support Laboratory Research and Education Fund". This support is gratefully acknowledged.

This dissertation was partially supported by the Office of Naval Research (ONR) under grant number N000142212459. This support does not constitute an endorsement by the funding agency of the opinions expressed in the dissertation.

The author acknowledges the [University of Maryland supercomputing resources](#) made

available for conducting the research reported in this dissertation.

This dissertation utilizes some data provided by E.U. Copernicus Marine Service Information; relevant DOIs: <https://doi.org/10.48670/moi-00016>, <https://doi.org/10.48670/moi-00017>, <https://doi.org/10.48670/moi-00305>.

Table of Contents

Foreword	ii
Acknowledgements	iv
Table of Contents	vi
List of Tables	ix
List of Figures	x
List of Abbreviations	xii
List of Symbols	xiii
Chapter 1: Introduction	1
1.1 Motivation and Objective	1
1.2 Research Thrusts	4
1.3 Organization of Dissertation	7
Chapter 2: Terminology and Literature Review	9
2.1 Definitions	9
2.2 Position with Respect to Existing Literature	19
2.2.1 Multi-Objective Robust Optimization	19
2.2.2 Utility-Based Multi-Objective Optimization	22
2.2.3 Surrogate Modelling and Optimization	24
2.2.4 Single-Objective Robust Optimization with Decision-Dependent Uncertainty	26
2.3 Research Gaps and Dissertation Aim	28
Chapter 3: Original Sampling-Based Multi-Objective Robust Optimization (OSB-MORO)	31
3.1 Overview of Approach	32
3.1.1 Computational Effort	38
3.2 Examples	39
3.2.1 MinEx - Illustrative Example	41
3.2.2 TNK - Numerical Example	45
3.2.3 CTP - Numerical Example	47
3.2.4 Three Bar Truss - Engineering Example	48

3.2.5	ZDT3 - Scalable Example	50
3.2.6	USV Case Study	51
3.3	Utility-Based Sampling-Based Multi-Objective Robust Optimization (USB-MORO)	56
3.3.1	USB-MORO Problem Formulation and Approach	56
3.3.2	USV Case Study	58
3.4	Discussion	62
Chapter 4:	Surrogate Sampling-Based Multi-Objective Robust Optimization (SSB-MORO)	66
4.1	Overview of Approach	67
4.1.1	Computational Effort	76
4.2	Examples	77
4.2.1	TNK - Numerical Example	78
4.2.2	CTP - Numerical Example	80
4.2.3	Three-Bar Truss - Engineering Example	82
4.2.4	ZDT3 - Scalable Example	83
4.2.5	USV Case Study	85
4.3	Comparison with Existing Approaches	89
4.4	Discussion	94
Chapter 5:	Sampling-Based Single-Objective Robust Optimization with Decision-Dependent Uncertainties (SB-SORO-DDU)	96
5.1	Overview of Approach	98
5.1.1	Computational Effort	103
5.2	Examples	104
5.2.1	Simple Circle Problem	104
5.2.2	USV Case Study	108
5.3	Discussion	112
Chapter 6:	Conclusions	114
6.1	Summary	114
6.2	Contributions	117
6.3	Directions for Future Work	119
Appendix A:	Example Problem Formulations	122
A.1	MinEx	122
A.2	TNK	123
A.3	CTP	124
A.4	Three-Bar Truss	125
A.5	ZDT3	127
Appendix B:	USV Case Study	129
B.1	ECCS Model and Surrogate	130
B.2	Hydrodynamics Model and Surrogate	131
B.3	Reliability Model	133
B.4	Models Integration	137

B.5	Considering Environmental Conditions Based on Real Data	138
B.6	Utility Function Creation	140
	Bibliography	145

List of Tables

3.1	Summary of OSB-MORO Test Problems	40
3.2	MOGA and RO option settings for numerical and engineering example problems	40
3.3	TNK Result Comparison - OSB-MORO	46
3.4	CTP Result Comparison - OSB-MORO	48
3.5	Three-Bar Truss Result Comparison - OSB-MORO	49
3.6	ZDT3 Result Comparison - OSB-MORO	51
3.7	MOGA and RO option settings for USV case study	53
3.8	fmincon and RO option settings for USV case study (USB-MORO approach) .	60
4.1	Summary of Test Problems	77
4.2	MOGA and RO option settings for numerical and engineering example problems	78
4.3	TNK Result Comparison - SSB-MORO	80
4.4	CTP Result Comparison - SSB-MORO	81
4.5	Three-Bar Truss Result Comparison - SSB-MORO	83
4.6	ZDT3 Result Comparison - SSB-MORO	84
4.7	MOGA and RO option settings for USV case study	85
4.8	Example Results Comparison	92
4.9	ANOVA Results for Example Problems	93
5.1	Summary of results for Circle problem	107
5.2	fmincon and RO option settings for USV case study (SB-SORO-DDU approach)	110
5.3	Comparison of objective function values for SB-SORO-DDU and USB-MORO .	110
B.1	Error Metrics on the Training and Testing Dataset for the ECCS Surrogate Model	131
B.2	Error Metrics on the Training and Testing Dataset for the Hydro Surrogate Model	133

List of Figures

1.1	The Sea Hunter, an Unmanned Surface Vehicle (USV). US Navy Photo [4]	2
2.1	Hyperarea Difference	15
2.2	Overall Spread	16
2.3	Spacing	18
3.1	Flowchart of OSB-MORO approach	33
3.2	OSB-MORO computational complexity with respect to number of uncertainty samples ($\mathcal{O}(S)$) and population size ($\mathcal{O}(N)$)	39
3.3	MinEx problem solution for iteration #1	42
3.4	MinEx problem constraint violations for iteration #1	43
3.5	MinEx problem solution for iteration #2	44
3.6	MinEx problem constraint violations for iteration #2	45
3.7	TNK solution for OSB-MORO	46
3.8	CTP solution for OSB-MORO	47
3.9	Three-bar truss solution for OSB-MORO	49
3.10	ZDT3 solutions for $N = 5, 10, 20$ for OSB-MORO	50
3.11	USV Pareto Fronts for $\alpha = 1$ and $\alpha = 4$	54
3.12	USV trajectories and control values for $\alpha = 1$ and $\alpha = 4$	55
3.13	USV inputs for USB-MORO (deterministic and robust cases)	61
3.14	USV paths for USB-MORO (deterministic and robust cases)	61
3.15	USV attribute and utility profiles for USB-MORO (deterministic and robust cases)	62
4.1	Flowchart of SSB-MORO approach	68
4.2	TNK solution for SSB-MORO	79
4.3	CTP solution for SSB-MORO	81
4.4	Three-bar truss solution for SSB-MORO	83
4.5	ZDT3 solutions with $N=5,10,20$ - range of SSB-MORO solutions from 10 independent runs with all points, and with infeasible points removed.	84
4.6	Pareto Front for USV case study	86
4.7	Optimized throttle, heading, and USV speed operational profiles	87
4.8	Optimized USV paths for deterministic and robust cases. The color of the path corresponds to the USV speed.	87

4.9	Solutions for TNK, CTP, and Three-Bar Truss problems - range of SSB-MORO solutions from 10 independent runs with all points, and with infeasible points removed.	90
4.10	Comparison of quality metrics (mean and standard deviation) for TNK, CTP and Three Bar Truss problems - lower HD and SP is better, higher OS is better.	92
5.1	Flowchart of SB-SORO-DDU Approach	99
5.2	Evolution of uncertainty space for Circle problem	105
5.3	Progression of objective function value for Circle problem - for SB-SORO-DDU and OSB-MORO approaches	106
5.4	Uncertainty space for all iterations of the Circle problem	107
5.5	USV inputs for SB-SORO-DDU (deterministic and robust cases)	111
5.6	USV paths for SB-SORO-DDU (deterministic and robust cases)	111
5.7	USV attribute and utility profiles for SB-SORO-DDU (deterministic and robust cases)	112
A.1	Three-bar truss configuration	126
B.1	Engine Cooling and Control System Design Presented in [5]	130
B.2	Framework of the development of the of reliability model	134
B.3	Structure of the developed BBN model for system-level reliability analysis	135
B.4	Adjusted reliability (R_{adj}) for different values of α	136
B.5	Integration of ECCS, Hydro, and Reliability Models	138
B.6	Real Data for Wave Current Environment	142
B.7	Profiles for Wave Current Environment	142
B.8	Real Data for Wave Environment	143
B.9	Profiles for Wind Environment	143
B.10	Real Data for Wind Environment	144
B.11	Profile for Significant Wave Height	144

List of Abbreviations

AOVR	Acceptable Objective Variation Range
DDU	Decision-Dependent Uncertainty
DOE	Design of Experiments
DOF	Degrees of Freedom
EA	Evolutionary Algorithm
GA	Genetic Algorithm
HD	Hyperarea/hypervolume Difference
LHS	Latin Hypercube Sampling
MAUF	Multi-Attribute Utility Function
MOGA	Multi-Objective Genetic Algorithm
MORO	Multi-Objective Robust Optimization
MSE	Mean Squared Error
OS	Overall Spread
OSB-MORO	Original Sampling-Based Multi-Objective Robust Optimization
OSB-SORO	Original Sampling-Based Single-Objective Robust Optimization
RO	Robust Optimization
SB-MORO	Sampling-Based Multi-Objective Robust Optimization
SB-SORO-DDU	Sampling-Based Single-Objective Robust Optimization with Decision-Dependent Uncertainty
SORO	Single-Objective Robust Optimization
SSB-MORO	Surrogate Sampling-Based Multi-Objective Robust Optimization
USB-MORO	Utility-Based Sampling-Based Multi-Objective Robust Optimization
USV	Unmanned Surface Vessel

List of Symbols

$\mathbf{f}$	Vector of M objective functions, $[f_1, \dots, f_m, \dots, f_M]$
$\mathbf{g}$	Vector of J inequality constraint functions, $[g_1, \dots, g_j, \dots, g_J]$
j	Index of inequality constraint functions
k	Index of optimizer population points
m	Index of objective functions
n	Index of decision variables
p	Index of uncertain parameters
s	Index of uncertainty samples
$\mathbf{u}$	Vector of P uncertainty parameters, $[u_1, \dots, u_p, \dots, u_P]$
$\mathbf{x}$	Vector of N decision variables, $[x_1, \dots, x_n, \dots, x_N]$
$\mathbf{x}^*$	Vector of N optimized non-dominated solutions, $[x_1^*, \dots, x_n^*, \dots, x_N^*]$

Chapter 1: Introduction

This chapter presents the motivation and objective for pursuing the proposed research in multi-objective robust optimization. The chapter also poses the three research thrusts that this dissertation aims to address.

This chapter is organized as follows. Section 1.1 presents a motivational example, based on an unmanned surface vessel (USV) case study, illustrating the importance and pertinence of multi-objective robust optimization (MORO) techniques in the realm of engineering design and systems' operations. Section 1.2 introduces the research thrusts that define the scope of this dissertation and set the foundation for the proposed approaches presented in the subsequent chapters. Finally, Section 1.3 outlines the organization of this dissertation.

1.1 Motivation and Objective

Multi-objective robust optimization (MORO) deals with the optimization of engineering systems that have multiple objectives that are at least partly in conflict and constrained, while simultaneously accounting for uncertainties. MORO methods aim to find a set of robust Pareto solutions, which are not only optimal with respect to multiple objectives, but also are rather insensitive (i.e., robust or have robustness with respect) to the variations in uncertain parameters.

Many engineering problems nowadays are very complex and often require the development



Figure 1.1: The Sea Hunter, an Unmanned Surface Vehicle (USV). US Navy Photo [4]

of simulations to model the design or operation of the system in question. Additionally, optimized decisions about the system may need to be made while considering uncertain conditions, which can significantly impact the performance and feasibility of solutions. Consider a mission planning case study of an unmanned surface vessel (USV) (see Figure 1.1). The USV needs to reach a destination in a certain amount of time, while ensuring a healthy operational state and maintaining a desirable component and system reliability threshold. Additionally, the operating environment of the USV needs to be considered, e.g., the surrounding temperatures, pressures, wave conditions - all of which are uncertain and uncontrollable. Of course, to optimize, various simulation models can be developed to estimate the performance and evaluate design or decision alternatives of a USV and its operation. However, due to the complexity of the actual USV system, it is reasonable to assume that a thorough and reliable model would also be complex, thus very demanding in terms of simulation time and computational resources.

While the literature reports many methods for multi-objective optimization [6, 7] and robust optimization for single-objective problems [8, 9], there have been less developments in the intersection of these two fields - namely, MORO. Of the existing methods there are approximation-based methods [10–13], and methods that solve the original problem directly [10, 14, 15]. The

vast majority of methods reported in the literature implement a bi-level approach - either nested [16, 17] or sequential [10], which utilize either single- or multi-objective solvers [10, 17] at each level. With respect to uncertainty, most approaches consider problems with either interval [11, 14, 18, 19] or distributional uncertainty [20–22]. In both cases, however, the uncertainty is usually considered to be fixed - i.e. the interval bounds are known a priori and do not change, or the distributional parameters describing the uncertainty are known and fixed.

The main challenges and shortcomings of the existing methods are related to the computational intensity of MORO algorithms and their limited or specific applicability. The bi-level optimization approaches can become a burden on time and computational resources if the problem is too large, has a complicated formulation, or has computationally intensive simulations; additionally, these algorithms can be confusing or challenging to implement. Considering fixed uncertainty, although reasonable for many applications, may not lend itself to some problems involving path planning or systems operational planning, where the uncertainty is dependent on the decision variables. The versatility of the approach also needs to be considered - many methods state specific assumptions that need to be met; otherwise the approach is not applicable.

Based on the aforementioned challenges and shortcomings, the objective of this dissertation is to present a framework for a general sequential, single-level, scenario-generation MORO approach, that can lay the foundation for several fine-tuned methods. The approach is designed to be general in nature, simple to implement, and can be applied to a range of problems - constrained and unconstrained, linear and nonlinear, as well as optimization problems where the objectives and constraints are black-box models. Although all the examples presented in this dissertation feature problems with continuous decision variables and functions, the proposed approach is flexible to handle discrete variables and functions - the only thing that needs to be changed is

the solver function, to one that is designed to solve discrete problems. Similarly with the uncertainties, although the focus of this dissertation is considering interval uncertainty, distributional uncertainty can be handled as well - instead of sampling the uncertainty space from an interval range, the sampling should be performed from a distributional range.

1.2 Research Thrusts

Based on the objective and challenges mentioned in the previous section, the proposed research aims to address three research thrusts.

Research Thrust 1: General Sampling-Based Multi-Objective Robust Optimization Method using Original Problem Functions (OSB-MORO)

Many existing MORO methods follow a bi-level or two-stage workflow, requiring two optimization problems to be solved [10, 16, 17, 23–26], thus doubling the computational cost per solver iteration. In these formulations, the first-stage problem is a deterministic multi-objective optimization problem, which is solved to obtain a set of optimal solutions; the second-stage problem is single-objective and evaluates the robustness of all solutions obtained from the first step, with respect to the uncertain parameters [11]. In the proposed MORO approach, optimization is performed in a single stage, bypassing the need to solve another optimization problem for handling the uncertainty, by replacing it with a sampling strategy. The uncertainty is assumed to belong to an interval range, which is known and fixed (although the approach can be easily modified to consider distributional uncertainty).

Additionally, the proposed approach can be considered from a single-objective point of view. The utility-based sampling-based MORO (USB-MORO) method addresses this by using

a utility function to reformulate the multi-objective problem as a single-objective one, which is often done in the literature to scalarize the problem, include decision maker preferences, and apply standard single-objective optimization techniques [27–29].

However, one challenging aspect of the proposed scenario generation based strategy in this thrust is balancing accuracy and computational demand, ensuring that the uncertainty is sampled thoroughly but efficiently.

Thus, the objective of Research Thrust 1 is to develop a general sampling based MORO approach (OSB-MORO), that effectively integrates a single-level optimization workflow with a scenario generation scheme to handle the robustness.

Research Thrust 2: Online Surrogate Sampling-Based Multi-Objective Robust Optimization (SSB-MORO)

The computational challenges associated with expensive functions (such as USV’s hydro-dynamic simulations) are only intensified when coupled with MORO - this can be mitigated by developing online surrogate models. The online surrogate models approximate the true function and continuously update as new data becomes available during the optimization process. Thus, the proposed surrogate sampling-based MORO (SSB-MORO) approach both speeds up the optimization process and reduces the computational costs. Building surrogate models for expensive functions has been actively studied in the literature [30–32], and methods such as Gaussian processes, polynomial regression, neural networks, and other machine learning techniques have been utilized to create accurate surrogates. Although the approach in this thrust has the potential to significantly reduce the computational intensity of solving the MORO problem, acquiring a truly robust result is highly dependent on the accuracy of the surrogate model, which can be a challenging problem in itself. Additionally, integration of a surrogate development scheme with the

optimization algorithm can pose some challenges as well, such as data management (e.g., deciding when, how and which new data points to incorporate to update the surrogate) and integration issues between the optimizer and surrogate (e.g., ensuring consistent data structures across models, or smooth communication between software platforms/languages).

Thus, the objective of Research Thrust 2 is to integrate online surrogates with a scenario generation scheme for handling uncertainty to develop an efficient MORO approach.

Research Thrust 3: Sampling-Based Single-Objective Robust Optimization with Decision-Dependent Uncertainty (SB-SORO-DDU)

Traditional robust optimization assumes uncertainty is independent of the decision variables, however, in many real-world applications (such as path planning or systems operations) that is not the case. For example, in deciding the route for an USV, the uncertainty in ocean waves becomes dependent on a particular USV route decision, i.e. uncertainty is dependent on the decision for the location and direction of the USV. The main challenges in this area include the increased computational complexity and the difficulty of modelling and reformulating uncertainty sets that change with decision variables [33–35]. While most decision-dependent robust optimization methods focus on polyhedral or distributionally robust sets with tractable reformulations [34, 36–38], a sampling-based method may offer a simpler, more straight-forward approach, in addition to more flexibility for complex problems. However, as with any sampling-based approach, the computational burden may become a challenge, especially as the dimensionality of the problem and uncertainty parameters increases.

Thus, the objective of Research Thrust 3 is to develop a sampling-based SORO approach that can account for variable interval uncertainty, i.e. decision-dependent uncertainty.

These research questions collectively aim to develop a robust and efficient framework that

handles fixed and variable uncertainty effectively, and addresses the challenges posed by expensive function evaluations.

1.3 Organization of Dissertation

This dissertation is organized as follows. Chapter 2 will define the relevant terminology used throughout this dissertation and introduce some background and literature review on multi-objective robust optimization, surrogate modelling and optimization, and robust optimization with decision-dependent uncertainty. The next few chapters will each cover one of the research thrusts presented in Section 1.2 and demonstrate the proposed approaches on some numerical examples and an USV case study. Chapter 3 will present the original sampling-based multi-objective robust optimization (OSB-MORO) approach - a single-level, sequential MORO approach which handles the robustness aspect via a scenario generation scheme. Due to the general nature of the OSB-MORO method, it lays the foundation for the approaches in the subsequent chapters. Next, Section 3.3 introduces a reformulation of the MORO problem to a single-objective robust optimization (SORO) problem by means of a utility function. Thus, the OSB-MORO approach is adapted to a single-objective problem. Chapter 4 integrates an online surrogate with the OSB-MORO approach to improve the computational efficiency. Chapter 5 further pushes the bounds of the sampling-based approach by considering decision-dependent uncertainty, rather than uncertainty that is independent of the decision variables. Finally, Chapter 6 summarizes the approaches proposed in this dissertation and highlights their contributions to the field of multi-objective robust optimization. The dissertation concludes with a discussion of potential improvements and directions for future work.

The next chapter will define the terminology used throughout this dissertation and provide a literature review to position the presented research directions.

Chapter 2: Terminology and Literature Review

This chapter will provide a review of some of the main terminology that will be referenced in this dissertation and will also position the proposed work with respect to existing literature. Additionally, the following notation will be used for describing variables: *non-bold italic* notation represents a single *variable* (e.g., x_n); ***bold-faced italic*** notation represents a ***vector*** of variables (e.g., $\mathbf{x} = [x_1, x_2, \dots, x_n]$); **bold-faced** notation represents a **matrix** of variables (e.g., $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k]^T$, where $\mathbf{x}_k = [x_{k1}, x_{k2}, \dots, x_{kn}]$, and the superscript T in $[\dots]^T$ denotes the transpose of the matrix: $[\dots]$).

This chapter is organized as follows. Section 2.1 will define the terminology used throughout this dissertation as well as descriptions of quality metrics used to assess the results from the proposed approaches. Section 2.2 will provide a review of the foundational topics presented in this dissertation to position the proposed approaches with respect to existing literature. Finally, Section 2.3 will discuss the research gaps identified from the literature review, which subsequently guided the development of the proposed approaches.

2.1 Definitions

In this section, some key definitions are provided.

General Formulation of Multi-Objective Robust Optimization Problem

The general formulation [17] of a constrained multi-objective robust optimization (MORO) problem is presented in Equation (2.1) below:

$$\begin{aligned} & \min_{\mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ & \text{subject to:} \\ & g_j(\mathbf{x}, \mathbf{u}) \leq 0; j = 1, \dots, J \\ & \mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}] \\ & \forall \mathbf{u} \in [\mathbf{u}_{lb}, \mathbf{u}_{ub}] \end{aligned} \tag{2.1}$$

It is assumed that the objective functions, $\mathbf{f} = [f_1, \dots, f_M]$, are to be minimized with respect to the decision variables, $\mathbf{x} = [x_1, \dots, x_N]$, and are at least partially in conflict with each other. The problem is constrained by J inequality constraints, $g_j(\mathbf{x}, \mathbf{u})$. The uncertainties, $\mathbf{u}$, can represent either uncertain parameters, or uncertainty in the decision variables, and are represented as an interval range. As shown in Eq. (2.1), the uncertainties can appear in the constraints as well as the objective functions.

Objective and Feasibility Robust Optimization

The uncertainty parameters in robust optimization (RO) problems can show up in the objective function, constraints, or both – and are referred to as objective robustness, feasibility robustness, and objective-feasibility robustness, respectively [13, 16].

The problem formulation for feasibility robustness is shown in Equation (2.2). Here, the uncertainties appear only in the constraints, and are handled according to the techniques used to solve the RO problem.

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to:

$$g_j(\mathbf{x}, \mathbf{u}) \leq 0; j = 1, \dots, J \quad (2.2)$$

$$\mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}]$$

$$\forall \mathbf{u} \in [\mathbf{u}_{lb}, \mathbf{u}_{ub}]$$

Objective robustness is handled by adding another constraint (Equation (2.3)), which limits the change in the objective function to be within an acceptable objective variation range (AOVR) [39], which is determined by the decision maker based on preference or problem specifics. As shown in the formulation in Equation (2.4), the change in objective function value is defined as follows:

$$\Delta f = f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}, \mathbf{u}_0) \quad (2.3)$$

In the first term the objective function is evaluated at the current values of the decision variables, $\mathbf{x}$, and current realization of the uncertainty parameters, $\mathbf{u}$. The second term evaluates the objective function using the nominal values for the uncertainty parameters, $\mathbf{u}_0$, instead. Note, the Δf term is signed, so it can be positive or negative. However, only the positive case matters, since the AOVR constraint in Equation (2.4) must be satisfied (and if Δf is negative, the constraint is satisfied automatically). Additionally, since the goal is to minimize the objectives, it is undesirable that the variation in any objective function increases its value beyond the AOVR.

$$\begin{aligned}
& \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{u}) \\
& \text{subject to:} \\
& g_j(\mathbf{x}) \leq 0; j = 1, \dots, J \\
& f_m(\mathbf{x}, \mathbf{u}) - f_m(\mathbf{x}, \mathbf{u}_0) \leq AOV R_m; m = 1, \dots, M \\
& \mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}] \\
& \forall \mathbf{u} \in [\mathbf{u}_{lb}, \mathbf{u}_{ub}]
\end{aligned} \tag{2.4}$$

Finally, as the name implies, objective-feasibility robustness considers uncertainty in both the objectives and constraints. The formulation is provided in Equation (2.5) below.

$$\begin{aligned}
& \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{u}) \\
& \text{subject to:} \\
& g_j(\mathbf{x}, \mathbf{u}) \leq 0; j = 1, \dots, J \\
& f_m(\mathbf{x}, \mathbf{u}) - f_m(\mathbf{x}, \mathbf{u}_0) \leq AOV R_m; m = 1, \dots, M \\
& \mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}] \\
& \forall \mathbf{u} \in [\mathbf{u}_{lb}, \mathbf{u}_{ub}]
\end{aligned} \tag{2.5}$$

Note, that in both objective robustness and objective-feasibility robustness, the AOV constraint is treated as an additional constraint. Hence when the constraints are evaluated for all realizations of the uncertain parameters to determine feasibility, the AOV constraint is included.

Inferiority, Non-inferiority, and Dominance

In multi-objective optimization, the concepts of dominance and inferiority are used to identify the optimal solutions to a problem [6, 40]. Consider two feasible design points, $\mathbf{x}_a$ and $\mathbf{x}_b$ - then $\mathbf{x}_a$ is said to be *inferior* with respect to $\mathbf{x}_b$ if, for a minimization problem, $f_m(\mathbf{x}_b) \leq f_m(\mathbf{x}_a), \forall m = 1, \dots, M$ and $f_m(\mathbf{x}_b) < f_m(\mathbf{x}_a)$ for at least one m . Correspondingly, $\mathbf{x}_b$ *dominates* $\mathbf{x}_a$. If $\mathbf{x}_a$ does not dominate and is not inferior to $\mathbf{x}_b$, then the two points are *non-inferior* with respect to each other.

Non-Dominated Set

Consider a set of K solutions. If a pair-wise comparison is performed for each solution with respect to all other solutions, one can find which solution dominates which and which solutions are non-dominated with respect to each other. In this way, one can find a new set (K') that is the *non-dominated* set - in which any two solutions do not dominate each other and there is no other solution in the set K that dominates this non-dominated set [6].

Pareto Optimality, Pareto Solutions/Set and Pareto Front/Frontier

The non-dominated set derived from the entire search space is the *Pareto-optimal* set. The points, $\mathbf{x}^*$, which belong to this set are the *Pareto solutions*. The set of points formed by these solutions is the *Pareto front or frontier* [6].

True and Observed Solutions

The *true solution* refers to the Pareto-optimal solution. The *observed solution* is the non-dominated set of solutions obtained by the optimization solver.

Deterministic Solution

The *deterministic solution* refers to the non-dominated set of solutions obtained by the optimization solver without considering uncertainty - i.e. the uncertain parameters are set to their

nominal (or assumed) values, u_0 .

Robust Solution

The *robust solution* refers to the non-dominated set of solutions obtained by the optimization solver *considering* uncertainty - i.e. the robust solution is optimal and feasible for all realizations of the uncertainty parameters.

Ideal (Good) and Nadir (Bad) Points

The Ideal and Nadir points (or their estimates: “good” and “bad” points, respectively), represent the best and worst possible values for each objective among all Pareto optimal solutions, respectively [6]. So, assuming all objectives are being minimized, the ideal point would be a vector $[f_1^*, \dots, f_m^*, \dots, f_M^*]$, where f_m^* is the minimum value for the m -th objective and the Nadir point would be a vector $[f_1^N, \dots, f_m^N, \dots, f_M^N]$, where f_m^N is the maximum value for the m -th objective. These points are often used to normalize the objective values for comparing solutions and evaluating their performance. The normalization would be performed following Equation (2.6):

$$\bar{f}_m = \frac{f_m - f_g}{f_b - f_g} \quad (2.6)$$

where $\bar{f}_m$ is the normalized m -th objective value, f_m is the original m -th objective value, and f_g and f_b are the “good” and “bad” points, respectively.

Hyperarea/Hypervolume Difference (HD)

The HD is a closeness metric proposed by Wu and Azarm [41], i.e. it evaluates how close the observed solutions are to the ideal (good) point. Assuming the objectives are normalized with respect to the “good” and “bad” points (Equation (2.6)), the HD can be calculated using Equation (2.7):

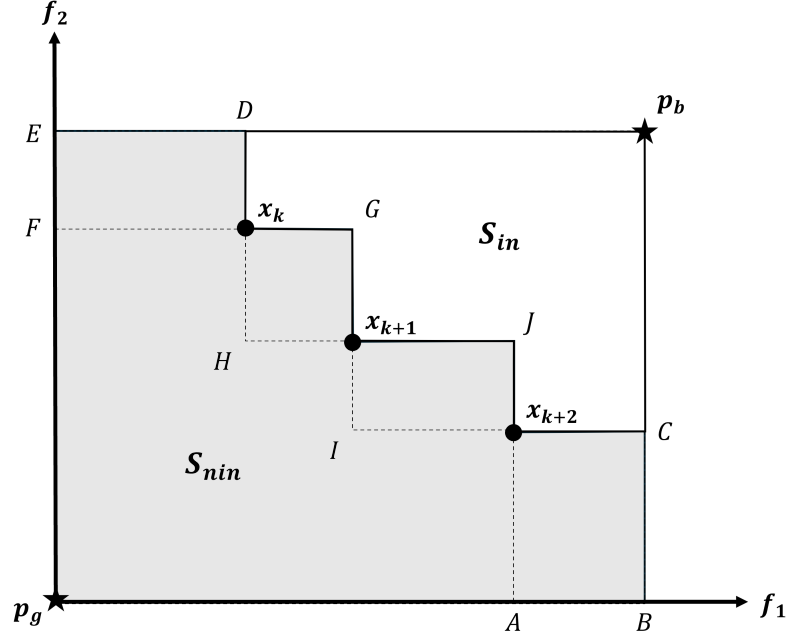


Figure 2.1: Hyperarea Difference

$$HD = 1 - \text{space}(S_{in}) = \text{space}(S_{nin}) \quad (2.7)$$

The term “*space*” refers to the area (or volume in a three-objective space) of a region, and S_{in} and S_{nin} refer to the sets of inferior and non-inferior regions, respectively.

Figure 2.1 illustrates a set of Pareto points, $K' = (x_k, x_{k+1}, x_{k+2})$ and its associated good (p_g) and bad (p_b) points. For a minimization problem, the inferior region of the Pareto set, S_{in} , is defined by the connected points $p_b C x_{k+2} J x_{k+1} G x_k D p_b$, and the non-inferior region of the Pareto set, S_{nin} , is defined by the connected points $p_g E D x_k G x_{k+1} J x_{k+2} C B p_g$. So, visually, the HD corresponds to the grey shaded area in Figure 2.1. Since HD is a measure of closeness to the ideal (good) point, a smaller value is desired.

Overall Spread (OS)

Another metric proposed by Wu and Azarm [41] is the OS, the overall spread metric, i.e.

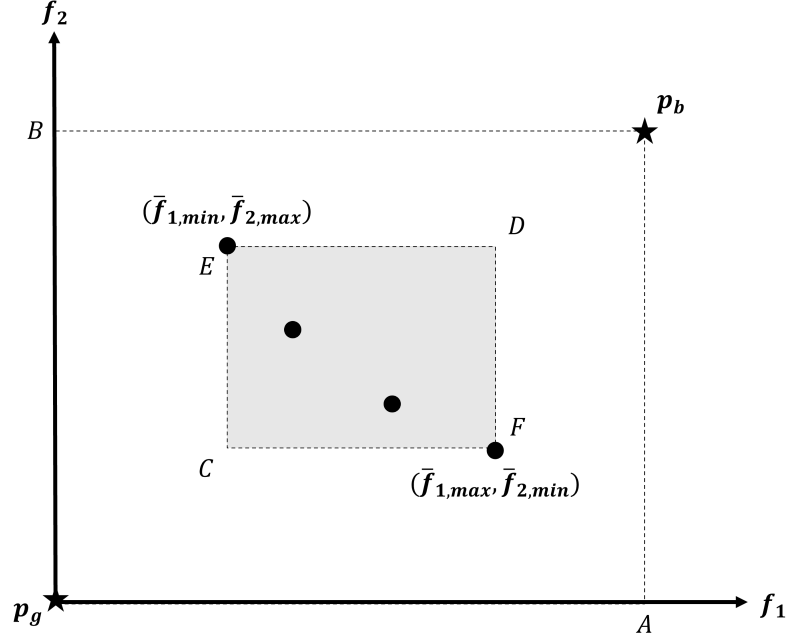


Figure 2.2: Overall Spread

it evaluates how diverse or spread out the observed solution is compared to the ideal/ (good) and nadir (bad) points. Assuming the objectives are normalized with respect to the “good” and “bad” points (Equation (2.6)), the OS can be calculated following Equation (2.8):

$$OS = \frac{HR_{ex}}{HR_{gb}} \quad (2.8)$$

For instance, for a bi-objective case, the OS is defined as a ratio of the areas of two rectangles (HR), with one defined by the extreme points of the observed Pareto set (HR_{ex}) and the other defined by the “good” and “bad” points (HR_{gb}). In the case that there are more than two objectives, the OS definition is extended to higher dimensions - i.e. instead of dealing with areas, we now have 3D or higher-dimensional *volumes*. Thus, the OS is defined as the ratio of the volumes of two hyper-volumes (one defined by the extreme points of observed Pareto solutions, the other defined by the “good” and “bad” points).

Figure 2.2 presents a visual representation of the elements defining the OS in a two-dimensional space. The black points represent the observed Pareto set (K'), normalized with respect to the good and bad points (p_g and p_b). The extreme points of the observed Pareto set are labelled as E and F and are equal to the normalized extreme objective function values - $(\bar{f}_{1,min}, \bar{f}_{2,max})$ and $(\bar{f}_{1,max}, \bar{f}_{2,min})$, respectively. The shaded area defined by those points ($EDFC$) corresponds to the term HR_{ex} in Equation (2.8). The hyper-rectangle defined by the good and bad points, HR_{gb} is represented by the rectangle $p_g B p_b A$. So, visually, the OS is the ratio of the areas of the shaded and white rectangles. Since OS is a measure of diversity, a larger OS is better.

Spacing (SP)

The spacing metric (SP) [42] is a distance metric that quantifies the relative distances between the nearest solutions in the observed Pareto set. The SP can be calculated using Equation (2.9):

$$SP = \sqrt{\frac{1}{|K'| - 1} \sum_{i=1}^{|K'|} (d_i - \bar{d})^2} \quad (2.9)$$

where

$$d_i = \min_{x_j \in K' \cap x_j \neq x_i} \sum_{m=1}^M |f_m(x_i) - f_m(x_j)|$$

In Equation (2.9), $|K'|$ is the cardinality of the observed non-dominated solution set, K' ; d_i is the minimum Manhattan distance (or L_1 norm) between the solution x_i and the remaining solutions, x_j ; $\bar{d}$ is the mean of d_i ; M is the number of objective functions; and $f_m(x_i)$, $f_m(x_j)$ are the m^{th} objective function values for the solutions x_i and x_j , respectively.

Figure 2.3 illustrates the concept of the SP metric, by showing the minimum Manhattan

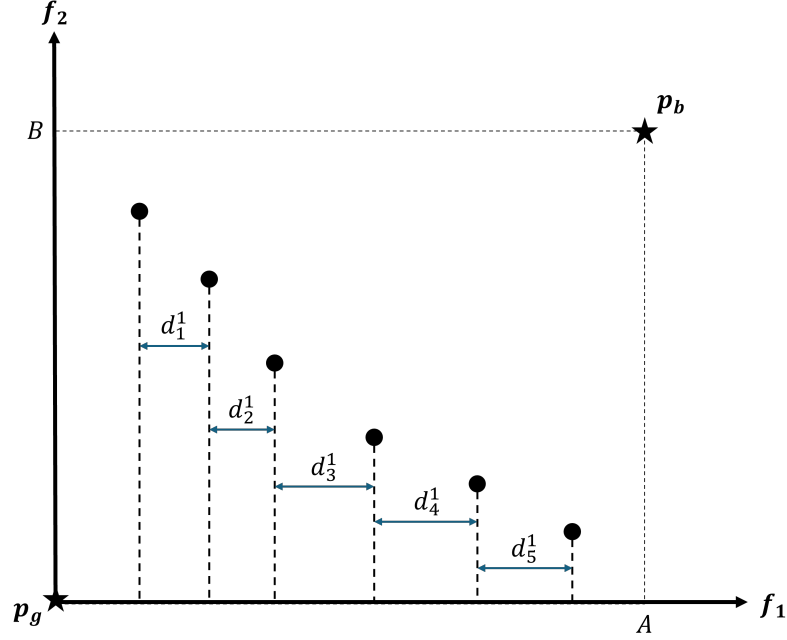


Figure 2.3: Spacing

distances, d , between each point of the obtained solution. So, the SP is essentially the standard deviation of those distances.

The SP metric measures the uniformity of the solutions obtained, so a smaller value for the SP represents a more evenly distributed set, which is preferable.

Function Calls

One way of measuring the computational efficiency of an algorithm is by reporting the number of function calls needed to converge to a solution. In this dissertation, since both surrogate-based and non-surrogate-based approaches are presented, the number of function calls is the same as the number of sample points. In the context of non-surrogate-based approaches, a sample point is a unique point that is passed through the optimizer, where a unique point consists of a set of decision variables and uncertain parameters - $[\mathbf{x}, \mathbf{u}]$. A function call is counted only when the point is evaluated using the original problem function or simulation - surrogate calls do

not contribute to the function count.

2.2 Position with Respect to Existing Literature

Multi-objective robust optimization (MORO) is a prominent field in optimization that seeks to address the complexities of real-world engineering problems involving multiple conflicting objectives and inherent uncertainties. The primary goal of MORO is to identify solutions that are optimal with respect to multiple objectives and maintain robustness against variations in the problem due to uncertainties.

The goal of this section is to position the proposed research directions (as posed in Section 1.2) with respect to the existing literature and review the three key relevant areas - namely, MORO approaches (Section 2.2.1) and the role of utility theory in the context of MORO (Section 2.2.2), online surrogate models for MORO (Section 2.2.3), and SORO with decision-dependent uncertainty (Section 2.2.4).

2.2.1 Multi-Objective Robust Optimization

The literature on robust optimization presents various methods for finding solutions that are both optimal and robust ("insensitive") to uncertainty [10, 13, 17, 18, 43–46].

Most MORO algorithms are bi-level - the upper-level problem (main problem) seeks to optimize the objectives with respect to the decision variables, while the lower-level problem (sub-problem) handles the uncertainty, given the solution from the upper level [11, 17, 39, 43, 47, 48]. The bi-level algorithms require two optimization problems to be solved, one for each level, and the algorithm terminates once the solutions from both levels converge. The two main strategies

for bi-level MORO are nested [11, 49, 50] and sequential approaches [11, 51, 52]. In nested bi-level MORO, the upper-level optimization problem is solved to search the design space. For each intermediate, or candidate, design solution found in the upper-level problem, the lower-level problem is solved to determine whether that point is robust. This process repeats iteratively until convergence is achieved. Generally, nested approaches offer better convergence properties but can be more computationally expensive, compared to sequential approaches. Sequential bi-level MORO, on the other hand, decouples the upper- and lower-level problems and solves them sequentially within each iteration. The upper-level problem is solved first, then its solution is used to solve the lower-level problem. Thus, only the optimal solutions (not the intermediate ones) are evaluated for robustness. The sequential approach is computationally less expensive than its nested counterpart, and may converge faster, but can be prone to premature convergence and may not guarantee optimality together with robustness of solutions.

Most MORO methods reported in the literature utilize either evolutionary algorithms (EAs) or decomposition based methods, due to their ability to obtain globally optimum solutions. Evolutionary algorithms (EAs) [6, 53–55] are widely used due to their ability to handle complex, nonlinear, and potentially discontinuous functions in problems. Examples include NSGA-II [56], which uses a fast non-dominated sorting approach and a crowding distance mechanism to maintain solution diversity; and SPEA2 (Strength Pareto Evolutionary Algorithm) [57], featuring a fine-grained fitness assignment strategy and a density estimation technique. Decomposition-based methods, as the name suggests, decompose the multi-objective problem into simpler subproblems which are solved independently. Some examples include MOEA/D [58] decomposes a multi-objective problem into several single-objective subproblems and optimizes them simultaneously; the weighted sum method [6] converts the problem into a single-objective one by assigning

weights to each objective; and the ϵ -constraint method [6] optimizes one objective while treating others as constraints with specified allowable thresholds.

In terms of handling the uncertainty and robustness aspect of the problem, the literature reports several methods as well, including stochastic methods (such as stochastic or chance-constrained programming) [20, 59, 60], implementation of constraint cuts and worst-case analysis [10, 46, 61], and scenario generation approaches [2, 9, 62], to name a few. In these cases, the uncertain parameters are typically described either as interval [14, 52, 61, 63–65], or distributional [66–71] uncertainties. Stochastic methods [59, 60, 72, 73] account for uncertainties by considering their probabilistic distribution and aim to find a solution that would remain feasible and optimal for any realization of the uncertainty parameter. Stochastic optimization methods [13] incorporate randomness directly into the optimization model, solving for solutions that perform well on average or with a certain probability. The uncertainty is typically modelled through random variables and the expected value is optimized, while chance-constrained programming [60] ensures constraints are satisfied with a specified probability, providing a probabilistic guarantee of feasibility. In the implementation of the worst-case analysis approaches [10, 16, 74], constraint cuts are utilized to iteratively refine the feasible solution space based on the worst-case scenario of uncertain parameters. The worst-case scenario is obtained by solving the lower-level problem of the bi-level MORO problem, which results in additional computational costs. On a similar note, min-max optimization algorithms [15, 74] seek to minimize the maximum possible loss, ensuring robustness against the worst-case scenario. In scenario-generation based approaches [2, 9, 62, 75, 76], the uncertainty is handled in a more straightforward way - a set of scenarios representing different realizations of uncertain parameters is generated to capture the range of possible outcomes. Then these scenarios are evaluated with respect to the candi-

date solutions and to the robustness against uncertainty, ensuring the solution remains feasible and optimal across all scenarios. However, most approaches still utilize another, lower-level optimization problem to refine the local solution, or to help select a new worst-case uncertainty sample.

In the MORO methods reviewed from the literature, bi-level optimization approaches have been implemented, which is computationally expensive. Moreover, all the methods make some assumptions about the problems and have some limitations or restrictions, for example on the uncertainty type. To address these drawbacks, a scenario generation based MORO algorithm is proposed.

Although a sampling-based technique offers simplicity and flexibility in handling the uncertainties, it may not scale well as the problem size increases and may become computationally expensive. There are two ways to address this challenge - one is to convert the problem to a single-objective formulation via a utility function (which will allow the use of a standard optimization method which is more efficient); the other is to develop surrogate-based optimization methods (which use surrogate (or meta-) models to replace the computationally expensive function evaluations, allowing faster optimization and uncertainty analysis).

2.2.2 Utility-Based Multi-Objective Optimization

Combining utility theory with mathematical programming has shown several advantages in the context of MORO. Firstly, a utility function provides a structured way to model and incorporate a decision maker's preferences, which can be an efficient way of aggregating multiple objectives into a single measure [29, 77, 78]. Secondly, using a utility-based function converts

the problem to a single-objective formulation, which makes it possible to apply efficient single-objective optimization techniques [27, 29, 79, 80].

The development of a utility function usually starts with creating a single-attribute utility function. A single-attribute utility function is used when an attribute can be evaluated based on a single criterion or performance measure, such as cost. The utility function $U(x)$ assigns a numerical value, i.e., utility, to each possible value of the attribute, x . The utility value is typically normalized on a scale range from 0 to 1, or from -1 to 0, where a higher utility value usually indicates a preferred outcome [81]. When there are multiple attributes, a multi-attribute utility function (MAUF) is created. A MAUF extends the concept of single-attribute utility by aggregating the utilities of each attribute into an overall utility value, as shown in Equation (2.10):

$$U(x_1, x_2, \dots, x_n) = f(u_1(x_1), u_2(x_2), \dots, u_n(x_n)) \quad (2.10)$$

where x_i represents the value of the i -th attribute; and $u_i(x_i)$ represents the single-attribute utility function of the i -th attribute.

When the attributes are mutually preferentially independent, an additive utility function is commonly used, as shown in Equation (2.11):

$$U(x_1, x_2, \dots, x_n) = \sum_{i=1}^n k_i u_i(x_i) \quad (2.11)$$

where k_i are the weights that reflect the importance of each attribute and $\sum_{i=1}^n k_i = 1$.

In cases when attributes are not mutually independent but exhibit some interactions, a multiplicative utility function is more appropriate, which can be written as shown in Equation (2.12):

$$kU(x_1, x_2, \dots, x_n) + 1 = \prod_{i=1}^n [kk_i u_i(x_i) + 1] \quad (2.12)$$

where k is an additional scaling constant determined by the decision maker.

Note, in utility theory, attributes are usually equivalent to objective functions - i.e., the utility function is a function of the objective functions, $U = f(f_1, f_2, \dots, f_m)$, where f_m is the m -th objective function of the optimization problem. In this dissertation, however, it is assumed that the attributes are equivalent to the decision variables, thus the utility function is a function of decision variables - $U = f(x_1, x_2, \dots, x_n)$, where x_n is the n -th decision variable.

Although the utility-based approach has its benefits, it does require a well-defined utility function, which may be challenging to develop in practice [80]. Moreover, we are not solving the *original* multi-objective problem when using the utility-based approach, since the objectives have been combined into one. Depending on what utility function is used, this can skew the final results.

2.2.3 Surrogate Modelling and Optimization

Surrogate modelling is a powerful tool in MORO to alleviate the computational burden of black-box or expensive functions, such as simulations. By constructing approximations of the objective functions and constraints, surrogate models allow for rapid evaluation of design alternatives.

There are two main methods for implementing surrogates with MORO - offline and on-line [82]. The offline surrogate approach builds a surrogate for the problem offline, prior to any optimization. Then the MORO algorithm is applied to the surrogate problem. The advantage of

this approach is that it is easy to implement and does not require specific integration with the MORO algorithm. However, the ease of integration comes at a cost - the surrogate has to be built for the entire design space, since the optimum is not known a priori, thus requiring more data (for exploration of design space) to develop an accurate surrogate, which may become expensive and even infeasible. Online surrogates address this limitation by adaptively updating the surrogate as the MORO algorithm is running. This provides the advantage for the surrogate to learn about the problem and sample points that are more promising, thus utilizing the concepts of exploration and exploitation to effectively improve the surrogate accuracy as the optimum approaches [64].

Some examples of surrogate modelling techniques in MORO include Kriging [11, 32, 83–85], radial basis functions [32, 86], and polynomial regression [30, 87].

Kriging, also known as Gaussian process regression, has become a popular surrogate modelling technique in MORO due to its ability to provide both predictions and uncertainty estimates, which is valuable for balancing exploration and exploitation in adaptive sampling strategies and surrogate development. Additionally, Kriging is favoured for its flexibility and ability to handle complex, nonlinear relationships between inputs and outputs with relatively few sample points [64, 88]. Knowles [89] proposed the ParEGO (Pareto Efficient Global Optimization) algorithm, which uses Kriging models and the expected improvement criterion to efficiently solve multi-objective optimization problems. This work laid the foundation for many subsequent studies in Kriging-based MORO ([61, 64, 85, 90]).

With the advent of machine learning and neural networks, many surrogate models began adopting those techniques, such as radial basis functions (RBFs) and support vector machines (SVMs), just to name a few. RBFs were introduced by Broomhead and Lowe [91] as a technique for interpolation in high-dimensional spaces. In the field of MORO, RBFs gained popularity

due to their flexibility, ability to handle nonlinear relationships, and computational efficiency (e.g. black-box simulations), although most of the literature focuses on classical multi-objective optimization ([92–94]). Similar to other machine learning techniques, SVMs are capable of handling high-dimensional spaces and non-linear relationships, but they are also less prone to the curse of dimensionality, can handle noisy data better, and are able to efficiently find the most critical points in the sampling set, while requiring relatively few training samples [95–97].

As modern problems become more complex, developing surrogate models becomes essential - the significant reduction in computational cost makes it feasible to tackle these problems. However, as with all approaches, surrogate modelling does not come without its limitations. The accuracy of the surrogate models has to be high, since an imperfect model can fail to capture the true nature of the problem, leading to potential errors in the optimization process and suboptimal or non-robust solutions. Additionally, constructing and updating surrogate models for multiple objectives and uncertainty scenarios can itself be computationally demanding, especially for large-scale problems. Finally, the performance of these methods heavily depends on the choice of surrogate model and supporting functions (e.g., acquisition, basis, or kernel functions) and relevant hyperparameters, which may require extensive tuning and validation.

2.2.4 Single-Objective Robust Optimization with Decision-Dependent Uncertainty

Traditionally, in robust optimization (RO) the uncertainty is assumed to be fixed and independent of the decision variables. However, in many real-world applications the uncertainty is influenced by the decision variables. This is known as endogenous, or decision-dependent

uncertainty (DDU) [37, 98]. Accounting for DDU, rather than assuming a uniform uncertainty across the entire design space, can allow for a more realistic modelling of uncertainty, reduce conservatism and can improve solution quality [34, 37, 99]. However, when DDUs are considered, traditional algorithms for solving RO problems with fixed uncertainty become inapplicable. This is because the problem structure and assumptions are altered - the previously selected worst-case scenarios may fall outside the bounds of the uncertainty set for the new decision variables, causing traditional algorithms to fail to converge [100].

The literature reports several approaches that have been developed to address DDU in SORO problems. In many works, the uncertainty sets are constructed as polyhedral sets or sets parameterized by binary decision variables [33, 37, 101]. This allows using linear programming, or mixed-integer linear programming techniques to solve the optimization problem. Two-stage and multistage robust optimization frameworks have been extended to handle DDU as well. In these models, the uncertainty set is explicitly linked to decision variables obtained from the first-stage optimization problem. Solution methods include iterative algorithms based on Benders decomposition and column-and-constraint generation [34, 100, 102]. Adjustable robust optimization frameworks, introduced by Ben-Tal [103], have also been extended to DDU problems. Decision rule approaches, which use parameterized policies for recourse actions, enable tractable reformulations and efficient solution of problems with both continuous and binary recourse decisions [99, 104].

While current research has provided some methods for solving robust problems with DDU, many challenges still remain. For one, a general-purpose, tractable framework has yet to be developed. Many current approaches and reformulations have strict assumptions or are problem-specific, thus limiting their application to a broader range of problems. Computational complexity

and scalability is another issue. Many RO problems with DDU make the problem non-convex or NP-hard, making them computationally intractable for larger-scale problems.

2.3 Research Gaps and Dissertation Aim

While there have been many advances in the field of MORO, especially in relation to improving the efficiency of MORO algorithms (both computational efficiency and efficiency in finding a robust solution) there are still some avenues to explore.

In the field of MORO, there is a significant reliance on bi-level optimization approaches. These approaches, although effective, can be computationally intensive, time consuming, and in some cases challenging to implement. Additionally, most existing methods assume certain problem characteristics (e.g., continuous, differentiable functions in optimization problems) and uncertainty types (e.g., following a normal distribution), which limits their applicability in more diverse and realistic scenarios. So, one of the primary research gaps lies in the need for more efficient or simpler algorithms that can handle the complexities of MORO problems without the extensive computational burden. And there is a need for more generalized approaches that can accommodate various types of problems and forms of uncertainty, without sacrificing the robustness of solutions.

Although surrogate models have become essential for approximating expensive functions and reducing computational costs (especially with the advent of machine learning techniques), some research gaps persist, specifically in the context of MORO. Generally, developing a surrogate model assumes a constant or fixed underlying problem, which does not change across iterations, allowing the surrogate to adapt and learn accordingly. Even in these cases, ensuring

high accuracy of surrogate models is challenging - there are a lot of dependencies that need to be considered and tuned, such as the choice of surrogate model, supporting functions for the surrogate, hyperparameter values, and sampling strategy for updating the surrogate model. However, in some MORO approaches, the underlying problem slightly changes every iteration (e.g., a new infeasible constraint is added) - the literature has yet to address the challenge of developing surrogate models that can dynamically adapt to the evolving MORO problem landscape, without compromising the accuracy and computational demand.

Another area of RO that requires further research is problems with decision-dependent uncertainty. Traditional RO methods fall short, and while several problem-specific reformulations exist, there are very few general-purpose and tractable frameworks for solving RO problems with DDU.

In view of the challenges and issues that have been uncovered in some of the current MORO methods reported in the literature, this dissertation aims to address the gaps described in this section. Thus, the key contributions of this dissertation are:

1. A general sampling-based multi-objective robust optimization method (OSB-MORO), which follows a single-level, iterative approach. Reducing the MORO algorithm to solving only one optimization problem per iteration and handling the uncertainty in a sampling based manner can help alleviate the computational burden and mitigate the limitations of other MORO approaches that rely on specific problem assumptions or characteristics. Moreover, due to its general nature, this approach provides a framework for solving MORO problems utilizing a scenario generation technique to handle the robustness. The framework and approach can be further modified to adapt to a specific problem.

2. An online surrogate sampling-based MORO (SSB-MORO) approach, which integrates a Kriging model to improve the computational effort of the OSB-MORO approach
3. A sampling-based single-objective robust optimization with decision-dependent uncertainty (SB-SORO-DDU) approach, which presents a simple and flexible way of handling endogenous uncertainty without significantly complicating the problem. This approach presents a tractable framework, that does not require a reformulation of the problem.

The next three chapters will dive into the details of the proposed approaches to address some of the research gaps discussed. The next chapter will describe the original sampling-based multi-objective robust optimization (OSB-MORO) approach, which constitutes the foundation of all the subsequent approaches.

Chapter 3: Original Sampling-Based Multi-Objective Robust Optimization (OSB-MORO) ¹

This chapter will focus on describing the original sampling-based multi-objective robust optimization (OSB-MORO) approach [2] for solving the problem formulated by Equation (3.1):

$$\begin{aligned} & \min_{\mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ & \text{subject to:} \\ & g_j(\mathbf{x}, \mathbf{u}) \leq 0; \quad j = 1, \dots, J \\ & f_m(\mathbf{x}, \mathbf{u}) - f_m(\mathbf{x}, \mathbf{u}_0) \leq AOV R_m; \quad m = 1, \dots, M \\ & \mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}] \\ & \forall \mathbf{u} \in [\mathbf{u}_0 + \Delta \mathbf{u}] \\ & \Delta \mathbf{u} \in [\Delta \mathbf{u}_{lb}, \Delta \mathbf{u}_{ub}] \end{aligned} \tag{3.1}$$

The formulation in Equation (3.1) assumes objective-feasibility robustness. However, if uncertainty is only present in the constraints, then the $AOVR$ constraint is removed. Likewise, if only objective robustness is considered, then the uncertainty, $\mathbf{u}$, is removed from the constraints $g_j(\mathbf{x})$.

¹A version of this chapter was presented at the ASME IMECE-2024 conference [1] and further revised and published in the ASME Journal of Computing and Information Science in Engineering [2]

The OSB-MORO algorithm will be described in detail in Section 3.1, and its performance will be demonstrated on a few examples (Section 3.2) and a USV case study (Section 3.2.6). Section 3.3 will introduce a reformulation of the MORO problem as a single-objective problem via a multi-attribute utility function (MAUF) and apply the proposed OSB-MORO approach to the USV case study with a MAUF formulation (Section 3.3.2). Section 3.4 will conclude this chapter with a summary and discussion of some of the drawback and limitations of the baseline OSB-MORO approach.

3.1 Overview of Approach

The OSB-MORO approach is a single-level, sequential method, based on a scenario generation scheme created by sampling uncertain parameters. A single-objective version of the approach was reported elsewhere [9]. The proposed method follows an iterative approach, where the optimization problem is first solved to get a preliminary solution, and then the uncertainty is handled by a sampling technique, using a fixed set of uncertainty samples.

A high-level flowchart of the proposed approach is shown in Figure 3.1, and its corresponding pseudocode is provided in Algorithm 1.

In the initialization step, all the uncertain parameters are set to their nominal values or assumed with initial values, $\mathbf{u}_0$. Then, the optimization problem is solved to obtain a set of non-dominated solutions, $\mathbf{x}^*$. Note, that for the first iteration, since the uncertain parameters are assumed to be taken at their nominal values, the solutions, $\mathbf{x}^*$, are the deterministic solutions to the optimization problem. Next, a number of new samples (or scenarios), $\mathbf{u}$, are generated. In this case, a Latin-Hypercube sampling (LHS) approach [105] is implemented for a fixed number

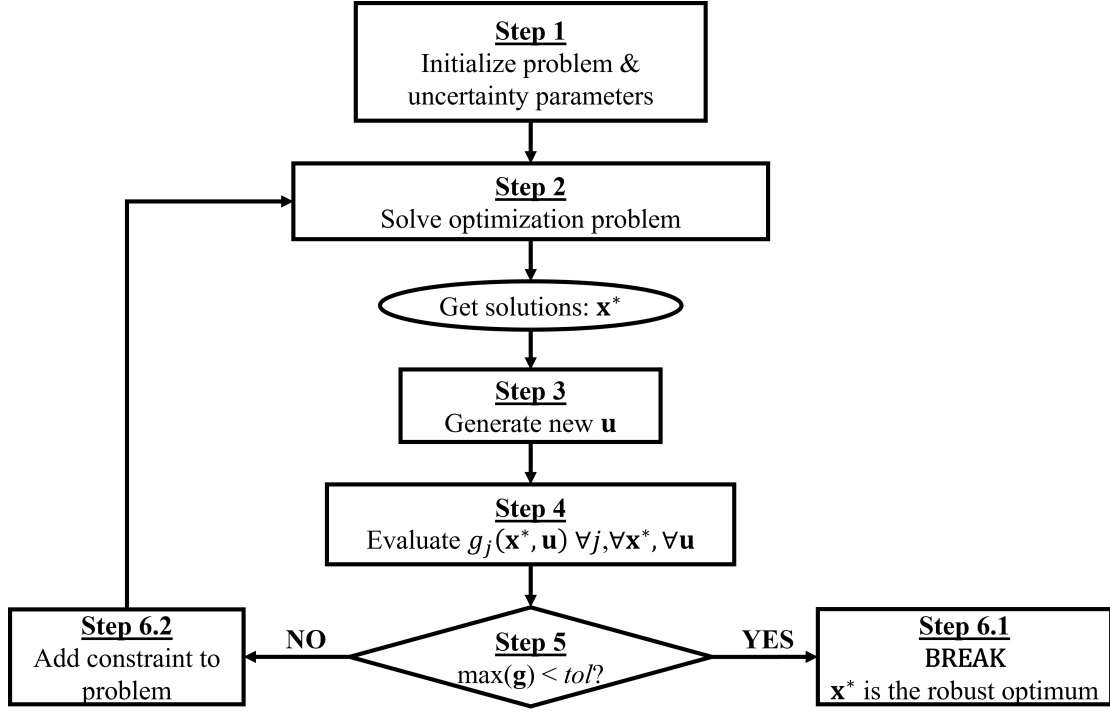


Figure 3.1: Flowchart of OSB-MORO approach

of samples. Then, all the constraints are evaluated for all non-dominated points, $\mathbf{x}^*$, for each realization of $\mathbf{u}$, and a feasibility check is performed. The constraint with the maximum violation is added to the optimization problem. At this point, as single MORO iteration is complete. The algorithm continues, so, for the next iteration, the problem is resolved (now with the additional constraint), a new set of non-dominated solutions is obtained, and the procedure is repeated. The algorithm stops once all the constraints are satisfied at the scenario generation stage – i.e., at this point no realization of $\mathbf{u}$ has violated any of the constraints for the current $\mathbf{x}^*$. Thus, the solutions $\mathbf{x}^*$ found in the current iteration are returned as the robust optimal solutions.

In further discussion of the details of the approach, **Step X** will refer to the corresponding step in the flowchart (Figure 3.1) and Line A.Y will refer to the line number Y of the pseudocode for Algorithm A (defined in Algorithm 1).

Algorithm 1: OSB-MORO algorithm

```
Input: problemDef, ROptions
Output: [detsoln, rosoln, allsoln]
/* Step 1 - Initialization */
1 uinfeas  $\leftarrow$  [];
2 coninfeas  $\leftarrow$  [];
3 u  $\leftarrow$  u0;
4 for iter = 1 to maxIter do
    /* Step 2 - Solve optimization problem */
    5 if iter = 1 then
    6     [x*, f]  $\leftarrow$  solverFctn(problemDef, u0, uinfeas, coninfeas) ;
    7     detsoln  $\leftarrow$  [x*, f] ;
    8 else
    9     [x*, f]  $\leftarrow$  solverFctn(problemDef, u0, uinfeas, coninfeas) ;
    10    allsoln(iter)  $\leftarrow$  [x*, f] ;
    /* Step 3 - Generate S uncertainty samples */
    11 unew  $\leftarrow$  generateNewU(problemDef, S) ;
    /* Step 4 - Evaluate constraints */
    12 g  $\leftarrow$  [] ;
    13 for sample = 1 to S do
    14     u  $\leftarrow$  unew(sample) ;
    15     g  $\leftarrow$  evaluateConstraints(problemDef, x*, u) ;
    /* Step 5 - Check constraint feasibility */
    16 if max(g) < tol then
    17     break ;
    /* Step 6 - Add worst-case constraint to problem */
    18 else
    19     [uinfeas, coninfeas]  $\leftarrow$  findMaxViolation(g, unew) ;
    20     problemDef.new_constr  $\leftarrow$  [uinfeas, coninfeas]
21 rosoln  $\leftarrow$  [x*, f] ;
```

Step 1: Initialization and Problem definition

In this step, the problem is defined and the options for running the MORO algorithm are set.

The input to the algorithm, *problemDef*, contains all the information related to the problem definition - namely, the formulation of the objective and constraint functions, the lower and upper bounds for the decision variables, the robustness type of the problem (e.g., feasibility or objective robustness), and the AOVR values (if applicable). Additionally, the uncertainty parameters

are defined - their nominal values and variation bounds. Finally, the problem solver options are defined - these are dependent on the solver algorithm selected (e.g., genetic algorithm or pareto search), the platform used to solve the problem (e.g., MATLAB or Python), and of course the nature of the problem itself, since different problems may require different solver option settings for best performance.

The other input to the algorithm, *ROptions*, allows the user to specify details of the MORO algorithm - e.g., the maximum number of iterations, *maxIter* ([Line 1.4](#)), to loop through before terminating the algorithm and reporting the final (robust) solution, how many samples, *S* ([Line 1.11](#)), to generate for the robustness evaluation, and the tolerance, *tol* ([Line 1.16](#)), for acceptable constraint violation.

[Line 1.1](#): $\mathbf{u}_{infeas}$ is the set of uncertainty parameters that led to a maximum constraint violation.

[Line 1.2](#): The set $\mathbf{con}_{infeas}$ keeps track of which constraints were violated.

[Line 1.3](#): For the first iteration all uncertainties are set to their nominal values. Additionally, since for the first iteration there are no additional infeasible constraints added to the problem yet, the reported solution is the deterministic solution to the problem.

Step 2: Solve optimization problem

As the name implies, during this step the optimization problem is solved.

[Lines 1.5-1.10](#): The `solverFctn` solves the optimization problem, with the uncertainties set to their nominal values, $\mathbf{u}_0$. The optimized decision variables (the non-dominated set), $\mathbf{x}^*$, as well as the associated optimal objective function values, $\mathbf{f}$, are returned. For this dissertation, MATLAB's multi-objective genetic algorithm (MOGA) solver, `gamultiobj`, was implemented [106], which is based on a non-dominated sorting algorithm (NSGA-II) [6]. Note, that for the first iteration, the deterministic solution is returned, since the uncertain parameters are set

to their nominal values and no infeasible uncertainties or constraints are present yet. Subsequent iterations return intermediate solutions, till the algorithm is terminated.

Step 3: Generate uncertainty samples

In this step, a fixed number of uncertainty samples are generated (where a sample is a single realization for each uncertainty parameter of the problem).

Line 1.11: The `generateNewU` function generates a set of uncertainty samples, using the Latin-Hypercube sampling method. The uncertainty space to sample from is defined by the nominal uncertainty value, $\mathbf{u}_0$, and the uncertainty variation range, $\Delta\mathbf{u}$, i.e. $\mathbf{u} \in [\mathbf{u}_0 + \Delta\mathbf{u}]$. The number of uncertainty samples to generate, S , is defined by the user.

Step 4: Evaluate constraints

Once the uncertainty samples have been generated, they can be evaluated for each constraint. Note, in Figure 3.1, the approach with feasibility robustness is considered (evaluating only the $g_j(\mathbf{x}^*, \mathbf{u})$ constraints). For problems with objective robustness, the AOVR constraint (Equation (2.3)) is evaluated as well.

Line 1.15: The `evaluateConstraints` function evaluates all the problem's constraints at the current solutions, $\mathbf{x}^*$, and current realization of the uncertainty parameter, $\mathbf{u}$. Note, that for the first iteration, the problem constraints contain only the constraints from the original problem formulation. For subsequent iterations, the number of constraints increases by one, since the maximum infeasible constraint is added to the problem by the end of each MORO iteration.

Step 5: Check constraint feasibility

After all constraints have been evaluated for each uncertainty sample, $\mathbf{u}$, and each solution point, $\mathbf{x}^*$, the maximum constraint violation can be determined.

Step 6.1: BREAK

Lines 1.16-1.17: If the maximum constraint value is less than the constraint tolerance, tol , specified by the user, then the procedure stops and the optimal solution has been found.

Line 1.21: The robust optimal solution is the solution found in the current iteration, where no constraint violations have been found.

Step 6.2: Add worst-case constraint to problem

If the maximum constraint violation is greater than the specified constraint tolerance, then that constraint is added to the problem, and the algorithm continues to the next iteration, starting from **Step 2**.

Line 1.19: In order to add the worst-case constraint to the problem, the algorithm needs to determine which constraint was violated (con_{infeas}) and which set of uncertainty parameters caused the maximum violation (u_{infeas}) - this is performed by the `findMaxViolation` function.

Line 1.20: Once the worst-case constraint has been identified, it can be appended to the problem as an additional constraint with the corresponding uncertainty values that led to the infeasibility.

The approach described above is very general in nature and can be applied to a range of problems – constrained and unconstrained, linear and non-linear, continuous and discrete, as well as problems where the objectives and constraints are black-box models. Additionally, although the description of the approach focused on interval uncertainty, in theory, any uncertainty type can be handled - such as distributional, or a mix of interval and distributional.

One note to consider is that, for the approach to find a truly robust solution, the uncertainty space should be well explored, which relates to determining the appropriate number of samples to consider for the uncertainty sample generation. There is no general rule to determine an optimal sample size, and it is evident that this number will be dependent on the size and nature of the problem – specifically, how many uncertain parameters are being considered, how many

constraints contain any of the uncertain parameters, whether there are any couplings between the uncertainties, etc. Naturally, as the size and complexity of the problem increases, the number of samples and thus the computational effort of this approach can increase substantially.

3.1.1 Computational Effort

For the OSB-MORO approach (Equation(3.2)), the computational complexity (number of function calls) depends on:

- a) the number of iterations of the MORO algorithm required to achieve a robust solution (I);
- b) the number of function calls required for the solver (G); in this approach, since the MOGA solver (NSGA-II) is used, the computational complexity is $\mathcal{O}(MN^2)$ [107], where N is the population size and M is the number of objectives;
- c) the number of uncertainty samples (S) generated to evaluate the robustness of the obtained non-dominated solutions (K)

$$\mathcal{O}(\text{OSB-MORO}) = I \times (M \times N^2 + S \times K) \quad (3.2)$$

Following the assumption that the NSGA-II solver is used, it can be seen that the driving factor of the computational complexity is the population size (N), which leads to a quadratic growth, rather than the number of samples (S), which yields a linear relationship with respect to complexity - as seen in Figure 3.2.

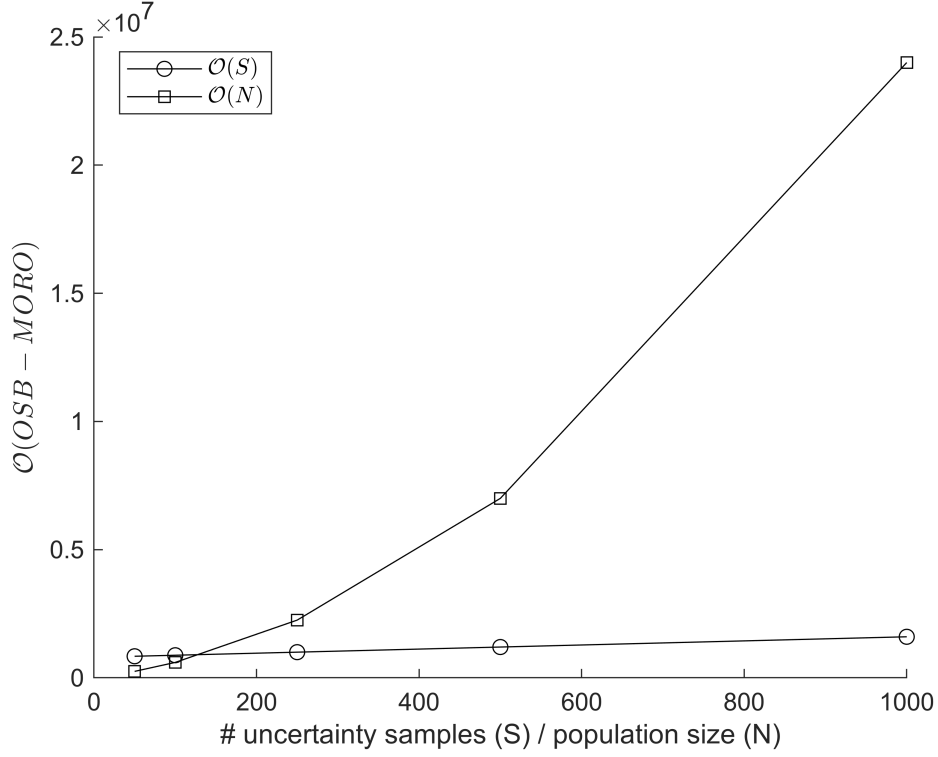


Figure 3.2: OSB-MORO computational complexity with respect to number of uncertainty samples ($\mathcal{O}(S)$) and population size ($\mathcal{O}(N)$)

3.2 Examples

This section will show applications of the proposed OSB-MORO approach to some numerical and engineering examples, as well as a USV case study. A summary of example problems considered is presented in Table 3.1.

The optimization solver implemented for all the example problems was MATLAB's multi-objective genetic algorithm (MOGA) solver `gamultiobj` (version 2022b) [106], and each example was solved 10 times to account for the stochasticity of MOGA and the scenario generation. The examples were solved with the following MOGA and RO algorithm options, as shown in Table 3.2. The remaining parameter settings for the MOGA solver were set to their default values. The number of uncertainty samples to generate for the robustness evaluation was derived based

Table 3.1: Summary of OSB-MORO Test Problems

	MinEx	TNK	CTP	Three-Bar Truss	ZDT3	USV
M	2	2	2	2	2	2
J	2	2	1	3	0	3
N	2	2	10	2	variable	10
P	1	2	2	2	2	1
type	obj-feas	feas	feas	obj-feas	obj	feas

M - number of objectives; J - number of constraints; N - number of decision variables; P - number of uncertain parameters; "type" refers to robustness type - feasibility ("feas") robustness, objective ("obj") robustness, or objective-feasibility ("obj-feas") robustness.

Table 3.2: MOGA and RO option settings for numerical and engineering example problems

MOGA options	
Maximum generations	500
Population size	200
Pareto fraction	0.35
RO options	
Maximum MORO iterations	15
Number of uncertainty samples	1000

on empirical studies performed on the test problems listed in Table 3.1 during the development of the approach. It was determined that the range of the uncertainty variation as well as the number of constraints the uncertainty appears in seem to have the most influence on the number of samples required to obtain a robust solution. So, to accommodate all example problems, the sample size was set to 1000.

In order to evaluate the performance of the proposed OSB-MORO method and compare it to an existing method, called Improved MORO, introduced by Hu et al. [63], three quality metrics were used - the hyperarea difference (HD) [41], the overall spread (OS) [41], and spacing (SP) [42]. Additionally, the number of function calls and the number of MORO iterations required to converge to a robust solutions were reported.

The Improved MORO [63] is an iterative sequential bi-level MORO approach, where the robustness is determined based on a worst-case analysis. The upper-level problem is solved to find an optimal solution, then the lower-level problem is solved to check whether all the solutions from the upper-level problem are robust. To obtain a robust solution, the feasible domain is iteratively confined by constraint cuts determined by the solutions from the lower-level problem.

3.2.1 MinEx - Illustrative Example

The first problem, MinEx, is presented as an illustrative example, to walk through the steps of the OSB-MORO algorithm. It was first introduced by Deb [6] and later adapted by Hu et al. [63] to include uncertainty. The formulation is presented in Appendix A.1.

Step 1: Initialization and Problem definition

The problem is defined as shown in Appendix A.1, Equation (A.1). The uncertainty is set to its nominal value.

Step 2: Solve optimization problem

The problem is solved using MATLAB's MOGA solver, and a set of non-dominated solutions, $\mathbf{x}^*$, are obtained, as show by the Pareto front in Figure 3.3. Note, that since the uncertainty was set to its nominal value, the obtained solution is the deterministic solution to the problem.

Step 3: Generate uncertainty samples

In this step, 100 uncertainty samples (just for illustrative purposes) are generated using the LHS technique. The uncertainty variation range is $\Delta u = \pm 0.3$, so the uncertainty was sampled from an interval range of $[0.7, 1.3]$.

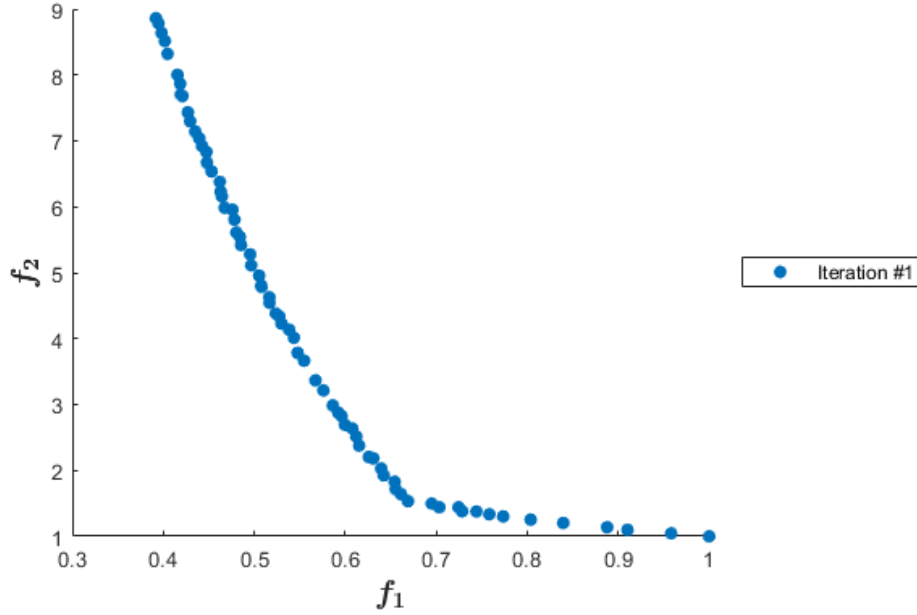


Figure 3.3: MinEx problem solution for iteration #1

Step 4: Evaluate constraints

In this step the constraints are evaluated for all obtained non-dominated solutions ($\mathbf{x}^*$) and all realizations of uncertainty ($\mathbf{u}$). Note, that since this problem is of objective-feasibility type, the AOVR constraint (Equation (2.3)) is also considered. Hence, the following constraints are evaluated:

$$\begin{aligned}
 g_1(\mathbf{x}^*, \mathbf{u}) &= 6 - 9x_1^* - \mathbf{u}x_2^* \leq 0 \\
 g_2(\mathbf{x}^*, \mathbf{u}) &= 1 - 9x_1^* + \mathbf{u}x_2^* \leq 0 \\
 \Delta f_1 &= f_1(\mathbf{x}^*, \mathbf{u}) - f_1(\mathbf{x}^*, \mathbf{u}) \leq 0.3 \\
 \Delta f_2 &= f_2(\mathbf{x}^*, \mathbf{u}) - f_2(\mathbf{x}^*, \mathbf{u}) \leq 0.6
 \end{aligned} \tag{3.3}$$

Figure 3.4 shows a heatmap of the constraint violations for each constraint. The x-axis represents the uncertainty sample, and the y-axis represents the non-dominated solution point. The magnitude of the constraint violation is represented by the colorbar, and the grey points represent feasible points. Thus, it can be seen, for example, that constraint 1 (g_1) has the most

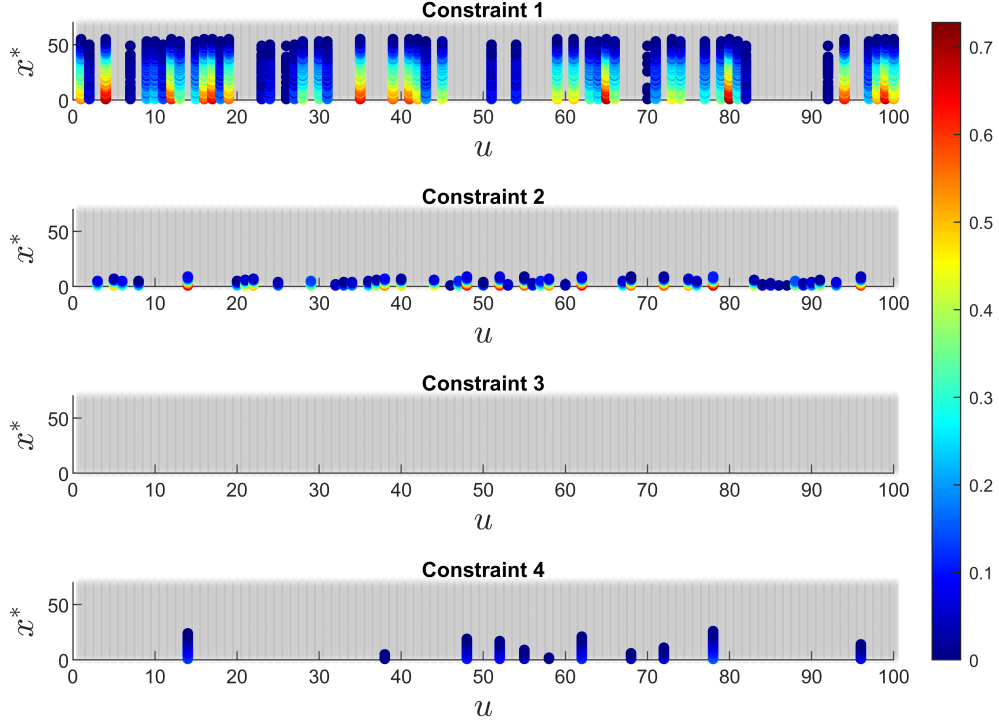


Figure 3.4: MinEx problem constraint violations for iteration #1

violations across all non-dominated points and uncertainty samples, and constraint 4 (Δf_2) is violated only for a few uncertainty samples and non-dominated points.

Step 5: Check constraint feasibility

Once all the constraints have been evaluated, the maximum constraint violation can be determined. In this case, the maximum constraint violation was for $g_1 = 0.728$ for $u = 0.705$.

Step 6.2: Add worst-case constraint to problem

From the previous step it was determined that the uncertainty value $u = 0.705$ led to the maximum constraint violation for g_1 . Thus, the infeasibility sets for the constraints and uncertainties are updated accordingly - $\mathbf{con}_{infeas} = [g_1]$, $\mathbf{u}_{infeas} = [0.705]$ - and the following constraint is added to the problem:

$$g_{1,infeas}^{(1)}(\mathbf{x}) = 6 - 9x_1 - 0.705x_2 \leq 0 \quad (3.4)$$

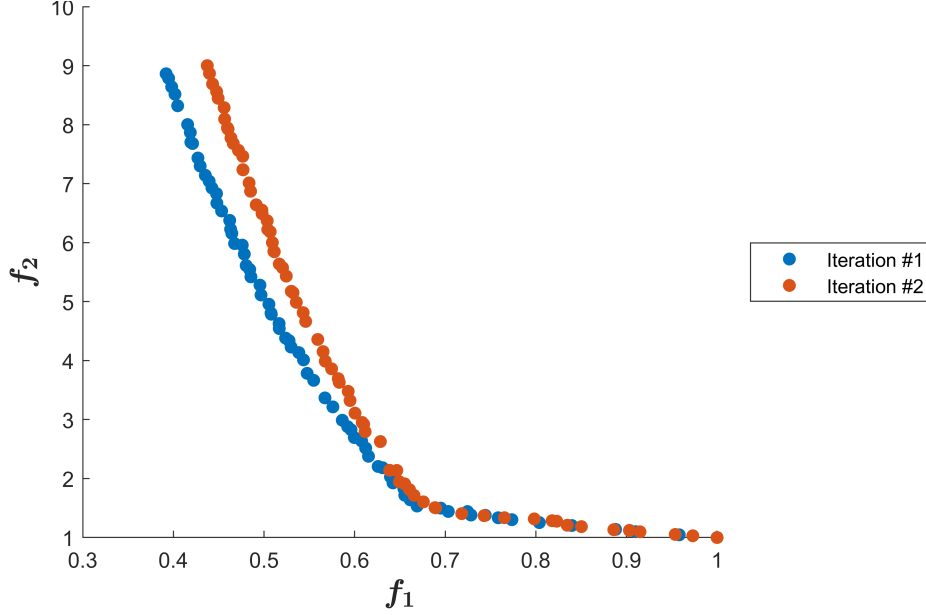


Figure 3.5: MinEx problem solution for iteration #2

This completes the first iteration of the OSB-MORO algorithm.

The second iteration of OSB-MORO returns to **Step 2** of the algorithm - however, now the problem is solved with the additional constraint, as shown in Equation (3.4). The resulting Pareto front is shown in Figure 3.5, in orange, whereas the points in blue show the solution from the first iteration. It can be seen that the additional constraint led to some separation between the two Pareto curves. Figure 3.6 shows the subsequent constraint violations calculated in **Step 4** for the second iteration. This time, most violations occur for constraint 2 (g_2), rather than constraint 1 (g_1). The maximum constraint violation happened for g_2 for $u = 1.298$ - these were appended to the infeasibility sets, i.e. $\text{con}_{infeas} = [g_1, g_2]$ and $\mathbf{u}_{infeas} = [0.705, 1.298]$. Thus, in **Step 6.2**, the following constraint is added to the problem:

$$g_{2,infeas}^{(2)}(\mathbf{x}) = 1 - 9x_1 + 1.298x_2 \leq 0 \quad (3.5)$$

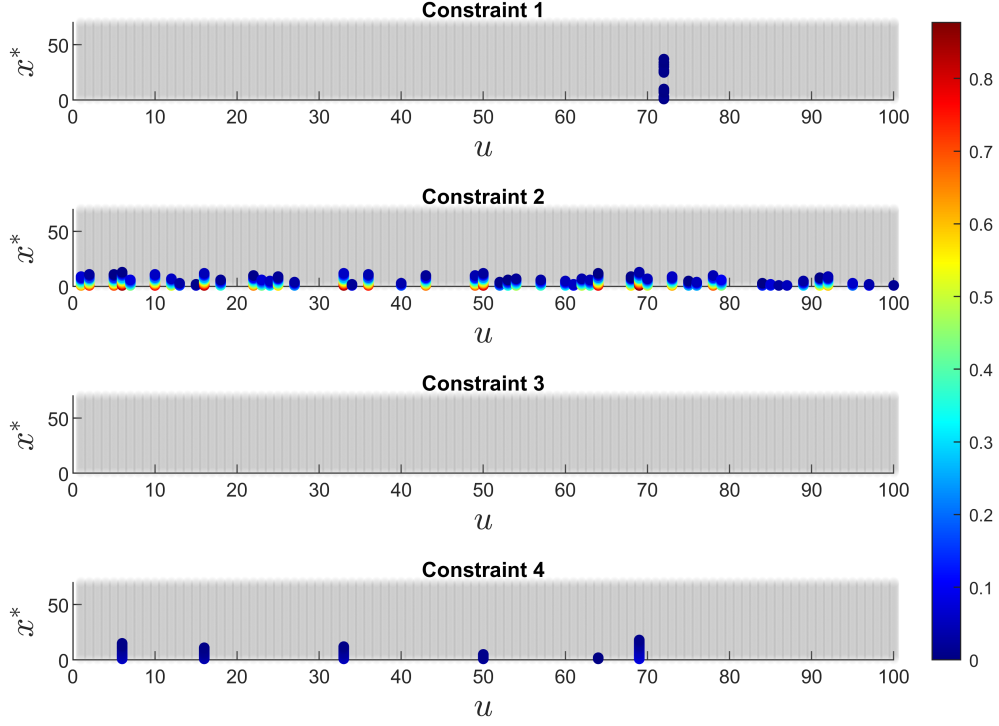


Figure 3.6: MinEx problem constraint violations for iteration #2

The algorithm repeats and stops after 6 iterations, once all the constraints are satisfied during the robustness evaluation step (**Step 4**).

3.2.2 TNK - Numerical Example

The next example presented is a simple numerical example, known in the literature as TNK, was first introduced by Tanaka et al. [108] and later modified to include uncertainty by Hu et al. [10]. The formulation is presented in Appendix A.2. It is a non-linear, constrained, bi-objective problem, with two decision variables and two uncertain parameters, which appear in one of the constraints.

The proposed OSB-MORO algorithm was applied to the problem, and the results are shown in Figure 3.7. The blue and red lines show the deterministic and robust solutions across all runs, respectively.

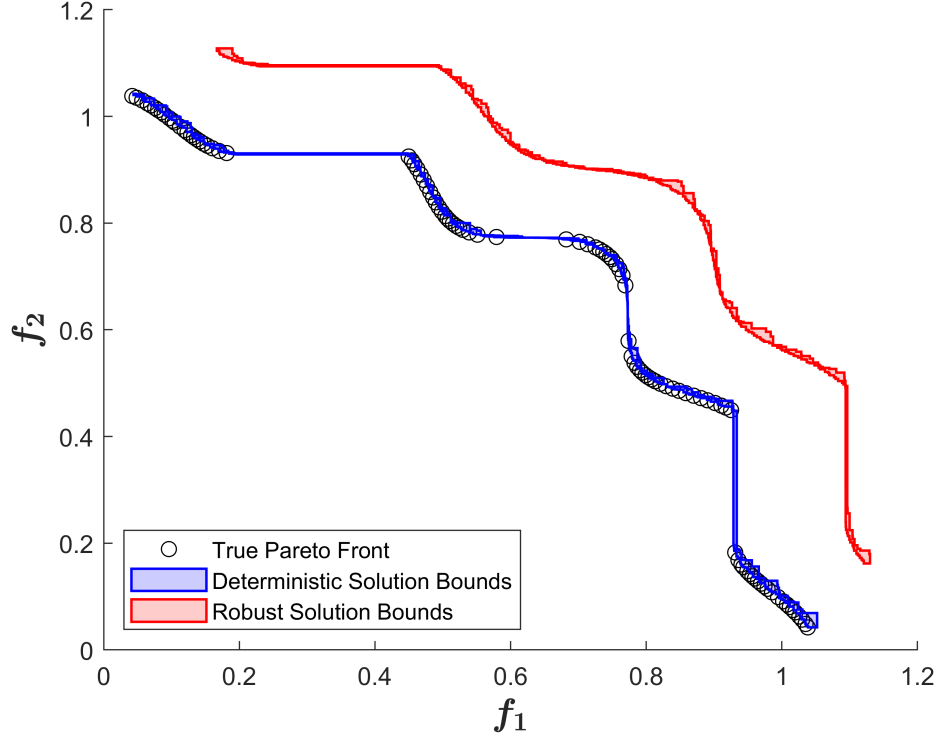


Figure 3.7: TNK solution for OSB-MORO

Table 3.3: TNK Result Comparison - OSB-MORO

Metric [mean (std)]	OSB-MORO	Improved MORO [63]
HD	0.580 (0.001)	0.581 (0.003)
OS	0.459 (0.008)	0.458 (0.011)
SP	0.004 (0.001)	0.005 (0.001)
Function Calls	344,316 (135,366)	817,740 (101,004)
MORO Iterations	3 (0.943)	3 (0)

The quality metrics were calculated and averaged across all runs and compared to the performance of the Improved MORO approach [63] - the results are presented in Table 3.3. It can be seen that the proposed approach outperforms Improved MORO with respect to the mean values, especially in terms of function calls, reducing the computational effort in half. The proposed approach also has a tighter standard deviation, leading the results to be more consistent across independent runs.

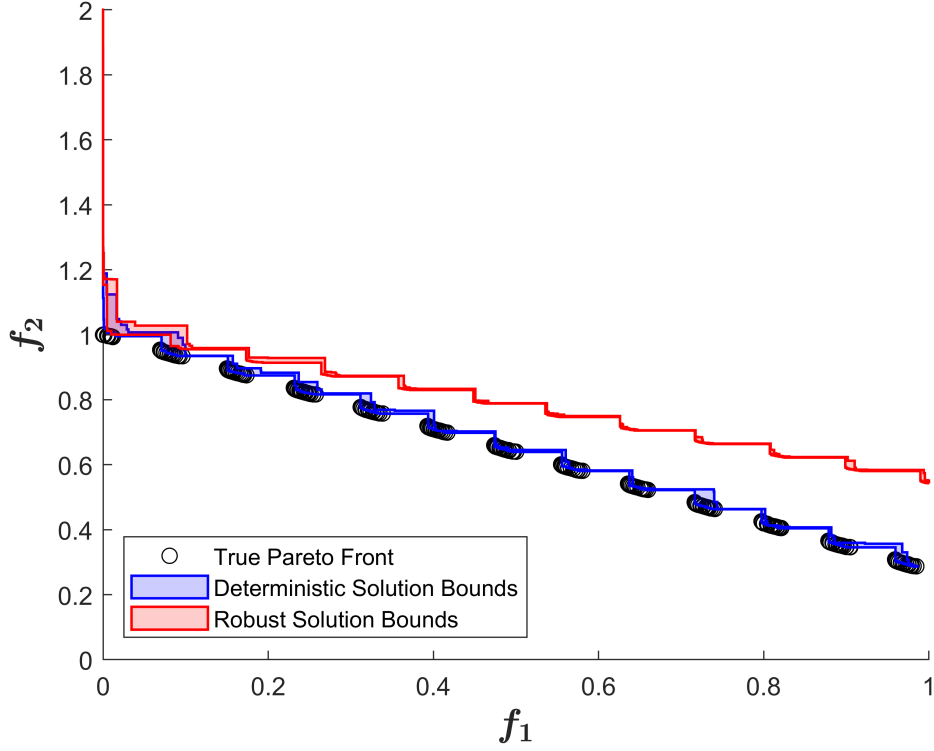


Figure 3.8: CTP solution for OSB-MORO

3.2.3 CTP - Numerical Example

The CTP example is a highly non-linear bi-objective problem, with one constraints, ten decision variables and two uncertain parameters. The problem was initially introduced by Deb [6] and later modified by Hu et al. [10] to include uncertainty. The formulation of the problem is presented in Appendix A.3.

The results are shown in Figure 3.8. The problem was run 10 times (independently) and the bounds for the deterministic and robust solutions across runs are shown in blue and red, respectively.

Table 3.4: CTP Result Comparison - OSB-MORO

Metric [mean (std)]	OSB-MORO	Improved MORO [63]
HD	0.712 (0.004)	0.760 (0.001)
OS	1.870 (1.022)	0.834 (0.102)
SP	0.034 (0.020)	0.006 (0.004)
Function Calls	2,176,400 (863,520)	3,598,252 (1,764,746)
MORO Iterations	10.8 (2.741)	5 (1.414)

The average and standard deviation values of the quality metrics are summarized in Table 3.4. The OSB-MORO approach performs better than the Improved MORO with respect to the HD, OS, and function call metrics.

3.2.4 Three Bar Truss - Engineering Example

OSB-MORO was also applied to an engineering example - the three-bar truss problem taken from [61]. The problem is bi-objective with three constraints, and two design variables. The uncertainty is connected to the design variables and appears in both the objective functions and constraints, making the problem an objective-feasibility MORO problem. The description and formulation of the problem can be found in Appendix A.4

The results are shown in Figure 3.9. The blue and red curves show the deterministic and robust solutions, respectively.

The quality metrics were calculated and averaged across all 10 runs and compared to the performance of Improved MORO [63] - the results are presented in Table 3.5. As shown in Table 3.5, the results from the proposed approach appears to show better performance than Improved MORO, particularly the function calls, somewhat comparable for HD and SP, and slightly worse with respect to OS.

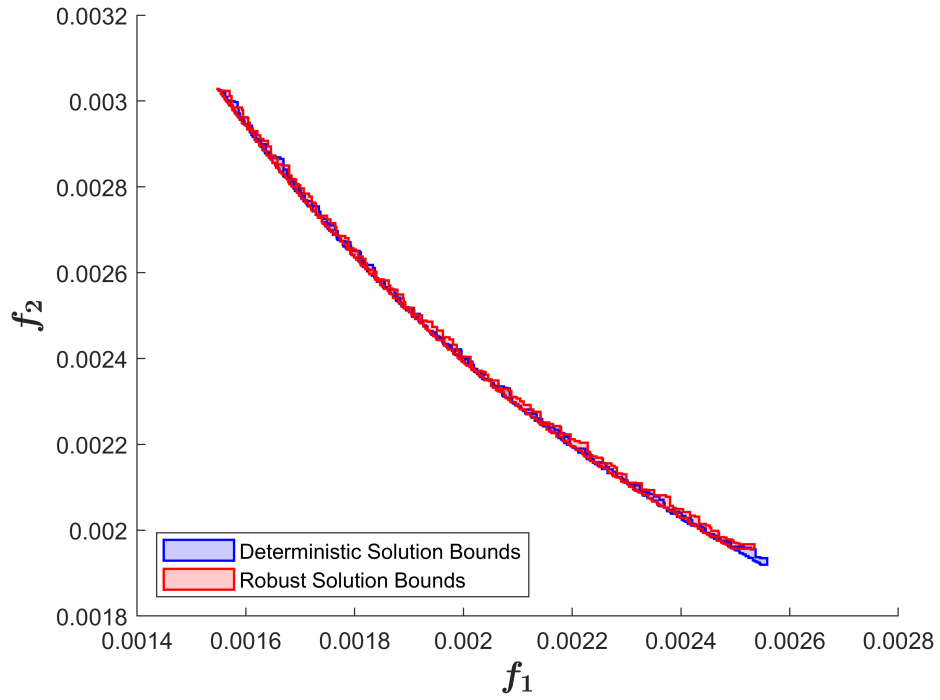


Figure 3.9: Three-bar truss solution for OSB-MORO

Table 3.5: Three-Bar Truss Result Comparison - OSB-MORO

Metric [mean (std)]	OSB-MORO	Improved MORO [63]
HD	0.487 (0.006)	0.483 (0.001)
OS	0.595 (0.034)	0.651 (0.001)
SP	0.008 (0.001)	0.008 (0.002)
Function Calls	343,862 (144,229)	712,297 (90,281)
MORO Iterations	3 (0.943)	3 (0)

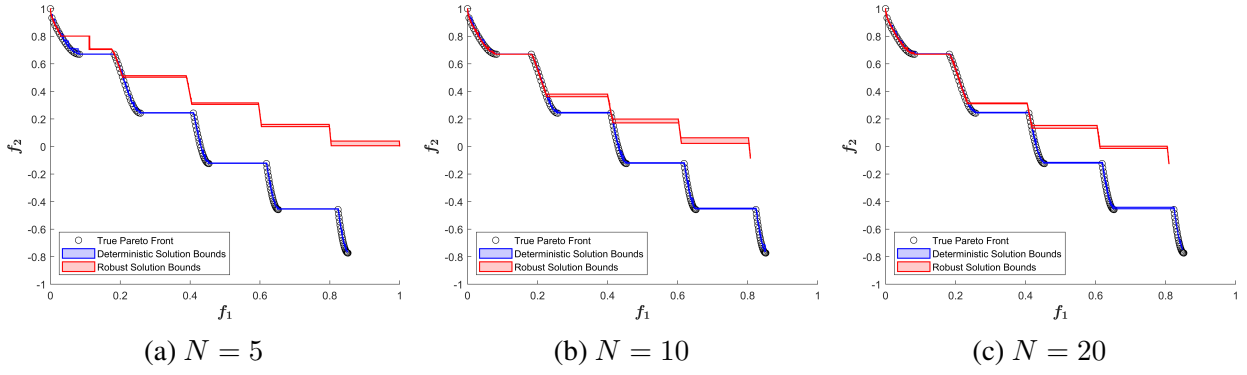


Figure 3.10: ZDT3 solutions for $N = 5, 10, 20$ for OSB-MORO

3.2.5 ZDT3 - Scalable Example

The OSB-MORO approach was applied to a scalable example problem - ZDT3 [6], which was adapted to include uncertainty in [10]. The example is an unconstrained bi-objective problem with a variable number of decision variables (N) and two uncertain parameters. The formulation of the problem is shown in Appendix A.5.

The ZDT3 problem was solved for $N = 5, 10, 20$ decision variables and the results and quality metrics are shown in Figure 3.10 and Table 3.6, respectively. It is interesting to observe that the HD metric improves as the number of variables increases. Based on the Pareto fronts in Figure 3.10, it makes sense - as the number of variables increases, the separation between the deterministic and robust solutions decreases (i.e. the robust solution is closer to the true Pareto front, thus leading to a smaller HD value). On the other hand, the OS metric seems to worsen as the number of decision variables increases. The remaining quality metrics are somewhat comparable.

Table 3.6: ZDT3 Result Comparison - OSB-MORO

Metric [mean (std)]	$N = 5$	$N = 10$	$N = 20$
HD	0.677 (0.002)	0.636 (0.017)	0.616 (0.013)
OS	0.500 (0.001)	0.378 (0.111)	0.395 (0.098)
SP	0.005 (0.002)	0.004 (0.001)	0.004 (0.001)
Function Calls	557,370 (162,070)	504,812 (165,498)	554,539 (205,527)
MORO Iterations	6.6 (1.140)	4.1 (0.994)	4.4 (1.174)

3.2.6 USV Case Study

The proposed OSB-MORO approach was also applied to a real world engineering problem - a case study of an unmanned surface vessel (USV) [2]. Details of the USV system and modelling are described in Appendix B.

The USV case study is formulated as a bi-objective RO problem, with the objectives being to maximize the minimum USV speed ($speed_{min}$) and maximize the final system reliability (R). Constraints are imposed on the sway force experienced by the USV (F_y), the final distance to the target destination (d), and the system reliability (R). The constraints have been normalized to improve the solver performance. The uncertainty parameter is manifested in the form of the wave current direction. The formulation of the problem is presented in Equation (3.6).

$$\min\{-f_1 = speed_{min}, -f_2 = R\}$$

subject to:

$$g_1(\mathbf{x}, u) = \max(F_y)/300kN - 1 \leq 0$$

$$g_2(\mathbf{x}, u) = d/5km - 1 \leq 0$$

$$g_3(\mathbf{x}, u) = -R/0.5 + 1 \leq 0 \tag{3.6}$$

$$0 \leq x_i \leq 1, \forall i = 1, 2, 3, 4, 5$$

$$-30^\circ \leq x_6 \leq 30^\circ; \quad -90^\circ \leq x_i \leq 90^\circ, \forall i = 7, 8, 9, 10$$

$$u = u_0 + \Delta u$$

$$u_0 = -45^\circ; \quad \forall \Delta u \in [-1^\circ, 1^\circ]$$

The decision variables, $x_i, \forall i = 1, \dots, 10$, represent the USV's throttle and heading values for each of the 5 profile stages. The uncertainty, u , is considered in the wave current direction, and set as an interval uncertainty, with the nominal value of u_0 . Some preliminary simulation analysis has shown that the second constraint, g_2 , is extremely sensitive to variations in the uncertainty value, so the allowable range was limited to ensure a viable solution could be found. The first objective, f_1 , is to maximize the minimum USV speed. This objective ensures that the USV gets to the destination as quickly as possible. The second objective, f_2 , is to maximize the final system reliability. Constraint g_1 ensures that the USV does not experience any wave forces above a certain threshold, and constraint g_3 enforces a limit on acceptable system reliability. Constraint g_2 ensures the USV arrives at the destination and inherently accounts for optimizing the route of the USV from the origin to the target destination.

As a reminder, since the system reliability was found to be fairly immutable with respect

Table 3.7: MOGA and RO option settings for USV case study

MOGA options	
Maximum generations	75
Population size	200
Pareto fraction	0.35
RO options	
Maximum iterations	3
Number of uncertainty samples	25

to varying input conditions (e.g., decision variables), a modified reliability value was used (following Equation (B.5) in Appendix B.3). The modification parameters $\alpha = 1$ and $\alpha = 4$ were used.

Due to the complexity of the problem and computational intensity, the MOGA and RO options were limited to the values set in Table 3.7. These measures were set to ensure the MORO algorithm can produce some sensible results in a reasonable amount of time.

Figure 3.11 shows the resulting observed Pareto fronts for the USV problem, for the cases when $\alpha = 1$ (circles) and $\alpha = 4$ (squares). The blue represent the deterministic solutions, and the red - the robust solutions. As it can be seen, some of the points of the robust solution overlap with the deterministic solution - this can be due to several factors, such as the premature termination of the optimizer, or the limited samples size in the uncertainty generation step.

One observation can be made about the degradation of the system (represented by the reliability value) and how various α values from the adjusted reliability calculation (Equation (B.5)) affect the solutions. The most noticeable difference is that for $\alpha = 4$, the system can maintain a higher reliability (above 0.75), compared to the case when $\alpha = 1$, where the reliability is under 0.65. This observation makes sense given the adjusted reliability calculation in Equation (B.5). In Figure B.4 it can be seen that the reliability decreases linearly w.r.t the throttle for $\alpha = 1$,

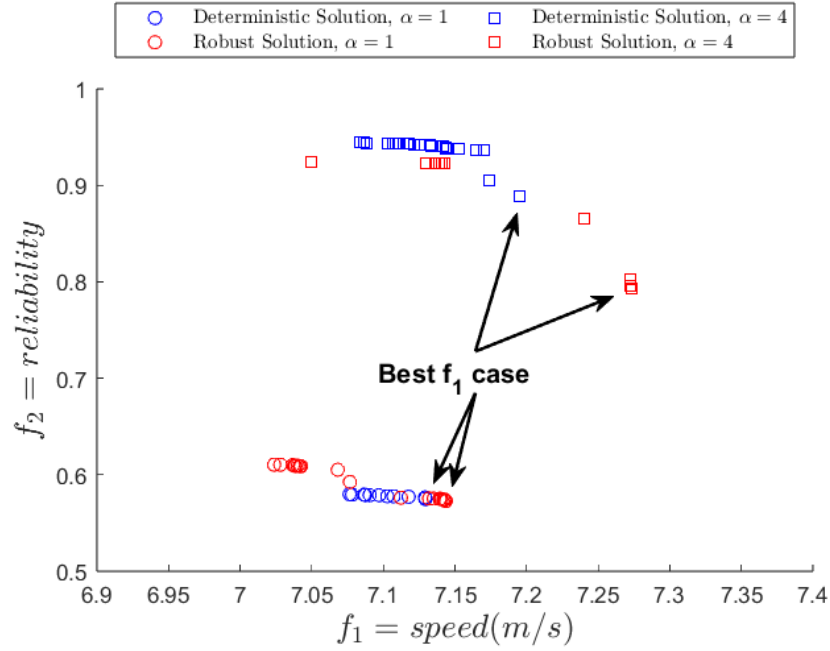
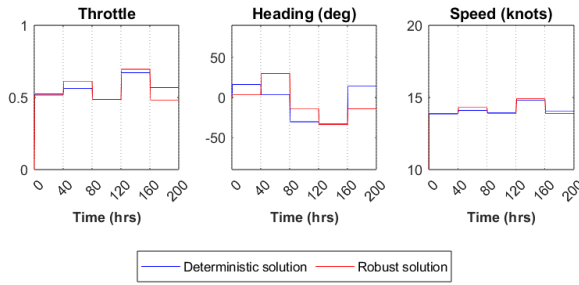


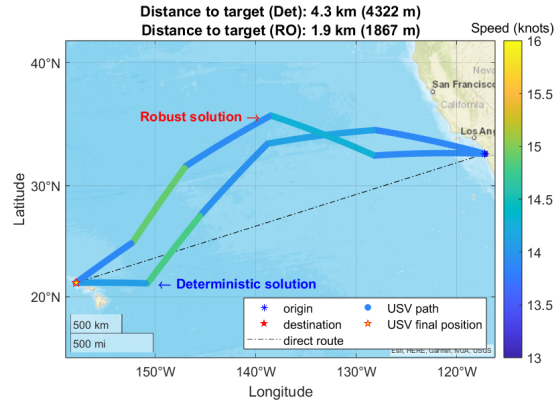
Figure 3.11: USV Pareto Fronts for $\alpha = 1$ and $\alpha = 4$

whereas for $\alpha = 4$, the reliability decreases at a significantly slower rate, at least initially. This implies that overall, the system reliability will be lower, when $\alpha = 1$ and higher when $\alpha = 4$. Also, constraint g_3 in Equation (3.6) enforces a 0.5 lower bound on the system reliability - so, for $\alpha = 1$, the throttle is limited to about 0.7, but for $\alpha = 4$, the throttle can reach a maximum of about 0.9. Since the USV speed is proportional to the throttle, it is reasonable to assume that the USV speed may be higher when $\alpha = 4$, compared to when $\alpha = 1$ - the Pareto front in Figure 3.11 exhibits this as well.

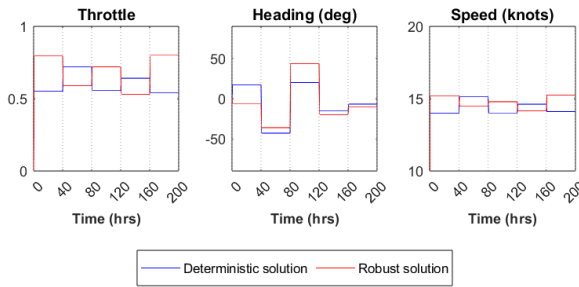
Another observation can be made regarding the spread of the solutions. The constraint on the system reliability (g_3 in Equation (3.6)) indirectly limits the maximum throttle and therefore the speed too; however the distance constraint (g_2 in Equation (3.6)) indirectly enforces a lower bound on the speed and consequently the throttle as well. So, the optimization solver must find a throttle range that is feasible for both constraints. Referring to Figure 3.11 and the discussion in



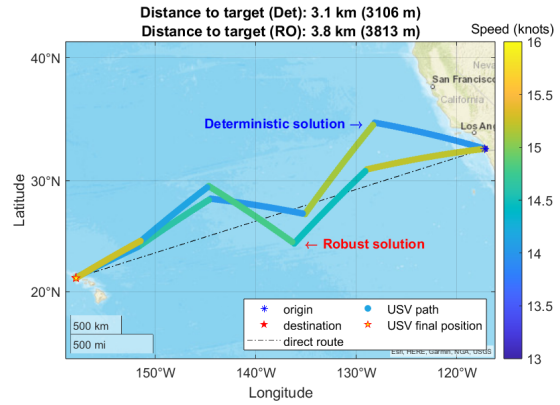
(a) USV Input and Output Values for $\alpha = 1$



(b) USV Path for $\alpha = 1$



(c) USV Input and Output Values for $\alpha = 4$



(d) USV Path for $\alpha = 4$

Figure 3.12: USV trajectories and control values for $\alpha = 1$ and $\alpha = 4$.

the paragraphs above, it is apparent that for $\alpha = 4$, the throttle has a greater range than for $\alpha = 1$, hence the solution has a greater spread as well.

To visualize the deterministic and robust solutions, the USV simulation was run with the corresponding inputs for one of the extreme cases - the best f_1 values. The results are shown in Figures 3.12a, 3.12b for $\alpha = 1$ and Figures 3.12c, 3.12d for $\alpha = 4$. Figures 3.12a and 3.12c show the decision variables (input throttle and heading) and speed profile (the objective value) during the mission. The blue and red lines represent the deterministic and robust solutions, respectively. As alluded to previously, the overall throttle and speed values are higher for the case when $\alpha = 4$, and the deterministic and robust solutions are more diverse. The heading for both

cases is different, which can be explained by the uncertainty parameter (wave current direction). Figures 3.12b and 3.12d show the resulting deterministic and robust paths of the USV for both cases. The speed of the USV (in knots) along the path is represented by the color according to the colorbar range on the right hand side of the figure. It is interesting to observe that the paths for both cases ($\alpha = 1$ and $\alpha = 4$) follow very different trajectories; however, the deterministic and robust solutions are somewhat similar for each respective case.

3.3 Utility-Based Sampling-Based Multi-Objective Robust Optimization (USB-MORO)

The OSB-MORO approach can also be applied to single-objective optimization problems. This can be achieved by reformulating the multi-objective problem as single-objective - by use of a multi-attribute utility function (MAUF). The use of utility functions as scalarization methods for multi-objective problems has been used in the literature extensively [27]. Utility functions are a foundational concept in decision theory, providing a structured way to model and quantify a decision maker's preferences over a set of choices or outcomes under uncertainty. Utility theory is commonly used and particularly useful when decisions must be made among alternatives that involve trade-offs between competing attributes.

3.3.1 USB-MORO Problem Formulation and Approach

The problem formulation for the utility-based single-objective version of OSB-MORO is shown in Equation (3.7):

$$\begin{aligned}
& \min_{\mathbf{x}} [-w_1 U(\mathbf{x}, \mathbf{u}) + w_2 \Delta f] \\
& \text{subject to:} \\
& g_j(\mathbf{x}, \mathbf{u}) \leq 0; j = 1, \dots, J \\
& \mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}] \\
& \forall \mathbf{u} \in [\mathbf{u}_0 + \Delta \mathbf{u}] \\
& \Delta \mathbf{u} \in [\Delta \mathbf{u}_{lb}, \Delta \mathbf{u}_{ub}] \\
& \text{where} \\
& \Delta f = U(\mathbf{x}, \mathbf{u}) - U(\mathbf{x}, \mathbf{u}_0) \\
& w_1 + w_2 = 1
\end{aligned} \tag{3.7}$$

In Equation (3.7), the objective function consists of two parts - a weighted combination of the utility function, $U(\mathbf{x}, \mathbf{u})$, and variation vector, Δf . The first term, $U(\mathbf{x}, \mathbf{u})$, represents the utility function. Since the goal is typically to maximize the utility, a negative sign is present. The second term, Δf , represents the amount of variation between the actual (due to uncertainty, $\mathbf{u}$) and nominal values for the utility function. This term is equivalent to the AOVR constraint in Equation (3.1). The reason for including the variation vector, Δf , in the objective function, rather than treating it as an AOVR constraint, is as follows. Since the objective is represented as a utility function, which combines several attributes and is usually scaled between 0 and 1 (or -1 and 0), the actual value of the utility loses its physical meaning, making it difficult to assign a single representative AOVR value. However, since it is still desirable to minimize the variation due to uncertainty, the term is included in the objective function.

Both terms in the objective function are assigned weights (the sum of which equal to 1), which are determined by the decision maker. If both terms are equally important, then set $w_1 = w_2 = 0.5$. If the utility term is more important, then set $w_1 > w_2$, and vice versa. Moreover, if the decision maker wants to see a tradeoff between the two terms, they can implement the weighted sum method [6] and run a sweep of weight values to obtain a curve of Pareto-optimal points.

The rest of the problem formulation is identical to the one presented for OSB-MORO in Equation (3.1).

The approach for USB-MORO follows the same steps as OSB-MORO (refer to Figure 3.1 and Section 3.1). The only difference is that the solver in **Step 2** should be a single-objective solver, such as genetic algorithms (GA) [109, 110], or gradient-based methods [111].

3.3.2 USV Case Study

The USB-MORO approach was applied to the USV case study, the details of which are described in Appendix B.

The USV case study is formulated as a single-objective problem, with the objective of maximizing the utility and minimizing the variation of utility due to uncertainty. Constraints are imposed on the sway force experienced by the USV (F_y), the final distance to the target destination (d), and the system reliability (R). The constraints have been normalized to improve solver performance. The uncertainty parameters are manifested in the current directions and current speeds. The formulation of the problem is presented in Equation (3.8). Note, the formulation of the problem used for USB-MORO is not equivalent to the formulation used for OSB-MORO - since the attributes of the utility function are based on the decision variables and uncertain

parameters, not the original objective functions.

$$\begin{aligned}
& \min \{-0.5U(\mathbf{x}, \mathbf{u}) + 0.5\Delta f\} \\
& \text{subject to:} \\
& g_1(\mathbf{x}, \mathbf{u}) = \max(F_y)/350kN - 1 \leq 0 \\
& g_2(\mathbf{x}, \mathbf{u}) = d/5km - 1 \leq 0 \\
& g_3(\mathbf{x}, \mathbf{u}) = -R/0.5 + 1 \leq 0 \\
& 0 \leq x_i \leq 1, \forall i = 1, 2, 3, 4, 5 \\
& -30^\circ \leq x_6 \leq 30^\circ; \quad -90^\circ \leq x_i \leq 90^\circ, \forall i = 7, 8, 9, 10 \\
& \mathbf{u} = \mathbf{u}_0 + \Delta \mathbf{u} \\
& \mathbf{u}_0 = [12.51^\circ, 30.81^\circ, 11.42^\circ, 7.9^\circ, 10.54^\circ, \dots \\
& \dots 0.06m/s, 0.06m/s, 0.11m/s, 0.09m/s, 0.14m/s] \\
& \forall \Delta \mathbf{u} \in \pm[5^\circ, 5^\circ, 5^\circ, 5^\circ, 5^\circ, \dots \\
& \dots 0.01m/s, 0.01m/s, 0.01m/s, 0.01m/s, 0.01m/s] \\
& \text{where} \\
& \Delta f = U(\mathbf{x}, \mathbf{u}) - U(\mathbf{x}, \mathbf{u}_0)
\end{aligned} \tag{3.8}$$

The decision variables, x_i , are the USV throttle and heading values for each section of the trip. The first 5 uncertain parameters represent the current directions, and the last 5 represent the current speeds for each section of the trip. The nominal values for the uncertainties are set to $\mathbf{u}_0$, with a variation of $\Delta \mathbf{u} \in [-5^\circ, 5^\circ]$ for the current directions and $\Delta \mathbf{u} \in [-0.01m/s, 0.01m/s]$ for the current speeds.

Table 3.8: `fmincon` and RO option settings for USV case study (USB-MORO approach)

fmincon options	
Algorithm	interior-point
Subproblem Algorithm	cg
Constraint Tolerance	1×10^{-3}
Finite Difference Step Size	1×10^{-4}
Optimality Tolerance	1×10^{-3}
Step Tolerance	1×10^{-6}
RO options	
Maximum iterations	10
Number of uncertainty samples	100

The utility function is developed as a multiplicative multi-attribute utility function (MAUF), combining 6 attributes - the USV speed (x_{11}), the sway forces acting on the USV (x_{12}), the wave height (x_{21}), the wind speed (x_{22}), the wave current speed (x_{23}), and the USV system reliability (x_3). The wave height and wind speed values were determined based on real environment data, as described in Appendix B.5 and the utility function development is described in Appendix B.6.

Constraint g_1 ensures that the USV does not experience any wave forces above a certain threshold, and constraint g_3 enforces a limit on acceptable system reliability. Constraint g_2 ensures the USV arrives at the destination and inherently accounts for optimizing the route of the USV from the origin to the target destination.

As a reminder, since the system reliability was found to be fairly immutable with respect to varying input conditions (e.g., decision variables), a modified reliability value was used (following Equation (B.5) in Appendix B.3). The modification parameter $\alpha = 1$ was used.

For the optimization solver, MATLAB's interior-point algorithm implemented via the `fmincon` function was used [112]. The solver and RO options are summarized in Table 3.8. The remaining options were set to their default values.

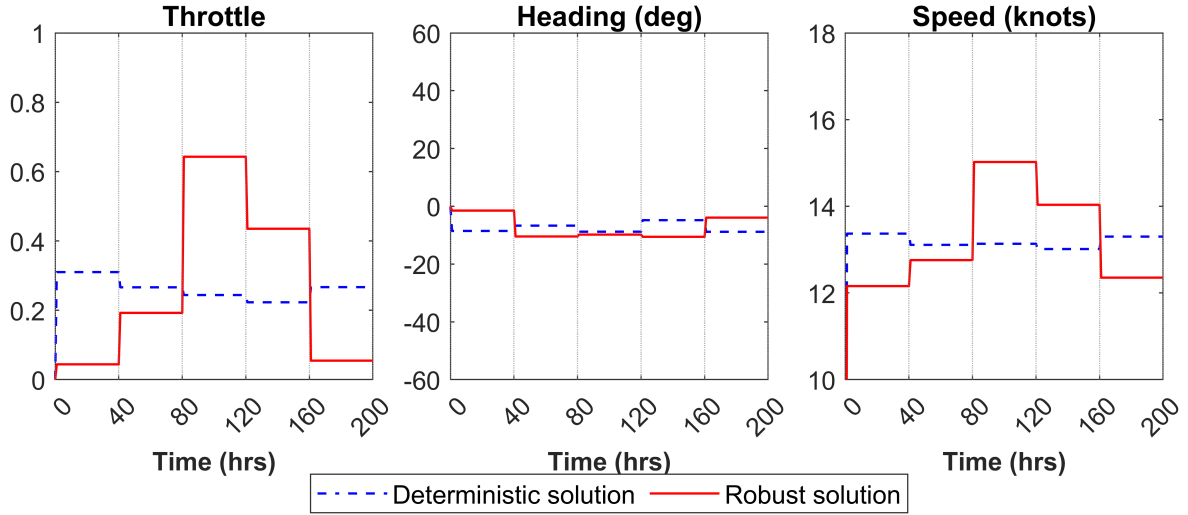


Figure 3.13: USV inputs for USB-MORO (deterministic and robust cases)

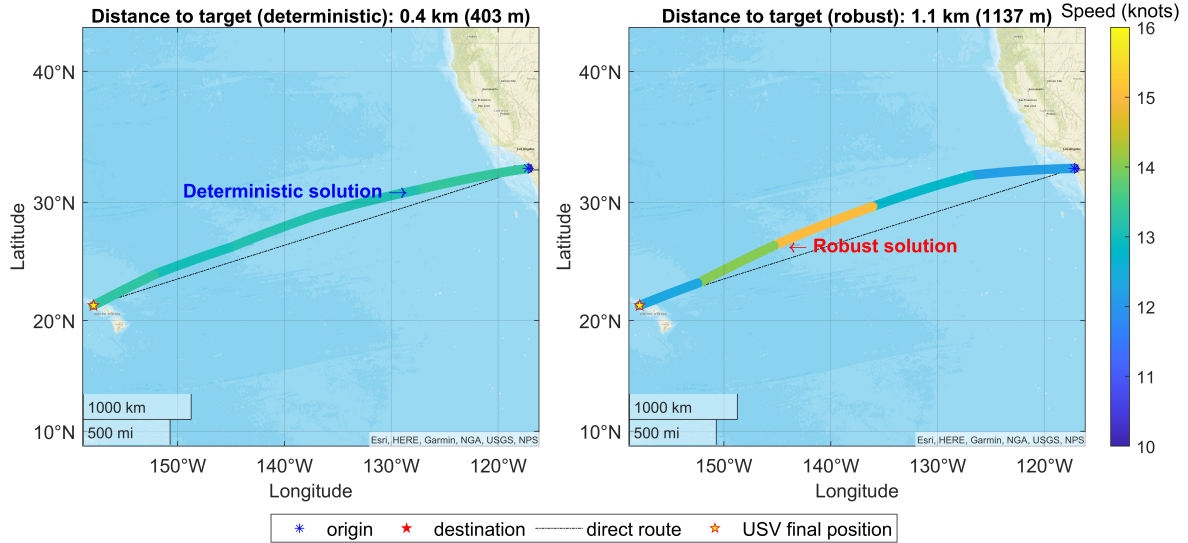


Figure 3.14: USV paths for USB-MORO (deterministic and robust cases)

The optimized decision variables (throttle and heading) and resulting USV speed are shown in Figure 3.13. The corresponding paths are shown in Figure 3.14.

Although the trajectories for both the deterministic and robust solutions are similar, the throttle (and consequentially the USV speed) show more variation, which can be attributed to the uncertain and changing environment.

Figure 3.15 shows the profiles of the attributes and utility. As expected, the utility attributes

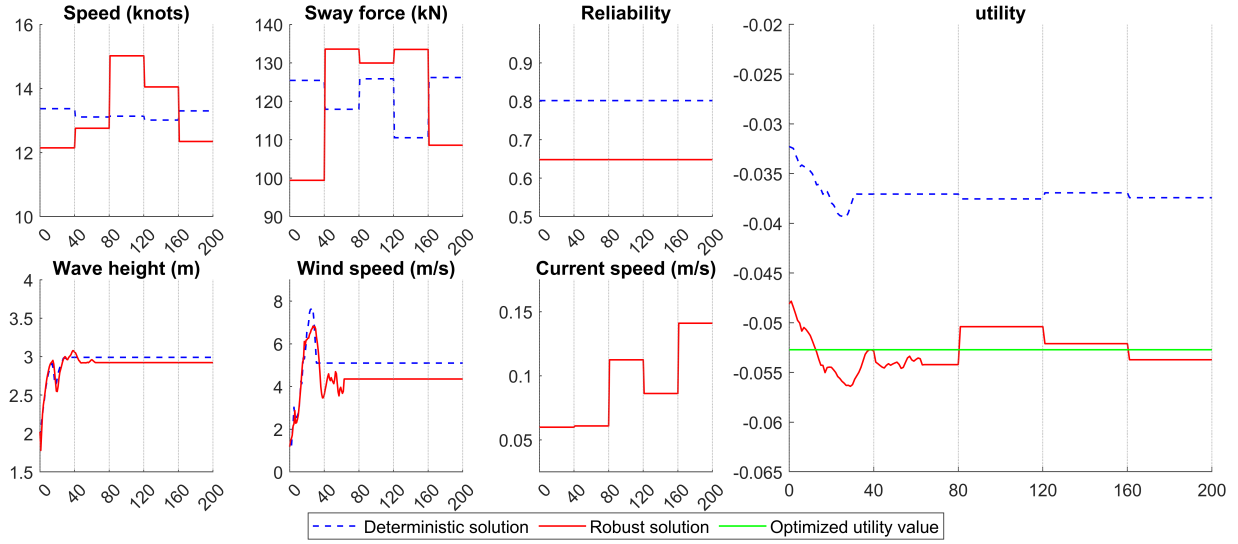


Figure 3.15: USV attribute and utility profiles for USB-MORO (deterministic and robust cases)

for the deterministic and robust solutions are unique. The USV speed for the robust case is higher than the deterministic (on average), which leads to a lower system reliability (recall the adjusted reliability calculation in Appendix B.3, Equation B.5). The differences in environment conditions (wave height and wind speed) are due to the different trajectories for the deterministic and robust cases. The profile for the utility value also makes sense - for the deterministic case it is higher (better), averaging at -0.0371, whereas for the robust case the average utility is -0.0527 ².

3.4 Discussion

This chapter presented a new sequential, single-level, scenario-generation based MORO approach. This is a truly single-level optimization approach, which does not require another optimization problem to be solved and utilizes solely a sampling technique to handle the robustness. The advantages of this method include its flexibility to handle a wide range of problem types (such as convex and non-convex, with objective or feasibility robustness, interval or distri-

²Note, since the optimizer for a single-objective problem can only handle a single value for the objective function, the mean aggregation of the utility value was used - as shown by the green line in the "utility" subplot of Figure 3.15.

butional uncertainty ranges), its simplicity, and its sequential single-level nature. The flexibility of the proposed approach stems from the generality of its framework - the sampling-based algorithm to handle robustness is not restrictive to specific problem types, and thus is applicable to any problem; likewise, the MOGA solver utilized is capable of handling a wide variety of problem types. The method was applied to a couple of example problems and a real-world case study featuring the operation of a USV, and has demonstrated its overall effectiveness.

Another advantage of OSB-MORO is that it only requires the optimization problem to be solved once per iteration, which can effectively reduce the computational cost, as demonstrated in the example problems. However, for problems with expensive objective and constraint functions, this method requires the development of (offline and/or online) surrogate models, which would replace the original functions in the optimizer. Moreover, the surrogates must be reasonably accurate for the optimizer and MORO algorithm to produce valid results. As demonstrated in the USV case study, this can be a challenging task.

Although the OSB-MORO approach is simple and straightforward to implement, it exhibits some drawbacks and limitations.

The sampling scheme is straightforward to implement and conceptually intuitive, however, it does not scale well, specifically with respect to the number of uncertainty samples to generate. A primary drawback is the challenge in determining the appropriate number of samples needed to fully explore the uncertainty space - this would have a direct impact on finding a truly robust solution. As presented, there is no general rule to determine an optimal sample size, and it is evident that this number will be dependent on the size and nature of the problem – specifically, how many uncertain parameters are being considered, how many constraints contain any of the uncertain parameters, whether there are any couplings between the uncertainties, etc. Naturally,

as the size and complexity of the problem (e.g., number of decision variables, or constraints) increases, the number of samples and thus the computational effort of this approach can increase substantially and become infeasible. Considering this, OSB-MORO is not an ideal approach for expensive or complex problems, such as those involving simulations or black-box functions.

Another aspect to consider is the sampling technique - the current implementation utilizes Latin-Hypercube sampling from an interval range, which is simple to implement, however, may not efficiently cover the uncertainty space, thus leading the algorithm to incorrectly converge to a non-optimal robust solution. Alternative sampling methods, such as other design of experiments (DOE) techniques, or adaptive sampling techniques could potentially offer better coverage and more efficient exploration of the uncertainty space.

Finally, the computational cost of this approach can also become significant for more complex problems - the approach utilizes the NSGA-II optimizer, which is an evolutionary algorithm (EA), thus inherently requires a lot of function calls to reach a solution (however, the wide applicability of EA's makes up for their computational cost in some respects). Additionally, the OSB-MORO algorithm makes the problem progressively more complex, since the approach adds a new constraint to the problem sequentially each iteration, thus making the feasible domain more restrictive.

The proposed approach could be extended in several ways to improve the computational efficiency and robustness of the final solution. One area of improvement would be the methodology for the uncertainty generation and handling. Another aspect to improve, as demonstrated by the case study, would be integrating a surrogate development model in the MORO algorithm framework.

In summary, while OSB-MORO provides a straightforward framework for handling un-

certainties in MORO, its drawbacks include difficulties in determining appropriate sample sizes, inefficient sampling techniques, and high computational costs. These limitations highlight the need for more advanced and adaptive algorithms that can dynamically adjust to the problem's characteristics and improve computational efficiency.

The next chapter will explore an improved OSB-MORO method - one that seamlessly integrates online surrogates to enhance the computational efficiency of a sampling-based MORO approach.

Chapter 4: Surrogate Sampling-Based Multi-Objective Robust Optimization (SSB-MORO)¹

This chapter presents a significantly enhanced version, particularly with respect to computational costs, of the original sampling-based approach presented in Chapter 3 - a surrogate-assisted sampling-based MORO (SSB-MORO) method. Similarly to OSB-MORO, the SSB-MORO approach aims to solve the problem formulated by Equation 4.1:

$$\begin{aligned}
 & \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{u}) \\
 & \text{subject to:} \\
 & g_j(\mathbf{x}, \mathbf{u}) \leq 0; \quad j = 1, \dots, J \\
 & f_m(\mathbf{x}, \mathbf{u}) - f_m(\mathbf{x}, \mathbf{u}_0) \leq AOV R_m; \quad m = 1, \dots, M \\
 & \mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}] \\
 & \forall \mathbf{u} \in [\mathbf{u}_0 + \Delta \mathbf{u}] \\
 & \Delta \mathbf{u} \in [\Delta \mathbf{u}_{lb}, \Delta \mathbf{u}_{ub}]
 \end{aligned} \tag{4.1}$$

However, unlike OSB-MORO, the proposed SSB-MORO approach utilizes surrogates, rather than the original problem functions, to perform computations for the optimization and

¹A version of this chapter will be presented at the ASME IDETC-CIE-2025 conference [3]

robustness evaluation.

This chapter is organized as follows. A detailed overview of the SSB-MORO approach will be described in Section 4.1. The performance of the proposed approach will be demonstrated on a few examples (Section 4.2) and a USV vase study (Section 4.2.5). Section 4.3 will provide a comparison between SSB-MORO and a few methods from the literature. Finally, Section 4.4 will conclude this chapter with a summary and discussion of some of the highlights and limitations of the SSB-MORO approach.

4.1 Overview of Approach

The proposed surrogate-assisted sampling-based MORO method (SSB-MORO) is an extension to a scenario generation MORO method [2], as presented in Chapter 3, which was not surrogate-based, and integrates an online surrogate model to improve computational efficiency.

A high-level flowchart of the approach is presented in Figure 4.1, and the pseudocode is provided in Algorithm 2.

The proposed algorithm closely resembles OSB-MORO (presented in Chapter 3), with the addition of an integrated online surrogate model to decrease the computational cost of evaluating expensive functions. The purpose of integrating an online surrogate, rather than developing one offline, is to be able to dynamically update the surrogate in real time, strategically targeting regions of high uncertainty and constraint violation. This adaptive refinement of both decision and uncertainty spaces ensures improved surrogate fidelity near the robust Pareto-optimal front.

The surrogate is developed using a Kriging model [113], which models the entire problem. The Kriging approach [114] is a two-part model - the first consisting of a regression model, and

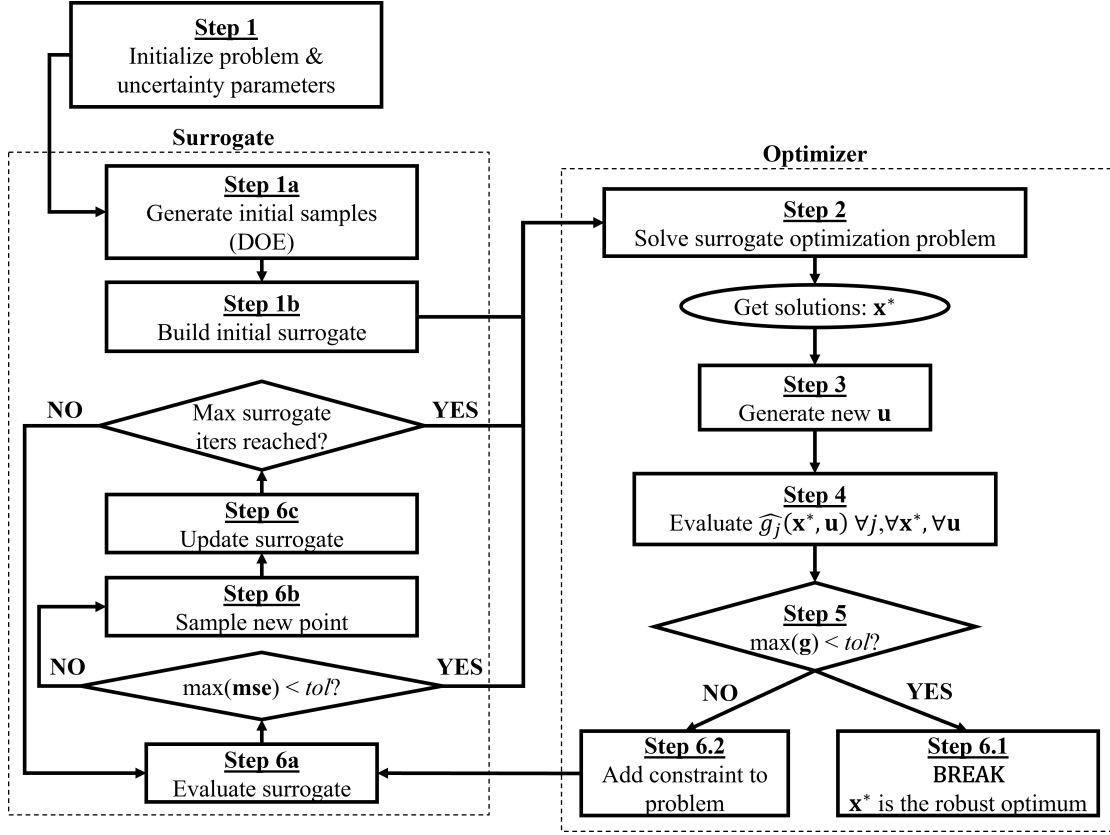


Figure 4.1: Flowchart of SSB-MORO approach

the second is a random Gaussian function, as shown in Eq. (4.2):

$$Y(x) = \sum_{j=1}^k \beta_j f_j(x) + Z(x) \quad (4.2)$$

where f_j are the known functions, β_j are the regression coefficients, and $Z(x)$ is the Gaussian function with a mean of zero and covariance of $\sigma^2 R$, where σ^2 is the process variance and R is the correlation.

Similar to OSB-MORO, the SSB-MORO approach is iterative in nature, handling the optimization and robustness evaluation sequentially. In the initialization step, the problem is defined and the uncertain parameters are set to their nominal values. A design of experiments (DOE) technique, capturing the entire design space, is used to generate some sample inputs and responses to

build an initial surrogate model for the problem. This surrogate model is then used for all subsequent function evaluations, until the surrogate is updated. In the optimization step, the problem is solved using a multi-objective solver, and a set of non-dominated solutions, $\mathbf{x}^*$, is obtained. Next, a fixed number of new uncertainty scenarios, $\mathbf{u}$, are generated using the Latin-Hypercube sampling (LHS) method. Then, the robustness evaluation is performed - all the constraints (for all non-dominated solutions, $\mathbf{x}^*$, and all realizations of uncertainty, $\mathbf{u}$) are evaluated. The constraint with the maximum violation is added to the optimization problem. Recall, the function evaluations are performed on the surrogate functions, not the original problem functions. This completes a single MORO iteration, and now the surrogate model can be updated, based on the current obtained solution and robustness evaluation. This aspect ensures that the surrogate is effectively sampling not only the decision variable space, but the uncertainty space as well. The surrogate is allowed to update a set number of times (determined empirically or by the decision maker), before continuing to the next steps of the algorithm. After the surrogate has been updated, the algorithm continues - the problem is resolved (with the added constraint and improved surrogate), a new set of non-dominated solutions is obtained, and the procedure is repeated. The algorithm stops once all the constraints are satisfied during the robustness evaluation. Thus, the solutions $\mathbf{x}^*$ found in the current iteration are returned as the robust optimal solutions.

In further discussion of the details of the approach, **Step X** will refer to the corresponding step in the flowchart (Figure 4.1) and Line A.Y will refer to the line number Y of the pseudocode for algorithm A (defined in Algorithm 2). Some of the steps are very similar to OSB-MORO, so the reader will be referred to them accordingly.

Algorithm 2: SSB-MORO algorithm

Input: $problemDef, ROptions, KrigingOptions$
Output: $[det_{soln}, ro_{soln}, all_{soln}]$

```
/* Step 1 - Initialization */
1  $u_{infeas} \leftarrow []$ ;
2  $con_{infeas} \leftarrow []$ ;
3  $u \leftarrow u_0$ ;
  /* Step 1a - DOE - Generate initial samples */
4  $DS \leftarrow initialDOE(problemDef, KrigingOptions)$ ;
  /* Step 1b - Build initial surrogate */
5  $surrModel \leftarrow buildSurrogate(DS, KrigingOptions)$ ;
6  $KrigingOptions \leftarrow [DS, surrModel]$ ;
7 for  $iter = 1$  to  $maxIter$  do
  /* Step 2 - Solve surrogate optimization problem */
  8 if  $iter = 1$  then
  9    $[x^*, f] \leftarrow$ 
      $solverFctn(problemDef, KrigingOptions, u, u_{infeas}, con_{infeas})$ ;
  10 else
  11    $[x^*, f] \leftarrow$ 
      $solverFctn(problemDef, KrigingOptions, u_0, u_{infeas}, con_{infeas})$ ;
  12    $all_{soln}(iter) \leftarrow [x^*, f]$ ;
  /* Step 3 - Generate  $S$  uncertainty samples */
  13  $u_{new} \leftarrow generateNewU(problemDef, S)$ ;
  /* Step 4 - Evaluate constraints */
  14  $g \leftarrow []$ ;
  15 for  $sample = 1$  to  $S$  do
  16    $u \leftarrow u_{new}(sample)$ ;
  17    $g \leftarrow evaluateConstraints(problemDef, KrigingOptions, x^*, u)$ ;
  /* Step 5 - Find maximum constraint violation */
  18 if  $\max(g) < tol$  then
  19   break;
  /* Step 6 - Add worst-case constraint to problem */
  20 else
  21    $[u_{infeas}, con_{infeas}] \leftarrow findMaxViolation(g, u_{new})$ ;
  22    $problemDef.new\_constr \leftarrow [u_{infeas}, con_{infeas}]$ 
  /* Algorithm continues on next page */
```

Algorithm 2: SSB-MORO algorithm - continued

```
22 for  $iter = 1$  to  $maxIter$  do
    /* continued from previous page */
23     $surrIter = 0$ ;
24    while  $surrIter < maxSurrIter$  do
        /* Step 6a - Evaluate surrogate */
25         $MSE \leftarrow$ 
            evaluateSurrogate( $\mathbf{x}^*, \mathbf{u}, \mathbf{u}_{infeas}, \mathbf{con}_{infeas}, KrigingOptions$ );
26        if  $\max(MSE) < tol$  then
27            | Continue to Step 2;
28        else
            /* Step 6b - Sample new point */
29             $[\mathbf{x}_{worst}, \mathbf{u}_{worst}] \leftarrow \text{findWorstPoint}(MSE, \mathbf{x}^*, \mathbf{u})$ ;
            /* Step 6c - Update surrogate */
30             $\mathbf{DS}_{new} \leftarrow \text{evaluatePoints}(\mathbf{x}_{worst}, \mathbf{u}_{worst}, problemDef)$ ;
31             $\mathbf{DS} \leftarrow [\mathbf{DS}, \mathbf{DS}_{new}]$ ;
32             $surrModel \leftarrow \text{updateSurrogate}(\mathbf{DS}, KrigingOptions)$ ;
33             $KrigingOptions \leftarrow [\mathbf{DS}, surrModel]$ ;
34             $surrIter \leftarrow surrIter + 1$ ;
35  $ro_{soln} \leftarrow [\mathbf{x}^*, \mathbf{f}]$ ;
    /* Optional - resolve with final surrogate for
       deterministic solution */
36  $det_{soln} \leftarrow \text{solverFctn}(problemDef, KrigingOptions, \mathbf{u}_0)$ 
```

Step 1: Initialize problem

In this step, the problem objective and constraint functions are defined and the uncertainty parameters are set to their nominal values. The inputs to the algorithm, *problemDef* and *ROptions* are the same as for OSB-MORO (see **Step 1** in Section 3.1). The other input, *KrigingOptions*, contains information related to the Kriging surrogate model - the regression function, correlation function, as well as the number of initial samples to generate (for the initial surrogate), the maximum number of times to update the surrogate per MORO iteration, *maxSurrIter* (Line 2.24), and the tolerance, *tol* (Line 2.26), for acceptable mean squared error (MSE) of the surrogate.

Step 1a: Generate initial samples for surrogate

In this step, a set of sample points is generated, using a LHS generation scheme [105], which are used in the next step to build an initial surrogate model. Note, other sampling methods could be used as well, such as random sampling or maximum entropy design [83].

Line 2.4: **DS** is the matrix containing the initial samples generated - the inputs (corresponding to the decision variables and uncertain parameters) and response variables (corresponding to the objective and constraint functions).

Step 1b: Build initial surrogate

The initial sample points are used to build the Kriging surrogate model [83, 114]. Recall, that the Kriging model is built for all of the objective and constraint functions of the problem. In theory, it would be possible to build surrogates only for the expensive functions, however, for simplicity, all functions are considered here.

Line 2.5: *surrModel* is the surrogate model built from the initial samples, **DS**

Line 2.6: The *surrModel* and **DS** objects are passed to the *KrigingOptions* object, which will carry the surrogate model throughout the SSB-MORO algorithm.

Steps 2-6 for SSB-MORO are the same as for OSB-MORO - refer to Steps 2-6 in Section 3.1 for details. For completeness, a brief overview of the steps will be presented here as well. The next step unique to SSB-MORO is **Step 6a**.

Step 2: Solve surrogate optimization problem

Once the Kriging model is built, it is passed into the optimizer, and the surrogate multi-objective optimization problem is solved to obtain a set of non-dominated points, $\mathbf{x}^*$. Note that since the uncertain parameters are initially set to their nominal values, the solution during this iteration is equivalent to a deterministic solution. However, since the initial surrogate may still be inaccurate

during the first iteration, the obtained solution may also not be the "true" deterministic solution (when compared to solving the original problem, without the surrogate and without uncertainty).

Step 3: Generate uncertainty samples

In this step, a new set of uncertain parameters is generated, using the LHS method. The uncertainty space is defined by an interval range, and the number of samples to generate is determined by the user.

Step 4: Evaluate constraints

To determine the robustness and feasibility of the current solution, each constraint is evaluated for all the solution points $\mathbf{x}^*$ and all uncertainty samples generated in **Step 3**. Note, the computation is performed on the surrogate functions, not the original problem functions.

Step 5: Check constraint feasibility

After all constraints have been evaluated for each uncertainty sample, $\mathbf{u}$, and each non-dominated point, $\mathbf{x}^*$, the maximum constraint violation can be determined.

Step 6.1: BREAK

If the feasibility check in **Step 5** is satisfied, then the procedure is terminated, and the solution obtained at the current iteration is returned as the robust optimal solution.

Step 6.2: Add worst-case constraint to problem

If the maximum constraint violation found in **Step 5** is greater than the specified constraint tolerance, then that constraint (with the associated uncertainty sample that led to infeasibility, $\mathbf{u}_{infeas}$) is appended to the problem. This completes a MORO iteration of the algorithm.

Step 6a: Evaluate surrogate

Once the MORO iteration has completed, the surrogate is evaluated for the problem *with the newly added constraint*, for all current solutions, $\mathbf{x}^*$.

Line 2.25: MSE is an estimate of the mean square errors (MSE) of the true vs. predicted values of the responses (objective and constraint function values) obtained from the Kriging model (hence the estimation did not require the evaluation of the true functions). The estimated MSE is calculated by Equation (4.3) [113]:

$$MSE = \sigma^2(1 + c^T R c - 2c^T r) \quad (4.3)$$

where σ^2 is the Kriging process variance, c are the Kriging coefficients, R is the correlation function used for the Kriging model, and r are the correlations between the new and observed sample points. Although an estimated MSE is provided for each response variable for each current solution, $\mathbf{x}^*$, it is more convenient to consider a single MSE value for each current solution, $\mathbf{x}^*$ - i.e. aggregate the MSE's for the response variables. The aggregation is defined as the L_2 norm of the normalized MSE values for the response variables.

Line 2.26-2.27: If the maximum estimated MSE value is less than a user-defined tolerance, then the algorithm continues to the next iteration of MORO, starting from **Step 2**. Otherwise, the algorithm proceeds to **Step 6.b** to improve the surrogate.

Step 6b: Sample new point to update surrogate

The procedure for sampling a new point to use for updating the surrogate is based on evaluating the L_2 norm of the MSE's for each observed solution point obtained so far, $\mathbf{x}^*$.

Line 2.29: $[\mathbf{x}_{worst}, \mathbf{u}_{worst}]$ is the set of decision variables and uncertain parameters that yielded the maximum L_2 norm of the MSE's.

Step 6c: Update surrogate

The selected point from **Step 6.b** is used to update the Kriging model.

Line 2.30: The original problem functions are evaluated for the point $[x_{worst}, u_{worst}]$ and the results are saved to DS_{new} .

Line 2.31: The original matrix with the design samples, DS is appended with the new sample, DS_{new} .

Line 2.32: The surrogate model, $surrModel$, is updated.

Line 2.33: The updated DS matrix and $surrModel$ are saved to the *KrigingOptions* object to be passed on to the next MORO iteration.

Line 2.34: The surrogate iteration count, $surrIter$ is updated by 1. This is done because a single point may not significantly improve the surrogate, so the surrogate updating process (**Steps 6a-c**) is repeated $maxSurrIter$ times (defined by the user), or until the maximum MSE value is less than the specified tolerance, tol , at which point the algorithm continues to the next iteration of MORO, starting from **Step 2**.

The SSB-MORO approach is also general in nature, similar to OSB-MORO. The implementation of the Kriging model is relatively straightforward and can be modified according to the problem at hand - for example, a first order or second order polynomial regressor can be used; or a gaussian vs. exponential correlation function can be used. The main drawback with this approach, as with any surrogate approach, is ensuring surrogate accuracy - since that plays a key role in obtaining a robust, feasible solution which is very close to the solution obtained from the original problem. Higher accuracy requires more computational effort, so it is important to consider the trade-off.

4.1.1 Computational Effort

For the proposed SSB-MORO approach (Eq.(4.4)), the computational complexity (number of function calls) depends on:

- a) the number of samples needed to develop the initial surrogate (D);
- b) the number of iterations of the MORO algorithm required to achieve a robust solution (I);
- c) the number of times the surrogate can be updated during a single iteration of the MORO algorithm (Q);

$$\mathcal{O}(\text{SSB-MORO}) = D + I \times Q \quad (4.4)$$

Recall, that the number of function calls is equivalent to the number of sample points, and that a function calls are only counted for original function evaluations, not surrogate evaluations. Hence, since the optimization solver and robustness evaluation are performed on the surrogate functions, the solver function calls and number of uncertainty sample points do not affect the computational complexity, as they did for the OSB-MORO approach (as discussed in Section 3.1.1). Thus, the computational complexity of the SSB-MORO approach scales linearly with respect to all the terms in Equation (4.4).

It is also worth mentioning that the computational advantage of the surrogate approach has the implicit assumption that the cost of sampling to build and update the surrogate would be much lower than the cost of the optimization on original functions.

Table 4.1: Summary of Test Problems

	TNK	CTP	Three-Bar Truss	ZDT3	USV
M	2	2	2	2	2
J	2	1	3	0	3
N	2	10	2	variable	10
P	2	2	2	2	1
type	feas	feas	obj-feas	obj	feas

M - number of objectives; J - number of constraints; N - number of decision variables; P - number of uncertain parameters; "type" refers to robustness type - feasibility ("feas") robustness, objective ("obj") robustness, or objective-feasibility ("obj-feas") robustness.

4.2 Examples

This section provides some examples to demonstrate the proposed approach and compare it with existing MORO methods from the literature.

The first part of this section (Sections 4.2.1- 4.2.4) will focus on some example problems, as well as an unmanned surface vessel (USV) case study (Section 4.2.5), while the second part (Section 4.3) will provide a systematic comparison with other approaches from the literature. In addition, a statistical analysis will be presented to illustrate the differences between the various MORO methods, for select problems.

A summary of the examples considered is provided in Table 4.1.

The solver implemented for the following example problems was MATLAB's multi-objective genetic algorithm (MOGA) solver (version 2022b) [106], and each example was solved 10 times to account for the stochasticity of MOGA and the scenario generation. Additionally, to verify the robustness of the obtained solutions, a Monte Carlo feasibility analysis was performed with 100,000 samples, for each design point in the robust non-dominated frontier.

The DACE toolbox [113] was used to build the Kriging model, with a second order poly-

Table 4.2: MOGA and RO option settings for numerical and engineering example problems

MOGA options	
Maximum generations	500
Population size	200
Pareto fraction	0.35
RO options	
Maximum iterations	15
Number of uncertainty samples	1000

nomial regression function and Gaussian correlation function. To build the initial surrogate, the number of initial samples was set as $10 \times (nvars + nparams)$, where $nvars$ is the number of decision variables of the problem and $nparams$ is the number of uncertainty parameters. The factor 10 was based on a rule-of-thumb for initial design of experiments (DOE) sample sizes [115] and trial and error. The maximum number of times the surrogate was allowed to update per iteration was set to 25 - this value was determined base on trial and error.

The examples were solved with the following MOGA and RO algorithm options, as shown in Table 4.2. The remaining parameter settings for the MOGA solver were set to their default values.

4.2.1 TNK - Numerical Example

The first example presented is a simple numerical example, known in the literature as TNK, was first introduced by Tanaka et al. [108] and later modified to include uncertainty by Hu et al. [10]. The formulation is presented in Appendix A.2.

This is a nonlinear bi-objective constrained optimization problem with two design variables, two inequality constraints, and two uncertain parameters. Since the uncertainties appear only in one of the constraints, only feasibility robustness is considered. The parameter uncer-

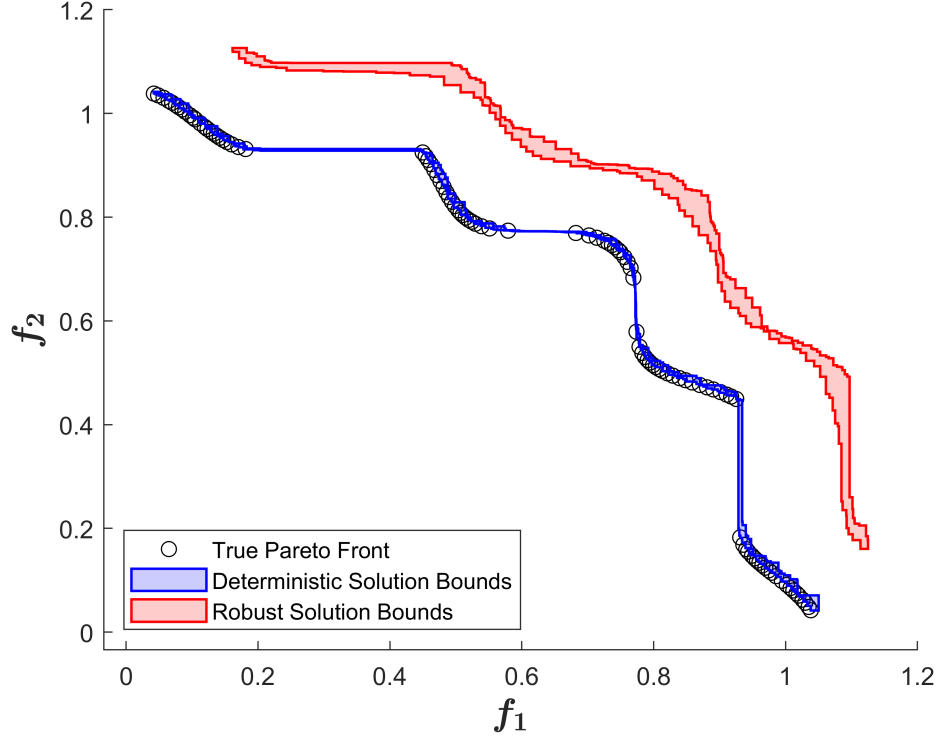


Figure 4.2: TNK solution for SSB-MORO

tainties are defined as an interval, with the nominal values, $u_{0,p}$, equal to zero, and the interval bounds defined by Δu_p .

The results are shown in Figure 4.2, with the blue and red curves representing the deterministic and robust solution bounds, respectively. Note, since the initial surrogate is still inaccurate when the deterministic solution is obtained (during the first MORO iteration), the deterministic version of the problem is re-solved with the final surrogate, after the robust solution is found. This ensures that both the robust and deterministic solutions shown are obtained with the same surrogate. Visually, the obtained solutions look similar to the solutions obtained from the OSB-MORO approach (see Section 3.2.2), however, the robust solution has more variability across runs for SSB-MORO. This can be attributed to the minor surrogate errors as well as the stochastic nature of the approach.

Table 4.3: TNK Result Comparison - SSB-MORO

Metric [mean (std)]	SSB-MORO	OSB-MORO
HD	0.578 (0.004)	0.580 (0.001)
OS	0.461 (0.007)	0.459 (0.008)
SP	0.005 (0.001)	0.004 (0.001)
Function Calls	415 (110)	344,316 (135,366)
MORO Iterations	9.3 (2.541)	3 (0.943)

The quality metrics for the SSB-MORO approach are summarized in Table 4.3 and compared with the quality metrics from OSB-MORO. Additionally, a Monte-Carlo feasibility check was performed with 100,000 samples, which showed that SSB-MORO obtained robust solutions that are feasible 99.7% of the time. Overall, both approaches have similar performance with respect to the HD, OS, and SP metrics. As expected, there is significant improvement with respect to function calls, although the SSB-MORO algorithm does require more MORO iterations to converge to the robust solution.

4.2.2 CTP - Numerical Example

The next example, CTP, is a highly non-linear, constrained bi-objective problem with ten design variables and two uncertain parameters. Both uncertainties appear in the constraints, making the problem a feasibility robustness type. It was originally formulated as a deterministic problem [6], then revised to include uncertainty [10]. The formulation is presented in Appendix A.3.

The results are shown in Figure 4.3 and the quality metrics are summarized in Table 4.4.

Visually, the results differ from the OSB-MORO solutions (refer to Section 3.2.3) - they exhibit more variability. Both the deterministic and robust solution are also worse than the true Pareto optimum. Given that the CTP problem has more decision variables and is highly non-

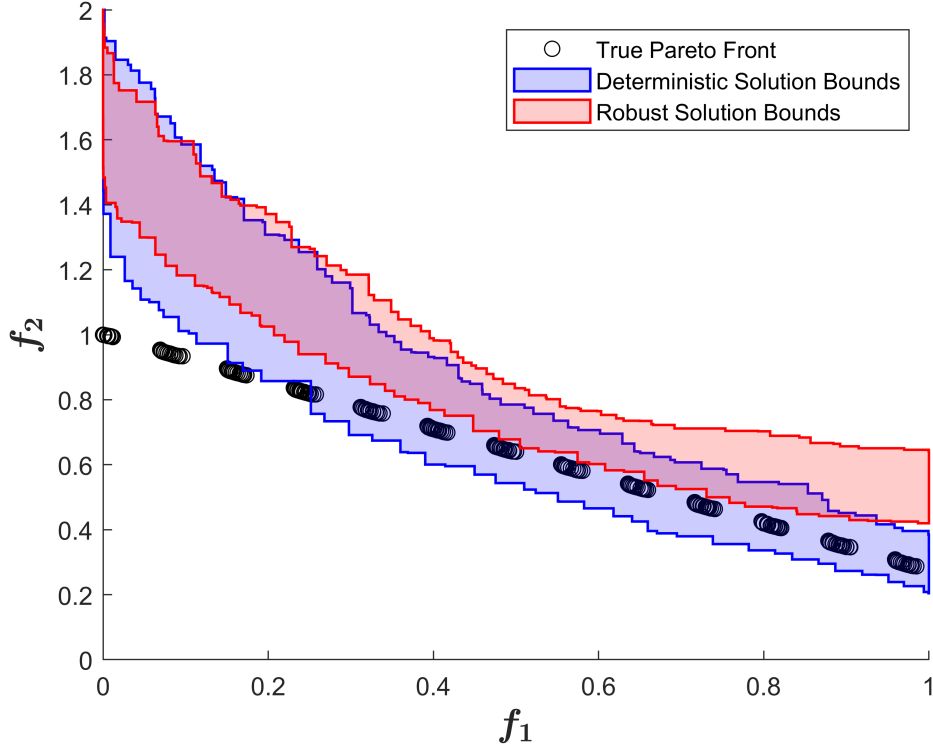


Figure 4.3: CTP solution for SSB-MORO

Table 4.4: CTP Result Comparison - SSB-MORO

Metric [mean (std)]	SSB-MORO	OSB-MORO
HD	0.862 (0.090)	0.712 (0.004)
OS	2.831 (0.729)	1.870 (1.022)
SP	0.024 (0.010)	0.034 (0.020)
Function Calls	1,820 (1,049)	2,176,400 (863,520)
MORO Iterations	10.7 (3.772)	10.8 (2.741)

linear, it is possible that the Kriging surrogate struggled to create an accurate approximation of the true functions.

With respect to the quality metrics, the HD is worse for SSB-MORO than it is for OSB-MORO, but the OS is better - which is consistent with the observation from Figure 4.3. The SP metric is comparable for both approaches, although the mean value for SSB-MORO is better. As expected, the function calls are significantly reduced for SSB-MORO, despite the similar count of MORO iterations. Additionally, a Monte-Carlo feasibility check was performed with 100,000

samples, which showed that SSB-MORO obtained robust solutions that are feasible 98.3% of the time.

4.2.3 Three-Bar Truss - Engineering Example

The proposed approach was also applied to an engineering example - the three-bar truss problem taken from [61]. The problem is bi-objective with three constraints, and two design variables. The uncertainty is connected to the design variables and appears in both the objective functions and constraints, making the problem an objective-feasibility MORO problem.

The problem formulation and the details of the truss configuration are provided in Appendix A.4, and the results and quality metrics are shown in Figure 4.4 and Table 4.5.

The results for SSB-MORO look similar to OSB-MORO (refer to Section 3.2.4), for the most part - except for the trailing tail at the bottom right corner of the figure. Based on the spread of the solutions and similarity to the solutions from OSB-MORO, it can be inferred that the surrogate model succeeded in approximating the problem functions accurately.

The quality metrics in Table 4.5 further support the similarity of performance between the two approaches. Although the HD is better for OSB-MORO, SSB-MORO performs better with respect to the OS and SP, and both approaches require a similar count of MORO iterations to obtain the robust solution. As shown in previous examples, the function call count is reduced by several orders of magnitude. A Monte-Carlo feasibility check, with 100,000 samples, showed that SSB-MORO obtained robust solutions that are feasible 99.5% of the time.

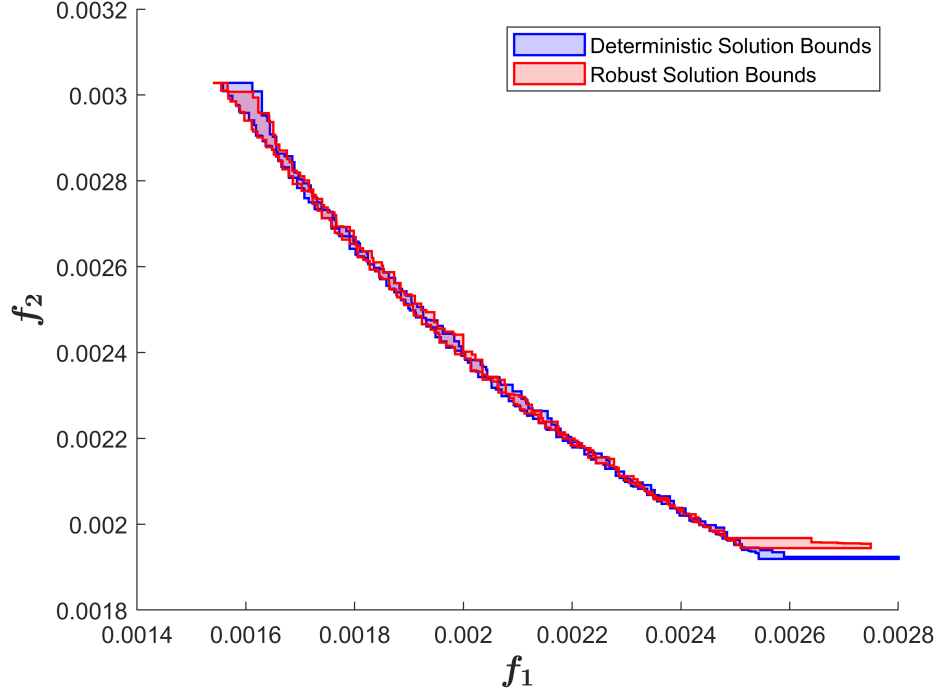


Figure 4.4: Three-bar truss solution for SSB-MORO

Table 4.5: Three-Bar Truss Result Comparison - SSB-MORO

Metric [mean (std)]	SSB-MORO	OSB-MORO
HD	0.492 (0.007)	0.487 (0.006)
OS	0.614 (0.093)	0.595 (0.034)
SP	0.007 (0.001)	0.008 (0.001)
Function Calls	80 (23)	343,862 (144,229)
MORO Iterations	4.6 (3.435)	3 (0.943)

4.2.4 ZDT3 - Scalable Example

The proposed approach was also applied to a scalable example problem ZDT3 [6]. The problem was adapted to include uncertain parameters in [10] as shown in Appendix A.5.

The ZDT3 problem is an unconstrained bi-objective problem, with a scalable number of decision variables (N) and two uncertain parameters. An AOV_R is specified only for the second objective.

The proposed SSB-MORO approach was applied to the ZDT3 problem for $N = 5, 10, 20$

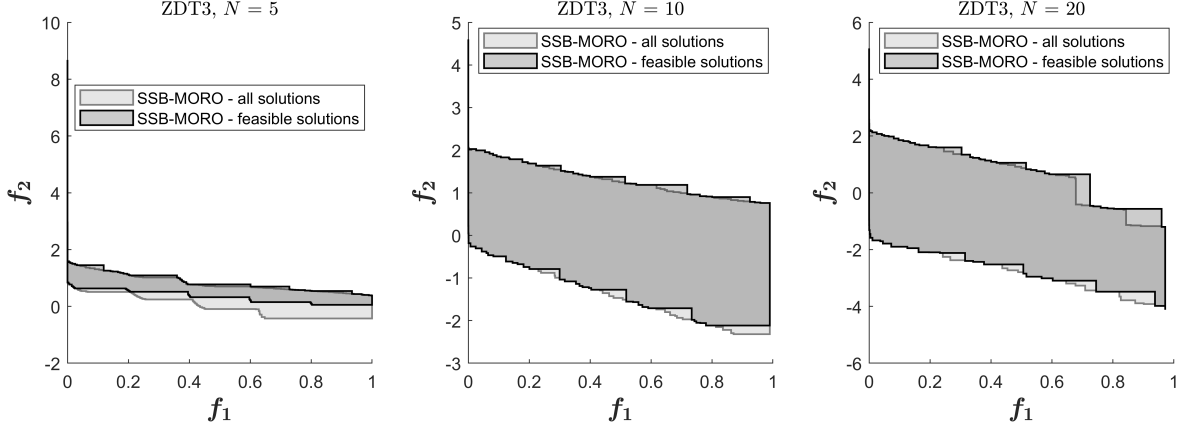


Figure 4.5: ZDT3 solutions with $N=5,10,20$ - range of SSB-MORO solutions from 10 independent runs with all points, and with infeasible points removed.

Table 4.6: ZDT3 Result Comparison - SSB-MORO

Metric [mean (std)]	$N = 5$	$N = 10$	$N = 20$
HD	0.645 (0.124)	0.668 (0.404)	0.261 (0.590)
OS	1.318 (0.841)	1.180 (0.609)	1.420 (1.148)
SP	0.056 (0.084)	0.015 (0.010)	0.013 (0.007)
Function Calls	175	300	825
MORO Iterations	2.2 (0.421)	2.6 (0.699)	3.5 (1.080)

decision variables. To accommodate the increasing size of the problem, the function call limit was set to $ds + 3(ds/2)$, where ds is the number of points to build the initial surrogate. This relationship was determined experimentally, based on the minimum needed function calls to achieve a reasonable solution.

The results are shown in Figure 4.5. The quality metrics were calculated and are presented in Table 4.6. A Monte-Carlo feasibility check was performed for each case, with 100,000 samples. SSB-MORO obtained robust solutions that are feasible 96.2% of the time for $N = 5$, 95% of the time for $N = 10$, and 97.3% of the time for $N = 20$.

As can be seen from the results, the algorithm performance degrades, yielding more variability in the results, as the problem size increases. The results become more dispersed and

Table 4.7: MOGA and RO option settings for USV case study

MOGA options	
Maximum generations	200
Population size	200
Pareto fraction	0.35
RO options	
Maximum iterations	3
Number of uncertainty samples	100

non-uniform across independent runs. A potential cause could be the surrogate inaccuracy. More complex problems may need more points to develop the surrogate and/or more update iterations allowed to converge to a better solution.

4.2.5 USV Case Study

The proposed SSB-MORO approach was also applied to a real-world case study for planning a mission of an unmanned surface vessel (USV), which is adapted from [2, 5]. For a detailed description of the individual models and integration, refer to Appendix B.

The formulation for the USV case study is the same as for the OSB-MORO approach - refer to Equation (3.6) and Section 3.2.6 for details.

The reliability adjustment factor, α , which was introduced to achieve a more exaggerated relationship between the reliability and throttle was set to $\alpha = 4$.

Due to the computational intensity of the USV problem, the MOGA (MATLAB's multi-objective genetic algorithm (version 2022b) [106]) and RO algorithm options were limited to the values set in Table 4.7. The remaining parameter settings for the MOGA solver were set to their default values. These measures were set to ensure the MORO algorithm can produce some sensible results in a reasonable amount of time.

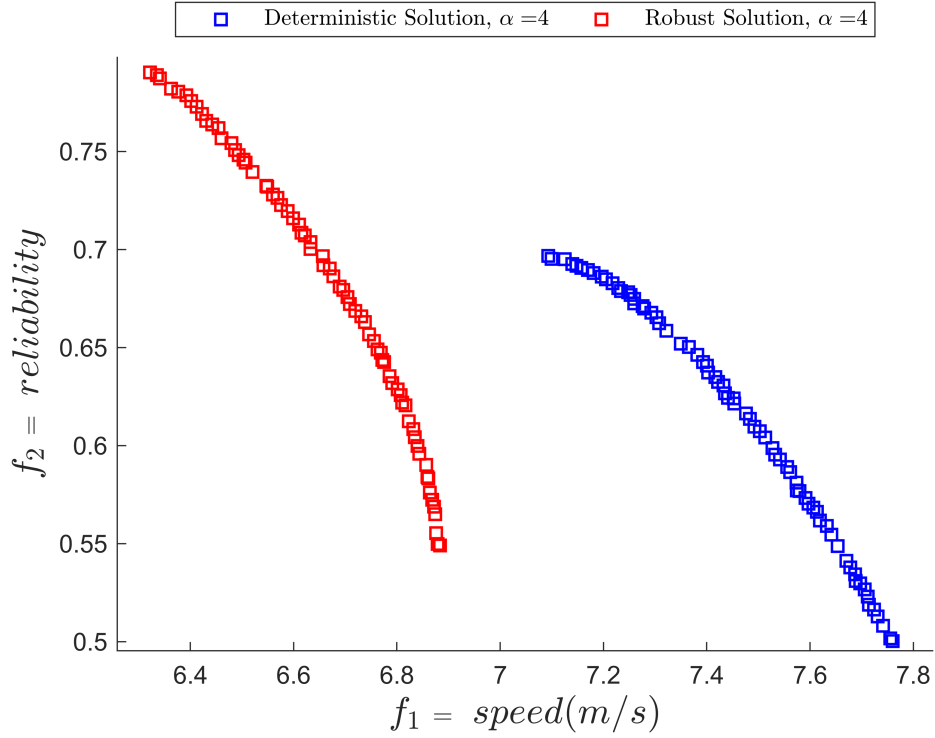


Figure 4.6: Pareto Front for USV case study

The results are shown in Figures. 4.6, 4.7, 4.8. Figure 4.6 shows the deterministic and robust solutions. It must be noted, however, that the MORO iterations were limited to obtain these results, hence the robust solutions presented here are "semi-robust" which means they have not fully converged to the final robustly optimal solutions. Nonetheless, the present solutions are feasible.

Figure 4.7 shows the throttle, heading and USV speed profiles for one of the optimal solutions (the best f_1 solution, i.e. the rightmost points of the Pareto fronts in Figure 4.6). Figure 4.8 shows the deterministic and robust paths for the optimal throttle and heading values presented in Figure 4.7.

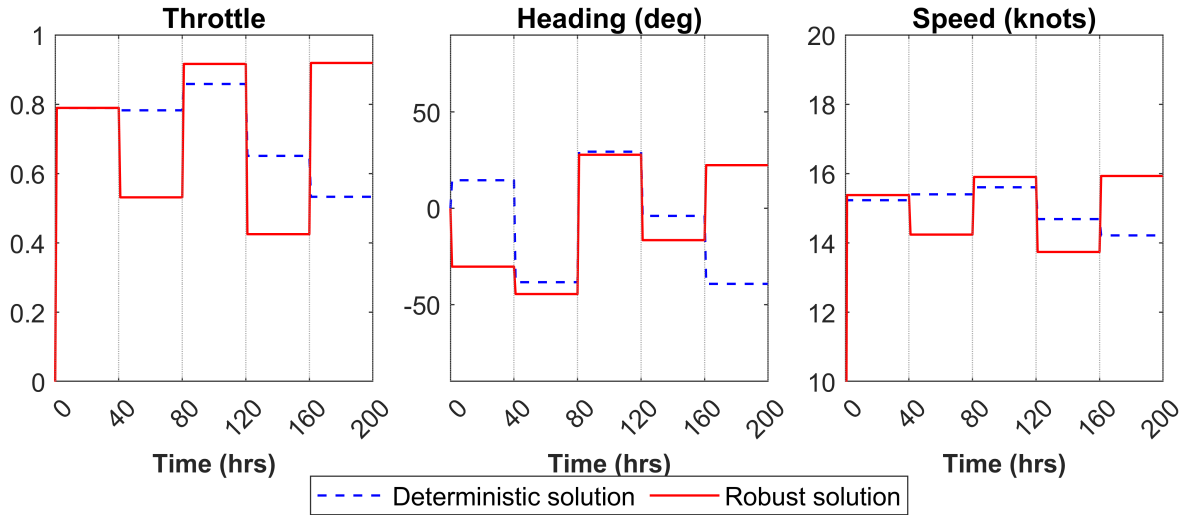


Figure 4.7: Optimized throttle, heading, and USV speed operational profiles

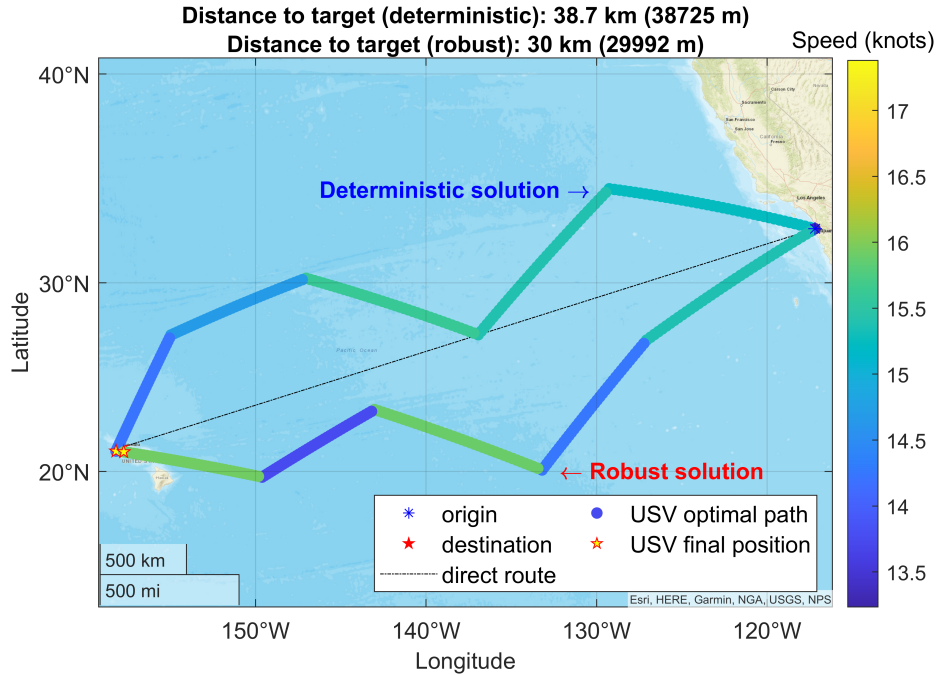


Figure 4.8: Optimized USV paths for deterministic and robust cases. The color of the path corresponds to the USV speed.

It is worth noting that, although both solutions are feasible with respect to the Kriging surrogate, the distances to target destination (as seen in Figure 4.8) for the solutions are greater than the 5km radius imposed in the problem constraint (3.6). This discrepancy is most likely due to

the inaccuracies of the surrogate models, and the "double surrogate" effect - recall, that offline surrogates were built to approximate the *USV model* (namely, the ECCS and hydrodynamic subsystems), and then the Kriging surrogate was built on top as part of SSB-MORO. Moreover, the Kriging surrogate was developed to model the *objective and constraint functions* of the problem - which do not mirror the USV model itself.

Upon further investigation, some insights were obtained revealing the challenges of developing the Kriging model for the USV case study. Firstly, the feasible design space is extremely small compared to the overall design space - hence, a lot more initial sample points need to be generated for building the initial surrogate to ensure that at least a few will cover the feasible domain. Secondly, the second constraint of the problem formulation, which governs the distance to target destination (Equation (3.6)), is extremely sensitive to variations in the uncertainty parameter. This makes the function very noisy, and leads to a high Kriging variance for that constraint, which makes the prediction less accurate. The DACE Kriging toolbox [113] that was used for this approach was originally developed for deterministic computer experiments, thus it does not natively accommodate highly noisy functions. This observation suggests that the Kriging model may not be the best one for the USV case study problem - perhaps an augmented Kriging model, which can support noisy data, would be more appropriate, or another surrogate model - such as Bayesian models or support vector machines.

Note also, that due to the surrogate inaccuracies, and since the MORO iterations were limited, the results here do not quite match the results for OSB-MORO. As seen in the Pareto plots (Figures 3.11 and 4.6), the extreme points for f_1 and f_2 do not match, hence, the throttle, heading and speed profiles, along with the USV paths also look different for SSB-MORO and OSB-MORO.

4.3 Comparison with Existing Approaches

The proposed SSB-MORO approach was compared with three other methods from the literature, which were selected based on the availability of published results. All of the following methods are iterative, sequential bi-level MORO methods, which have integrated surrogates. The bi-level MORO structure assumes that there is an upper-level problem, which searches the design space to find an optimum, and a lower-level problem, which performs the robustness evaluations on the obtained optimum points. To ensure robustness, the feasible design space is gradually reduced, by iteratively adding constraint cuts based on worst-case analysis of the uncertainty. The main difference between the following approaches is the surrogate model and its implementation. The first approach is an approximation-assisted MORO (AA-MORO), which utilizes a Kriging surrogate model and a constraint cut approach to find the robust solution [10]. The surrogate model is updated based on the optimal solution points obtained from the upper level problem. The second approach, B-MORO, uses a Bayesian framework to solve the MORO problem via a modified expected improvement Bayesian acquisition function [18]. The surrogate model is updated based on the optimal solution points obtained from the upper level problem as well as the solutions from the lower-level problem (the worst-case uncertainties). The third approach, MORO-KS, adopts the optimization structure based on constraint cuts and utilizes Kriging to approximate the objective functions and a support vector machine (SVM) classifier to determine the feasibility of design alternatives [61]. This approach differs from the rest in that it develops two sets of surrogates and handles them separately. The Kriging regression model to approximate the objectives is updated based on the optimal solutions from the upper-level problem; and the SVM classifier for the constraints is updated based on the uncertainty samples.

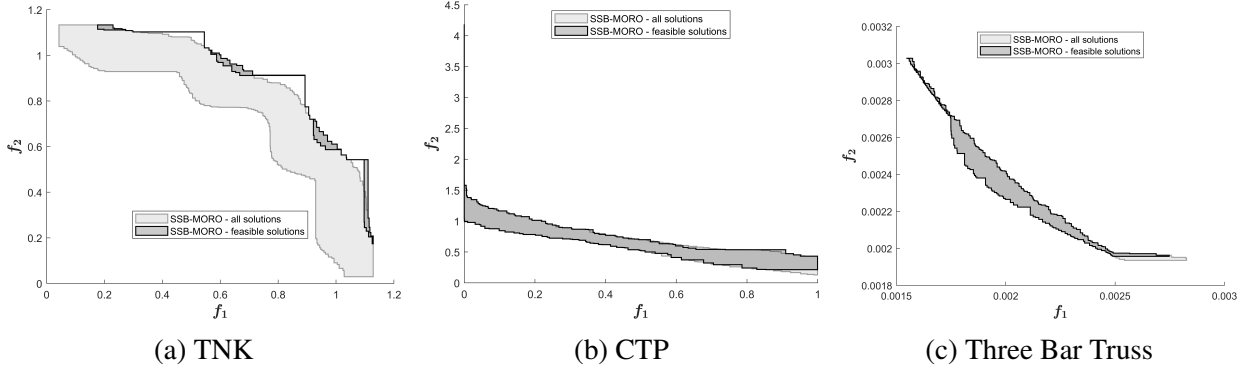


Figure 4.9: Solutions for TNK, CTP, and Three-Bar Truss problems - range of SSB-MORO solutions from 10 independent runs with all points, and with infeasible points removed.

To provide a fair and more accurate comparison among approaches, the proposed method was implemented under the same conditions as the others, as much as possible - i.e., with the same solver settings and function call limits as presented in [12] (refer to Tables 4.2 and 4.8 for details). Intuitively, it is obvious that increasing the function call limit would yield better converged solutions - however, then it would be difficult to provide a fair comparison, since the results would be obtained under different assumptions.

Three example problems - TNK, CTP, and Three-Bar Truss - were solved with the SSB-MORO algorithm under the conditions specified (see Tables 4.2 and 4.8). The results are shown in Figure 4.9. The shaded areas contain the range of solutions obtained from SSB-MORO (after solving the problem 10 times). The light grey area represents all the solutions obtained from SSB-MORO, whereas the dark grey area represents the solutions obtained from SSB-MORO with the infeasible points removed (where the feasibility of a point was determined from the Monte Carlo verification). From the figures it can be seen that not all of the solutions obtained by SSB-MORO are truly robust - this can be explained by the limit imposed on function calls, which was done to provide a fair comparison with the other approaches from the literature. Since the function calls were limited, it is natural to assume that the algorithm did not have enough time to converge to a

more accurate final solution - hence the reduced number of feasible, truly robust, points.

In order to evaluate the performance of the proposed SSB-MORO method, three quality metrics were used - the hyperarea difference (HD) [41], the overall spread (OS) [41], and spacing (SP) [42]. The quality metrics were calculated and averaged across all runs and compared between selected approaches - the results are presented in Table 4.8. Figure 4.10 provides a visual comparison of the means and standard deviations of each quality metric for the example problems. Recall, that a higher OS is better and lower HD and SP values are better. Note that the quality metrics values should be between 0 and 1, however, in some cases they exceeded their limit. This is likely due to the estimation error of the "good" and "bad" points. The quality metric calculations are performed on the normalized objective function values (between 0 and 1) - hence an estimate of the "good" and "bad" points must be provided beforehand. However, if some points from the solution set yield objective function values that are outside of the bounds of the "good" and "bad" points, then the normalization becomes inaccurate, forcing some normalized values to be lower than 0 or greater than 1, which would lead to errors in the quality metric calculation.

As can be seen, SSB-MORO outperforms the other approaches across all example problems (except CTP) in terms of the SP metric - meaning SSB-MORO is able to consistently provide more uniform results. For the OS metric, the proposed approach is better for the TNK and CTP problems, and is second-best for the three-bar truss problem. Finally, with respect to the HD metric, the proposed approach outperformed the rest only for the TNK problem.

An additional statistical analysis was performed to determine whether there was a significant difference between the various approaches. Due to the non-homogeneous variances, Welch's ANOVA [116] was performed, considering the means and standard deviations of the quality met-

Table 4.8: Example Results Comparison

Example	Approach	Function Calls	HD mean (std)	OS mean (std)	SP mean (std)
TNK	SSB-MORO	130	0.513 (0.076)	0.470 (0.027)	0.005 (0.001)
	AA-MORO	130	0.634 (0.224)	0.176 (0.176)	0.122 (0.197)
	B-MORO	130	0.618 (0.044)	0.402 (0.118)	0.043 (0.016)
CTP	SSB-MORO	200	0.527 (0.103)	3.047 (1.180)	0.058 (0.059)
	AA-MORO	200	0.571 (0.268)	0.122 (0.103)	0.029 (0.016)
	B-MORO	200	0.345 (0.014)	0.186 (0.120)	0.181 (0.090)
Three-Bar Truss	SSB-MORO	130	0.488 (0.006)	0.604 (0.130)	0.007 (0.002)
	AA-MORO	130	0.496 (0.015)	0.506 (0.086)	5.776 (0.991)
	B-MORO	130	0.491 (0.003)	0.719 (0.071)	14.375 (2.917)
	MORO-KS	2240	0.358 (0)	0.304 (0)	6.826 (0)

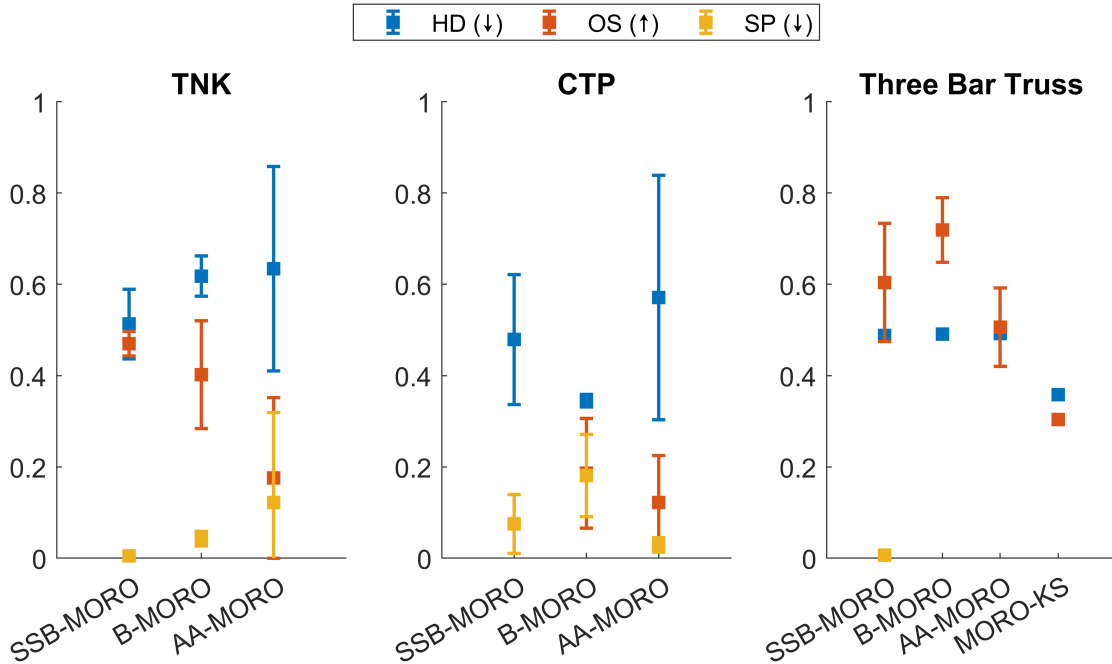


Figure 4.10: Comparison of quality metrics (mean and standard deviation) for TNK, CTP and Three Bar Truss problems - lower HD and SP is better, higher OS is better.

rics and the different approaches. Then, Games-Howell's [117] post-hoc test was applied to determine which approaches were significantly different from each other.

The results are summarized in Table 4.9 for problems TNK, CTP, and Three-Bar Truss.

The rows and columns of the tables represent the approaches. The cells of the intersections show

Table 4.9: ANOVA Results for Example Problems

Approach	AA-MORO	B-MORO	MORO-KS
TNK			
SSB-MORO	OS	HD, SP	–
AA-MORO	–	OS	–
CTP			
SSB-MORO	OS	HD, OS, SP	–
AA-MORO	–	SP	–
Three-Bar Truss			
SSB-MORO	SP	SP	HD, OS, SP
AA-MORO	–	OS, SP	HD, OS, SP
MORO-KS	–	HD, OS, SP	–

whether there is a difference between the two approaches, statistically speaking. The dash (“–”) means the comparison is not applicable; if the approaches were deemed statistically different, then the cell contains the quality metrics for which a difference was determined.

Based on the results, it can be inferred that overall the approaches are different - the analysis from Welch’s ANOVA suggests that there are statistically significant differences between the quality metrics across the compared approaches. When considering the various approaches, it appears that more differences arise for harder problems, such as CTP and Three-Bar Truss, whereas the approaches yield more similar results for simpler problems, such as TNK. For example, in Table 4.9, for the TNK problem - the results for SSB-MORO and AA-MORO are statistically different only with respect to the OS quality metric; with respect to the HD and SP metrics, SSB-MORO and AA-MORO yield results that are not statistically different. Similarly, when comparing SSB-MORO and B-MORO for the TNK problem, the HD and SP metrics are statistically different, while the OS is not. Now, if we consider the more complex three-bar truss problem, there is a significant difference in the SP metric across SSB-MORO, AA-MORO, and B-MORO; and when comparing SSB-MORO and MORO-KS, there is a significant difference for

all quality metrics.

The ANOVA results from Table 4.9 can be used in tandem with the quality metrics comparison (Table 4.8, Figure 4.10) to determine the better approach for a certain problem. For example, consider the TNK problem. The mean HD value is best for SSB-MORO, followed by B-MORO and then AA-MORO (recall, smaller HD is better). Now, referring to Table 4.9, there is only a difference between SSB-MORO and B-MORO with respect to HD; there is no statistical difference between SSB-MORO and AA-MORO with respect to HD. Hence, for the TNK problem, considering the HD metric, it can be concluded that SSB-MORO performs better than B-MORO, but similar to AA-MORO. Following the same logic, with respect to the OS metric, SSB-MORO performs better than AA-MORO (higher OS mean value and significant statistical difference) and similar to B-MORO (although the OS mean value is higher, it is not statistically significantly different).

4.4 Discussion

The approach proposed in this chapter contributes to the existing literature by developing a new online surrogate sampling-based MORO approach. The proposed approach, SSB-MORO, is an extension of a fixed sampling-based approach (OSB-MORO, presented in Chapter 3) [2], which performs calculations on the original problem functions. SSB-MORO also follows an iterative sequential approach, where the optimization problem is first solved, then the robustness aspect is handled via a scenario generation scheme. However, unlike OSB-MORO, the proposed approach integrates the implementation of an online surrogate model to improve computational effort. The surrogate is based on a Kriging model and is adaptively updated during each MORO

iteration, while considering the decision variable and uncertainty spaces to ensure a promising next point selection.

The proposed approach was demonstrated on a few example problems and compared with some existing methods from the literature - overall, it produces comparable results, and excels in obtaining a more uniform solution set (as shown with the SP quality metric).

However, SSB-MORO has some limitations and areas of improvement. The most prominent concern is the surrogate accuracy. As seen for more complex problems, SSB-MORO can return sub-optimal or inconsistent results. Future work can include exploring tuning the Kriging model parameters (such as the correlation function), as well as alternative surrogate modelling strategies and integration methods to enhance the convergence behavior and robustness of the final solution.

The next chapter will dive into the final research thrust of this dissertation, where decision-dependent uncertainty is considered in the MORO formulation.

Chapter 5: Sampling-Based Single-Objective Robust Optimization with Decision-Dependent Uncertainties (SB-SORO-DDU)

This chapter will focus on describing a sampling-based single-objective robust optimization method with decision-dependent uncertainties (SB-SORO-DDU). The problem formulation is provided in Equation (5.1):

$$\begin{aligned}
 & \min_{\mathbf{x}} f(\mathbf{x}, \mathbf{u}) \\
 & \text{subject to:} \\
 & g_j(\mathbf{x}, \mathbf{u}) \leq 0; j = 1, \dots, J \\
 & \mathbf{x} \in [\mathbf{x}_{lb}, \mathbf{x}_{ub}] \\
 & \forall \mathbf{u} \in [\mathbf{u}^* + \Delta \mathbf{u}] \\
 & \Delta \mathbf{u} \in [\Delta \mathbf{u}_{lb}, \Delta \mathbf{u}_{ub}]
 \end{aligned} \tag{5.1}$$

where

$$\mathbf{u}^* = h(\mathbf{x})$$

Note, although the formulation in Equation (5.1) contains uncertainty in the objective function, only feasibility robustness is considered. The effect of uncertainty on the objective function is ignored - there is no additional AOVR constraint to limit the variation in the objective function

value due to uncertainty, and the objective function variation, Δf , is not included as an additional term to the problem objective function. Note, that the uncertainty range, $\mathbf{u} \in [\mathbf{u}^* + \Delta \mathbf{u}]$, is comprised of the nominal uncertainty value, $\mathbf{u}^*$, and the uncertainty variation, $\Delta \mathbf{u}$. The nominal uncertainty value is dependent on the decision variables (i.e., is a function of the decision variables) - $\mathbf{u}^* = h(\mathbf{x})$, where $h(\mathbf{x})$ is the function that describes the relationship between the uncertainties and decision variables. In this formulation, we assume the same function for all uncertain parameters, however, the approach can be extended to consider unique functions for each decision-dependent uncertain parameter. The uncertainty variations, $\Delta \mathbf{u}$, are assumed to be fixed for each uncertain parameter. Thus, the overall uncertainty bounds are defined as follows in Equation (5.2):

$$\mathbf{u} \in [\mathbf{u}_{lb}, \mathbf{u}_{ub}]$$

where

$$\mathbf{u}_{lb} = \mathbf{u}^* - \Delta \mathbf{u}_{lb} \tag{5.2}$$

$$\mathbf{u}_{ub} = \mathbf{u}^* + \Delta \mathbf{u}_{ub}$$

$$\mathbf{u}^* = h(\mathbf{x})$$

Given the relationship between the uncertain parameters and decision variables, it is apparent that the interval bounds become *variable*, rather than *fixed*, as in previous approaches presented in this dissertation.

The details of the SB-SORO-DDU algorithm will be presented in Section 5.1, followed by an illustrative example (Section 5.2.1) and application to the USV case study (Section 5.2.2). A summary and discussion of the proposed approach - highlights, limitations, and future directions

- concludes this chapter (Section 5.3).

5.1 Overview of Approach

The SB-SORO-DDU approach, similar to OSB-MORO, is a single-level, iterative method, where the uncertainty is handled via a scenario generation scheme. On a high level, the two approaches are very similar and follow the same steps.

A flowchart of the SB-SORO-DDU approach is shown in Figure 5.1 and the pseudocode is provided in Algorithm 4. The main (and significant) difference is in how the newly generated uncertainty samples are handled in the robustness evaluation step (**Step 4** in Figure 5.1). The details will be described later in this chapter.

Similar to the other approaches presented in this dissertation, the first step of SB-SORO-DDU is to initialize the problem and define the relationship between the uncertain parameters and decision variables, $\mathbf{u}^* = h(\mathbf{x})$. Then, the optimization problem can be solved to obtain the optimal solution, $\mathbf{x}^*$, and the associated nominal uncertain parameters, $\mathbf{u}^*$. It is important to note that the function defining the relationship between the uncertainty parameters and decision variables is not treated as additional inequality constraint for the optimization problem - it is merely a *calculation* that is performed inside, or as part of, the constraint function. Next, new uncertainty scenarios are generated using the LHS method. The samples are generated from an interval range that is defined by the current value of the nominal uncertainty parameter, $\mathbf{u}^*$, and the uncertainty variation bounds, $\Delta \mathbf{u}$. Following the scenario generation step, is the robustness evaluation - each constraint is evaluated for all realizations of the uncertainty parameters. The constraint with the maximum violation is added to the optimization problem. This concludes a SORO iteration. For

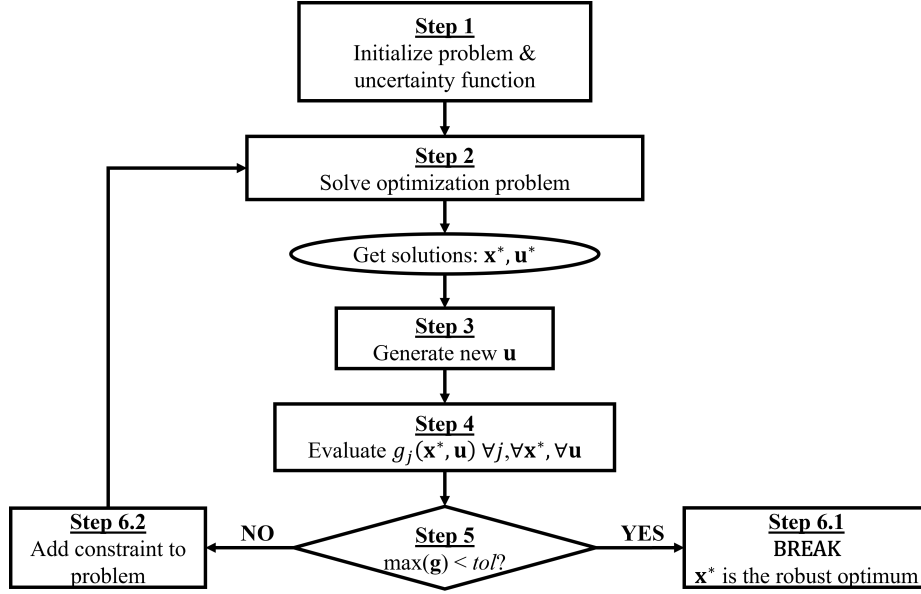


Figure 5.1: Flowchart of SB-SORO-DDU Approach

the next iteration, the procedure repeats, starting from solving the optimization problem, now with the added constraint. The algorithm stops when none of the constraints are violated during the robustness evaluation stage. Thus, the solution obtained at the current iteration, $\mathbf{x}^*$, is the robust optimum.

In further discussion of the details of the approach, **Step X** will refer to the corresponding step in the flowchart (Figure 5.1) and Line A.Y will refer to the line number Y of the pseudocode for Algorithm A (defined in Algorithm 4). Most of the steps overlap with the OSB-MORO approach significantly, so the reader is encouraged to refer to the appropriate steps in Section 3.1 for more details, however, the key differences will be highlighted here.

Algorithm 3: SB-SORO-DDU algorithm

Input: $problemDef, ROptions$
Output: $[det_{soln}, ro_{soln}, all_{soln}]$

```
/* Step 1 - Initialization */
1  $u_{infeas} \leftarrow []$ ;
2  $con_{infeas} \leftarrow []$ ;
3 for  $iter = 1$  to  $maxIter$  do
    /* Step 2 - Solve optimization problem */
    4 if  $iter = 1$  then
        5  $[x^*, f] \leftarrow solverFctn(problemDef, u_{infeas}, con_{infeas})$ ;
        6  $u^* \leftarrow h(x^*)$ ;
        7  $det_{soln} \leftarrow [x^*, f]$ ;
    8 else
        9  $[x^*, f] \leftarrow solverFctn(problemDef, u_{infeas}, con_{infeas})$ ;
        10  $u^* \leftarrow h(x^*)$ ;
        11  $all_{soln}(iter) \leftarrow [x^*, f]$ ;
    /* Step 3 - Generate  $S$  uncertainty samples */
    12  $u_{new} \leftarrow generateNewU(problemDef, S)$ ;
    /* Step 4 - Evaluate constraints */
    13  $g \leftarrow []$ ;
    14 for  $sample = 1$  to  $S$  do
        15  $u \leftarrow u_{new}(sample)$ ;
        16  $\hat{u} \leftarrow checkUsample(u, x^*)$ ;
        17  $g \leftarrow evaluateConstraints(problemDef, x^*, \hat{u})$ ;
    /* Step 5 - Check constraint feasibility */
    18 if  $\max(g) < tol$  then
        19 break;
    /* Step 6 - Add worst-case constraint to problem */
    20 else
        21  $[u_{infeas}, con_{infeas}] \leftarrow findMaxViolation(g, u_{new})$ ;
        22  $problemDef.new\_constr \leftarrow [u_{infeas}, con_{infeas}]$ ;
23  $ro_{soln} \leftarrow [x^*, f]$ ;
```

Algorithm 4: checkUsample function

Input: u, x^* **Output:** $\hat{u}$

```
1  $\hat{u} \leftarrow u$  ;  
2  $u^* \leftarrow h(x^*)$  ;  
3  $u_{lb} \leftarrow u^* - \Delta u$  ;  
4  $u_{ub} \leftarrow u^* + \Delta u$  ;  
5 for  $i \leftarrow 1$  to  $\text{len}(\hat{u})$  do  
6   if  $\hat{u}[i] < u_{lb}[i]$  then  
7      $\hat{u}[i] \leftarrow u_{lb}[i]$  ;  
8   if  $\hat{u}[i] > u_{ub}[i]$  then  
9      $\hat{u}[i] \leftarrow u_{ub}[i]$  ;
```

Step 1: Initialize problem

In this step, the problem objective and constraint functions are defined, as well as the function governing the relationship between the uncertainty parameters and decision variables. The inputs to the algorithm, *problemDef* and *ROoptions* are the same as for OSB-MORO (see **Step 1** in Section 3.1).

Step 2: Solve optimization problem

In this step the optimization problem is solved. Recall, the formulation for the problem is as shown in Equation (5.1), where the uncertainty is variable, rather than fixed.

Lines 3.4-3.11: The `solverFctn` solves the optimization problem, with the uncertainty calculation, $u^* = h(x)$, performed inside the constraint function (and not as an additional equality constraint to the problem). Depending on the problem, one of two solvers were implemented - MATLAB's `fmincon`, which is an interior-point algorithm [112], or `ga`, which is a genetic algorithm [118]. Since this solution was obtained during the first iteration of the SORO algorithm, before any additional constraints are added to account for uncertainty variation, this is the deterministic solution.

Line 3.6: Once the optimal solution is found, the associated new nominal, or "optimal", uncertainty parameters can be calculated.

Step 3: Generate uncertainty samples

In this step, a set of new uncertain parameters is generated via the LHS method. Another difference to note (compared to the OSB-MORO approach), is that the uncertainty space to sample from is dependent on the "optimal" uncertainty value, u^* , and is defined by Equation (5.2).

Step 4: Evaluate Constraints

For the constraint evaluation step, each constraint is evaluated at the optimal solution, x^* , and for all realizations of the uncertainty parameter.

Line 3.16: Since the uncertainty is decision-dependent, each time a new sample is about to be evaluated, it needs to be checked, to ensure it is within bounds of the for the current solution, x^* . If it is not within bounds, the outlying uncertain parameter is set to the value of the bound it is closest to. The details of the `checkUsample` function are outlined in Algorithm 5.

Step 5: Check constraint feasibility

Once all of the constraints have been evaluated for all uncertainty scenarios, the maximum constraint violation can be determined.

Step 6.1: BREAK

The procedure is terminated if the feasibility check from **Step 5** is satisfied. The solution obtained at the current iteration is returned as the robust optimal solution.

Step 6.2: Add worst-case constraint to problem

The constraint with the maximum violation is added to the problem, with the associated uncertainty sample that led to infeasibility. This completes a SORO iteration of the algorithm. The next iteration continues from **Step 2**. Note, however, that since the uncertainty is decision-dependent,

the added constraint (with a specific u_{infeas}) may not be feasible for a new point, x . In other words, for a new point, x , the associated uncertainty bounds are within $u \in [h(x) \pm \Delta u]$, however, u_{infeas} may be outside of those bounds. In that case, the u_{infeas} is clipped to be within the current uncertainty bounds (as performed in **Step 4**, Line 3.16, see Algorithm 5).

5.1.1 Computational Effort

The computational effort of the SB-SORO-DDU approach (Equation (5.3)) depends on a few factors:

- a) the number of iterations of the SORO algorithm required to achieve a robust solution (I);
- b) the number of function calls required for the solver (G);
- c) the number of uncertainty samples (S) generated to evaluate the robustness of the obtained solution

$$\mathcal{O}(\text{SB-SORO-DDU}) = I \times (G + S) \quad (5.3)$$

It can be seen that the complexity of the SB-SORO-DDU algorithm scales linearly with respect to the number of uncertainty samples, and how it scales with respect to the solver depends on the solver chosen for the optimization. Thus, the driving factor of the algorithm complexity would depend on the solver chosen for the optimizer - if the solver's complexity is better than linear, then the driving factor is the number of uncertainty samples; if the solver's complexity is worse than linear, then it governs the complexity of the SB-SORO-DDU algorithm.

5.2 Examples

This section demonstrates the proposed SB-SORO-DDU approach to some example problems - an illustrative numerical example and USV case study.

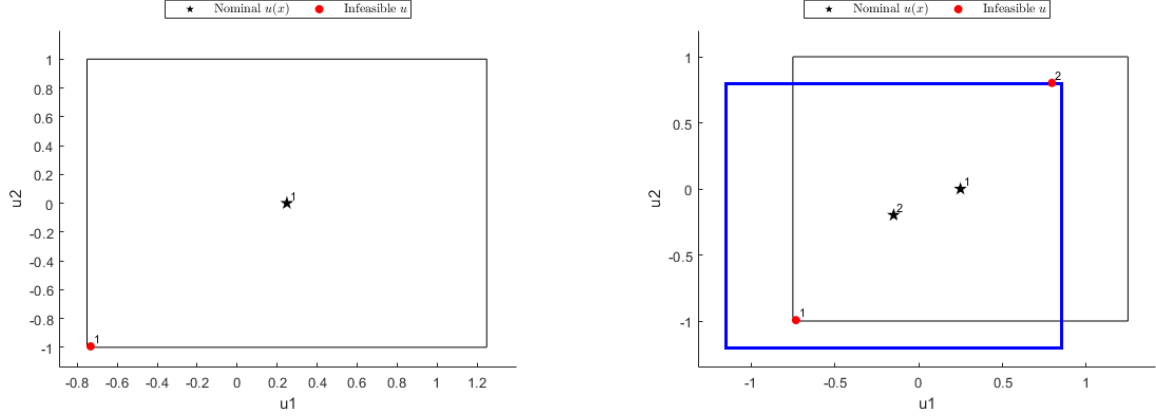
5.2.1 Simple Circle Problem

The proposed approach was applied to a simple numerical problem to illustrate the steps of the approach and provide a clearer picture of the shifting, or variable, uncertainty space.

The circle problem was adapted from Ref. [9] to include DDU. This is a simple non-convex problem, with a single constraint, two decision variables and two uncertain parameter. The formulation is presented in Equation 5.4.

$$\begin{aligned} & \min -x_1^2 - x_2^2 \\ & \text{subject to:} \\ & g(\mathbf{x}, \mathbf{u}) = (x_1 - u_1)^2 + (x_2 - u_2)^2 - 5 \leq 0 \\ & -5 \leq x_i \leq 5, \forall i = 1, 2 \\ & \forall \mathbf{u} \in [\mathbf{u}^* + \Delta \mathbf{u}] \\ & \Delta \mathbf{u} \in [-1, 1] \\ & \text{where} \\ & \mathbf{u}^* = \mathbf{x}/10 \end{aligned} \tag{5.4}$$

The problem was solved via MATLAB's `fmincon` function [112], which is an implementation of the interior-point algorithm. The initial point for the problem was set to $\mathbf{x}_0 = [0.5, 0]$



(a) Uncertainty space for iteration #1 of the Circle problem

(b) Uncertainty space for iteration #2 of the Circle problem

Figure 5.2: Evolution of uncertainty space for Circle problem

and the maximum number of iterations for the interior-point algorithm ('MaxIterations' option) was set to 100. The remaining solver options were left at their default values.

The first iteration of SB-SORO-DDU returned the solution $\mathbf{x}^* = [2.485, 4.369 \times 10^{-4}]$ with the objective function value $f = -6.173$. The associated nominal uncertainty value was calculated as $\mathbf{u}^* = [0.248, 4.369 \times 10^{-5}]$ and is shown as the star in Figure 5.2a, with the bounds of the uncertainty space defined by the black rectangle. After the robustness evaluation was performed, the constraint with the maximum violation was added to the problem - the uncertainty value that led to the infeasibility was $\mathbf{u}_{infeas} = [-0.733, -0.993]$ and is shown as the red dot in Figure 5.2a.

Now, for the second iteration the problem was resolved with the added constraint, as shown in Equation (5.5):

$$\begin{aligned}
 g_1(\mathbf{x}, \mathbf{u}) &= (x_1 - u_1)^2 + (x_2 - u_2)^2 - 5 \leq 0 \\
 g_2(\mathbf{x}, \mathbf{u}) &= (x_1 + 0.733)^2 + (x_2 + 0.993)^2 - 5 \leq 0
 \end{aligned} \tag{5.5}$$

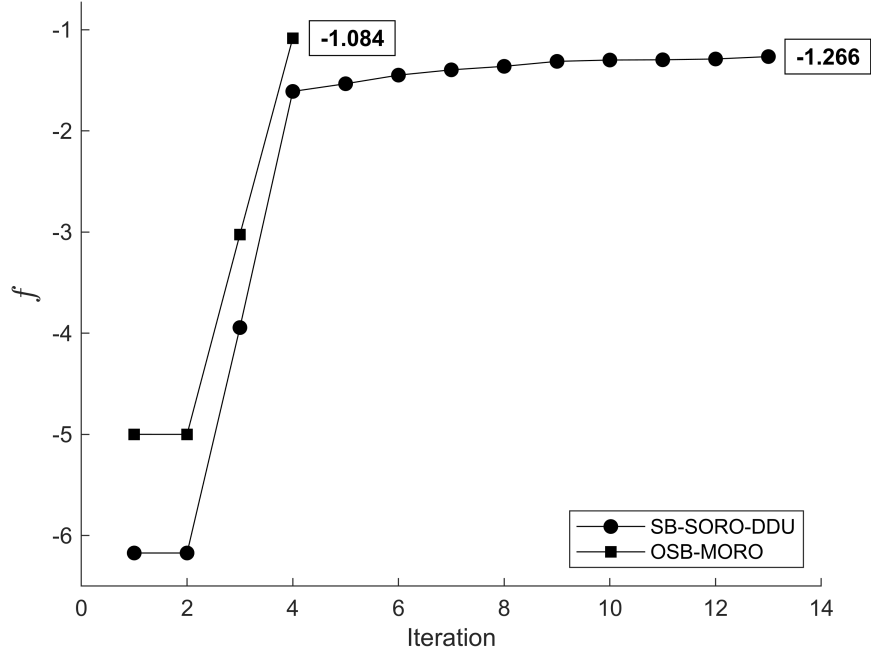


Figure 5.3: Progression of objective function value for Circle problem - for SB-SORO-DDU and OSB-MORO approaches

The obtained solution was $\mathbf{x}^* = [-1.502, -1.979]$ with the objective function value $f = -6.173$. The associated uncertainty was $\mathbf{u}^* = [-0.150, -0.198]$ and is shown as the star labelled '2' in Figure 5.2b, with the bounds of the uncertainty space defined by the blue rectangle. The uncertainty that led to the maximum constraint violation was $\mathbf{u}_{infeas} = [-0.733, -0.993]$ and is shown as the red dot labelled '2' in Figure 5.2b. It can be seen how the uncertainty space has shifted due to the new nominal uncertainty value, $\mathbf{u}^*$.

The SB-SORO-DDU algorithm continues, gradually converging to a robust optimal solution $\mathbf{x}^* = [1.125, -0.003]$ and $f = -1.266$, as shown in Figure 5.3. It can be seen that it takes a few iterations for the objective function value to gradually converge to its final value. Note, as the algorithm progresses, the uncertainty space also converges - as seen in Figure 5.4, by the overlapping black stars after iteration 3.

The problem was run 10 times to account for stochasticity, and the results (mean and stan-

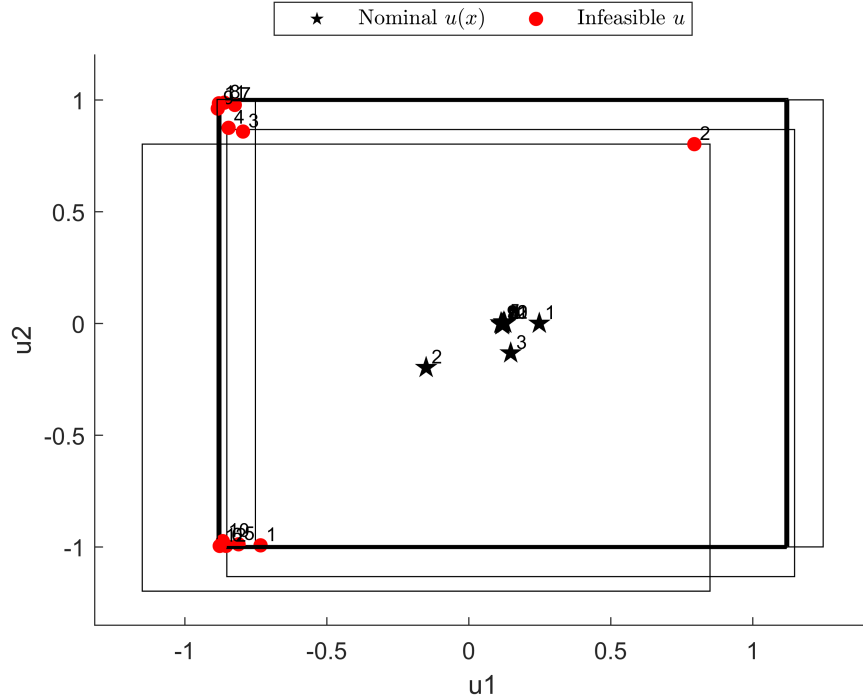


Figure 5.4: Uncertainty space for all iterations of the Circle problem

Table 5.1: Summary of results for Circle problem

	Function Calls	Final Objective Function Value
SB-SORO-DDU mean (std)	8,838 (1,965)	-1.319 (0.030)
OSB-SORO mean (std)	5,364 (1,073)	-1.066 (0.022)

dard deviation (std)) are presented in Table 5.1. The problems was also solved using a single-objective implementation of the OSB-MORO approach, without considering the decision dependent uncertainty to compare the solutions.

Although the formulations for the Circle problem for each approach (SB-SORO-DDU and OSB-SORO) are slightly different (one with variable, the other with fixed interval uncertainty), a comparison of the results can provide some high-level general insights.

The number of function calls for SB-SORO-DDU is higher than for OSB-SORO. As shown in Figure 5.3, the OSB-MORO approach takes 4 MORO iterations to converge to a robust solution, whereas the SB-SORO-DDU approach requires 13, thus yielding a higher function call

count. This behaviour is expected, since the DDU approach takes longer to converge due to the constantly shifting uncertainty space. The objective function value, however, is better for the SB-SORO-DDU approach. This is consistent with the theory found in the literature - when uncertainty is modelled as decision-dependent, the uncertainty sets are more realistic and adaptive, thus leading to less conservative and often better solutions, compared to traditional RO approaches where the uncertainties are fixed.

5.2.2 USV Case Study

The proposed SB-SORO-DDU approach was also applied to the USV case study. For a detailed description of the individual models and integration, refer to Appendix B.

The formulation for the USV case study is similar to the one for the USB-MORO case (see Section 3.3.2), and is shown in Equation (5.6).

The objective is to maximize the utility function (which is defined in Equation (B.8)). Constraints are imposed on the sway force (F_y), the final distance to the destination (d)m and the system reliability (R). As a reminder, the system reliability was modified following Equation (B.5) in Appendix B.3 with $\alpha = 1$. The uncertainties are manifested in the current directions and current speeds. Note, since in this formulation we are considering decision dependent uncertainty, the current directions and speeds are determined from the environment map, based on the position of the USV (see Appendix B.5 for details).

$$\begin{aligned}
& \min\{U(\mathbf{x}, \mathbf{u})\} \\
& \text{subject to:} \\
& g_1(\mathbf{x}, \mathbf{u}) = \max(F_y)/350kN - 1 \leq 0 \\
& g_2(\mathbf{x}, \mathbf{u}) = d/5km - 1 \leq 0 \\
& g_3(\mathbf{x}, \mathbf{u}) = -R/0.5 + 1 \leq 0 \\
& 0 \leq x_i \leq 1, \forall i = 1, 2, 3, 4, 5 \\
& -30^\circ \leq x_6 \leq 30^\circ; \quad -90^\circ \leq x_i \leq 90^\circ, \forall i = 7, 8, 9, 10 \\
& \mathbf{u} = \mathbf{u}^* + \Delta\mathbf{u} \\
& \mathbf{u}^* = h(\mathbf{x}) \\
& \forall \Delta\mathbf{u} \in \pm[5^\circ, 5^\circ, 5^\circ, 5^\circ, 5^\circ, 0.01m/s, 0.01m/s, 0.01m/s, 0.01m/s, 0.01m/s]
\end{aligned} \tag{5.6}$$

The optimization solver used was MATLAB `fmincon` function [112], which is an implementation of the interior-point algorithm. The solver and RO options are summarized in Table 5.2. The remaining options were set to their default values.

The optimized decision variables (throttle and heading) and resulting USV speed are shown in Figure 5.5, along with the corresponding paths in Figure 5.6. Figure 5.7 shows the profiles of the attributes and utility.

It is interesting to see that the USV headings are the same for both the deterministic and robust cases, and the throttle and speed profiles are very similar as well. One possible explanation for the lack of uniqueness in the solutions is the presence of decision-dependent uncertainty. In contrast, the solutions appeared to be more unique for the USB-MORO approach (refer back to

Table 5.2: `fmincon` and RO option settings for USV case study (SB-SORO-DDU approach)

fmincon options	
Algorithm	interior-point
Subproblem Algorithm	cg
Constraint Tolerance	1×10^{-3}
Finite Difference Step Size	1×10^{-4}
Optimality Tolerance	1×10^{-3}
Step Tolerance	1×10^{-6}
RO options	
Maximum iterations	10
Number of uncertainty samples	100

Table 5.3: Comparison of objective function values for SB-SORO-DDU and USB-MORO

	Deterministic	Final
SB-SORO-DDU	-0.041	-0.038
USB-MORO	-0.037	-0.053

Section 3.3.2). There are a few additional comparisons that can be made between the results from the SB-SORO-DDU approach and the USB-MORO approach. Table 5.3 summarizes the mean utility values for the two approaches. Perhaps the first observation would be regarding the utility values for the deterministic cases. It is expected that the solutions will be different since two different problems were solved - one with variable uncertainty, the other with fixed uncertainty. Thus, the nominal values of the uncertain parameters are different for both problems, which would yield different results - compare the "Current speed" attribute profile for SB-SORO-DDU and USB-MORO in Figures 5.7 and 3.15, respectively. The utility values for the robust solutions also make sense - as demonstrated in the Circle problem and as supported by the literature, the robust solution with DDU is expected to be better than its fixed uncertainty counterpart.

Again, some robust attribute values are very similar to their deterministic counterparts, which, given the same heading, is not entirely surprising. For the deterministic case, the utility

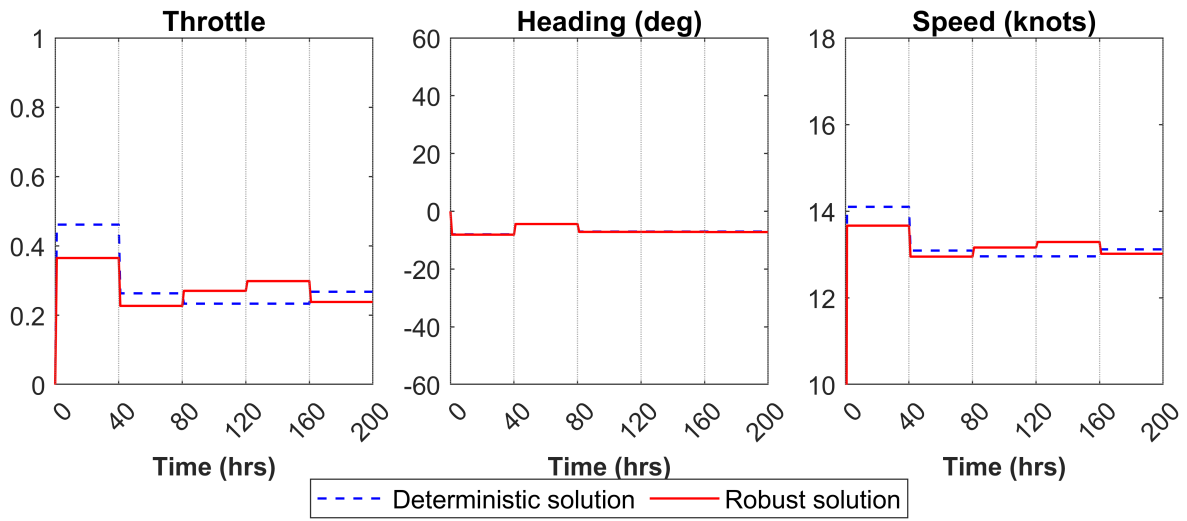


Figure 5.5: USV inputs for SB-SORO-DDU (deterministic and robust cases)

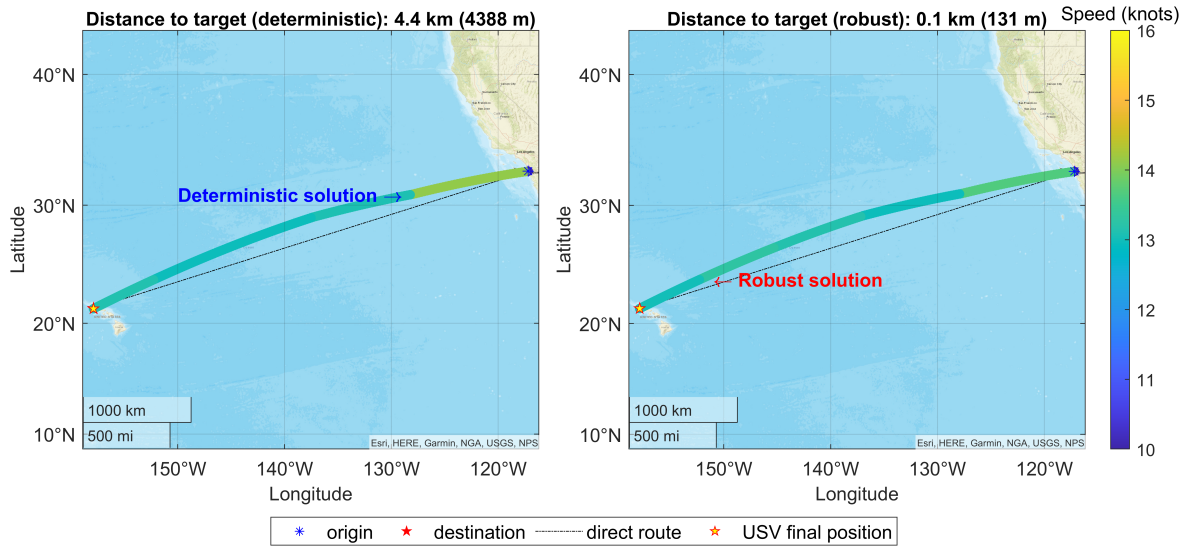


Figure 5.6: USV paths for SB-SORO-DDU (deterministic and robust cases)

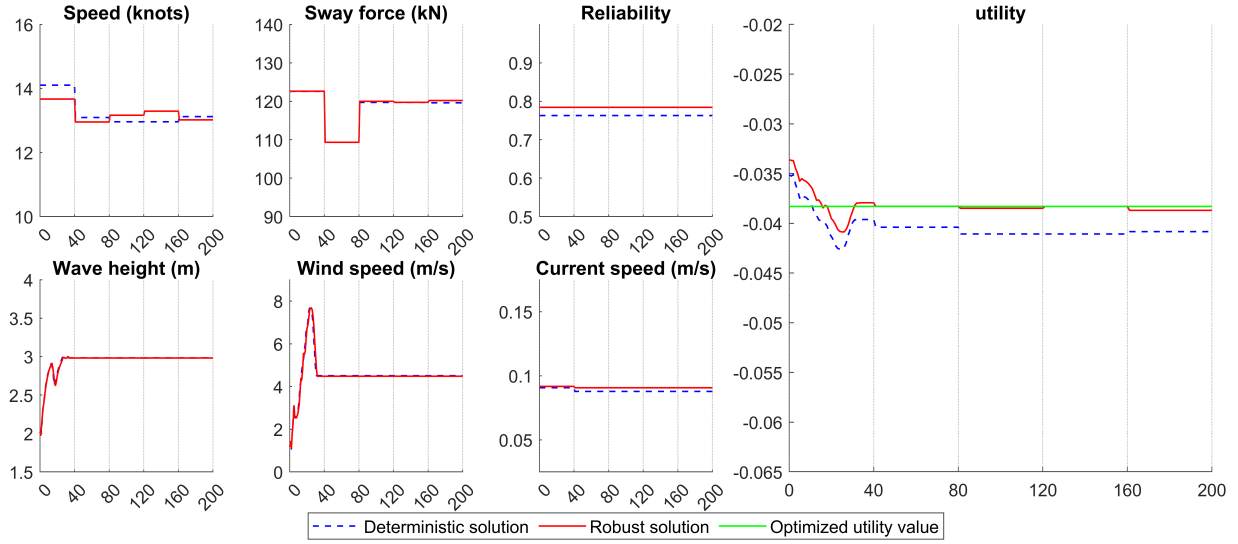


Figure 5.7: USV attribute and utility profiles for SB-SORO-DDU (deterministic and robust cases)

value averages at -0.041, whereas for the robust case the average utility is -0.038. Although a better objective function value for the robust case is atypical for traditional MORO approaches, it is possible that the same rules do not apply to RO problems with DDUs. Especially if the uncertain parameters also appear in the objective function.

5.3 Discussion

This chapter introduced a novel sampling-based robust optimization method, for solving problems with decision-dependent uncertainty (SB-SORO-DDU). Unlike previous approaches, the proposed method addresses optimization problems in which the uncertain parameters are dependent on the decision variables, leading to variable (rather than fixed) uncertainty bounds.

One of the main advantages of the proposed approach is its simplicity - due to the sampling-based nature of handling uncertainties, the problem characteristics do not change - i.e. the problem does not become NP-hard due to the decision-dependent uncertainty.

The algorithm was demonstrated with two examples: a simple Circle problem and a USV

case study. Results showed that SB-SORO-DDU can successfully converge to robust solutions under decision-dependent uncertainty, though at the cost of increased function evaluations due to the dynamically shifting uncertainty space. However, the results highlight that incorporating DDU has potential of improving the solution quality, compared to problems with fixed uncertainty sets, as it leverages the interactions between the decision variables and uncertainty parameters.

Overall, this chapter demonstrated the feasibility and flexibility of the SB-SORO-DDU approach and laid the foundation for extending robust optimization techniques to more complex, real-world systems where uncertainty is inherently tied to design choices.

The next chapter will conclude this dissertation and discuss the main highlights, contributions, and future research directions based on the approaches proposed in this dissertation.

Chapter 6: Conclusions

This dissertation introduced new methods for solving single- and multi-objective robust optimization (SORO and MORO) problems with fixed and variable interval uncertainty. The proposed methods are built upon a single-level iterative framework and manage uncertainty robustness using a scenario generation technique.

6.1 Summary

The core approaches presented in this dissertation include:

Original Sampling-Based Multi-Objective Robust Optimization (OSB-MORO)

This method handles robust optimization under fixed interval uncertainty. It follows a sequential, single-level approach that uses a multi-objective genetic algorithm solver for the optimization stage and a Latin-Hypercube sampling technique for handling the robustness evaluation stage.

The main advantage of the OSB-MORO approach is its generality, simplicity of implementation, and flexibility, making it applicable to a wide array of problems, including constrained and unconstrained, linear and nonlinear, and those involving black-box models. It can also theoretically accommodate problems with discrete variables and discontinuous functions - as long as the appropriate optimization solver is used. And in addition to interval uncertainty, distributional uncertainty, or a mix of both, can be easily accommodated.

Moreover, the OSB-MORO approach can be adapted to single objective problems. In that regard, a utility-based adaptation, USB-MORO, was explored, reformulating the MORO problem into a single-objective one via a multi-attribute utility function.

Given the general and modular framework of the approach, it is also possible to utilize information about the problem to potentially improve the approach's performance. For example, a specific optimization solver (besides NSGA-II) can be used, if it is known to work better for a given type of problems. Or, if the problem is convex, then firstly, the obtained solution is guaranteed to be the global optimum; secondly, the problem can be converted to a single-objective problem via classical methods (e.g., weighted sum or epsilon-constrained methods), and a more efficient single-objective solver, such as a gradient-based solver, can be applied. Utilizing this knowledge could help reduce the computational cost significantly. Another valuable insight to a problem would be monotonicity. If the constraints are linear or monotonic with respect to the uncertainty parameters, then the worst-case scenario can be quickly determined via a gradient calculation, thus eliminating the uncertainty sample generation step and subsequent robustness evaluation step entirely, which would lead to a significant reduction in computational cost.

The approach has demonstrated its overall effectiveness in various examples and a real-world Unmanned Surface Vessel (USV) case study. A comparison to a previous approach indicates that OSB-MORO can reliably produce comparable results in terms of solution quality and significantly decreases the number of function calls.

A significant limitation of OSB-MORO is its computational intensity due to its reliance on evaluating original problem functions. The sampling scheme, while straightforward, does not scale well, and poses challenges in determining an optimal sample size to thoroughly explore the uncertainty space and potentially leading to sub-optimal robust solutions if not adequately

covered. Additionally, the sequential addition of new constraints can progressively make the feasible domain more restrictive.

Surrogate Sampling-Based Multi-Objective Robust Optimization (SSB-MORO)

This method enhances OSB-MORO by integrating online surrogates to improve computational efficiency under fixed interval uncertainty. SSB-MORO utilizes a Kriging surrogate to model the problem. As the MORO iterations progress, the surrogate dynamically updates, strategically targeting regions of high uncertainty and constraint violation, which helps improve surrogate fidelity near robust optimal fronts.

The approach was applied to several example problems as well as a USV case study. The method generally produces comparable results to existing literature approaches and excels in achieving a more uniform solution set.

The primary concern with SSB-MORO is surrogate accuracy, which can lead to sub-optimal or inconsistent results, especially for more complex problems. Issues like the "double surrogate" effect (using surrogates on top of other pre-built surrogates, as seen in the USV case study) and limited MORO iterations can also contribute to discrepancies in results.

Sampling-Based Single-Objective Robust Optimization with Decision-Dependent Uncertainties (SB-SORO-DDU)

This approach extends the sampling-based framework to address single-objective problems where uncertainty is dependent on the decision variables.

SB-SORO-DDU provides an efficient method for solving problems where the uncertainty interval itself is a function of the decision variables, common in real-world applications like path planning (e.g. the USV case study). The approach follows similar steps to OSB-MORO, with key modifications in how uncertainty samples are handled during robustness evaluation, ensuring

they remain within the bounds defined by the current decision variables. A significant advantage of the SB-SORO-DDU approach is its relatively straightforward implementation and there is no need to reformulate the problem to account for the variable uncertainty bounds.

The literature on decision-dependent uncertainty is relatively light, and there is a lack of standard test problems to evaluate new methods. So the proposed approach was applied on a simple Circle problem to demonstrate the steps and illustrate how the shift in uncertainty is handled. SB-SORO-DDU was also applied to the USV case study, and showed promising results.

Some drawbacks of the approach relate to the computational expense of sampling-based techniques - although for cheap and small problems it works great, it does not scale well. Moreover, due to the constantly shifting uncertainty space, the SB-SORO-DDU approach generally requires a higher number of function calls to converge compared to methods with fixed interval uncertainty.

6.2 Contributions

The research efforts discussed in this dissertation make the following contributions to the field of MORO:

1. A general framework for scenario generation based MORO.

- This work presents a simple, modular framework for a general sequential, single-level, sampling-based MORO approach, which serves as a foundation for developing more fine-tuned and specialized robust optimization techniques, as demonstrated by the proposed methods in this dissertation.

2. *A new single-level, general sampling-based multi-objective robust optimization (OSB-MORO) approach*

- *Improved computational cost:* This method provides a significant improvement in computational efficiency by introducing a single-level iterative framework, which means the optimization problem only needs to be solved once per iteration. This contrasts with many existing multi-objective robust optimization (MORO) methods that rely on computationally expensive bi-level or two-stage workflows.
- *Enhanced generality and applicability:* OSB-MORO offers a highly flexible and general framework that can be applied to a wide array of problems, including constrained and unconstrained, linear and nonlinear, and those involving black-box models. This addresses a key research gap where existing methods often impose specific assumptions on problem characteristics.

3. *An online surrogate sampling-based multi-objective robust optimization (SSB-MORO) approach*

- *Reduced computational cost:* SSB-MORO addresses the challenge of applying MORO to problems involving expensive functions or simulations. By integrating an online surrogate model (specifically Kriging) with a sampling-based MORO approach, this method significantly reduces the computational time and effort required for both optimization and uncertainty analysis.
- *Improved solution uniformity:* The proposed approach not only achieves comparable results to existing literature methods, but also excels in obtaining a more uniform distribution of

solutions within the Pareto front, as indicated by the Spacing (SP) quality metric.

4. A novel sampling-based single-objective robust optimization approach with decision-dependent uncertainty (SB-SORO-DDU)

- *Simplified tractability:* SB-SORO-DDU offers a simple, flexible, and tractable framework for handling variable uncertainty. Unlike many existing RO methods with DDU that require complex problem reformulations, this sampling-based strategy avoids complicating the problem structure or making it NP-hard.
- *Improved uncertainty consideration:* This approach addresses a critical limitation in traditional RO methods by accounting for endogenous, or variable uncertainty, where uncertainty parameters are dependent on the decision variables. This enables more realistic and less conservative modelling of uncertainty in real-world applications. By leveraging the interactions between decision variables and uncertain parameters, this approach has the potential to yield better solutions compared to traditional RO problems with fixed uncertainty sets.

6.3 Directions for Future Work

The methodologies presented in this dissertation lay the groundwork for several avenues of future research and potential improvements:

Improving sampling strategies

While all the proposed methods integrate a simple LHS approach, further improvements can be explored. This includes refining the methodology for uncertainty generation and handling (e.g.,

exploring adaptive sampling techniques beyond Latin-Hypercube sampling for better and more efficient and effective coverage). Although a fixed sampling strategy (such as LHS, Taguchi, factorial designs [105, 119]) may be easy to implement, it can be challenging to determine the number of samples to use to ensure the entire uncertainty space is efficiently explored. Additionally, a fix sampling technique would not scale well, and it would be necessary to perform some analysis to determine the tradeoff between the cost of computation and cost of exploration and accuracy. To address that limitation, adaptive sampling techniques have been developed [115, 120], where new samples are adaptively selected based on the problem characteristics and current information.

Enhancing surrogate model accuracy

The accuracy of surrogate models remains a key area for improvement, especially for complex problems. Future work can involve tuning Kriging model parameters (e.g., correlation functions) and exploring alternative surrogate modelling strategies and integration methods to enhance the convergence behavior and robustness of the final solutions. For example, a surrogate model could be developed for each function individually, rather than for all functions at once, thus allowing for more granular fine-tuning of surrogate model parameters. Additionally, it is important to consider the strategies for both creating the initial DOE samples, and selecting the best points to use for updating the surrogate.

Extending robust optimization with decision-dependent uncertainty to multi-objective formulation

The SB-SORO-DDU method addresses single-objective problems. A natural extension would be to investigate multi-objective robust optimization problems with decision-dependent uncertainties, building upon the insights gained from the single-objective approach. There are several

ways this could be addressed. Possibly the simplest approach would be to apply scalarization techniques to reformulate the multi-objective problem to a single-objective formulation, e.g. via the weighted sum method [6] or via utility functions [29]. Then, as already demonstrated in this dissertation, single-objective optimization techniques can be used to solve the problem. Other potential approaches include developing bi-level optimization formulations [35, 121] or extending evolutionary algorithm approaches, decomposition methods, or distributionally robust optimization models [36, 122].

Consideration of time-dependent uncertainty

Future studies could explore incorporating time-dependent uncertainty, thus bridging the gap between traditional MORO and dynamic optimization or optimal control problems. This could have many applications in routing, scheduling, or planning problems. Current approaches utilize the concepts of dynamic optimization and the use of evolutionary algorithms to solve these problems [123, 124]. Many time-dependent problems have been considered in the area of reliability, where consideration of time-dependent reliability provides a more realistic and comprehensive assessment of system safety and performance, which is essential for design, maintenance, and lifecycle management decisions [125, 126]. However, as with many other approaches, a primary challenge is the high computational cost, especially for high-dimensional or complex problems, as robust optimization often requires extensive simulations or sampling to account for uncertainties over time. Many existing methods are limited to time-invariant objectives or can only handle specific types of constraints and disturbances, restricting their applicability to real-world, dynamic scenarios [127].

Appendix A: Example Problem Formulations

This appendix presents the formulations and backgrounds of the example problems presented in this dissertation - MinEx, TNK, CTP, Three-Bar Truss and ZDT3.

A.1 MinEx

A deterministic version of the MinEx problem was presented in [6] and later revised to include uncertainty in [63]. It is a bi-objective problem with two constraints and two decision variables. There is a single uncertain parameter, which appears in both the objectives and constraints, thus making the problem an objective-feasibility robustness type. The nominal uncertainty value is set to 1, with its variation defined by an interval range. The acceptable objective variation range (AOVR) is set to 0.3 for the first objective, f_1 , and 0.6 for the second objective, f_2 . The formulation of the problem is presented in Equation (A.1)

$$\min \begin{cases} f_1(\mathbf{x}, u) = ux_1 \\ f_2(\mathbf{x}, u) = \frac{u+x_2}{x_1} \end{cases}$$

subject to:

$$g_1(\mathbf{x}, u) = 6 - 9x_1 - ux_2 \leq 0$$

$$g_2(\mathbf{x}, u) = 1 - 9x_1 + ux_2 \leq 0$$

(A.1)

$$0.1 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 5$$

where

$$u_0 = 1 \quad \Delta u \in [-0.3, 0.3]$$

$$u = u_0 + \Delta u$$

$$AOVR = [0.3, 0.6]$$

A.2 TNK

The TNK problem was originally formulated by Tanaka [108] as a bi-objective problem with two constraints and two decision variables. The problem was then adapted by Hu [10] to include uncertainty, as show in Eq. (A.2). There are two uncertain parameters, u_1 and u_2 , whose nominal values are set to zero. The uncertainty is defined as an interval range, and appears in only one of the constraints, thus only feasibility robustness must be considered.

$$\min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = x_2 \end{cases}$$

subject to:

$$\begin{aligned} g_1(\mathbf{x}, \mathbf{u}) = & -x_1^2 - x_2^2 + 0.1 \cos \left(16 \arctan \left(\frac{x_1}{x_2} \right) \right) + \\ & + 1 + 0.35 \sin(u_1) \cos(u_2) \leq 0 \end{aligned} \tag{A.2}$$

$$g_2(\mathbf{x}) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5$$

$$0 \leq x_n \leq \pi, \forall n$$

where

$$u_{0,p} = 0, \forall p; \quad \Delta u_p \in [-2, 2], \forall p$$

$$u_p = u_{0,p} + \Delta u_p, \forall p$$

A.3 CTP

The CTP example is a highly non-linear, constrained bi-objective problem with ten design variables and two uncertain parameters. Both uncertainties appear in the constraints, making the problem a feasibility robustness type. It was originally formulated as a deterministic problem in [6], then revised to include uncertainty in [10].

$$\min \begin{cases} f_1 = x_1 \\ f_2 = k \left(1 - \sqrt{\frac{f_1}{k}} \right) \end{cases}$$

subject to:

$$g = a |\sin(b\pi(\sin(\theta)(f_2 - e) + \cos(\theta)f_1)^c)|^d$$

$$-\cos(\theta)(f_2 - e) + \sin(\theta)f_1 \leq 0$$

$$0 \leq x_1 \leq 1; -5 \leq x_i \leq 5, i = 2, \dots, 10 \quad (\text{A.3})$$

where

$$k = \left| 1 + \left(\sum_{i=2}^{10} x_i \right)^{0.25} \right|$$

$$\theta = -0.2\pi + u_1; a = 0.2; b = 10; c = 1; d = 6 + u_2; e = 1;$$

$$u_0 = [0, 0]; \quad \Delta u_1 \in [-0.2, 0.2]; \quad \Delta u_2 \in [-2, 2]$$

$$u_p = u_{0,p} + \Delta u_p, \forall p$$

A.4 Three-Bar Truss

The Three-Bar Truss example problem aims to optimize the design of the three-bar truss, such that the horizontal deflection, u , and total volume of the components, V , are minimized. The design is subject to constraints on the yield stress of the components, and the vertical deflection of the overall assembly.

The configuration of the truss (see Figure A.1) is symmetric, with the side components having a cross-sectional area $A_1 \text{ cm}^2$ and the middle component having a cross-sectional area $A_2 = 2 \text{ cm}^2$. The components are hinged to a wall at nodes 1, 2, and 3 and are spaced $l = 1\text{m}$

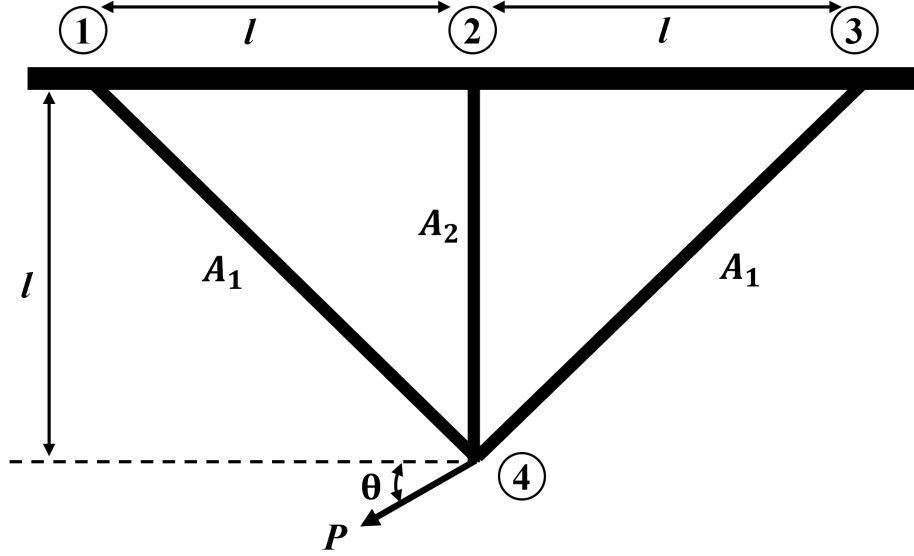


Figure A.1: Three-bar truss configuration

apart. The components are connected at node 4, where a force $P = 100\text{kN}$ is applied at an angle of θ below the horizon.

The objective of the problem is to optimize the design of the three-bar truss, such that the horizontal deflection, u , and total volume of the components, V , are minimized. The design is subject to constraints on the yield stress of the components, and the vertical deflection of the overall assembly.

The formulation of the problem is shown in Eq. (A.4). The decision variables for this problem are the cross-sectional area of the side components, A_1 , and the angle, θ , both of which contain uncertainty and may vary by ΔA_1 and $\Delta \theta$, respectively. The uncertainties appear in both the objective functions and constraints, thus requiring objective-feasibility robustness consideration. The allowable variations in the objectives are given by $AOVR$. The first and second constraints dictate that the stress on the side and middle components, respectively, must be less than the yield stress, $\sigma_{1,0} = \sigma_{2,0} = 140\text{MPa}$, and the third constraint ensures that the vertical

deflection is less than $\nu_0 = 0.5cm$. In the formulation, $E = 70GPa$ is Young's modulus.

$$\min\{f_1, f_2\}$$

subject to:

$$g_1 = \sigma_1 \leq \sigma_{1,0}; \quad g_2 = \sigma_2 \leq \sigma_{2,0}; \quad g_3 = \nu \leq \nu_0$$

$$2cm^2 \leq A_1 \leq 10cm^2, \quad 30^\circ \leq \theta \leq 40^\circ$$

$$\forall \Delta A_1 \in [-0.1cm^2, 0.1cm^2], \quad \forall \Delta \theta \in [-5^\circ, 5^\circ]$$

$$AOVR = [0.07cm, 400cm^3] \tag{A.4}$$

where

$$f_1 = u(A_1, \theta) = \frac{\sqrt{2}lP \cos(\theta)}{A_1 E}$$

$$f_2 = V(A_1, \theta) = 2\sqrt{2}A_1 l + A_2 l$$

$$\sigma_1 = \frac{P}{\sqrt{2}} \left[\frac{\cos(\theta)}{A_1} + \frac{\sin(\theta)}{(A_1 + \sqrt{2}A_2)} \right]$$

$$\sigma_2 = \frac{\sqrt{2}P \sin(\theta)}{(A_1 + \sqrt{2}A_2)}; \quad \nu = \frac{\sqrt{2}lP \sin(\theta)}{(A_1 + \sqrt{2}A_2)E}$$

A.5 ZDT3

The ZDT3 example is bi-objective, unconstrained problem, which is scalable with respect to the number of decision variables (N). It was originally formulated as a deterministic problem in [6] and later adapted to include uncertainty in [10]. The formulation is presented in Eq. (A.5)

$$\min \begin{cases} f_1 = x_1 + u_1 \\ f_2 = gh \end{cases}$$

$$0 \leq x_i \leq 1, i = 1, \dots, N$$

where

$$g = 1 + \frac{9}{N-1} \left(\sum_{i=2}^N x_i + u_1 + u_2 \right) \tag{A.5}$$

$$h = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g} \sin(10\pi f_1)$$

$$u_0 = [0, 0]; \quad \Delta u_1 \in [0, 0.1]; \quad \Delta u_2 \in [-0.2, 0.2]$$

$$u_p = u_{0,p} + \Delta u_p, \quad \forall p$$

$$AOVR_2 = 0.5$$

Appendix B: USV Case Study

A case study of an unmanned surface vessel (USV) was considered as a real-world application to test the approaches proposed in this dissertation. The following sections of this appendix present a description of the USV case study - an overview of the problem, the assumptions, setup, and details of the development of the USV simulation model.

An earlier version of this case study was presented in [5]. For this work, a few key subsystems in the USV model are considered, namely the engine cooling and control system (ECCS), the reliability system, and hydrodynamics subsystem [128] to account for external wave forces and to incorporate a routing aspect to the problem.

The USV is assumed to complete a 200-hour mission, going from San Diego, California to Honolulu, Hawaii. The USV is exposed to some environmental conditions (e.g., temperatures and pressures) and wave conditions (e.g., wave spectrum and current direction). The USV is also equipped with sensors, which record its operational state surroundings and monitor the critical components - this data is subsequently used to assess the health state of the USV and predict the component and system reliability. The USV mission trip is assumed to be split into five 40-hour sections, or legs, during each of which the operational profile remains fixed. In other words, over the course of the 200-hour mission, the USV can change its operational profile by changing engine throttle input and heading direction 5 times, every 40 hours during its mission.

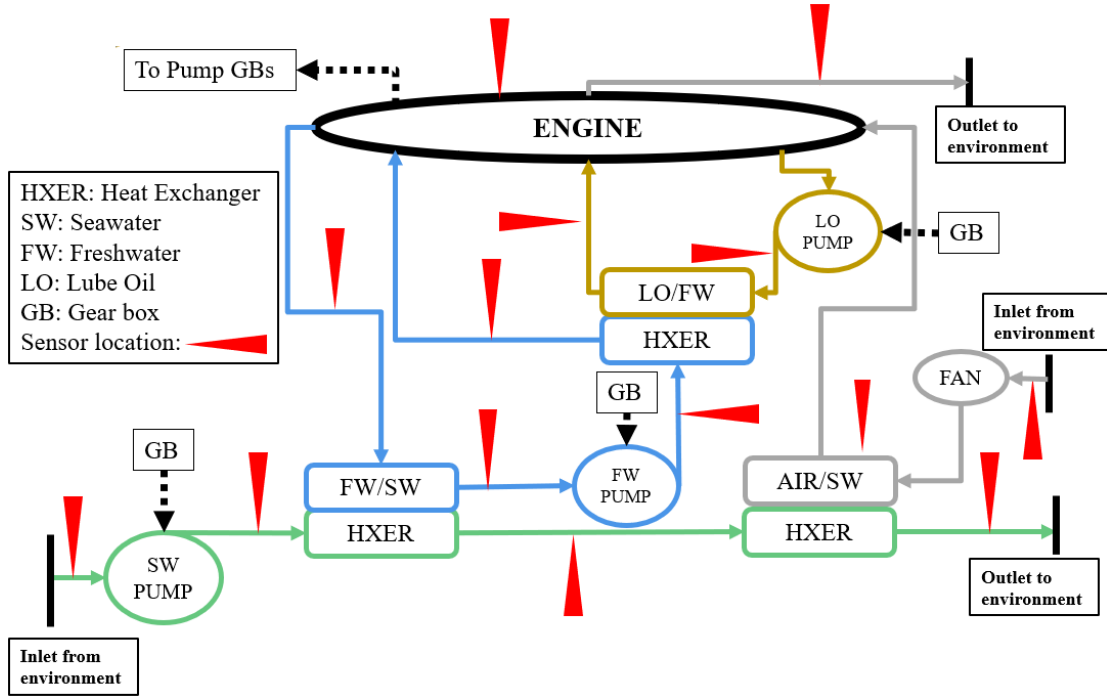


Figure B.1: Engine Cooling and Control System Design Presented in [5]

B.1 ECCS Model and Surrogate

The ECCS was an extension of the simulation model shown in [5]. The model was used to represent some of the sensors on a USV, such as different temperature and pressure sensors located at the red-flagged locations shown in Figure B.1.

The model used in [5] was designed for a medium-sized ship (100m long), whereas the ship model used in this case study is for a large container ship (175m long). This change was made due to the model availability apart of the Marine System Simulation (MSS) Toolbox in MATLAB [128]. The increase in size required a higher amount of required thrust to control the ship. The power for the increased thrust demand needed to be provided by the ship's engine. The increased power demand would result in the engine producing more heat that would need to be dissipated by the cooling system. As a result, the cooling system needed to be expanded. This

Table B.1: Error Metrics on the Training and Testing Dataset for the ECCS Surrogate Model

	R^2	MAPE	RMSE
Training	1.0	9.982×10^{-9}	6.807×10^{-6}
Testing	0.961	0.00323	8.652

was done by adding a secondary cooling system, identical to the first attached to the engine as well.

Much like the ECCS model used in [5], this model had similar computational intensity. As a result a surrogate model needed to be developed. The method used for building a surrogate model for the updated ECCS system was a K-Nearest Neighbors regression model, based on the 15 nearest neighbors. This model was chosen because it proved to be fast for making predictions and had reasonable accuracy. An 80/20 train/test split was applied to 24,766 samples with the errors (Eq. (B.2), (B.1), (B.3)) for the training and testing datasets shown in Table B.1.

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \frac{1}{N} \sum_{i=1}^N y_i)^2} \quad (\text{B.1})$$

$$MAPE = \frac{1}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (\text{B.2})$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad (\text{B.3})$$

B.2 Hydrodynamics Model and Surrogate

In order to make the overall USV model more realistic, a hydrodynamics (hydro) model was integrated with the ECCS model, which allowed to account for external wave forces and

facilitate the routing aspect. The hydro model was based on Fossen's Marine System Simulator (MSS) [128], adapted for compatibility with the existing USV model framework and case study specifics. The MSS is a MATLAB and Simulink library for marine systems, which includes hydrodynamic models for ships, underwater vehicles, and floating structures, as well as guidance, navigation, and control (GNC) algorithms for real-time simulation. The modifications to the MSS included adding origin and target destination information to track the USV's position with respect to those locations.

The Hydro model requires a set of inputs to be run. For this case study, some inputs were fixed, and others were allowed to be modified by the user. The fixed inputs included the USV hydrodynamic data (such as the wave spectrum and other wave characteristics). The Modified Pierson-Moskowitz (MPM) spectrum was used to model open sea conditions [129]. The user was allowed to control the following inputs: USV input throttle, USV heading, wave current direction, wave current speed, and coordinates of origin and target destinations.

The outputs of the Hydro simulation model included the wave forces on the USV (in 6 degrees of freedom (DOF)), the USV's positions and velocities in 6 DOF, and the USV speed. Some additional modifications were made to calculate the distances from the USV's current position to the origin and target destination, respectively.

Running the Hydro simulation for the full mission time (200 hours) proved to be computationally expensive, so an offline surrogate model was developed, using a multilayer perceptron (MLP) regression model [130]. The architecture of the model contained two hidden layers, with 512 and 128 neurons, respectively, and both employed LeakyReLU (with the negative slope parameter set to 0.2) as the activation function. Since the original simulation provides a timeseries output with 5 distinct stages (corresponding to the 5 profile stages), and the output parameter

Table B.2: Error Metrics on the Training and Testing Dataset for the Hydro Surrogate Model

	R^2	MAPE	RMSE
Training	0.998	0.396	0.350
Testing	0.999	0.175	0.354

values at each stage are relatively constant, the MLP was trained to make stage-wise predictions, which were then stitched together to produce the final full timeseries prediction. The model was built on 50,000 samples, with an 80/20 train/test split. The evaluation metrics for determining the accuracy of the model predictions were calculated using Eqs. (B.1), (B.2), (B.3) and are summarized in Table B.2.

B.3 Reliability Model

The reliability model of the ECCS system for the USV is summarized in Figure B.2, which is adapted from the framework proposed by Hazra et al. [5].

As shown in Figure B.2, two main stages are included in this framework. In the first stage, the subsystems' health state is diagnosed using deep learning models - one model for each subsystem. The convolutional neural network (CNN) models were trained for the health state classification of the seven crucial subsystems of the ECCS: the marine diesel engine, two fresh-water cooling subsystems, two seawater cooling subsystems, and two lube oil subsystems. The training data was obtained from the operation data generated from the ECCS+hydro simulation models, as discussed in Sections B.1 and B.2. The operation data used for CNN training includes temperatures, pressures, and other measurements from different locations of the ECCS+hydro system. Simulation results of 24,766 sampled missions with 200 hours each were used in the development of CNN models, while 80% of samples were selected for training purposes, and

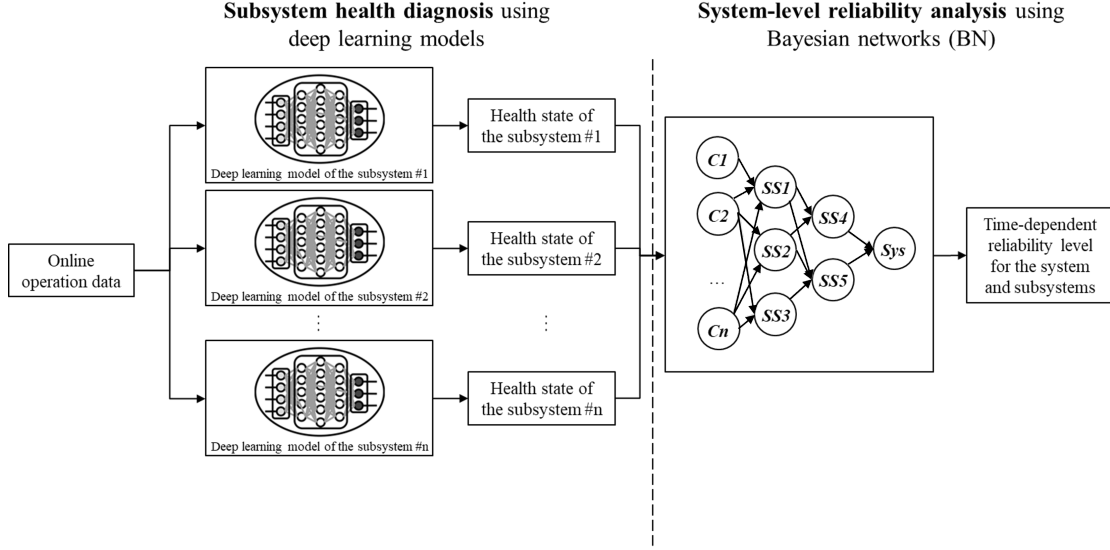


Figure B.2: Framework of the development of the of reliability model

the rest were used for validation. The simulated dataset comprises ten health state classes, each corresponding to different subsystem efficiencies. These efficiencies range from 100% down to 10%, decreasing in increments of 10%. The output of CNN models (the first stage) are probability distributions of the health state classes of each subsystem. It provides real-time health state information for each subsystem under the given input environmental and operational uncertainties. All seven CNN models showed high accuracy rates, approximately 96% for both training and validation data sets according to Eq.(B.4). The health state information is further input to Bayesian Belief networks (BBNs) as virtual evidence for updating the system-level reliability.

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i) \quad (B.4)$$

In the second stage, a BBN is constructed as shown in Figure B.3. This is developed using GeNIe Modeler by BayesFusion [131]. This BBN structure is adapted from Hazra et al. [5], considering the new design of the ECCS system. The main seven subsystems are highlighted in

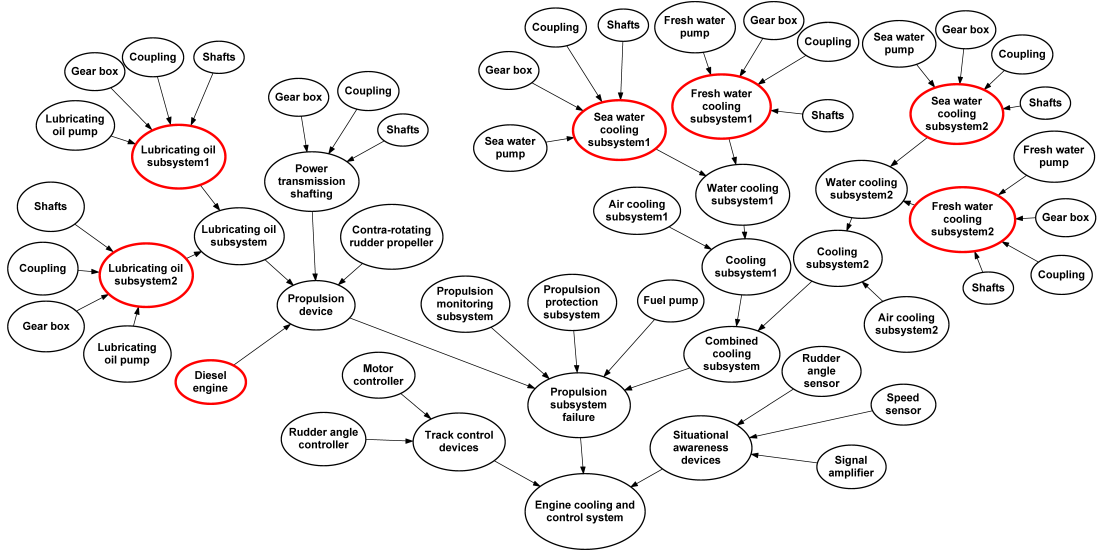


Figure B.3: Structure of the developed BBN model for system-level reliability analysis

red. The failure rates for the ECCS components and conditional probability tables (CPTs) can be found in Hazra et al. [5].

Utilizing health state information obtained through deep learning models in the first stage, the developed BBN model can update the reliability profile of the ECCS of a USV. Using prior information obtained from the reliability data handbooks, the BBN model of the ECCS was able to evaluate the health state of a 200-hour mission every hour.

Upon further analysis of the reliability model, it was discovered that the final system reliability was fairly immutable with respect to varying input conditions, e.g., decision variables. In the interest of producing some sensible results, the system reliability (R) was modified to be, among others, a function of the input throttle ($throttle$). The transformation was performed following Eq. (B.5), to achieve a more exaggerated relationship between the reliability and throttle and scale the values so that they were within 0 and 1 range.

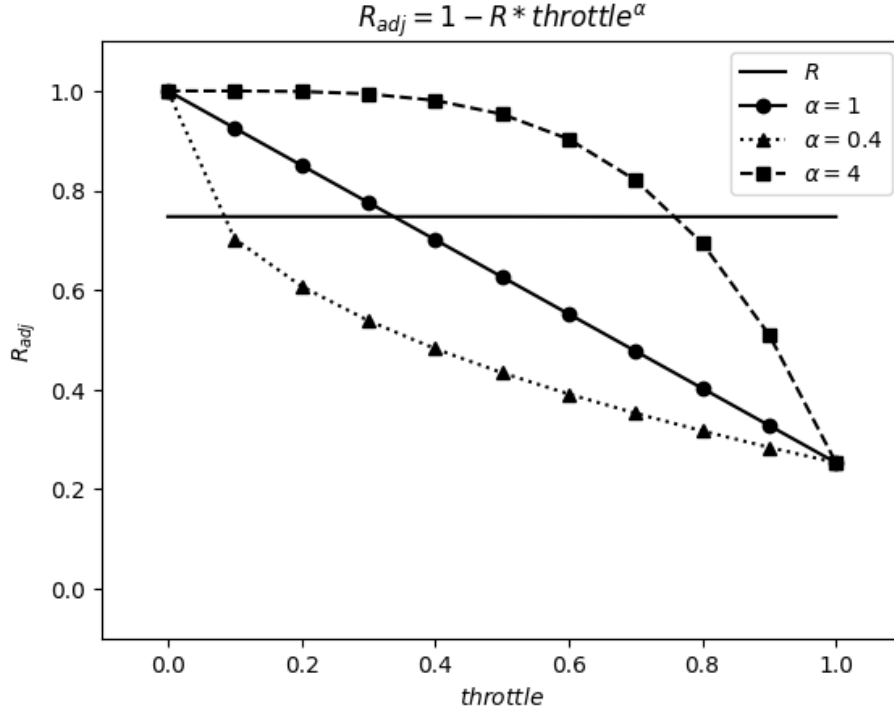


Figure B.4: Adjusted reliability (R_{adj}) for different values of α

$$R_{adj} = 1 - R \times throttle^{\alpha} \quad (B.5)$$

Figure B.4 illustrates how the value of α in Eq. (B.5) has a significant effect on the relationship between the throttle and adjusted reliability value, R_{adj} . For $0 < \alpha < 1$ the relationship between the adjusted reliability and throttle is decreasing and convex; for $\alpha = 1$, the adjusted reliability and throttle exhibit a linearly decreasing relationship; and for $\alpha > 1$ the relationship between the adjusted reliability and throttle is decreasing and concave. For this case study, the values of 1 and 4 have been used for α .

B.4 Models Integration

One of the many challenges of implementing this case study was integrating all the models in question - namely, the ECCS, system reliability, and Hydro surrogate models.

The overall integration framework for the aforementioned components of the USV model is shown in Figure B.5.

The white blocks (rectangular blocks labeled “Model”) represent the models, the light blue blocks (rectangular blocks with curved edges labeled “Data”) represent input/output parameters, the green blocks (irregular pentagon arrow blocks labeled “DV”) represent the input parameters that are the decision variables for the optimization problem, and the purple blocks (oval blocks labeled “Output”) represent the parameters that will serve as the objectives or constraints in the optimization problem. The environment data sets the stage for the ECCS simulation, which provides USV component sensor data to be passed through a reliability model for determining the USV system reliability. The ECCS model also provides USV thrust values, which are fed to the hydrodynamics model, along with wave data, to produce positional data of the USV, its speed, and wave forces on the USV. The ECCS is composed of two sub-systems - the propulsion and cooling systems. The inputs that affect the two, are the throttle and environmental conditions, respectively. The cooling system of the ECCS model outputs synthetic sensor data, which represents readings of the surrounding environment and ECCS components’ health states. This sensor data is then fed into the reliability model to get the USV system reliability. The propulsion sub-system of the ECCS essentially converts the input throttle to a thrust value, which is then fed into the Hydro model. When integrated with the Hydro model, the thrust values provided by the ECCS simulation yielded USV speeds that were below the typical speed range (10-25 knots) of

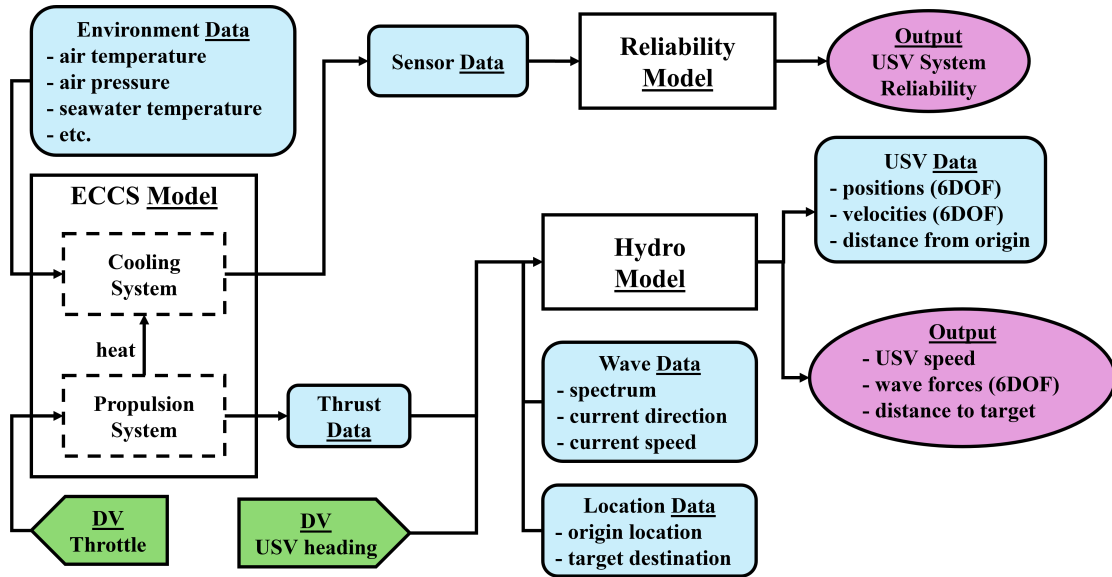


Figure B.5: Integration of ECCS, Hydro, and Reliability Models

the ship that was being modelled, so a booster factor of 30 was applied to mitigate this discrepancy. The Hydro model requires additional inputs, such as the USV heading, wave spectrum, wave current direction and speed, and coordinates of the origin and target destinations. The main outputs of interest of the Hydro model are the USV's speed and position relative to the origin and target, as well as the wave forces acting on the USV.

B.5 Considering Environmental Conditions Based on Real Data

Most of the inputs to the USV model that describe the surrounding environment are fixed or assumed with some values determined by the user. However, to provide a more realistic environment, real data for the currents, waves, and wind were integrated. The environment data was obtained from the Marine Data Store provided by the E.U. Copernicus Marine Service. The following datasets were used:

- Global Ocean Physics Analysis and Forecast [132] - was used for obtaining the wave cur-

rent directions and current speeds

- Global Ocean Waves Analysis and Forecast [133] - was used for obtaining the significant wave heights
- Global Ocean Hourly Sea Surface Wind and Stress from Scatterometer and Model [134] - was used for obtaining the wind directions and wind speeds

All of the datasets provided a monthly average of the data, so the monthly average for December 2024 was used. The area selected for consideration was defined by the latitude range of [17, 42] and longitude range of [-162, -116].

Figures B.6, B.8, B.10 illustrate the environment data on a map for the current, wave height, and wind, respectively. Thus, at any location on the map, the USV would experience a specific environment (defined by the environment maps). However, in order to integrate the environment data into the USV model (which is time-based at its core), it had to be transformed to a timeseries. Since at any given point in time, the position of the USV is known, the environment parameters can be easily estimated and "connected" to that same point in time. Thus, for a given path, a continuous profile for the environment parameter can be obtained, as shown by the dotted line in Figures B.7, B.9, B.11. The final step needed for integration is making a 5-stage profile - since the USV model is design to accommodate 5 operational profiles for the 200 hour mission. So, a mean aggregation was performed for each 40-hour section of the mission, as shown by the solid line in Figures B.7, B.9, B.11.

B.6 Utility Function Creation

The USV case study was generally formulated as a bi-objective problem. However, a single-objective reformulation was developed via a multi-attribute utility function (MAUF).

To create the MAUF for the USV problem, first a utility function for each attribute is developed, as shown in Equation (B.6). The utility functions were derived to be within the range of $[-1, 0]$, with 0 being the preferred value.

$$\begin{aligned}U_{11} &= 1.288(1 - \exp(-0.0823x_{11})) - 0.9345 \\U_{12} &= 0.0483(1 - \exp(0.00878x_{12})) \\U_{21} &= 0.652(1 - \exp(0.065x_{21})) \\U_{22} &= 0.579(1 - \exp(0.051x_{22})) \\U_{23} &= 0.262(1 - \exp(0.532x_{23})) \\U_3 &= x_3 - 1\end{aligned}\tag{B.6}$$

Then, the 6 single-attribute utility functions are grouped into 3 multi-attribute utility functions according to category and are shown in Equation (B.7):

- Operations, U_1 - groups the utility functions for USV speed (U_{11}) and sway force (U_{12})
- Environment, U_2 - groups the utility functions for wave height (U_{21}), wind speed (U_{22}), and wave current speed (U_{23})
- Reliability, U_3 - is the utility function for the system reliability

$$\begin{aligned}
U_1 &= 0.449U_{11} + 0.332U_{12} - 0.22U_{11}U_{12} \\
U_2 &= 0.252U_{21} + 0.15U_{22} + 0.113U_{23} \\
&\quad - 0.178U_{21}U_{22} - 0.134U_{21}U_{23} - 0.08U_{22}U_{23} + 0.094U_{21}U_{22}U_{23} \\
U_3 &= x_3 - 1
\end{aligned} \tag{B.7}$$

Finally, the overall MAUF for the USV problem objective is shown in Equation (B.8):

$$\begin{aligned}
U &= 0.049U_1 + 0.055U_2 + 0.061U_3 \\
&\quad - 0.131U_1U_2 - 0.146U_1U_3 - 0.164U_2U_3 + 0.394U_1U_2U_3
\end{aligned} \tag{B.8}$$

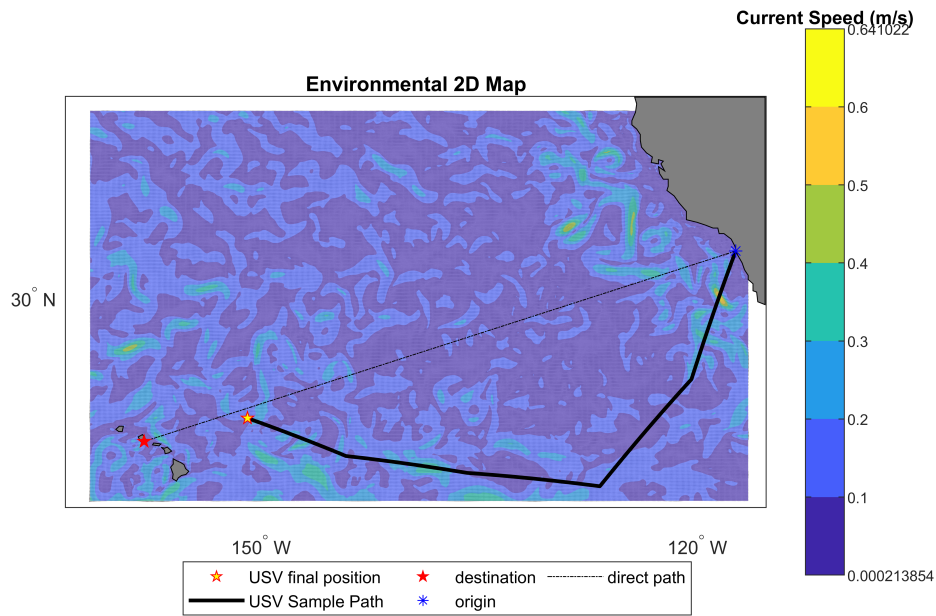


Figure B.6: Real Data for Wave Current Environment

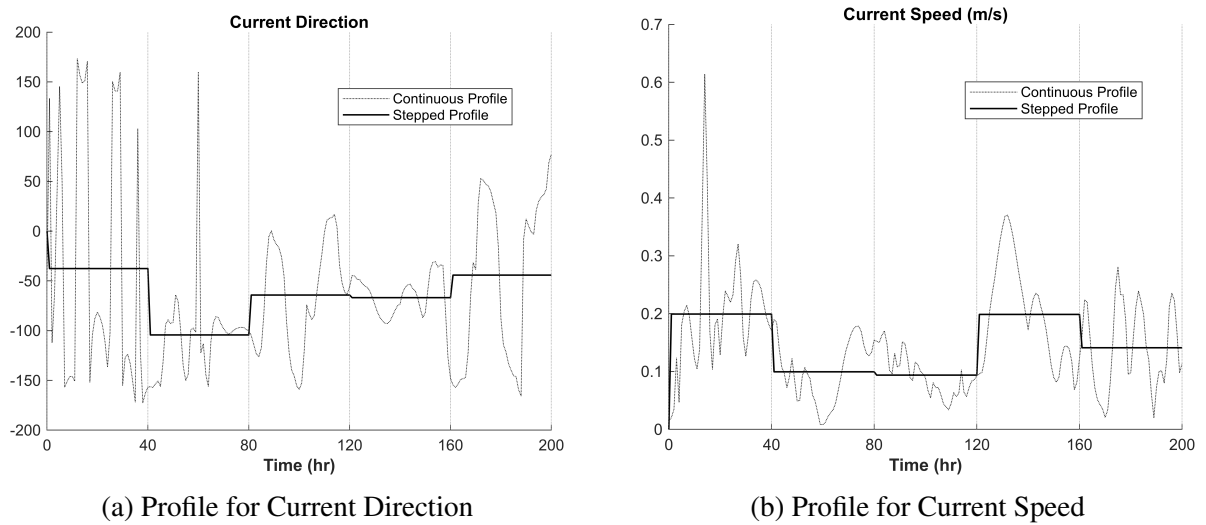


Figure B.7: Profiles for Wave Current Environment

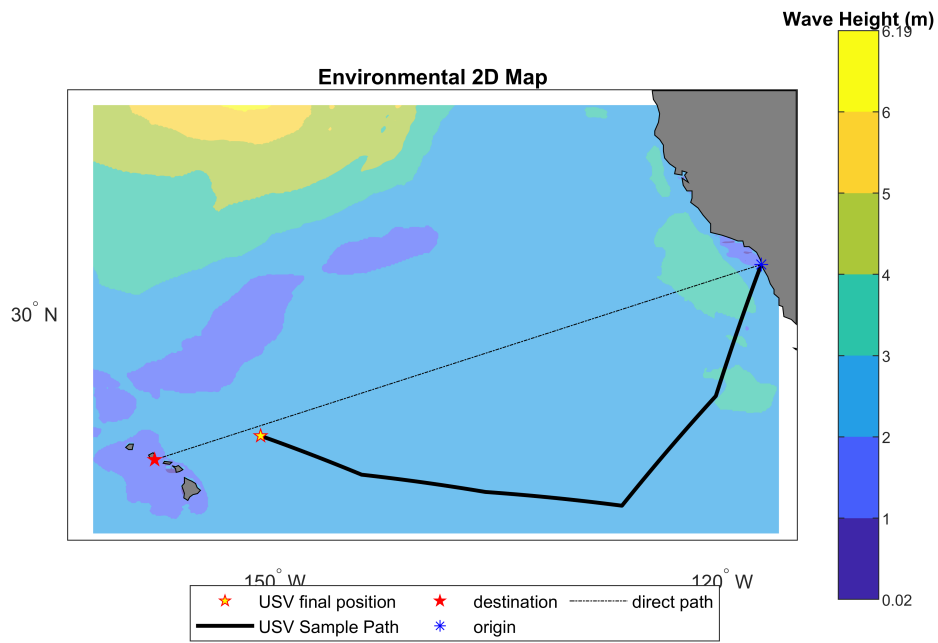


Figure B.8: Real Data for Wave Environment

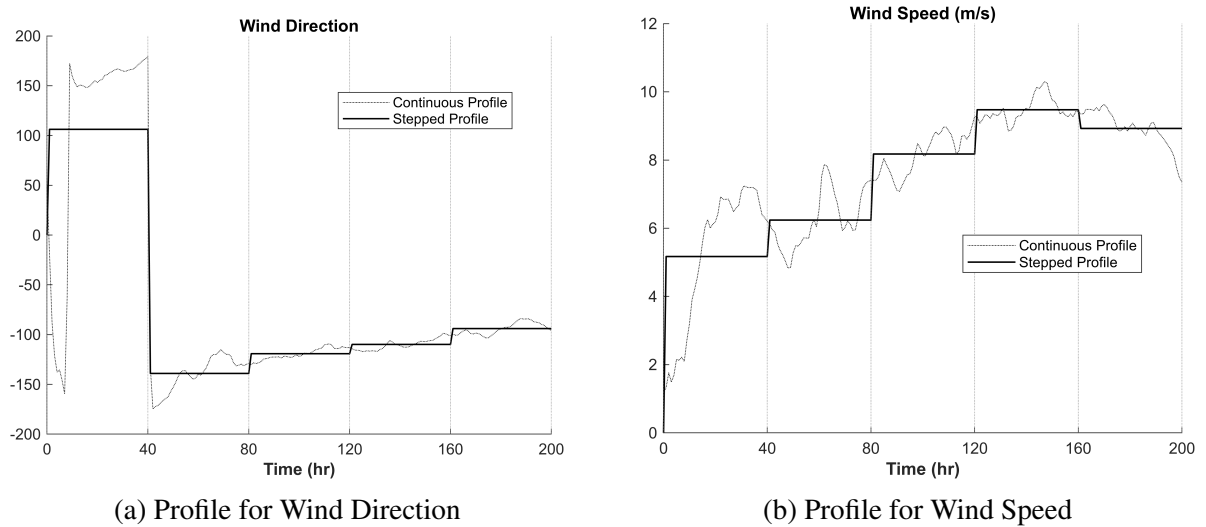


Figure B.9: Profiles for Wind Environment

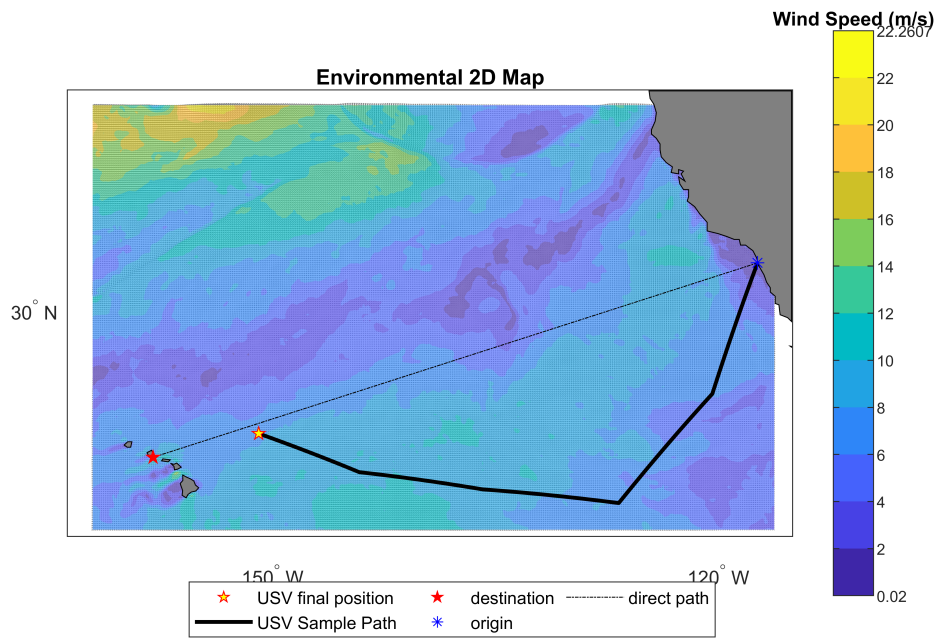


Figure B.10: Real Data for Wind Environment

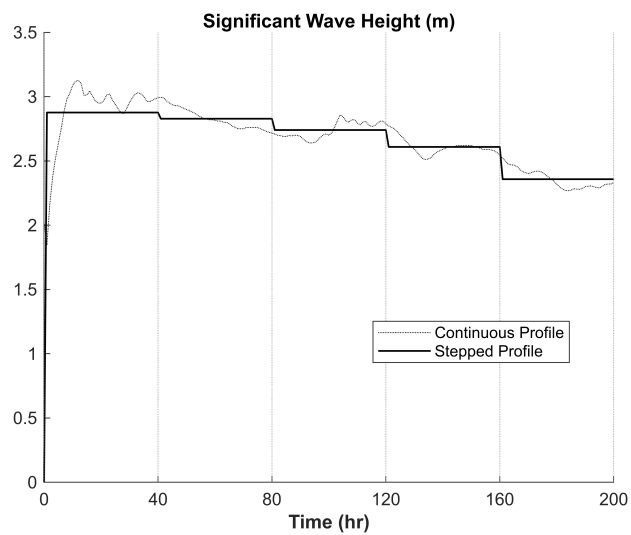


Figure B.11: Profile for Significant Wave Height

Bibliography

- [1] E. Jordan, A. Chatterjee, R. Yang, K. Groth, and S. Azarm, “On a Sampling Based Method for Multi-Objective Robust Optimization,” in *Proceedings of the ASME 2024 International Mechanical Engineering Congress and Exposition*. Portland, OR, USA: ASME, Nov. 2024.
- [2] —, “On a Sampling-Based Method for Multi-Objective Robust Optimization,” *Journal of Computing and Information Science in Engineering*, vol. 25, no. 4, p. 044501, Feb. 2025. [Online]. Available: <https://doi.org/10.1115/1.4067706>
- [3] E. Jordan and S. Azarm, “Online Surrogate Sampling-Based Multi-Objective Robust Optimization with Interval Uncertainty,” in *Proceedings of the ASME 2025 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Anaheim, CA, USA: ASME, Aug. 2025.
- [4] M. Eckstein, “Navy Claims a Strong Technical Foundation Ahead of Testing New Classes of Unmanned Ships,” Sep. 2019. [Online]. Available: <https://news.usni.org/2019/09/09/navy-claims-a-strong-technical-foundation-ahead-of-testing-new-classes-of-unmanned-ships>
- [5] I. Hazra, A. Chatterjee, J. Southgate, M. Weiner, K. Groth, and S. Azarm, “A Reliability-Based Optimization Framework for Planning Operational Profiles for Unmanned Systems,” *Journal of Mechanical Design*, vol. 146, no. 5, pp. 1–12, Oct. 2023. [Online]. Available: <https://doi.org/10.1115/1.4063661>
- [6] K. Deb, *Multi-objective optimization using evolutionary algorithms*, 1st ed., ser. Wiley-Interscience series in systems and optimization. Chichester ; New York: John Wiley & Sons, 2001.
- [7] N. Gunantara, “A review of multi-objective optimization: Methods and its applications,” *Cogent Engineering*, vol. 5, no. 1, p. 1502242, Jan. 2018. [Online]. Available: <https://doi.org/10.1080/23311916.2018.1502242>
- [8] A. Ben-Tal, L. El Ghaoui, and A. S. Nemirovskii, *Robust optimization*, ser. Princeton series in applied mathematics. Princeton: Princeton University Press, 2009.

- [9] E. Rudnick-Cohen and S. Azarm, “Non-Convex Feasibility Robust Optimization Via Scenario Generation and Local Refinement,” *Journal of Mechanical Design*, vol. 142, no. 5, Nov. 2019. [Online]. Available: <https://doi.org/10.1115/1.4044918>
- [10] W. Hu, M. Li, S. Azarm, and A. Almansoori, “Multi-objective robust optimization under interval uncertainty using online approximation and constraint cuts,” *Journal of Mechanical Design*, vol. 133, no. 6, p. 061002, Jun. 2011. [Online]. Available: <https://doi.org/10.1115/1.4003918>
- [11] W. Hu, “Approximation assisted multiobjective and collaborative robust optimization under interval uncertainty,” Ph.D., University of Maryland, College Park, United States – Maryland, 2012.
- [12] R. J. Kania, “Single- and multi-objective feasibility robust optimization under interval uncertainty with surrogate modelling,” Ph.D., University of Maryland, College Park, United States – Maryland, 2022.
- [13] V. Gabrel, C. Murat, and A. Thiele, “Recent advances in robust optimization: An overview,” *European Journal of Operational Research*, vol. 235, no. 3, pp. 471–483, Jun. 2014. [Online]. Available: <https://doi.org/10.1016/j.ejor.2013.09.036>
- [14] M. Li, S. Azarm, N. Williams, S. Al Hashimi, A. Almansoori, and N. Al Qasas, “Integrated multi-objective robust optimization and sensitivity analysis with irreducible and reducible interval uncertainty,” *Engineering Optimization*, vol. 41, no. 10, pp. 889–908, Oct. 2009. [Online]. Available: <https://doi.org/10.1080/03052150902853005>
- [15] M. Ehrgott, J. Ide, and A. Schöbel, “Minmax robustness for multi-objective optimization problems,” *European Journal of Operational Research*, vol. 239, no. 1, pp. 17–31, Nov. 2014. [Online]. Available: <https://doi.org/10.1016/j.ejor.2014.03.013>
- [16] 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, pp. 874–883, Jul. 2006. [Online]. Available: <https://doi.org/10.1115/1.2202884>
- [17] S. Gunawan and S. Azarm, “Multi-objective robust optimization using a sensitivity region concept,” *Structural and Multidisciplinary Optimization*, vol. 29, no. 1, pp. 50–60, Jan. 2005. [Online]. Available: <https://doi.org/10.1007/s00158-004-0450-8>
- [18] R. J. Kania and S. Azarm, “Bi-Objective Surrogate Feasibility Robust Design Optimization Utilizing Expected Non-Dominated Improvement With Relaxation,” *Journal of Mechanical Design*, vol. 145, no. 3, p. 031703, Nov. 2022. [Online]. Available: <https://doi.org/10.1115/1.4055738>
- [19] T. Assavapokee, M. J. Realff, and J. C. Ammons, “Min-max regret robust optimization approach on interval data uncertainty,” *Journal of Optimization Theory and Applications*, vol. 137, no. 2, pp. 297–316, Dec. 2007. [Online]. Available: <https://doi.org/10.1007/s10957-007-9334-6>

- [20] H. Arellano-Garcia and G. Wozny, "Chance constrained optimization of process systems under uncertainty: I. Strict monotonicity," *Computers & Chemical Engineering*, vol. 33, no. 10, pp. 1568–1583, Oct. 2009. [Online]. Available: <https://doi.org/10.1016/j.compchemeng.2009.01.022>
- [21] Z. Chen, P. Yu, and W. B. Haskell, "Distributionally robust optimization for sequential decision-making," *Optimization*, vol. 68, no. 12, pp. 2397–2426, Aug. 2019. [Online]. Available: <https://doi.org/10.1080/02331934.2019.1655738>
- [22] C. Shang and F. You, "Distributionally robust optimization for planning and scheduling under uncertainty," *Computers & Chemical Engineering*, vol. 110, pp. 53–68, Feb. 2018. [Online]. Available: <https://doi.org/10.1016/j.compchemeng.2017.12.002>
- [23] A. Gu, S. Lu, P. Ram, and T.-W. Weng, "Min-Max Multi-objective Bilevel Optimization with Applications in Robust Machine Learning," May 2023. [Online]. Available: <https://openreview.net/forum?id=PvDY71zKsvP>
- [24] A. Biswas, Y. Chen, N. Gibson, and C. Hoyle, "Bilevel Flexible-Robust Optimization for Energy Allocation Problems," *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 6, no. 031002, May 2020. [Online]. Available: <https://doi.org/10.1115/1.4046269>
- [25] J. Wu, G. De, Z. Tan, and Y. Li, "Study on bi-level multi-objective collaborative optimization model for integrated energy system considering source-load uncertainty," *Energy Science & Engineering*, vol. 9, no. 8, pp. 1160–1179, 2021. [Online]. Available: <https://doi.org/10.1002/ese3.880>
- [26] T. Giovannelli, G. D. Kent, and L. N. Vicente, "Bilevel optimization with a multi-objective lower-level problem: risk-neutral and risk-averse formulations," *Optimization Methods and Software*, vol. 0, no. 0, pp. 1–23, 2024. [Online]. Available: <https://doi.org/10.1080/10556788.2024.2318707>
- [27] L. Lampariello, S. Sagratella, V. G. Sasso, and V. Shikhman, "Scalarization via utility functions in multi-objective optimization," Jan. 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2401.13831>
- [28] J.-P. Dubus, C. Gonzales, and P. Perny, "Multiobjective optimization using gai models," in *International Joint Conference on Artificial Intelligence*, Jul. 2009. [Online]. Available: <https://dl.acm.org/doi/10.5555/1661445.1661750>
- [29] R. E. Rosenthal, "Concepts, theory, and techniques principles of multiobjective optimization," *Decision Sciences*, vol. 16, no. 2, pp. 133–152, Apr. 1985. [Online]. Available: <https://doi.org/10.1111/j.1540-5915.1985.tb01479.x>
- [30] P. S. Palar, R. P. Liem, L. R. Zuhal, and K. Shimoyama, "On the use of surrogate models in engineering design optimization and exploration: the key issues," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 1592–1602. [Online]. Available: <https://doi.org/10.1145/3319619.3326813>

- [31] O. Roeva, *Real-World Applications of Genetic Algorithms*. IntechOpen, 2012.
- [32] R. Alizadeh, J. K. Allen, and F. Mistree, “Managing computational complexity using surrogate models: a critical review,” *Research in Engineering Design*, vol. 31, no. 3, pp. 275–298, Jul. 2020. [Online]. Available: <https://doi.org/10.1007/s00163-020-00336-7>
- [33] O. Nohadani and K. Sharma, “Optimization under decision-dependent uncertainty,” *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1773–1795, Jan. 2018. [Online]. Available: <https://doi.org/10.1137/17M1110560>
- [34] Y. Zhang, F. Liu, Y. Su, Y. Chen, Z. Wang, and J. P. S. Catalao, “Two-stage robust optimization under decision dependent uncertainty,” *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 7, pp. 1295–1306, Jul. 2022. [Online]. Available: <https://ieeexplore.ieee.org/document/9754328>
- [35] H. Lefebvre, M. Schmidt, S. Stevens, and J. Thürauf, “Solving decision-dependent robust problems as bilevel optimization problems,” Mar. 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2503.01559>
- [36] X. Yu and B. Basciftci, “Distributionally robust optimization with multimodal decision-dependent ambiguity sets,” Apr. 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2404.19185>
- [37] N. H. Lappas and C. E. Gounaris, “Robust optimization for decision-making under endogenous uncertainty,” *Computers & Chemical Engineering*, vol. 111, pp. 252–266, Mar. 2018. [Online]. Available: <https://doi.org/10.1016/j.compchemeng.2018.01.006>
- [38] F. Luo and S. Mehrotra, “Distributionally robust optimization with decision dependent ambiguity sets,” *Optimization Letters*, vol. 14, no. 8, pp. 2565–2594, Apr. 2020. [Online]. Available: <https://doi.org/10.1007/s11590-020-01574-3>
- [39] M. Li, “Robust optimization and sensitivity analysis with multi-objective genetic algorithms: Single- and multi-disciplinary applications,” Ph.D., University of Maryland, College Park, United States – Maryland, 2007.
- [40] S. Gunawan, “Parameter sensitivity measures for single objective, multi-objective, and feasibility robust design optimization,” Ph.D., University of Maryland, College Park, College Park, MD, United States, 2004.
- [41] J. Wu and S. Azarm, “Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set,” *Journal of Mechanical Design*, vol. 123, no. 1, pp. 18–25, Mar. 2001. [Online]. Available: <https://doi.org/10.1115/1.1329875>
- [42] K. Zheng, R.-J. Yang, H. Xu, and J. Hu, “A new distribution metric for comparing Pareto optimal solutions,” *Structural and Multidisciplinary Optimization*, vol. 55, no. 1, pp. 53–62, Jan. 2017. [Online]. Available: <https://doi.org/10.1007/s00158-016-1469-3>

- [43] A. Ben-Tal and A. Nemirovski, “Robust optimization – methodology and applications,” *Mathematical Programming*, vol. 92, no. 3, pp. 453–480, May 2002. [Online]. Available: <https://doi.org/10.1007/s101070100286>
- [44] H.-G. Beyer and B. Sendhoff, “Robust optimization – A comprehensive survey,” *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 33-34, pp. 3190–3218, Jul. 2007. [Online]. Available: <https://doi.org/10.1016/j.cma.2007.03.003>
- [45] S. Leyffer, M. Menickelly, T. Munson, C. Vanaret, and S. M. Wild, “A survey of nonlinear robust optimization,” *INFOR: Information Systems and Operational Research*, vol. 58, no. 2, pp. 342–373, Apr. 2020. [Online]. Available: <https://doi.org/10.1080/03155986.2020.1730676>
- [46] 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. [Online]. Available: <https://doi.org/10.1007/s00158-011-0631-1>
- [47] Y. Beck, I. Ljubić, and M. Schmidt, “A Brief Introduction to Robust Bilevel Optimization,” Jan. 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2211.16072>
- [48] M. Lombardi and R. T. Haftka, “Anti-optimization technique for structural design under load uncertainties,” *Computer Methods in Applied Mechanics and Engineering*, vol. 157, no. 1, pp. 19–31, Apr. 1998. [Online]. Available: [https://doi.org/10.1016/S0045-7825\(97\)00148-5](https://doi.org/10.1016/S0045-7825(97)00148-5)
- [49] H. K. Singh, M. M. Islam, T. Ray, and M. Ryan, “Nested evolutionary algorithms for computationally expensive bilevel optimization problems: Variants and their systematic analysis,” *Swarm and Evolutionary Computation*, vol. 48, pp. 329–344, Aug. 2019. [Online]. Available: <https://doi.org/10.1016/j.swevo.2019.05.002>
- [50] J.-A. Mejía-De-Dios, A. Rodríguez-Molina, and E. Mezura-Montes, “Multiobjective Bilevel Optimization: A Survey of the State-of-the-Art,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 9, pp. 5478–5490, Sep. 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10121701>
- [51] T. Xia, M. Li, and J. Zhou, “A Sequential Robust Optimization Approach for Multidisciplinary Design Optimization With Uncertainty,” *Journal of Mechanical Design*, vol. 138, no. 111406, Sep. 2016. [Online]. Available: <https://doi.org/10.1115/1.4034113>
- [52] H. Wei, L. Shu, Y. Yang, Q. Zhou, L. Zhong, and P. Jiang, “An improved sequential multi-objective robust optimisation approach considering interval uncertainty reduction under mixed uncertainties,” *Journal of Engineering Design*, vol. 32, no. 2, pp. 61–89, Feb. 2021. [Online]. Available: <https://doi.org/10.1080/09544828.2020.1858475>
- [53] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, “Evolutionary Algorithms,” *WIREs Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 178–195, 2014. [Online]. Available: <https://doi.org/10.1002/widm.1124>

- [54] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. Springer Science & Business Media, 2010.
- [55] K. Deb, “Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction,” in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, L. Wang, A. H. C. Ng, and K. Deb, Eds. London: Springer, 2011, pp. 3–34. [Online]. Available: https://doi.org/10.1007/978-0-85729-652-8_1
- [56] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II,” in *Parallel Problem Solving from Nature PPSN VI*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer, 2000, pp. 849–858. [Online]. Available: https://doi.org/10.1007/3-540-45356-3_83
- [57] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength pareto evolutionary algorithm,” ETH Zurich, Report, May 2001. [Online]. Available: <https://doi.org/10.3929/ethz-a-004284029>
- [58] Q. Zhang and H. Li, “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec. 2007. [Online]. Available: <https://ieeexplore.ieee.org/document/4358754>
- [59] E. Erdoğan and G. Iyengar, “Ambiguous chance constrained problems and robust optimization,” *Mathematical Programming*, vol. 107, no. 1-2, pp. 37–61, Jun. 2006. [Online]. Available: <https://doi.org/10.1007/s10107-005-0678-0>
- [60] Y. Xu, G. H. Huang, X. S. Qin, and M. F. Cao, “SRCCP: A stochastic robust chance-constrained programming model for municipal solid waste management under uncertainty,” *Resources, Conservation and Recycling*, vol. 53, no. 6, pp. 352–363, Apr. 2009. [Online]. Available: <https://doi.org/10.1016/j.resconrec.2009.02.002>
- [61] 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, Aug. 2018. [Online]. Available: <https://doi.org/10.1115/1.4040710>
- [62] G. C. Calafiore and M. C. Campi, “The Scenario Approach to Robust Control Design,” *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 742–753, May 2006. [Online]. Available: <http://ieeexplore.ieee.org/document/1632303/>
- [63] 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 *Volume 5: 35th Design Automation Conference, Parts A and B*. San Diego, California, USA: ASMEDC, Sep. 2009, pp. 1193–1203. [Online]. Available: <https://doi.org/10.1115/DETC2009-87312>

- [64] Q. Zhou, P. Jiang, X. Huang, F. Zhang, and T. Zhou, “A multi-objective robust optimization approach based on Gaussian process model,” *Structural and Multidisciplinary Optimization*, vol. 57, no. 1, pp. 213–233, Jan. 2018. [Online]. Available: <https://doi.org/10.1007/s00158-017-1746-9>
- [65] S. Cheng, J. Zhou, and M. Li, “A New Hybrid Algorithm for Multi-Objective Robust Optimization With Interval Uncertainty,” *Journal of Mechanical Design*, vol. 137, no. 2, Feb. 2015. [Online]. Available: <https://doi.org/10.1115/1.4029026>
- [66] F. Yin, Y. Liu, and Y. Chen, “Distributionally robust optimisation model for multi-objective hub location problem via considering ambiguousness,” *Transportmetrica A: Transport Science*, vol. 19, no. 3, p. 2094494, May 2023. [Online]. Available: <https://doi.org/10.1080/23249935.2022.2094494>
- [67] M. G. Kapteyn, K. E. Willcox, and A. Philpott, “A Distributionally Robust Approach to Black-Box Optimization,” in *2018 AIAA Non-Deterministic Approaches Conference*. Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2018. [Online]. Available: <https://doi.org/10.2514/6.2018-0666>
- [68] P. Zhang, Y. Liu, G. Yang, and G. Zhang, “A multi-objective distributionally robust model for sustainable last mile relief network design problem,” *Annals of Operations Research*, vol. 309, no. 2, pp. 689–730, Feb. 2022. [Online]. Available: <https://doi.org/10.1007/s10479-020-03813-3>
- [69] W. He, J. Zhao, L. Zhao, Z. Li, M. Yang, and T. Liu, “Data-driven two-stage distributionally robust optimization for refinery planning under uncertainty,” *Chemical Engineering Science*, vol. 269, p. 118466, Apr. 2023. [Online]. Available: <https://doi.org/10.1016/j.ces.2023.118466>
- [70] T. Nguyen, S. Gupta, H. Ha, S. Rana, and S. Venkatesh, “Distributionally robust bayesian quadrature optimization,” in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, Jun. 2020, pp. 1921–1931. [Online]. Available: <https://proceedings.mlr.press/v108/nguyen20a.html>
- [71] H. Rahimian and S. Mehrotra, “Distributionally Robust Optimization: A Review,” *Open Journal of Mathematical Optimization*, vol. 3, pp. 1–85, Jul. 2022. [Online]. Available: <https://doi.org/10.48550/arXiv.1908.05659>
- [72] Z. Li and Z. Li, “Optimal robust optimization approximation for chance constrained optimization problem,” *Computers & Chemical Engineering*, vol. 74, pp. 89–99, Mar. 2015. [Online]. Available: <https://doi.org/10.1016/j.compchemeng.2015.01.003>
- [73] P. Parpas, B. Rustem, and E. N. Pistikopoulos, “Global optimization of robust chance constrained problems,” *Journal of Global Optimization*, vol. 43, no. 2, pp. 231–247, Mar. 2009. [Online]. Available: <https://doi.org/10.1007/s10898-007-9244-z>
- [74] P. E. Yau, “On Algorithms for Nonlinear Minimax and Min-Max-Min Problems and Their Efficiency,” Ph.D. dissertation, Naval Postgraduate School, Monterey, CA, Mar. 2011.

- [75] B. Xu, Y. Sun, X. Huang, P.-a. Zhong, F. Zhu, J. Zhang, X. Wang, G. Wang, Y. Ma, Q. Lu, H. Wang, and L. Guo, "Scenario-Based Multiobjective Robust Optimization and Decision-Making Framework for Optimal Operation of a Cascade Hydropower System Under Multiple Uncertainties," *Water Resources Research*, vol. 58, no. 4, p. e2021WR030965, 2022. [Online]. Available: <https://doi.org/10.1029/2021WR030965>
- [76] B. Wang, X. Xia, H. Meng, and T. Li, "Bad-scenario-set Robust Optimization Framework With Two Objectives for Uncertain Scheduling Systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 143–153, 2017. [Online]. Available: <https://www.ieee-jas.net/en/article/id/fcba7ece-d92f-42d5-ae9d-a88683b743c7>
- [77] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Cambridge University Press, 1993.
- [78] J.-B. Yang and P. Sen, "Preference modelling by estimating local utility functions for multiobjective optimization," *European Journal of Operational Research*, vol. 95, no. 1, pp. 115–138, Nov. 1996. [Online]. Available: [https://doi.org/10.1016/0377-2217\(96\)00300-1](https://doi.org/10.1016/0377-2217(96)00300-1)
- [79] R. Rădulescu, P. Mannion, D. M. Roijers, and A. Nowé, "Multi-objective multi-agent decision making: a utility-based analysis and survey," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 1, Dec. 2019. [Online]. Available: <https://doi.org/10.1007/s10458-019-09433-x>
- [80] I. Dewancker, M. McCourt, and S. Ainsworth, "Interactive preference learning of utility functions for multi-objective optimization," Dec. 2016. [Online]. Available: <https://doi.org/10.48550/arXiv.1612.04453>
- [81] V. Chankong and Y. Haimes, *Multiobjective decision making*, ser. North-Holland series in system science and engineering, A. P. Sage, Ed. New York [u.a.]: North-Holland, 1983, no. 8.
- [82] G. Li, "Online and Offline Approximations for Population Based Multi-Objective Optimization," Ph.D. dissertation, University of Maryland, College Park, 2007.
- [83] J. R. Koehler and A. B. Owen, "9 Computer experiments," in *Handbook of Statistics*, ser. Design and Analysis of Experiments. Elsevier, Jan. 1996, vol. 13, pp. 261–308. [Online]. Available: [https://doi.org/10.1016/S0169-7161\(96\)13011-X](https://doi.org/10.1016/S0169-7161(96)13011-X)
- [84] A. J. Keane and I. I. Voutchkov, "Robust design optimization using surrogate models," *Journal of Computational Design and Engineering*, vol. 7, no. 1, pp. 44–55, Feb. 2020. [Online]. Available: <https://doi.org/10.1093/jcde/qwaa005>
- [85] M. Ribaud, C. Blanchet-Scalliet, C. Helbert, and F. Gillot, "Robust optimization: A kriging-based multi-objective optimization approach," *Reliability Engineering & System Safety*, vol. 200, p. 106913, Aug. 2020. [Online]. Available: <https://doi.org/10.1016/j.ress.2020.106913>

- [86] A. Bhosekar and M. Ierapetritou, “Advances in surrogate based modeling, feasibility analysis, and optimization: A review,” *Computers & Chemical Engineering*, vol. 108, pp. 250–267, Jan. 2018. [Online]. Available: <https://doi.org/10.1016/j.compchemeng.2017.09.017>
- [87] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker, “Surrogate-based analysis and optimization,” *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1–28, Jan. 2005. [Online]. Available: <https://doi.org/10.1016/j.paerosci.2005.02.001>
- [88] A. Forrester, A. Sóbester, and A. Keane, *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, 2008.
- [89] J. Knowles, “ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, Feb. 2006. [Online]. Available: <https://ieeexplore.ieee.org/document/1583627>
- [90] M. Moustapha, A. Galimshina, G. Habert, and B. Sudret, “Multi-objective robust optimization using adaptive surrogate models for problems with mixed continuous-categorical parameters,” *Structural and Multidisciplinary Optimization*, vol. 65, no. 12, p. 357, Dec. 2022. [Online]. Available: <https://doi.org/10.1007/s00158-022-03457-w>
- [91] D. S. Broomhead and D. Lowe, “Multivariable Functional Interpolation and Adaptive Networks,” *Complex Systems*, vol. 2, pp. 321–355, 1988.
- [92] R. Jin, W. Chen, and T. Simpson, “Comparative studies of metamodeling techniques under multiple modelling criteria,” *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 1–13, Dec. 2001. [Online]. Available: <https://doi.org/10.1007/s00158-001-0160-4>
- [93] I. R. Meneghini, F. G. Guimarães, and A. Gaspar-Cunha, “Competitive coevolutionary algorithm for robust multi-objective optimization: The worst case minimization,” in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2016, pp. 586–593. [Online]. Available: <https://ieeexplore.ieee.org/document/7743846>
- [94] R. G. Regis, “Particle swarm with radial basis function surrogates for expensive black-box optimization,” *Journal of Computational Science*, vol. 5, no. 1, pp. 12–23, Jan. 2014. [Online]. Available: <https://doi.org/10.1016/j.jocs.2013.07.004>
- [95] J. M. Moguerza and A. Muñoz, “Support Vector Machines with Applications,” *Statistical Science*, vol. 21, no. 3, pp. 322–336, Aug. 2006. [Online]. Available: <https://doi.org/10.1214/088342306000000493>
- [96] S. M. Clarke, J. H. Griebisch, and T. W. Simpson, “Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses,” *Journal of Mechanical Design*, vol. 127, no. 6, pp. 1077–1087, Aug. 2004. [Online]. Available: <https://doi.org/10.1115/1.1897403>

- [97] Y. Yun, H. Nakayama, and M. Arakawa, "Generation of pareto frontiers using support vector machine." Whistler, B. C., Canada: MCDM, Aug. 2004.
- [98] T. Tan, M. Yang, R. Xie, Y. Cao, and Y. Chen, "Adjustable robust optimization with decision-dependent uncertainty for power system problems: A review," *Energy Conversion and Economics*, May 2025. [Online]. Available: <https://doi.org/10.1049/enc2.70010>
- [99] Q. Zhang and W. Feng, "A unified framework for adjustable robust optimization with endogenous uncertainty," *AIChE Journal*, vol. 66, no. 12, Oct. 2020. [Online]. Available: <https://doi.org/10.1002/aic.17047>
- [100] T. Tan, R. Xie, X. Xu, and Y. Chen, "A robust optimization method for power systems with decision-dependent uncertainty," *Energy Conversion and Economics*, vol. 5, no. 3, pp. 133–145, Jun. 2024. [Online]. Available: <https://doi.org/10.1049/enc2.12117>
- [101] H. K. Chen, X. A. Sun, and H. Yang, "Robust optimization with continuous decision-dependent uncertainty with applications to demand response management," *SIAM Journal on Optimization*, vol. 33, no. 3, pp. 2406–2434, Sep. 2023. [Online]. Available: <https://doi.org/10.1137/22M1502082>
- [102] X. Yu and S. Shen, "Multistage distributionally robust mixed-integer programming with decision-dependent moment-based ambiguity sets," *Mathematical Programming*, vol. 196, no. 1–2, pp. 1025–1064, Oct. 2020. [Online]. Available: <https://doi.org/10.1007/s10107-020-01580-4>
- [103] A. Ben-Tal, A. Goryashko, E. Guslitser, and A. Nemirovski, "Adjustable robust solutions of uncertain linear programs," *Mathematical Programming*, vol. 99, no. 2, pp. 351–376, Mar. 2004. [Online]. Available: <https://doi.org/10.1007/s10107-003-0454-y>
- [104] A. Selvi, A. Ben-Tal, R. Brekelmans, and D. den Hertog, "Convex maximization via adjustable robust optimization," *INFORMS Journal on Computing*, vol. 34, no. 4, pp. 2091–2105, Jul. 2022. [Online]. Available: <https://doi.org/10.1287/ijoc.2021.1134>
- [105] M. Cavazzuti, *Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. [Online]. Available: <https://doi.org/10.1007/978-3-642-31187-1>
- [106] MathWorks, "Find Pareto front of multiple fitness functions using genetic algorithm - MATLAB gamultiobj," 2022. [Online]. Available: <https://www.mathworks.com/help/gads/gamultiobj.html>
- [107] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: <https://ieeexplore.ieee.org/document/996017>
- [108] 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*,

- vol. 2. Vancouver, BC, Canada: IEEE, 1995, pp. 1556–1561. [Online]. Available: <http://ieeexplore.ieee.org/document/537993/>
- [109] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.
- [110] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [111] M. J. Colaço and G. S. Dulikravich, “A Survey of Basic Deterministic, Heuristic, and Hybrid Methods for Single-Objective Optimization and Response Surface Generation,” in *Thermal Measurements and Inverse Techniques*. CRC Press, 2011.
- [112] MathWorks, “Constrained Nonlinear Optimization Algorithms - MATLAB & Simulink,” 2022. [Online]. Available: <https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html#brnpd5f>
- [113] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, “DACE: A MATLAB Kriging Toolbox,” Technical University of Denmark, Denmark, Technical Report IMM-TR-2002-12, Aug. 2002.
- [114] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, “Design and Analysis of Computer Experiments,” *Statistical Science*, vol. 4, no. 4, pp. 409–423, Nov. 1989. [Online]. Available: <https://doi.org/10.1214/ss/1177012413>
- [115] J. N. Fuhg, A. Fau, and U. Nackenhorst, “State-of-the-art and comparative review of adaptive sampling methods for kriging,” *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2689–2747, Jun. 2021. [Online]. Available: <https://doi.org/10.1007/s11831-020-09474-6>
- [116] B. L. Welch, “On the Comparison of Several Mean Values: An Alternative Approach,” *Biometrika*, vol. 38, no. 3/4, pp. 330–336, 1951. [Online]. Available: <https://doi.org/10.2307/2332579>
- [117] P. A. Games and J. F. Howell, “Pairwise Multiple Comparison Procedures with Unequal N’s and/or Variances: A Monte Carlo Study,” *Journal of Educational Statistics*, vol. 1, no. 2, pp. 113–125, Jun. 1976. [Online]. Available: <https://doi.org/10.3102/10769986001002113>
- [118] MathWorks, “ga - find minimum of function using genetic algorithm.” [Online]. Available: <https://www.mathworks.com/help/gads/ga.html>
- [119] D. C. Montgomery, *Design and analysis of experiments*, 9th ed. Hoboken, NJ: John Wiley & Sons, Inc., 2017, includes bibliographical references and index. - Print version record and CIP data provided by publisher; resource not viewed.
- [120] V. C. Aute, “Single and multiresponse adaptive design of experiments with application to design optimization of novel heat exchangers,” phdthesis, University of Maryland, College Park, 2009.

- [121] M. Goerigk, J. Kurtz, M. Schmidt, and J. Thurauf, “Connections between Robust and Bilevel Optimization,” Nov. 2024.
- [122] R. Paradiso, A. Georghiou, S. Dabia, and D. Tönissen, “Exact and approximate schemes for robust optimization problems with decision-dependent information discovery,” *INFORMS Journal on Computing*, Dec. 2024. [Online]. Available: <https://doi.org/10.1287/ijoc.2023.0290>
- [123] D. Yazdani, M. N. Omidvar, D. Yazdani, J. Branke, T. T. Nguyen, A. H. Gandomi, Y. Jin, and X. Yao, “Robust optimization over time: A critical review,” *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 5, pp. 1265–1285, Oct. 2024.
- [124] R. Azzouz, S. Bechikh, and L. Ben Said, “Dynamic Multi-objective Optimization Using Evolutionary Algorithms: A Survey,” in *Recent Advances in Evolutionary Multi-objective Optimization*, S. Bechikh, R. Datta, and A. Gupta, Eds. Cham: Springer International Publishing, 2017, pp. 31–70.
- [125] K. Zhang, N. Chen, P. Zeng, J. Liu, and M. Beer, “An efficient reliability analysis method for structures with hybrid time-dependent uncertainty,” *Reliability Engineering & System Safety*, vol. 228, p. 108794, Dec. 2022.
- [126] V. Pandey, Z. P. Mourelatos, and A. Skowronska, “Flexible design of systems considering time-dependent reliability,” in *Volume 2B: 41st Design Automation Conference*, ser. IDETC-CIE2015. American Society of Mechanical Engineers, Aug. 2015.
- [127] L. Wei and L. Jin, “Robust neural differential solution for time-dependent nonlinear optimization with noises rejection,” *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 418–432, 2025.
- [128] T. I. Fossen and T. Perez, “Marine Systems Simulator (MSS),” 2004. [Online]. Available: <https://github.com/cybergalactic/MSS>
- [129] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 1st ed. John Wiley & Sons Ltd., 2011.
- [130] The PyTorch Foundation, “MLP — Torchvision main documentation,” 2024. [Online]. Available: <https://pytorch.org/vision/main/generated/torchvision.ops.MLP.html>
- [131] BayesFusion, LLC, “GeNIe Modeler: Programmer’s Manual,” Sep. 2023. [Online]. Available: <https://support.bayesfusion.com/docs/GeNIe.pdf>
- [132] European Union-Copernicus Marine Service, “Global Ocean Physics Analysis and Forecast.” [Online]. Available: <https://doi.org/10.48670/moi-00016>
- [133] —, “Global Ocean Waves Analysis and Forecast.” [Online]. Available: <https://doi.org/10.48670/moi-00017>
- [134] —, “Global Ocean Hourly Sea Surface Wind and Stress from Scatterometer and Model.” [Online]. Available: <https://doi.org/10.48670/moi-00305>