Design Space Covering for Uncertainty: Exploration of a New Methodology for Decision Making in Early Stage Design

by

Lauren Rose Claus

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Naval Architecture and Marine Engineering)
in The University of Michigan
2019

Doctoral Committee:

Associate Professor Matthew Collette, Chair
Professor Brian Denton
Associate Professor David Singer
Professor Armin Troesch

Lauren Claus
clausl@umich.edu
ORCID iD: 0000-0002-9694-8298

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

DSC-U Design Space Covering for Uncertainty

DSR Design Space Reduction

LP Linear Programming

LR Lagrangian Relaxation

LSC Level Set Covering

PBD Point Based Design

RBDO Reliability Based Design Optimization

SBD Set Based Design

SCP Set Covering Problem

ABSTRACT

Decisions made in early-stage design are of vital importance as they significantly impact the quality of the final design. Despite recent developments in design theory for early-stage design, designers of large complex systems still lack sufficient tools to make robust and reliable preliminary design decisions that do not have a lasting negative impact on the final design. Much of the struggle stems from uncertainty in early-stage design due to loosely defined problems and unknown parameters. Existing methods to handle this uncertainty in point-based design provide feasible, but often suboptimal, solutions that cover the range of uncertainty. Robust Optimization and Reliability Based Design Optimization are examples of point-based design methods that handle uncertainty. To maintain feasibility over the range of uncertainty, these methods accept suboptimal designs resulting in a design margin. In set-based design, design decisions are delayed preventing suboptimal final designs but at the expense of computational efficiency. This work proposes a method that evaluates a compromise between these two methodologies by evaluating the trade off of the induced regret and computational cost of keeping a larger design space. The design space covering for uncertainty (DSC-U) problem defines the metrics regret, which measures suboptimality, and space remaining, which quantifies the design space size after it is reduced. Solution methods for the DSC-U problem explore the trade space between these two metrics. When there is uncertainty in a problem, and the design space is reduced, there is the possibility that the optimal solution for the realized values of the uncertainty parameters has been eliminated; but without performing the design space reduction, it is computationally expensive to properly explore the original de-

sign space. Because of this, smart design space reductions need to be made to avoid the elimination of the optimal solution. To make smart design space reductions, designers need information regarding the design space and the trade-offs between the computational efficiency of a smaller subspace and the expected regret, or suboptimality, of the final design. As part of the DSC-U defitition, two separate spaces for the design variables and the uncertain parameters are defined. Two algorithms are presented here that solve the DSC-U problem as it is defined. A nested optimizer algorithm using a single objective optimization problem, nested in a multi-objective optimization problem is capable of finding the Pareto front in the regret-space remaining trade space for small problems. The nested optimizer algorithm is used to solve a box girder design and a cantilever tube design problems. The level set covering (LSC) algorithm solves for the Pareto front by solving the set covering problem with level sets corresponding to allowable regret levels. The LSC is used to solve a 7-variable Rosenbrock problem and a midship design problem. The presented solutions show that the DSC-U problem is a valid approach for handling uncertainty in early-stage design.

CHAPTER I

Introduction

1.1 Background and motivation

Decisions made in early stage design are of vital importance as they significantly impact the quality of the final design. Despite recent developments in design theory for early stage design, designers of large, complex systems are still lacking sufficient tools to make robust and reliable preliminary design decisions that do not have a lasting negative impact on the final design. Much of the struggle stems from uncertainty in early stage design due to loosely defined problems and unknown parameters. Existing methods to handle this uncertainty in Point Based Design (PBD), such as robust optimization and reliability based design optimization, provide feasible, but often suboptimal, solutions that cover the range of uncertainty. To maintain feasibility over the range of uncertainty, these methods accept suboptimal designs introducing a design margin. In Set Based Design (SBD), design decisions are delayed resulting in a design space being propagated through the design process instead of a single design point. This method avoids the suboptimal results of the point based design methods, but at the cost of computational efficiency.

Many of the differences between SBD and PBD result in extremes for the optimality of the design and the size of the design space. In SBD the solution will be optimal, or near-optimal, but at the price of additional evaluations for more of the

design space. In PBD the solution is not likely to be optimal, but comparatively little of the design space is evaluated. When dealing with extreme design space reduction that often occurs in PBD, the optimal solution may be eliminated from further evaluation due to the uncertainty in the problem. There is no existing method to evaluate the trade off of the computational cost of delaying decisions and the suboptimality resulting from premature design decision.

This work proposes a method that can be a tool to designers by providing information on this trade off. The Pareto front in the trade space between how large of a design space to keep, allowing design decisions to be delayed, and the expected suboptimality, due to decision making under uncertainty, of the final design is found.

1.2 Design space reduction covering uncertainty

There is a significant amount of uncertainty in early stage design due to multiple sources. The uncertainty is reduced through the design process as the design problem becomes better defined and parts of the design are finalized. The design decisions made in early stage design are important, as changes to the design are costly later in the design process. A graph depicting the general level of uncertainty and the cost attributed to making changes to the design is shown in Figure 1.1. In point based design, uncertainty is typically handled by adding margins to constraints to maintain feasibility, but this reduces the performance of the final design. In set based design, uncertainty is handled by maintaining a set of possible solutions and waiting until uncertainty is reduced to make further design decisions. The existing point-based and set-based design methods have opposite benefits and drawbacks. In point-based design methodology a single design is selected in the early stages of the design process ,and this design is then progressively changed through the design process. This method typically results in a suboptimal design because important design decisions are made while there is still a high level of uncertainty in the problem

definition, but the method has a relatively low computational cost. In set-based design,a set of designs, or a set of design elements, is created in the early stages of the design process, and is progressively reduced until a final solution is selected. This results in a near optimal final design, because design decisions are made under less uncertainty, but the large number of designs to evaluate is computationally expensive.

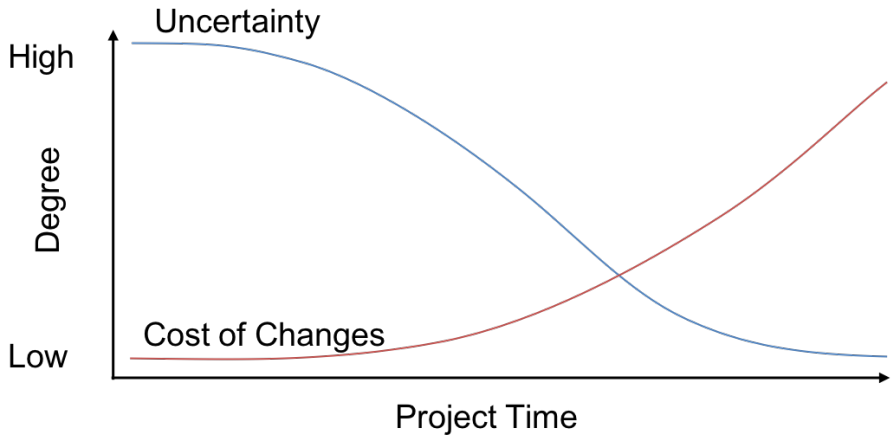

Figure 1.1: Early stage design is characterized by a high degree of uncertainty and a low cost incurred by making changes to the design.

The methods discussed above to handle uncertainty in early stage design require significant sacrifices to afford the advantages that they have. In PBD methods, design decisions are not able to be delayed, forcing decisions to be made while the problem is still loosely defined. On the other end of the spectrum is SBD, which sacrifices computational efficiency for the ability to delay design decisions. Neither of these methods are able to compromise having the ability to delay design decisions while maintaining computational efficiency.

1.3 Contributions

A new method to handle uncertainty in early stage design is defined as part of this dissertation. This work also includes two algorithms to solve this new problem definition. The major contributions of this dissertation are summarized below.

1. **Definition of the design space covering for uncertainty problem.** The design space covering for uncertainty (DSC-U) problem is a new problem definition to examine early stage design for uncertainty. This problem implements a novel decomposition of the design space for handling design variables and uncertain parameters for design space exploration in early stage design. The problem defines a trade space between regret and space remaining to be used as a tool for design space reduction decision making in early stage design.

2. **Development of the nested algorithm.** The nested algorithm was developed to solve the DSC-U problem. This algorithm uses nested single and multi objective optimization methods to solve for the Pareto front in the regret and space remaining trade space defined by the DSC-U problem. The algorithm has proven successful in solving small problems.

3. **Development of the level set covering algorithm.** The level set covering (LSC) algorithm was developed to solve the DSC-U problem for larger design problems. The algorithm uses a regret threshold to define level sets that are used to find a subspace by solving the set covering problem. This solves for the Pareto front in the trade space between regret and space remaining.

1.4 Overview of Dissertation

This dissertation use the following five chapters to present the design space covering for uncertainty problem and two solution methods. The design space covering for uncertainty problem is described in Chapter III. Chapter II provides background information for the work presented in this dissertation. The two solution methods are presented in Chapters IV and V. A nested algorithm that is suitable for small problems is presented in Chapter IV along with the results from two small test problems. A level set covering algorithm that is suitable for larger problems is presented

in Chapter V with results from a medium problem and large problem. Chapter VI discusses the conclusions of this work, the author's research contributions, and the author's recommendations for future work.

CHAPTER II

Background

2.1 Marine Design

2.1.1 Point Based Design Methodology

The standard design method for naval ship design is Point Based Design (PBD). In PBD, a single design point is selected and progressively changed until a final solution is decided upon. In *Evans* (1959) this iterative process was described as a design spiral. Each of the 'spokes' of the spiral correspond to different aspects of the design. As the design progresses through the spiral, each aspect of the design is sequentially refined to improve the design performance. These decisions are made considering a single aspect of the design, and not the design as a whole. A ship is a large complex problem and each of the aspects of the design are related to each other, which creates the need to go through the different aspects of the design again and account for the changes that were made in the last pass through the spiral. This process is continued, typically, only until the time allocated to each stage of the design is up and a 'good enough' solution has been found.

Many of the different aspects of the design are coupled to each other, making a design decision for one aspect affect the analysis and optimal decisions for another aspect. This creates unknown parameters, or uncertainty, in the analyses for each

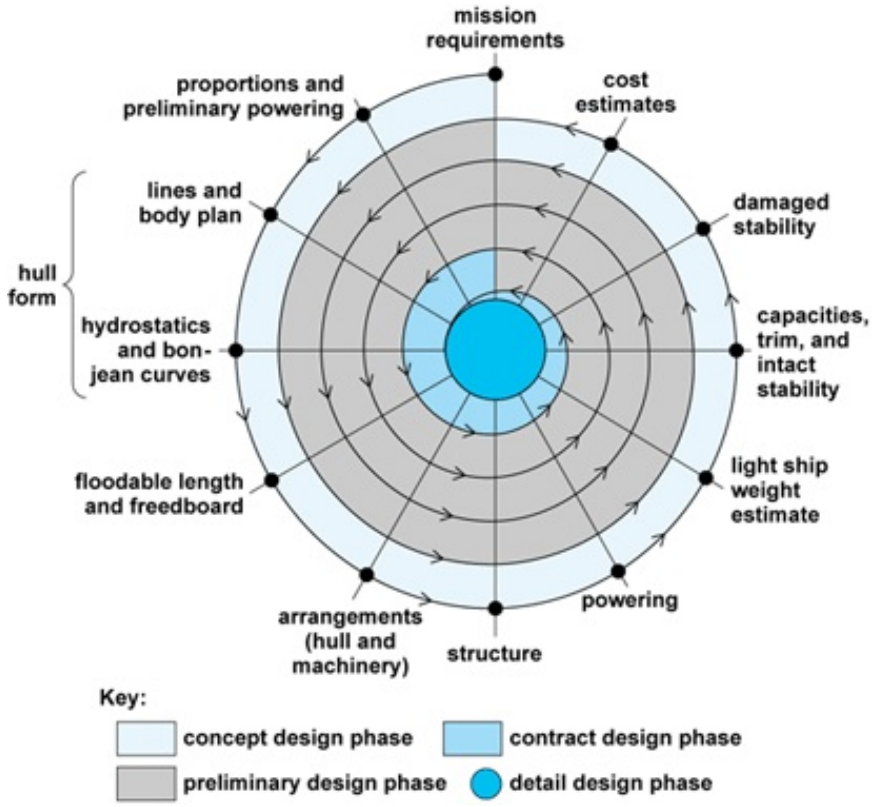

Figure 2.1: The design spiral based on the one presented by *Evans* (1959).

aspect of the design. Multiple methods have been created and improved to account for this uncertainty in decision making.

One existing solution to problems with unknown parameters is Robust Optimization. The main characteristics of robust optimization is the goal of finding an optimal solution such that the feasibility of the design is minimally affected by parameter uncertainty (*Bertsimas et al.*, 2011).

The concept of robust design has been credited to G. Taguchi who has been called the "father of robust design" (*Byrne and Taguchi*, 1986; *Beyer and Sendhoff*, 2007). In Taguchi's design methodology, performance variations were considered as *noise factors* to the *control parameters* or design variables (*Beyer and Sendhoff*, 2007). Robust design is able to handle many uncertainty types such as changing environmental or operating conditions, production tolerances, uncertainties in system output, and feasibility uncertainties (*Beyer and Sendhoff*, 2007). A simple example of robust optimization with three uncertain parameters is shown in Figure 2.2. The robust solution is the best worst-case, so the minimum (best case) of the supremum (worst case) of the three functions is considered to be the robust optimum.

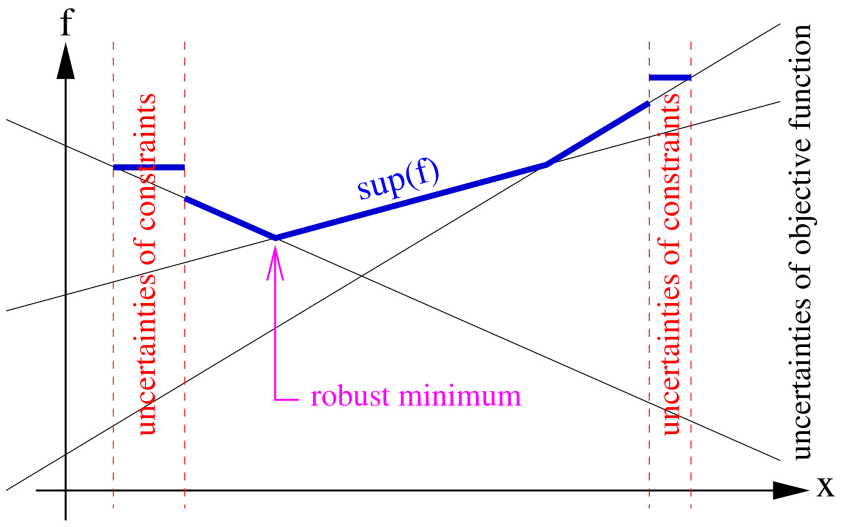

Figure 2.2: Example of robust linear optimization from *Beyer and Sendhoff* (2007).

Many different objective functions have been suggested in the literature such as minimizing the worst case regret and minimizing the worst case objective function, known as minmax robustness (*Ehrgott et al.*, 2014). Another approach is to formulate the problem as a multi objective optimization problem with the mean and standard deviation of the objective function as the two objectives (*Wang and Shan*, 2004). One major drawback of robust optimization is that the solutions are suboptimal for many of the possible realized values of the uncertain parameters (*Ehrgott et al.*, 2014). Another limitation of robust optimization is the lack of established methods for multi-objective problems (*Ehrgott et al.*, 2014). In *Goerigk and Schöbel* (2014) an approach called recovery-to-optimality is presented which gives a solution that minimizes the recovery cost to the optimal solution when the uncertain parameters are realized. While the method generates solutions that have good objective values, the feasibility of the initial solution (pre-recovery) is not guaranteed (*Goerigk and Schöbel*, 2014). While this provides the designer with information about how a decision will impact options later in the process, the typical cost of changing a major design parameter is high, making this method unsuitable for application to early stage design of large projects.

Reliability Reliability Based Design Optimization (RBDO) is another approach that is able to optimize problems that have uncertain parameters. RBDO methods approach uncertainty by optimizing an objective function with a constraint to the reliability of the problem; this is done by using a reliability assessment to calculated the probability that constraints will be satisfied under uncertainty (*Wang and Shan*, 2007; *Yao et al.*, 2011). Traditionally, RBDO methods require a known distribution for uncertainty parameters but there has been work into alternative methods that also include interval uncertainty (*Du*, 2012; *Huang et al.*, 2017). As with robust optimization, a major drawback of RBDO is that the method is designed to give a single solution, or solution set, that is feasible, but suboptimal, for the entire interval.

2.1.2 Set Based Design Methodology

Another approach that has been used for early stage design is SBD which differs from PBD methods in that it is a convergent method (*Mckenney*, 2013). Solutions in SBD are found by eliminating infeasible and dominated alternatives from an initial broad set of design values (*Mckenney*, 2013). Fundamentally, SBD and point based optimization methods solve the design problem from different directions, SBD eliminates the worst designs from a set of designs while point based optimization methods are trying to search for the best solution (*Mckenney*, 2013). The driving forces in designer's interest in SBD are the ability to delay decisions until later in the design process when uncertainty is reduced, and allowing stakeholders the ability to influence the design later in the design process (*Singer et al.*, 2009). Figure 2.3 shows how the committed costs and management influence through the design process are altered in SBD versus PBD. Unfortunately, the delayed design decisions also result in large design spaces that are kept further into the design process resulting in a high computational cost.

It has been shown that the SBD process is robust to requirement changes through the design process and allows the designer to see the impact of those changes to the design (*Mckenney et al.*, 2011). This is important because, as shown in figure 2.3, the requirements are not fully defined in early stage design. The first application of SBD to naval design is of the Ship to Shore Connector which was successfully designed using SBD in the preliminary design stage(*Mebane et al.*, 2011).

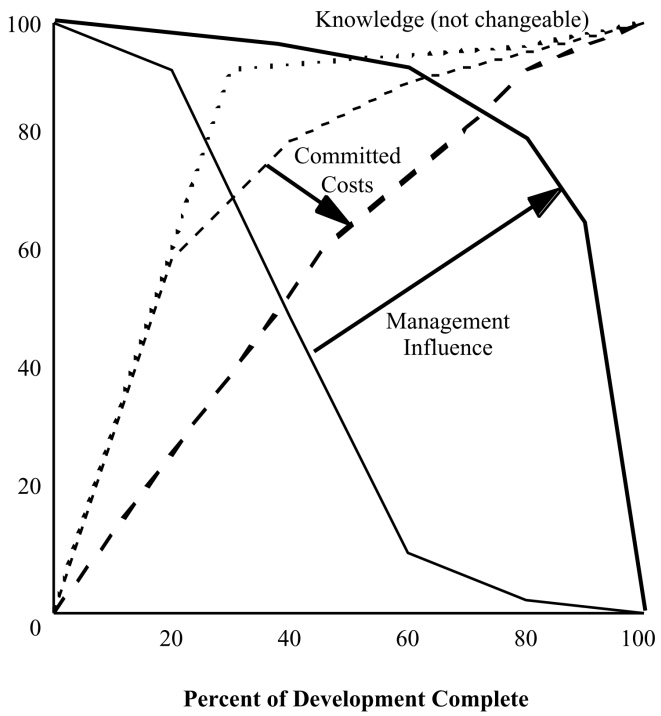

Figure 2.3: Graphic showing how the committed costs and management influence are changed between PBD and SBD from *Bernstein* (1998).

2.2 Metaheuristic Optimization Methods

Metaheuristic optimization frameworks are a group of optimization algorithms that have been created to solve problems in a reasonable time when gradient methods cannot be used (*Altay and Alatas*, 2018). There are many different types of these algorithms, and there are eleven main groupings of these algorithms: physics, sociology, music, swarm, chemistry, biology, mathematics, plant, water, sports, and hybrid based (*Altay and Alatas*, 2018).

2.2.1 Genetic Algorithms

Genetic algorithms are a subcategory of evolutionary algorithms and are considered to be a metaheuristic method. Genetic algorithms (GA) are inspired by Darwin's theory of natural selection and the theory is applied to a population of individuals. Each individual in the population has a string of genes making up a chromosome,

which is the set of variable values belonging to that individual. All forms of GAs have four phases: mutation, crossover, fitness evaluation, and selection. These operations are repeatedly performed on a fixed size, finite population until a convergence criterion is met. The objective function, or functions, are evaluated for every individual in the population during the fitness evaluation operation to determine the fitness for each individual. The mutation operation models random changed to an individual's genetic makeup and can change randomly selected genes for some individuals in the population. The crossover operation is an exchange of genetic information between individuals and during the operation new children individuals are created as a combination of multiple parent individuals of the previous generation *Schmitt* (2001). The selection operation uses the fitness of each individual to select which will be in the next generation.

2.2.1.1 Single Objective Genetic Algorithm

A single objective genetic algorithm (SOGA) framework was used to solve the single objective problems present in this work. The in-house SOGA used in this work has constraint handling, elitism, mutation, and crossover. Both quadratic and feasible-penalty constraint handling were used in this work. The quadratic penalty is calculated as

$$p = \frac{P}{2} \sum_{i=1}^{n} v_i^2 \tag{2.1}$$

where p is the penalty amount, P is the constraint parameter constant, n is the number of constraints, and v_i is the violation for constraint i. Elitism is implemented in the algorithm by the best individuals for each generation being cloned to the next generation; the elitism parameter gives the percent of the population to be cloned. A two-pass tournament selection was implemented to select which individuals will be used for the crossover operation. In the two-pass tournament selection the population is randomly mixed, and two individuals are selected to compete , the one with the

better fitness is added to the set of chosen individuals; this is the first pass for the tournament selection. The population is randomly mixed again, and the same process is repeated to select individuals to compete and add the better performing individual to the set of chosen individuals. This set of chosen individuals is then used for crossover where a multi-point crossover with two parents is implemented. In the crossover step two parents are selected from the set of chosen individuals and their chromosomes are mixed to create two new individuals per the crossover methods selected for the binary and continuous genes. The crossover only happens with the crossover probability c_p, with probability $1 - c_p$ the parents are cloned to the next generation. The new generation of children is then mutated with probability c_m; note this probability is typically low and most commonly $\leq 1\%$. Mutation of the population is done by flipping a digit in a binary string for binary genes, and by adding or subtracting a random scaled value for continuous genes. The described algorithm is a standard SOGA method that was developed by *Temple* (2015).

2.2.1.2 Multi Objective Genetic Algorithm

In this work, the specific GA framework that was used for multi-objective problems was the nondominated sorting genetic algorithm II (NSGA-II) by *Deb et al.* (2002). The NSGA-II was developed to handle constrained problems while retaining a diverse population and elitism in the population. The algorithm employs a sorting method based on individual feasibility, domination status, and crowding distance. The NAGA-II utilizes an efficient sorting method to determine the Pareto front rank of each individual where F_1 is the true Pareto front of the solution, F_2 is the front of those individuals only dominated by the individuals in F_1, etc. To establish an individual p's non-domination rank, the number of solutions which dominate the individual, n_p, and the set of solutions dominated by the individual, S_p, are found. If an individual's domination count is 0 the individual is in the first nondominated front,

F_1, and is done being sorted. For each individual p in F_1 the domination count n_q for each solution q in S_p is reduced by one. This updates the domination count for the individuals to exclude the non-dominated individuals in F_1. If an unsorted individual now has a domination count of 0 it is in the second nondominated front, F_2. This process continues until N individuals have been sorted into nondominated fronts. This operation is performed on a combined population of parents (P_t) and children (Q_t) as shown in figure 2.4. To reduce this combined population to the size of the original population the points in each Pareto front are added to the next generation starting with F_1.

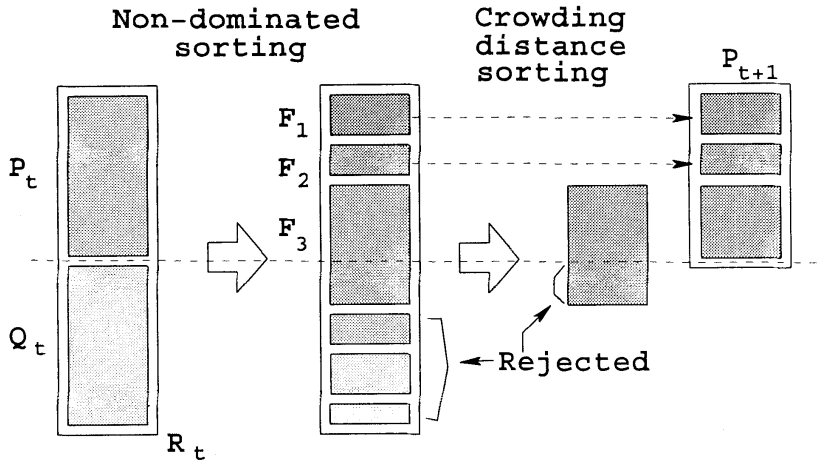

Figure 2.4: Procedure for the NSGA-II reproduced from *Deb et al.* (2002)

When only part of a non-dominated front is passed on to the next generation, a crowding metric is used to sort within the non-dominated front. The crowding distance metric is a measure of the perimeter of the cuboid created with the two nearest points in the front. This is calculated by first sorting the individuals in ascending order of fitness for each objective function. For each objective function the boundary solutions, those that have the highest or lowest function values, the crowding distance is set to infinity. For all other points the crowding distance is set to $\sum_{j=0}^{m} |f_j(x_{i-1}) - f_j(x_{i+1})$ where m is the number of objective functions. Figure 2.5

shows the crowding distance of the i^{th} point. The points with the highest crowding distance are selected for the next generation.

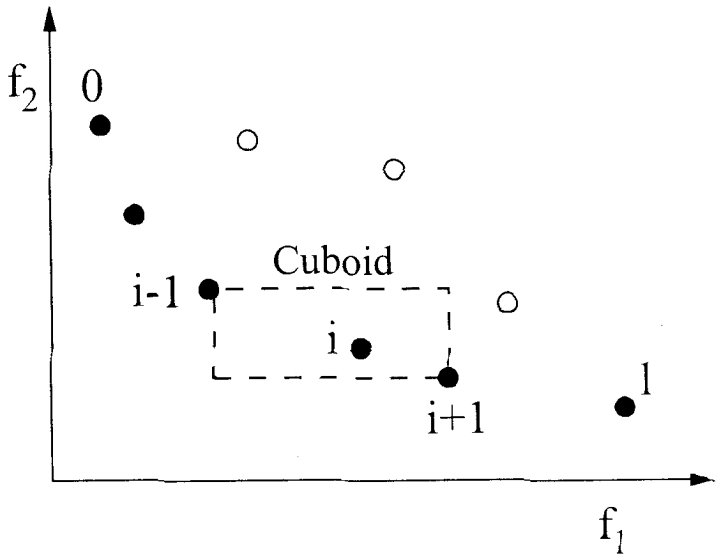

Figure 2.5: Crowding distance calculation for selecting points when a partial front is passed on to the next generation in NSGA-II reproduced from *Deb et al.* (2002)

The above description of NSGA-II is able to handle unconstrained problems, and it is also adaptable to constrained problems. The constrained NSGA-II uses individual feasibility, and magnitude of constraint violation, to sort individuals into non-dominated fronts. When sorting, there are three possible conditions that make solution i constrained-dominate solution j *Deb et al.* (2002):

1. Solution i is feasible and solution j is not.

2. Solutions i and j are both infeasible, but solution i has a smaller overall constraint violation.

3. Solutions i and j are feasible and solution i dominates solution j.

From here, the process from the unconstrained NSGA-II can be used to select individuals for the next generation.

2.3 Design Space Reduction

In Design Space Reduction (DSR) the design space is made smaller by either eliminating one or more variables to lower the dimension of the space, or reducing the range of one or more variables to reduce the size of the space. DSR is often used as a preliminary step before large optimization problems to improve computation time, and before creating surrogate models to improve model accuracy (*Qiu et al.*, 2016; *Liu and Collette*, 2014). In *Wang and Shan* (2004) a design space is reduced by identifying regions, through sampling, where the objective function is below a specified value (for minimization problems). This approach is extended to multi objective problems by identifying the intersection of the subspaces created by each objective (*Wang and Shan*, 2004). In *Tseng et al.* (2014) a simplex based DSR method is presented to use a collection of simplexes to locate a smaller promising area prior to implementing a simulated annealing algorithm. Another approach to DSR is to reduce the dimension of the design space. In *Viswanath et al.* (2009) this is done by transforming some variables in the design space rather than removing them, which avoids losing information relating to all the variables. Again, this was done as a preliminary search for a global optimum and not as a way to retain good solutions under uncertainty. In *Qiu et al.* (2016) self organizing maps and fuzzy clustering are used to reduce a design space prior to building a surrogate model to improve computation time and accuracy.

2.4 Set Covering Problem

The SCP is one of the problems proven to be NP-complete by Karp (*Karp*, 1972). The problem has many applications such as crew scheduling and location selection for facilities (*Crawford et al.*, 2014).The problem is to find the minimum cost associated with selecting a number of sets from a list of sets such that all the elements of an

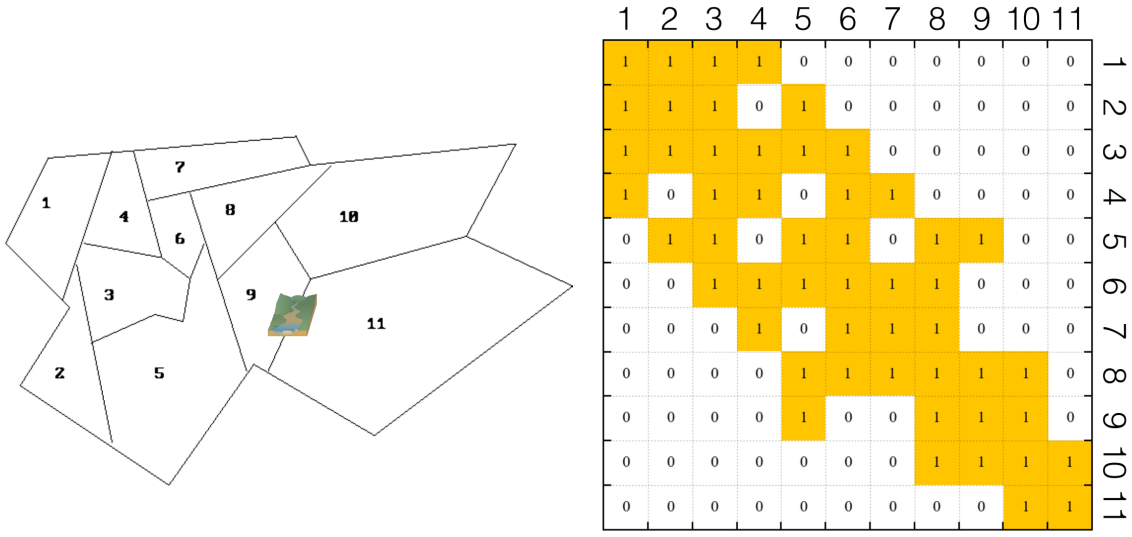

Figure 2.6: Graphical interpretation of the school locating example of the SCP with the A matrix corresponding the the problem.*Zhu* (2018)

input are contained in the union of the selected sets.The mathematical model for the Set Covering Problem (SCP) is

$$\begin{aligned} \min \quad & f(x) = \sum_{j=1}^{n} c_j^T x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad \forall i \in I \\ & x_j \in \{0,1\}, \quad \forall j \in J \end{aligned} \tag{2.2}$$

In this formulation columns of A are selected such that every row in A is covered. In this formulation n is the number of columns in A, c_j is the cost of selecting column j, $a_{i,j}$ is the entry in the boolean array if column j covers row i, and x_j is the integer solution array depicting which columns have been selected.

To further explain the SCP a small example shown in Figure 2.6 will be used. For this problem, there is a finite set of possible school locations which must 'cover' the entire town. A small theoretical town is shown on the left in figure 2.6 with the possible school locations and the precinct area that it can serve; this example is uni-

cost meaning a school will cost the same in each precinct. In addition, each school is able to serve the students in the neighboring precincts unless there is a barrier between the precincts such as the river between precincts 9 and 11. To solve the problem the A matrix must be created to represent the map, this is shown in figure 2.6. In the matrix $a_{i,j} = 1$ =TRUE if a school in precinct j (the column) can serve the students of precinct i (the row) and $a_{i,j} = 0$ =FALSE otherwise. The solution to this small example is three and there are multiple solutions including $\{1, 6, 10\}$, $\{3, 8, 11\}$, and $\{4, 5, 11\}$. While this simple problem can be solved by inspection, other techniques are needed to solve any more significant problems.

2.4.1 Existing Solutions for Set Covering Problem

There are many heuristic approximation methods in the literature that can find a near-optimal solution for the SCP. Many of the methods are based on linear programming relaxation or Lagrangian relaxation. The Linear Programming (LP) relaxation of the SCP eliminates the integer requirement for x_j making the requirements of the problem $0 \leq x_j \leq 1,\ \forall j \in N$ instead of $x_j \in \{0, 1\},\ \forall j \in N$. The problem can now be solved by general-purpose LP solution methods but these methods are typically computationally expensive (*Caprara and Toth*, 2000). With Lagrangian Relaxation (LR) the constraint that every row must be covered is relaxed and a penalty for uncovered rows is added to the objective function.

Many greedy heuristic versions have been shown to be quick to find solutions to the SCP but they are not always the optimal solution (*Crawford et al.*, 2014; *Álvarez-sánchez et al.*, 2015). While they all use different metrics for measuring what the best next step is they follow the same general structure. They start with an empty solution set, S, and a set containing the uncovered rows, M'. Each column j has a score σ_j, which is calculated differently or each method, and the column with the best score is added to the solution set S. The set of uncovered rows M' is updated to remove

the rows that are covered by the new column in S and the procedure continues until all the rows are covered. The solution vector S typically contains columns, which if removed, would not change the feasibility of the solution set; a procedure to remove some of these columns from the solution set can then performed (*Caprara and Toth*, 2000).

There have been several heuristic methods developed to solve the SCP. A genetic algorithm based method is presented in *Beasley and Chu* (1996) where a crossover method similar to what is found in a genetic algorithm is used with a reduction method to eliminate redundant columns in the solution. Methods based on simulated annealing, a metaheuristic, are also present in the literature (*Jacobs and Brusco*, 1995; *Brusco et al.*, 1999).

2.4.2 Implemented Solution to Set Covering Problem

In this work the python module SetCoverPy developed by Zhu was used to solve the set cover problem (*Zhu*, 2018). This method uses a Lagrangian Relaxation (LR) heuristic method, greedy heuristic, and the Lagrangian dual to solve the problem. As a LR method, the inequality constraint that every row must be covered is relaxed to become a penalty in the objective function using the Lagrangian multiplier vector $\mathbf{u}$. The solution of the LR heuristic method is a lower bound on the original SCP. The selection of the Lagrangian multiplier $\mathbf{u}$ for the LR heuristic is done by solving for the Lagrangian multiplier that maximizes $L(\mathbf{u})$ to push the lower bound solution from the LR heuristic to be nearly equal to the solution of the original problem. The subgradient method is used to solve the Lagrangian Dual problem. With the solution $\mathbf{u}$ to the Lagrangian Dual optimization problem, a new score metric for the greedy algorithm can be used shown in equation 2.3. With this new score metric, the greedy

algorithm is now equivalent to solving the Lagrangian subproblem with the constraint that all rows must be covered.

$$\begin{aligned} \sigma_j &= \begin{cases} \frac{\gamma_j}{\mu_j}, & \text{if } \gamma_j > 0 \\ \gamma_j \mu_j, & \text{if } \gamma_j < 0 \end{cases} \\ \gamma_j &= c_j - \sum_{i \in I_j^*} u_i^k \\ \mu_j &= |I_j \cap M^*| \end{aligned} \tag{2.3}$$

2.5 Level Set

In this work the level set for for the level r is defined as

$$S_r = \{x \in X : \ f(x) \leq r\} \tag{2.4}$$

A graphical example of the level set is shown in figure 2.7. The level set is the black set that corresponds to where the function value is less than the threshold. This method is used in Chapter V to create a set of designs that have a regret under a threshold regret r.

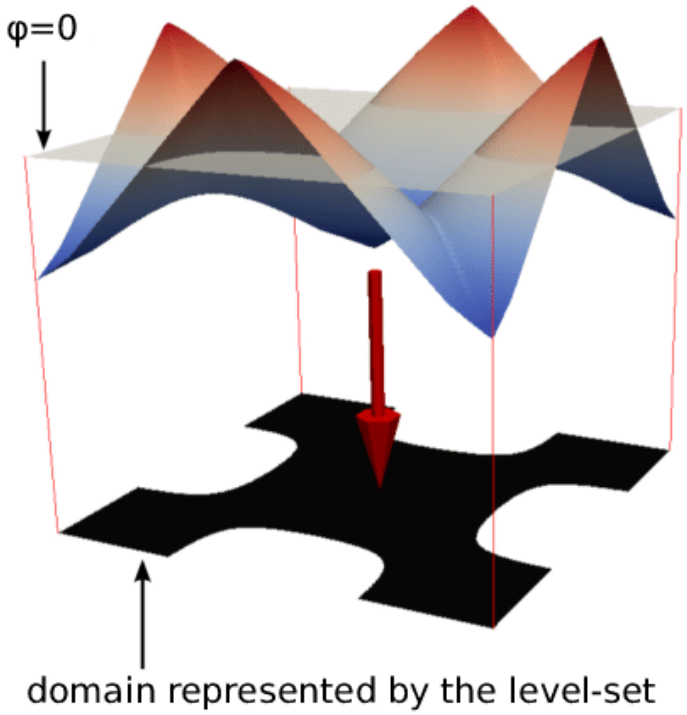

Figure 2.7: Graphical depiction of a level set from‘*Faure et al.* (2016)

CHAPTER III

Design Space Covering for Uncertainty

New methods for design space exploration and decision making in early stage design are being developed for large design problems, in particular marine design problems. There are two major existing design methodologies for marine design: point based design and set based design.

Point based design has been, and remains to be, the standard marine design methodology. The defining factor of point based design is that a single design point is initially selected and then incrementally changed through the design process. This method exploits the information known regarding the performance of the design being evaluated, resulting in the initial design point having a significant influence on the final design. This also results in no significant exploration of the design space. The initial design points are typically selected using the knowledge of experienced designers and the performance of past designs; this means that even between problems, exploitation of known designs is used over exploration of the design space. Compared to set based design, the computation cost is lower, but the lack of exploration means that the final solution is likely not the optimal solution.

In contrast, set based design stresses exploration of the design space over exploitation of design points. In set based design, decisions regarding design variables are delayed until further in the design process when uncertainty is reduced. This provides

a better final design, but the computational cost is high as all points remaining in the design space must be reevaluated at each step as the uncertainty is progressively reduced.

While both point and set based design are valid methodologies, they both have different strengths and weaknesses. This leads to the question of if there is an unexplored methodology that would bridge the gap between these two existing ones. A design methodology that is able to delay design decisions and explore the design space as in set based design but with a lower computational cost as in point based design. The exploration of such a design methodology is presented here.

3.1 Problem Description

The Design Space Covering for Uncertainty (DSC-U) problem is a new way to analyze decisions in early stage design under uncertainty. This problem formulation explores the trade offs between making design space reductions and the effect on the final design. The solution of this problem is the Pareto front in the trade space between the regret and space remaining metrics explained in sections 3.3 and 3.4. When a design space is reduced, there is the possibility that the global optimal solution has been eliminated making the optimal solution of the subspace worse than the global optimal solution. The optimal solution of the original design space is $\tilde{x}$ and the optimal solution in a subspace is $\bar{x}$. The regret for a design space is defined as $\bar{x} - \tilde{x}$; this will always be non-negative and will be zero if the global optimal solution remains in the subspace. The solution to the DSC-U problem an be an aid to designers by providing insight to how much regret will occur due to design space reductions.

Figure 3.1 shows a representation of the trade space and the Pareto front within it. At the extremes of the trade space are point based design and set based design. In point based design a single design is selected and is improved through the design

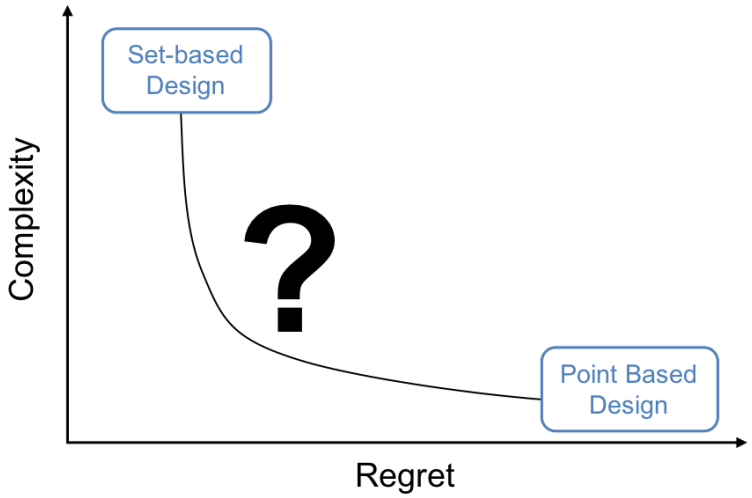

Figure 3.1: This problem is trying to find the Pareto front in the trade space between regret and the amount of the design space remaining

spiral; this method for design results in a small design space that must be further explored, but also tends to result in sub-optimal design due to only a small portion of the design space being explored. In set based design the entire design space is evaluated and progressively reduced until a single design is selected; this method for design results in a large computationally expensive design space that must be evaluated, but theoretically will have minimal regret in the performance of the final solution. This methodology aims to bridge the gap between point based and set based design by finding the Pareto front in this regret and space remaining trade space.

An existing method that informs the designer with the results of a trade study between performance and uncertainty is presented in *Liu* (2016). The method uses an optimizer that returns a robust solution that is the best worst-case performance over the range of uncertainty (*Liu et al.*, 2014). The best worst-case is found using a double-loop process where the inner loop evaluates a design for the worst case over the range of uncertainty and the outer loop evaluates these worst case scenarios to find the best option; this is a maximization problem inside of a minimization problem (*Liu*, 2016). In order to efficiently solve the framework, a new variable fidelity optimization

framework was developed; to improve the accuracy of the surrogate model multiple local Kriging surrogate models were built and updated online in the optimizer (*Liu et al.*, 2014).

One limitation of this work is that only the worst case performance for a design over the range of uncertainty is evaluated; this leads to a loss of information on the performance through the whole range of uncertainty. The DSC-U problem is set up to evaluate the performance of a design over the entire uncertainty range. In *Liu et al.* (2014) it was shown that finding a robust solution resulted in 15% regret compared to the optimal deterministic solution, but the deterministic solution performed extremely poorly over the range of uncertainty due to infeasibility. This work examines if the level of regret is reduced by selecting more than one solution to cover the range of uncertainty. By selecting more than one design point, a subspace of the original design space is defined.

In early stage design decisions are made while there is still uncertainty in the problem. Many of these decisions reduce the range for design variables, and possibly fix the value of design variables, ultimately performing a design space reduction. The decisions that reduce the design space can lead to the optimal solution being eliminated from the design space for specific realizations of the uncertain parameters. This method aims to avoid eliminating optimal solutions by delaying design decisions as is done in set based design.

3.2 Division of design space

An important aspect of the DSC-U problem is the division of the design space into two spaces. The original design space contains the design variables and uncertain parameters that are required to evaluate a design's fitness. The design variables are under the designers control while the uncertain parameters are not, requiring these parameters to be handled differently in design space exploration. This leads to the

decomposition of the design space into separate spaces, the X-space which contains the design variables, and the A-space which contains the uncertain parameters. For a simple naval design problem, the design variables could be geometry parameters such as the length, beam, and draft, and the uncertain parameters could be design requirements such as operating speed, range, and required weapons systems. With the division of the design space, the problem can now be described as trying to find reductions to the X-space while still covering the A-space with feasible, and hopefully near optimal, solutions. While these spaces are separate they are still related in that to fully evaluate a design's fitness a point from both the X and A spaces is needed. This is illustrated in figure 3.2 where the X-space is explored for every point in the A-space. After this exploration it is known how each design in the X-space will perform for any possible realization of the uncertainty (point in the A-space) and possible design space reductions of the X-space can be evaluated accordingly.

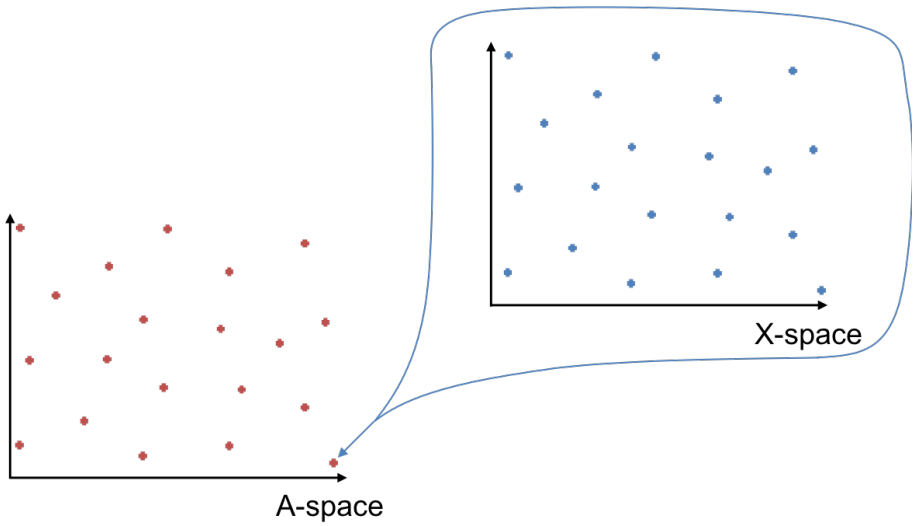

Figure 3.2: The separate X-space and A-space are still related to each other. When evaluating the X-space the design fitness is dependent on the point in the A-space.

3.3 Regret

The regret in the final design is a way to quantify how much performance has been left on the table for the design. For this application the regret is limited to the loss in performance due to making design decisions under uncertainty, specifically design space reduction decisions. For a given possible realization of the uncertain parameters there is a global optimum in the original design space. Unfortunately, often times design decisions are made in early stage design which eliminate this optimal solution when reducing the design space. The regret for a specific realization of the uncertainty is the difference in design fitness between the global optimal in the original design space and the new optimal in the reduced design space for that point in the uncertainty space.

Each of the two methods in this work use a different metric for the regret; the algorithm presented in Chapter IV uses the average regret while the algorithm in Chapter V uses a maximum threshold for regret. The regret of a design is always defined as $f(x,a) - f(\tilde{x},a)$ where $\tilde{x}$ is the optimal solution in the original design space for the realization of the uncertain parameters a. Figure 3.3 gives a graphical example of what the regret metric is measuring for a simple one variable problem.

For the nested algorithm in Chapter IV the regret metric is the average regret calculated by

$$F_R(\bar{x}) = \sum_{j=1}^{n} \frac{f(\bar{x}_j, a_j) - f(\tilde{x}_j, a_j)}{n} \tag{3.1}$$

where $f(\bar{x}_j)$ is the design fitness for the optimal solution $\bar{x}_j$ in the reduced design space for the sampled uncertainty point j, $f(\tilde{x}_j)$ is the design fitness for the optimal solution $\tilde{x}_j$ of the original design space for the sample uncertainty point j, and n is the number of sampled points from the uncertainty space.

For the level set covering algorithm in Chapter V, the regret metric is set as a threshold for finding the level sets that contain the designs in the X-space. This

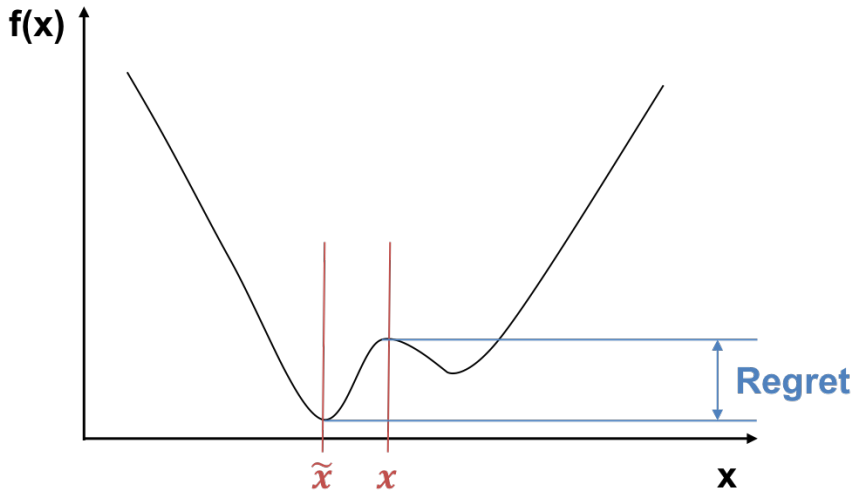

Figure 3.3: The value for the regret for the point x is the difference in the design fitness $f(x)$ between the optimal solution $\tilde{x}$ and the point x.

means that all designs in the X-space which have a regret below the threshold are considered equally feasible options for that point in the A-space.

3.4 Space remaining

The space remaining metric is used to quantify how much the design space has been reduced. The size of a design space that must be explored is important because it is directly related to the computational time needed. This is increasingly important as designers wish to use increasingly higher fidelity models in early stage design. In general, the space remaining is a measure of the reduced design space size normalized by the original design space size. This gives a measure of the percentage of the design space that is remaining to be further explored. As part of the nested algorithm presented in Chapter IV the space remaining is calculated as

$$F_S(\bar{x}) = \prod_{i=1}^{m} s^i \tag{3.2}$$

$$s^i = \sum_{h=1}^{k_i} \begin{cases} 1 & \text{if } X^{i,h} \text{ is in } \bar{x}^i \\ 0 & \text{otherwise} \end{cases}$$

where $\bar{x}$ is the optimal solution within the selected reduced design space found by the inner optimizer, m is the number of points sampled from the uncertain space, and k^i is the number of possible values for design variable i, and $\bar{x}^i$ is the set of optimal solutions for variable i for all sampled points j.

In the level set covering algorithm presented in Chapter V the space remaining is simply the percent of the design points used compared to the original number of sampled points in the X-space.

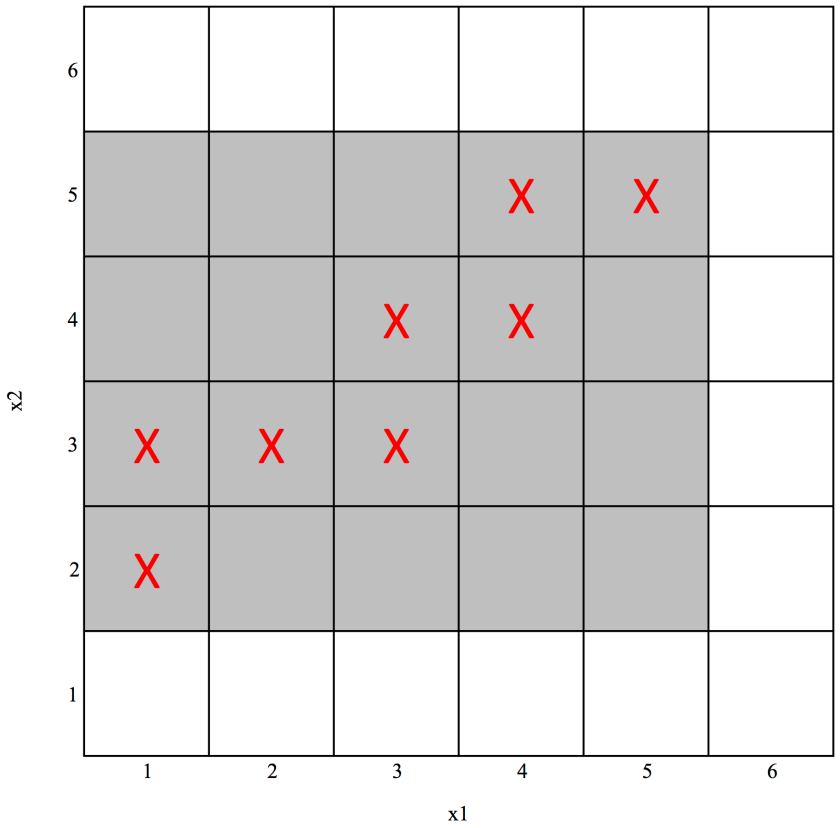

Figure 3.4: A graphical description of the space remaining metric for the nested algorithm in Chapter IV where the red X's are the optimal solutions and the gray box is the design space remaining. The area of the gray box divided by the area of the original design space is the value for the space remaining metric.

CHAPTER IV

Nested Design Space Exploration

Presented here is a method to solve the design space reduction for uncertainty (DSR-U) problem. This method aims to find a reduced design space, by narrowing the range of possible values for design variables, while maintaining a given level of optimality. In this case, the level of optimality is measured by the regret metric, which is a quantification of the loss in optimality due to a design space reduction. The magnitude of the reduction to the design space is measured by the space remaining metric, which measures the size of a remaining design space. The output of the algorithm is the trade space between the space remaining and regret for a design problem.

For this method, problems are defined in such a way that there are controllable design variables and uncontrollable uncertainty parameters. For example, in early stage design of a ship, things such as length, beam, and draft are design variables and things such as loosely defined operating speed and required weapons systems are uncertain parameters. Compared to other methods, this decomposition of the design space allows for the variables and parameters to be handled differently and more appropriately. The controllable design variables, $x_i \in X$ are the design variables for which the designer is selecting values and these are handled much like the design variables in traditional optimization methods. The uncontrollable uncertain

parameters, $a_i \in A$ are the uncertain parameters which are modeled with interval uncertainty as interval uncertainty is a valid method to handle both variability and ignorance (*Ferson and Ginzburg*, 1996). With interval uncertainty, only the maximum and minimum bounds are needed, and there is no information regarding the probability distribution within the range. The method structure can be summarized as:

1. **Problem definition:** Set up the problem by defining the design variables and uncertain parameters and their possible values as discrete parameters. Define the function that is used to measure the design fitness.

2. **Preprocessing:** Solve for the optimal solution in the original design space for each possible realization of the uncertain parameters. This is used to caluclate the regret metric.

3. **Set up the outer optimizer:** Each of the variables of the outer optimizer correspond to a design variable, and the variable value of the outer optimizer describes a range of possible values for the inner optimizer. There is a list of predetermined ranges for each design variable which the outer optimizer selects from and solves for the resulting regret and space remaining. The outer optimizer will have as many variables as there are design variables in the inner optimizer.

4. **Initialize population:** Create the first population for the outer optimizer.

5. **Solve inner optimizer:** For each individual in the population of the outer optimizer, solve for the optimal solution of each possible realization of the uncertain parameters for each individual in the population of the outer optimizer. This is done using a single objective solver. Each individual of the outer optimizer defines a subspace of the original design space to search.

6. **Calculate outer optimizer objective functions for each individual:** Use the list of optimal solutions for each possible realization of uncertain parameters in the subspace to calculate the space remaining metric F_S for each individual in the population. Also, for each individual calculate the regret metric, F_R which is the average regret for all the possible realizations of the uncertain parameters.

7. **Crossover, selection, and mutation:** Perform the crossover, selection, and mutation operations on the population to generate next population in the outer optimizer.

8. **Evaluate next generation:** Return to step 4 to evaluate the next population until stopping criteria is met.

4.1 Nested Algorithm

The algorithm is structured as a multi-objective optimization problem that contains a single objective optimization problem within its objective function calculation. This outer optimization problem is evaluating the regret and space remaining of the reduced design spaces. In this work regret is a metric that quantifies how sub-optimal a design is using the difference in the design fitness value. The space remaining metric quantifies the size of a reduced design space compared to the original design space. The inner optimization problem, that is within the objective function calculation of the outer optimization problem, is evaluating the design fitness for designs within a given reduced design space. Both optimization problems can be solve by any method; exhaustive sampling and genetic algorithms have been used in this work. The problem as a traditional optimization problem is

$$\begin{aligned} \text{minimize} \quad & F_R, F_S \\ \text{w.r.t} \quad & x^i_{range} \; \forall \; x^i \in X \end{aligned} \tag{4.1}$$

where F_R and F_S are the regret and space remaining objectives respectively, X is the original design space, X^i is the range for variable i in the original design space, and x^i_{range} is the reduced range for variable i. The space remaining objective is calculated as

$$F_S(\bar{x}) = \prod_{i=1}^{m} s^i \tag{4.2}$$

$$s^i = \sum_{h=1}^{k_i} \begin{cases} 1 & \text{if } x^{i,h} \text{ is in } \bar{x}^i \\ 0 & \text{otherwise} \end{cases}$$

where $\bar{x}$ is the optimal solution within the selected reduced design space found by the inner optimizer, m is the number of design variables, and k^i is the number of possible values for design variable i, and $\bar{x}^i$ is the set of optimal solutions for variable i for all sampled points j. The space remaining of this sample is the product of the number of discrete values of each design variable that are used in at least one solution; here the space remaining is the area of the dark gray rectangle. The space remaining function can easily be explained graphically for a 2-dimensional problem. In figure 4.1 a simple discrete 2-dimensional design space is shown with the optimal solution for sampled uncertainty points shown by red X's. The number of discrete values used in at least one optimal solution are counted off in red on the axis.

The regret objective is calculated as

$$F_R(\bar{x}) = \sum_{j=1}^{n} \frac{f(\bar{x}_j) - f(\tilde{x}_j)}{n} \tag{4.3}$$

where $f(\bar{x}_j)$ is the design fitness for the optimal solution $\bar{x}_j$ in the reduced design space for the sampled uncertainty point j, $f(\tilde{x}_j)$ is the design fitness for the optimal solution $\tilde{x}_j$ of the original design space for the sample uncertainty point j, and n is the number of sampled points from the uncertainty space.

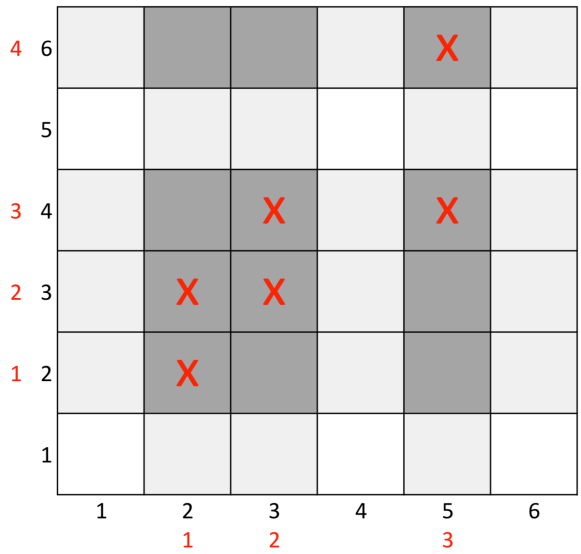

Figure 4.1: Simple graphical example of space reduced calculation with a value of 0.33 and a reduced design space size of $3 \times 4 = 12$ and an original design space size of $6 \times 6 = 36$.

To calculate both the space remaining and regret of the design space, the optimal solutions for each sampled uncertainty point must be found for both the original design space and the reduced design space. The optimal solution of the original design space is static and therefore calculated as a preprocessing step, and the optimal solution of the reduced design space is calculated by the inner optimizer. The outer optimizer defines a subspace of the original design space, and the inner optimizer is used to solve for the optimal solution within that space.

4.2 Box Girder

4.2.1 Problem Description

The design of a simple t-stiffened box girder resembling a ship hull girder was used to validate the nested approach. This simple design problem contained six design variables, one uncertain parameter, one constraint, and two objectives which were equally weighted and treated as a single objective function. The design variables were the plate thickness, stiffener spacing, and stiffener size for the horizontal and

vertical plates. The uncertain parameter was the required bending moment for the box girder and was used in the constraint parameter calculation. The problem of the inner optimizer in standard form is shown below.

$$\begin{aligned}
\text{minimize: :} \quad & C_p^{norm} + C_w^{norm} \\
\text{with respect to:} \quad & t_h, t_v, s_h, s_v, b_h, b_v \\
\text{such that:} \quad & M_z \geq M_{req} \\
& t_h^{min} \leq t_h \leq t_h^{max} \\
& t_v^{min} \leq t_v \leq t_v^{max} \\
& s_h^{min} \leq s_h \leq s_h^{max} \\
& s_v^{min} \leq s_v \leq s_v^{max} \\
& b_h^{min} \leq b_h \leq b_h^{max} \\
& b_v^{min} \leq b_v \leq b_v^{max}
\end{aligned} \tag{4.4}$$

The $3m \times 3m$ box girder shown in figure 4.2 is made of 6061 aluminum and is symmetrical about the x- and y-axes. The length, beam, and depth of the box girder were fixed to 5m, 3m, and 3m, respectively. The six design variables were the plate thickness, stiffener spacing, and stiffener size for the horizontal and vertical plates and the uncertain parameter was the required bending moment; these were all treated as discrete variables. The possible values for all problem parameters are shown in table 4.1.

Table 4.1: Variables for box girder example

Variable		Possible Values
Required Bending Moment (MNm)	a	1, 5, 10, 15, 20, 30, 40, 50
Plate thickness (mm)	t_h, t_v	3, 4, 5, 6, 7, 8, 9, 10
Number of stiffeners on plates	s_h, s_v	8, 9, 10, 11, 12, 13, 14, 15
Stiffener size on plates	b_h, h_v	3×8.625, 4×11.5, 5×17.5, 6×25, 7.5×25, 9×35, 10×37.5, 12×45

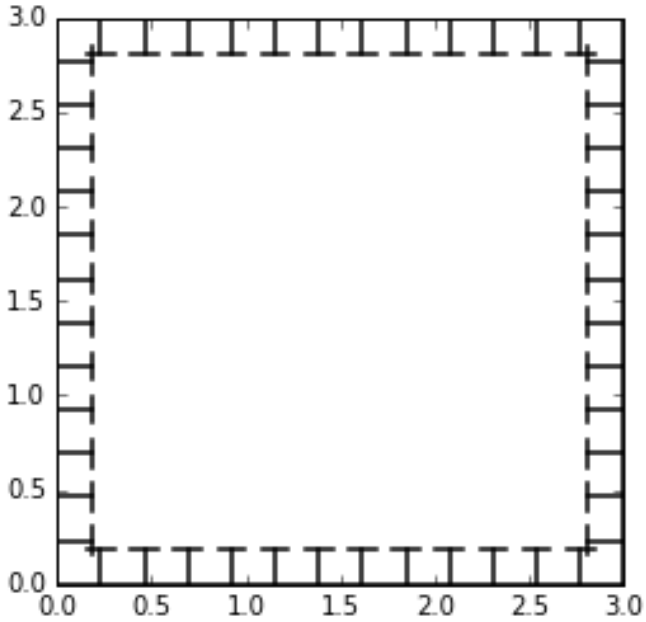

Figure 4.2: Cross section of the box girder with t-stiffened panels

The weight and production cost are used to measure the design fitness of a box girder design. The uniform density of the material and normalization of the objective allows the weight objective to be simplified to a normalized measure of the cross-sectional area. The cross-sectional area is simply calculated as the sum of the $width \times height$ of each plate and stiffener. The cost of the box girder is calculated with the method originally presented in *Rahman and Caldwell* (2012), adapted in *Liu* (2016) and *Temple* (2015), and evaluated in *Rigterink et al.* (2013) shown in equation 4.5. This cost calculation method includes five cost components which account for the material and labor costs for the specific design. The five cost components, $C_{i,j}$ are in table 4.2 and the total cost is calculated as

Table 4.2: Description of cost terms for box girder calculation

Cost term	Description
$C_{i,1}$	material cost for hull plates
$C_{i,2}$	material cost for longitudinal stiffeners
$C_{i,3}$	material cost for longitudinal framers
$C_{i,4}$	welding cost for longitudinal stiffeners
$C_{i,5}$	electricity and electrodes cost

Table 4.3: Constant values used for the cost calculation

Variable	Value	Description
P_a	860	material price (U.S\$/ton)
P_s	27	labor rate (US\$/hr)
r	7.85	specific weight of the material (ton/m^3)
l	5	panel length (m)
B	3	panel breadth (m)
C_{lm}	1.05	material cost coefficient for longitudinal stiffeners
C_{ls}	1.2	labor hour required per meter welding of stiffeners to plate
C_{ee}	0.9	labor hour equivalent required per meter of stiffeners implementing electricity
C_{fb}	1.5	labor hour required per meter of stiffeners for fabrication
W_p	-	Weight of plate
W_l	-	weight of longitudinal stiffeners
n_l	-	number of longitudinal stiffeners

$$C_p = \sum_{i=1}^{n} \sum_{j=1}^{6} C_{i,j} \tag{4.5}$$

The design of the box girder is constrained by a required bending moment, which is the uncertain parameter in this problem. The maximum bending moment of the design is calculated by equation 4.6 where σ_{min} is the ultimate compression strength of the weakest panel. If a design is infeasible, in that the required bending moment is not met, a constraint violation is added to the objective function. This constraint violation is given by the absolute difference between the box girder maximum bending moment and the required bending moment for ultimate compression strength calculation which is weighted with a multiplier (*Paik and Duran*, 2004).

$$M_z = \frac{\sigma_{min} I_{NA}}{y} \tag{4.6}$$

The ordered nature of the uncertain parameter in this example was exploited as another way to reduce the designs in the design space. The uncertain parameter is ordered such that A_1 was the smallest and A_8 was the largest required bending moment; this makes it that if a design is feasible for A_i, that same design will also be

feasible for A_j if $j \leq i$. Given this, an additional grouping of solutions was employed as a variable of the outer optimizer For example, with grouping label 1, the optimal solution would be found for A_4 and A_8, and the optimal solution for A_4 would be used as the solution for A_1, A_2, A_3 and A_4 and the optimal solution for A_8 would be used as the solution for A_5, A_6, A_7 and A_8. There were four possible groupings which are shown in table 4.4.

Table 4.4: Grouping of uncertain parameters to exploit the ordered nature of the uncertain parameter.

Grouping Label	Grouping of points in A	Solution used for all points
0	$[A_1 - A_8]$	A_8
1	$[A_1 - A_4]$	A_4
	$[A_5 - A_8]$	A_8
2	$[A_1 - A_2]$	A_2
	$[A_3 - A_4]$	A_4
	$[A_5 - A_6]$	A_6
	$[A_7 - A_8]$	A_8
3	$[A_1]$	A_8
	$[A_2]$	A_8
	$[A_3]$	A_8
	$[A_4]$	A_8
	$[A_5]$	A_8
	$[A_6]$	A_8
	$[A_7]$	A_8
	$[A_8]$	A_8

4.2.2 Results

The box girder design problem was solved using a MOGA based on the NSGA-II sorting algorithm (*Deb et al.*, 2002) for the outer optimizer and a brute force table search for the inner optimizer. A brute force table search was utilized for the inner optimizer as the limited range of discrete variables kept the table to a tractable size and computational effort to a minimum. The run parameters for the outer MOGA are shown in Table 4.5.

Table 4.5: Run parameters for the MOGA used to solve the regret and space remaining optimization problem.

$\|P\|$	N_G	p_c	p_m
60	75	0.8	0.001

The Pareto front from the final generation of the outer optimizer is shown in Figure 4.3. The regret axis has been normalized and depicts the percentage of regret compared to the optimal objective values. It should be noted that since the space remaining objective value will always be an integer, the solution will have a seemingly sparse Pareto front. The rightmost point of the Pareto front, with a space remaining of one, is the robust solution where one design which is feasible for all possible realizations of the uncertain parameter is selected. The leftmost point of the Pareto front, with a space remaining of 100 and regret of zero, is the solution where eight distinct solutions are kept, one for each possible realization of the uncertain parameter. The other points along the Pareto front correspond to solutions that have reduced the design space which has excluded some of the optimal solutions resulting in some regret. The shape of the Pareto front informs designers where reductions of the design space will create a significant or negligible increase in regret. The Pareto front is steepest from the points with a space remaining of 100-60 and 40-24; at these points, the design space is reduced by 40% and the average regret increases by only 0.06538%, a negligible amount for the significant reduction to the design space. The Pareto front is shallower as the space remaining is smaller and closer to 1; this informs the designers that reducing the design space past a space of 4 will result in significant regret from even a small space remaining reduction.

Information that would be of interest of the designer can be gathered from different points from the Pareto front. Three points from the Pareto front will be discussed here; the points selected are marked by orange circles in figure 4.4. Details of the subspace for these three points are shown in table 4.6. An interesting aspect of these

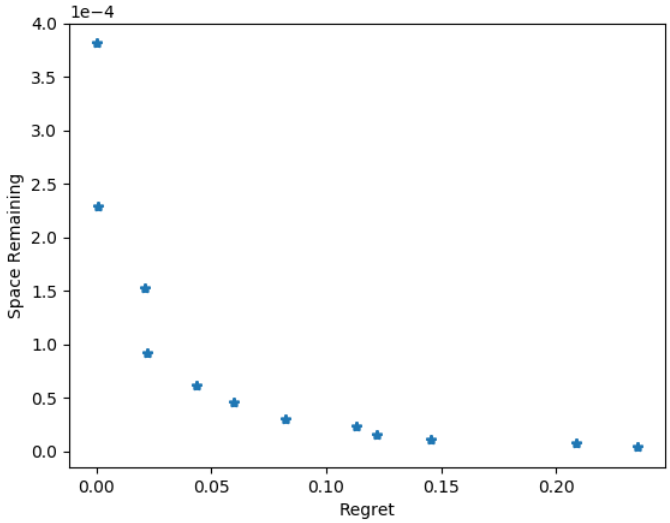

Figure 4.3: Pareto front between the space remaining and regret objectives for the box girder example.

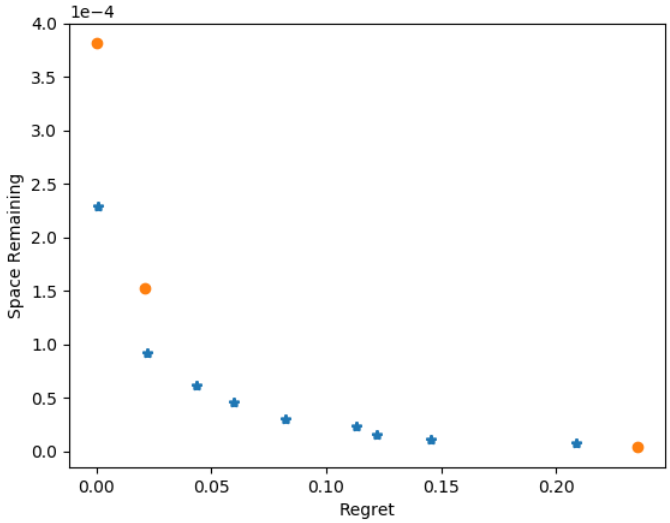

Figure 4.4: The solution for the box girder example showing three points on the Pareto front that will be further discussed.

design spaces is that all three, including the space that results in zero regret, have fixed the value for b_v and s_v to the smallest option; these variables correspond to the number of stiffeners for the vertical plate and stiffener size. This is not surprising given that these do not increase the ultimate bending strength significantly. The variables associated with the plate thicknesses, t_h and t_v, also experience a significant range reduction, even in the case of 0% regret. This leaves the stiffener size and spacing on the horizontal plates as the most important to keep options open for; this is not surprising as these stiffeners have the greatest affect on the ultimate bending strength of the box girder. Given this deeper analysis of the results along the Pareto front, it can be said that the values for b_v and s_v are not very important and can be fixed to a specific value without causing regret, even before the uncertainty is reduced. Having flexibility for the values of b_v and s_v is critical to minimizing regret, and while they can be reduce from the initial eight possible design values, delaying further design decisions for these variables is needed to avoid significant regret.

Table 4.6: Details for three points in the Pareto front shown in figure 4.4

		Dimension of subspace					
Space Remaining	Regret	t_h	b_h	x_h	t_v	b_v	s_v
0.0000938147	24%	1	1	1	1	1	1
0.0001525879	2%	1	4	5	2	1	1
0.0003814697	0%	2	5	5	2	1	1

4.3 Cantilever Pipe

4.3.1 Problem Description

The design space for a cantilever tube from *Du* (2007) was examined using this method. This design problem had an objective of minimizing weight with a constraint of keeping a sufficient reliability index while the applied forces are not fully defined. This problem contains uncertainty due to manufacturing tolerances and lack

of knowledge. The uncertainty due to manufacturing tolerances is handled by a First Order Reliability Method (FORM) within the design fitness calculation of the inner optimization problem and independently of the ignorance uncertainty handled by the nested algorithm. This program contains two controllable design variables X, which are the average thickness (μ_t) and diameter (μ_d) of the tube, and two uncertain parameters A, which are the angles that the forces are acting through (θ_1, θ_2). The design problem is formally defined by

$$\begin{aligned} \min: \quad & \text{Area} = \frac{\pi}{4}(d^2 - (d-2t)^2) \\ \text{with respect to:} \quad & \mu_t, \mu_d \\ \text{such that:} \quad & \beta \geq 3 \\ & t_{min} \leq \mu_t \leq t_{max} \\ & d_{min} \leq \mu_d \leq d_{max} \end{aligned} \tag{4.7}$$

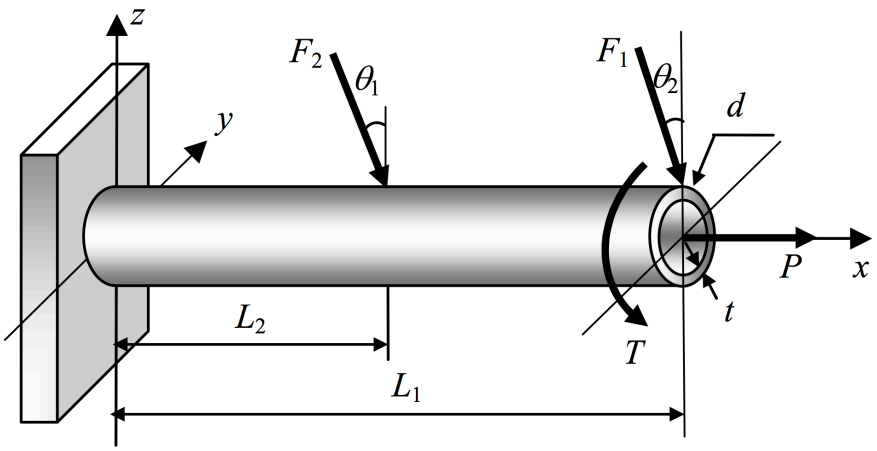

Figure 4.5: Geometry of the cantilever tube from *Du* (2007)

The geometry of the cantilever tube is shown in figure 4.5. All of the parameters that define the geometry are random variables and are included in the reliability analysis of the problem constraint; the random variables and their parameters are shown in table 4.7. It should be noted that these random variables are known parameters, but have a variance due to manufacturing tolerances. While these variances can be

a source of uncertainty, they are handled by reliability analysis within the constraint calculation as opposed to the uncertain parameters due to design uncertainty which are handled as the parameters in the A-space. The design variables, μ_t and μ_d, and the uncertain parameters, θ_1 and θ_2, are discrete variables; the possible values are presented in table 4.8.

Table 4.7: Random variables for the cantilever tube reliability calculation

Variables	Parameter 1*	Parameter 2*	Distribution
t	μ_t (mm)	0.1 (mm)	Normal
d	μ_d (mm)	0.5 (mm)	Normal
L_1	119.75 (mm)	120.25 (mm)	Uniform
L_2	59.75 (mm)	60.25 (mm)	Uniform
F_1	3.0 (kN)	0.3 (kN)	Normal
F_2	3.0 (kN)	0.3 (kN)	Normal
P	12.0 (kN)	1.2 (kN)	Gumbel
T	90.0 (Nm)	$9,0$ (Nm)	Normal
S_y	220.0 (Nm)	22.0 (Nm)	Normal

*For uniform distributions Parameters 1 and 2 are the lower and upper bounds, respectively. For all other distributions Parameters 1 and 2 are the mean and standard deviation, respectively.

Established methods in classical structural mechanics, shown in equation 4.8, were used to calculate the stress in the tube (*Du*, 2007). All reliability simulations

to calculate β were calculated using the PyRe (PythonReliability) module (*Hackl*, 2013).

$$\begin{aligned}
\sigma_{max} &= \sqrt{\sigma_x^2 + 3\tau_{xy}^2} \\
\sigma_x &= \frac{P + F_1 \sin\theta_1 + F_2 \sin\theta_2}{A} + \frac{Mc}{I} \\
M &= F_1 L_1 \cos\theta_1 + F_2 L_2 \cos\theta_2 \\
A &= \frac{\pi}{4}[d^2 - (d-2t)^2] \\
c &= d/2 \\
I &= \frac{\pi}{64}[d^4 - (d-2t)^4] \\
\tau_{xy} &= \frac{Td}{4I}
\end{aligned} \tag{4.8}$$

The design variables, μ_t and μ_d, and the uncertain parameters, θ_1 and θ_2, are discrete variables; the possible values are in table 4.8. The possible ranges given to the inner optimizer are shown in table 4.9.

Table 4.8: Variables for cantilever tube example

Variable		Possible Values
Angle of F_1 (deg)	θ_1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Angle of F_2 (deg)	θ_2	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
Average thickness of tube (mm)	μ_t	3, 4, 5, 6
Average diameter of tube (mm)	μ_d	38, 39, 40, 41, 42, 43, 44

4.3.2 Results

The small size of this problem, of only 64 individuals, suits it to be solved by enumerating all individuals instead of using an evolutionary algorithm to find the Pareto front. Many of these individuals did not produce feasible designs and therefore are severely penalized. Similarly to the previous example, the resulting Pareto front has limited points due to the integer space remaining objective value and the small

Table 4.9: Possible ranges given to inner optimizer for cantilever tube example

Range Label	Range for μ_t (mm)	Range for μ_d (mm)
0	3	38
1	4	39
2	5	40
3	6	41
4	3-4	42
5	5-6	43
6	4-5	44
7	3-6	38-44

design space. The Pareto front is shown in blue in figure 4.6 with the points in and near the front.

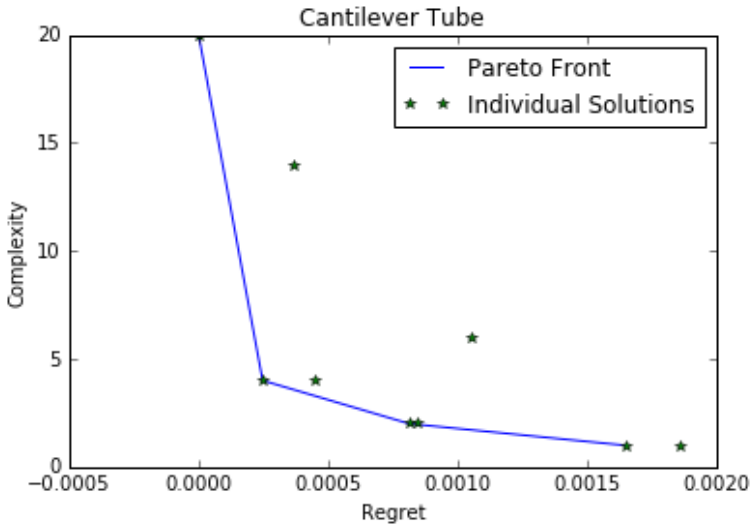

Figure 4.6: Pareto front for cantilever tube example

The steep Pareto with small levels of regret front means that the designer can significantly reduce the design space without a significant increase in regret. Specifically, the designer can reduce the design space by 80% with minimal regret. The endpoint of the Pareto front with a space remaining of twenty and regret of zero is a solution that includes the optimal design for all 121 possible combinations of θ_1 and θ_2; this solution has a design space that has been reduced by 28% but does not elim-

inate any optimal solutions for the possible realizations of the uncertain terms. The other endpoint of the Pareto front with a space remaining of one is the robust solution with a single solution that is feasible for all possible realizations of the uncertain parameter.

4.4 Summary

A novel method has been developed for design optimization in early-stage design with uncertainty. The method uses nested optimizers to analyze design space reductions for the resulting regret and space remaining, which are metrics for suboptimality and the size of the design space. Since the regret and space remaining of a reduced design space are competing objectives, the method gives the Pareto front in the trade space of the two objectives.

Two examples have been presented for proof of concept of the method. These two examples were the design of a box girder, with a larger design space and a single uncertain parameter, and the design of a cantilever tube, with a smaller design space and multiple uncertain parameters. Both of these examples showed that this method is a valid approach to evaluate design space reduction options while minimizing space remaining and regret for small problems. Given this information, designers will be able to make more informed design space reduction decisions in early-stage design.

This method is able to solve small DSC-U problems, but has proven to scale poorly. The nested structure requiring several single objective genetic algorithm solutions for each individual in a multi objective genetic algorithm is extremely computationally expensive. The number of inner optimization problems needed to solve each individual of the outer optimizer scales with the size of the uncertain parameter space and the computational effort to solve each inner optimization problem typically increases with the number of design variables; not only do these aspects individually scale poorly to larger problems when combined the poor scaling is compounded. Unfortunately,

these factors that result in the algorithm scaling poorly cannot be remedied by the use of surrogate models or other computation efficiency improvement. It was found that for a moderately sized 10-bar truss problem with 10 design variables and 2 uncertain parameters the computation time would be in the order of months and remained so even when implementing a surrogate model to quickly compute the design fitness. This extremely long computation time for this moderately sized problem shows that this method is not feasible to solve any problems larger than that are presented previously in this chapter. The results of the small problems presented in sections 4.2 and 4.3 show that the solution of the DSC-U problem is of interest but other more efficient methods will be needed.

CHAPTER V

Level Set Covering Algorithm

5.1 Introduction

Presented here is another algorithm to solve the design space covering for uncertainty problem (DSC-U), the Level Set Covering (LSC) algorithm. The first algorithm presented to solve the DSC-U problem in Chapter IV proved to scale poorly for large problems presenting the need for another algorithm to solve larger problems. In the LSC, a sampling method has been implemented for both the X and A spaces to limit the size of the problem regardless of the size of the spaces.

Similar to the nested algorithm presented in Chapter IV, the design space is separated into two spaces creating a design variable space and uncertain parameter space. The controllable design variables, $x_i \in X$, are under the control of the designer and are treated similarly to the design variables in a typical design optimization problem. The uncertain parameters, $a_i \in A$, are the uncontrollable parameters that model uncertainty in the problem. The controllable design variables make up the X-space while the uncertain parameters make up thee A-space. Both of these spaces are continuous for this algorithm, unlike the discrete space of the algorithm in Chapter IV. Again, the uncertainty of the parameters is treated as interval uncertainty where only the maximum and minimum bounds are given with no information regarding a probability distribution within the range. For each sampled point in the A-space the

algorithm finds a level set of designs that corresponds to a given level of regret; meaning that the algorithm defines a set containing the points in the design space who's design fitness is within a threshold regret percentage of the optimal for each sampled point in the A-space. In general, LSC solves the set covering problem with level sets to find the fewest number of designs that will cover all points in the uncertainty space for a given level of regret.

5.2 Framework

Just like the algorithm presented in Chapter IV, the design space of the problem is divided into two spaces: the X space of the design variables and the A space of the uncertain parameters. The resulting trade space from this algorithm gives information to the designer regarding the possible regret that will be in the final design due to a design space reduction. Only the X space is available to the designer to for size reduction as the A space is the uncertain parameters, and by the problem definition all possibilities must be covered. This decomposition of the design space from an $m+n$-dimensional space to an m-dimensional X-space and a n-dimensional A-space allows the algorithm to handle the design variables and uncertain parameters differently and more appropriately.

The algorithm can be divided into three major steps shown in figure 5.1. The preprocessing step is the most computationally expensive, as all of the design fitness function calculations are in that step. With all of the design fitness calculations complete, finding the level sets for each sampled point in the A space becomes relatively computationally inexpensive. With the level sets defined the set covering problem can be solved to calculate the percent of the design space remaining for a given regret value. For a discussion of level sets see section 2.5, and for a discussion on the set covering problem see section 2.4.

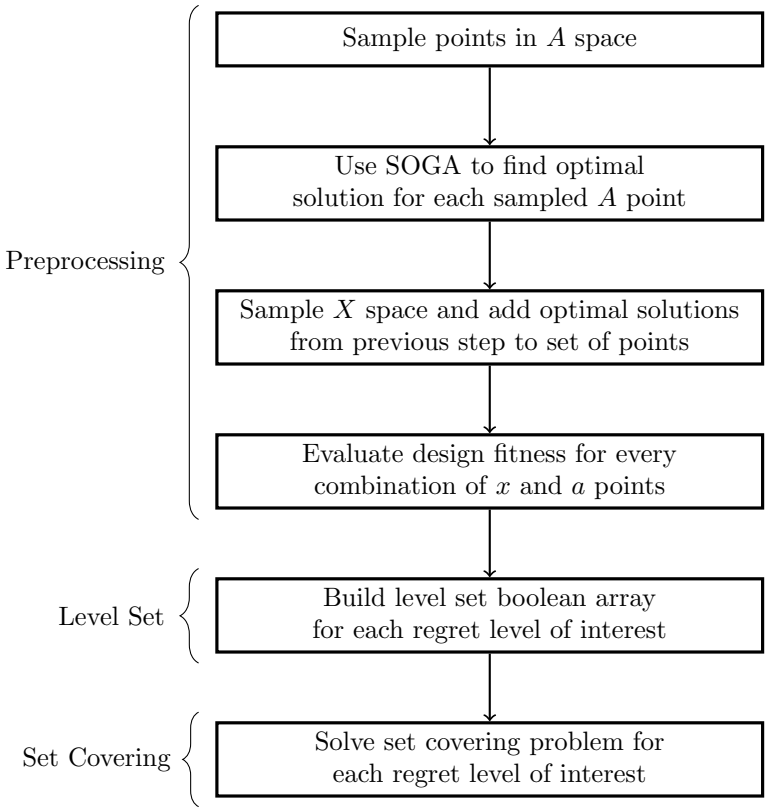

Figure 5.1: Framework of the level set covering algorithm to solve the design space covering for uncertainty problem.

5.2.1 Preprocessing

The first step of the algorithm contains the preprocessing calculations which calculate the objective values. These objective values are then used to find the level sets used in the set covering problem. Both the A and X spaces are defined as continuous spaces by the user, and each space is then sampled using Latin Hypercube Sampling (LHS) to create a set of points that can be used to evaluate the spaces *Mckay et al.* (1979). From this point on, the originally continuous A and X-spaces are described by these two sets of sampled points.

To calculate the regret the optimal design (in the X-space) must be found for each sampled point in the A-space; to calculate this, a single objective genetic algorithm (SOGA) is used to find the optimal design solution for each sampled point in A. At this step, there is a set of points in the A-space, for which the optimal solution is known for each point, along with a set of points in the X-space. To ensure that even extremely small levels of regret can be solved for, the optimal design solution for each sampled point in A (found previously using a SOGA) are appended to the list of sampled points in the X-space. For example, if there were 200 sampled points in the A-space and 1000 sampled points in the X-space, the new set of points used for the X-space for the rest of the algorithm would contain 1200 points.

The design fitness objective function is then calculated for each combination of sampled points in the X and A-spaces, including the optimal points appended to the set of points in the X-space. The calculation of the $n(m+n)$ design fitness values to create this matrix, along with the SOGA in the previous step, is what makes the preprocessing step so computationally expensive. From these calculations a large matrix is created to be a look-up table eliminating the need for any further design fitness calculations This large matrix of design fitness values, and the known optimal solutions, are used in the next step to solve for the level sets.

5.2.2 Calculating Level Sets

Level sets are used to find sets of design points (in the X-space) that are within an allowable regret amount, where regret is measured as the percent difference between the optimal solution and the selected design as shown in equation 5.1. In this equation $f(\tilde{x})$ is the design fitness of the optimal solution in the original design space, $\tilde{x}$, and $f(x)$ is the design fitness of the point, x. For each sampled point in A, a level set is found for each regret value of interest. To prepare for the set covering problem in the next step, the level sets are described by a boolean array the length of the set of sampled points in the X-space. This determination is made by iterating through the matrix of solutions from the preprocessing step and creating a boolean matrix for each regret value of interest where it is TRUE if the regret is within the allowable regret value and FALSE if not.

$$f_R(x) = \frac{100(f(x) - f(\tilde{x}))}{f(\tilde{x})} \tag{5.1}$$

5.2.3 Solving the Set Covering Problem

The final step of the LSC is to solve the set covering problem (SCP) for each regret value of interest. The SCP tries to find the minimum number of points in the X-space that will 'cover' all the points in the A-space, meaning that the solution set contains at least one point from the level set for every sampled point in the A-space. A simple example is shown in figure 5.2 where the level sets for 10% regret are shown for two points. The solution of the SCP is the point that is contained within both sets. In this work, the SCP is solved using three different methods: two greedy methods and the SetCoverPy module by *Zhu* (2018). A comparison of these methods will be presented in the test problem results. The size of the solution set from the SCP is then normalized by the original set of sampled points in the A-space to calculate the percent of the space remaining after the space reduction. The space

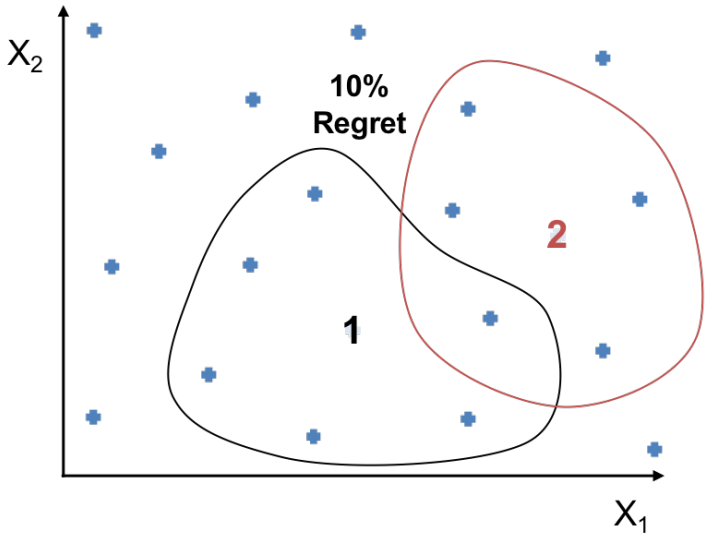

Figure 5.2: A simple example of the set-covering problem with two sets and a solution of one.

is normalized by the number of points in the A-space because when the allowable regret is 0% the remaining space will contain as many points as were sampled in the A-space. These points would be the optimal solutions for the sampled points. A discussion of additional solution methods for the SCP can be found in section 2.4.1.

5.2.4 Differences from Nested Algorithm

To combat the scaling problems of the nested algorithm, some fundamental changes were made in the development of the LSC algorithm. The structure of the nested algorithm resulted in a 'guess and check' structure to the algorithm which is not an efficient way to search the space and resulted in too many calculations of the design fitness function. The LSC algorithm no longer uses a 'guess and check' structure to evaluate certain subspaces, but builds subspaces from known acceptable designs. The algorithm uses level sets to define acceptable designs and solves the SCP to build a subspace that includes an acceptable design for all of the A-space.

This also changes the method forcing the search. In the nested algorithm, the algorithm is specifying a subspace making the space remaining metric the forcing

agent. In the LSC algorithm, the algorithm is searching for the smallest subspace that will cover the uncertainty space with a solution that is under a threshold of regret making the regret metric the forcing agent.

The different structure also required that the regret and space remaining metrics be calculated slightly differently. In the nested algorithm the regret is the average regret over the entire A-space and in the LSC algorithm the regret is the upper threshold of regret over the entire A-space. In the nested algorithm the space remaining metric creates a space that surrounds the included points while the LSC algorithm simply used the number of points included in the subspace compared to the original number of points for the space remaining metric. The space remaining metric no longer includes the space surrounding and between the included points as part of the subspace.

With these changes that were made in the development of the LSC, the computation cost was reduced and the algorithm will scale better. The LSC algorithm has been tested on the Rosenbrock function with five design variables and two uncertain parameters as well as on the design of a midship section with 36 design variables and four uncertain parameters. With the LSC the larger midship section problem is able to be solved in about a day with parallel processing and in less than a week with a single processor; a problem of this size would take several months to solve using the nested algorithm with parallel processing.

5.3 Rosenbrock Function

The Rosenbrock function was selected as the mathematical model to be used as an initial test of the LSC. The Rosenbrock function shown in equation 5.2 was first presented in *Rosenbrock* (1960), and has since become a common benchmark problem for optimization methods. The problem is a minimization problem with the optimal solution at $(1, 1)$. The plot in figure 5.3 shows the Rosenbrock function with its unique

parabolic valley. This problem was selected for this parabolic valley as it showcases the fact that the framework uses level sets and not a distance metric in the design variable space when applying a level of acceptable regret.

$$f(z) = 100(z_1^2 - z_2)^2 + (z_1 - 1)^2 \tag{5.2}$$

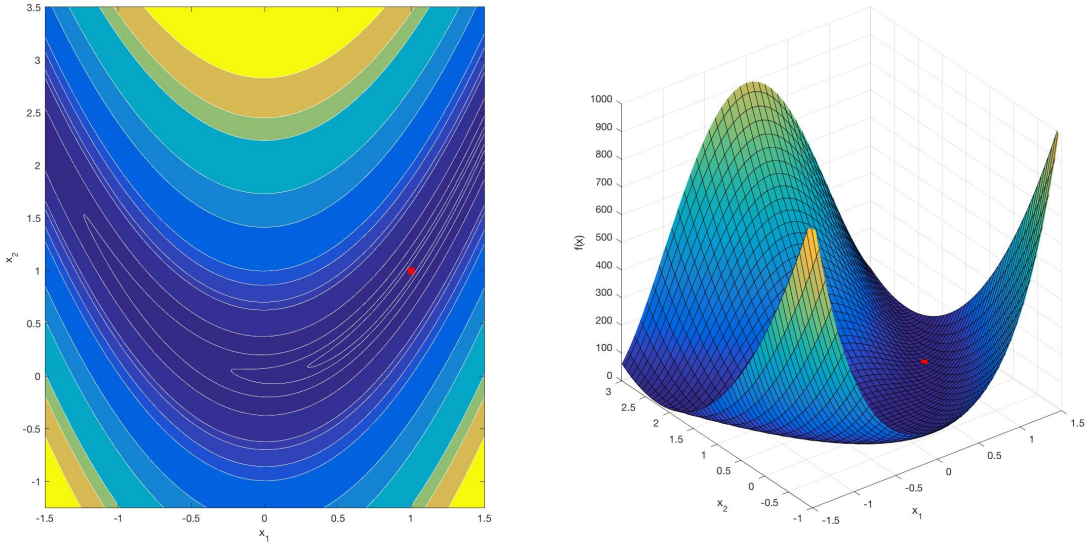

Figure 5.3: Contour and surface plots of the original Rosenbrock function, equation 5.2, with the optimal solution $(1, 1)$ marked.

The equation has been extended to be a higher dimension problem, this version is shown in equation 5.3 (*Kok and Sandrock*, 2009; *Shang and Qiu*, 2006). This extended n-dimensional Rosenbrock function is used in this work to test the framework.

$$f(z) = \sum_{i=1}^{n-1} 100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2 \tag{5.3}$$

5.3.1 Problem Description

A 7-dimensional design space is used for the application of the Rosenbrock problem to test the LSC framework. The specific form of the Rosenbrock equation used is shown in equation 5.4. There were two forms of the problem: the first being where

Table 5.1: Variable ranges for the Rosenbrock function and relationship of the z variable vector to the a and x vectors.

Form	Variable Vector z	a_{range}	x_{range}
1	$[a_1, a_2, x_1, x_2, x_3, x_4, x_5]$	$[0.25, 1.75]$	$[0, 2]$
2	$[x_1, a_1, x_2, a_2, x_3, x_4, x_5]$	$[0.25, 1.75]$	$[0, 2]$

z is a simple combination of the **a** and **x** vectors, and the second being where **z** is formed by mixing together the **a** and **x** vectors as shown in table 5.1. The parameters used for the SOGA to find the optimal solution in the original design space during the preprocessing step are shown in table 5.2.

$$f(z) = \sum_{i=1}^{6} 100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2 \tag{5.4}$$

Table 5.2: Parameters for the SOGA used in the preprocessing of the Rosenbrock problem

Parameter	Value
Variable Tolerance	1×10^9
Convergence Tolerance	1×10^{12}
Crossover Percent	0.9
Mutation Percent	0.01
Real Parameter	2
Mutation Parameter	1.0
Population Size	200
Constant Generations	25
Max Generations	1000
Elitism	1.0

5.3.2 Results

The resulting Pareto front for the first form of the problem is shown in figure 5.4. The initial trade off for a reduced design space is significant, there is only 0.5% regret with 20% of the original design space remaining. To reduce the design space to be much smaller than 5% significant regret will occur. One interesting range of the solution is between 10% and 20% regret where the size of the reduced design space

stays constant; it would be valuable information for designers to know that by even allowing twice as much regret the design space cannot be reduced.

The resulting Pareto front for the second form of the problem is shown in figure 5.5. The construction of the $\mathbf{z}$ vector from the $\mathbf{x}$ and $\mathbf{a}$ vectors results in a more tightly coupled problem. This means a larger design subspace is required for the same level of regret compared to the first form of the problem. The more tightly coupled problem results in the trade offs at either end of the Pareto front not being as severe; fairly significant design space reduction can still be seen through 50% regret; and at that point the design subspace is still 11% of the original design space.

The Pareto front for both forms of the problem are shown in figure 5.6. The second form of the equation requires a much larger subspace than the first form of the problem for the same level of regret. This is due to the design variables and uncertain parameters being more tightly coupled in the second form of the problem. Each term of the Rosenbrock problem use $\mathbf{z}_i$ and $\mathbf{z}_{i+1}$ for $i \in [1, 2, 3, 4, 5, 6]$, so, in the second form where $\mathbf{z}$ is made by mixing the $\mathbf{x}$ and $\mathbf{a}$ arrays, the variables are more tightly coupled. In the first form of the problem the uncertainty parameters are only in the first two terms, but are in the first four terms for in the second form of the problem.

This comparison of the results with how tightly coupled the design variables and uncertain parameters are leads to some interesting findings. The second form the of the equation, which is more tightly coupled, requires a larger design space for a regret threshold though the entire range of regret, and the Pareto front has a much more rounded shape. For a 60% reduction to the design space (40%-space remaining) the associated regret is approximately 50 times higher for the second form of the problem; resulting in 5% regret versus 0.1% regret for the first form of the problem.

Three points along the Pareto front were examined further for the second form of the problem. These points are the end points of the front and a middle point and

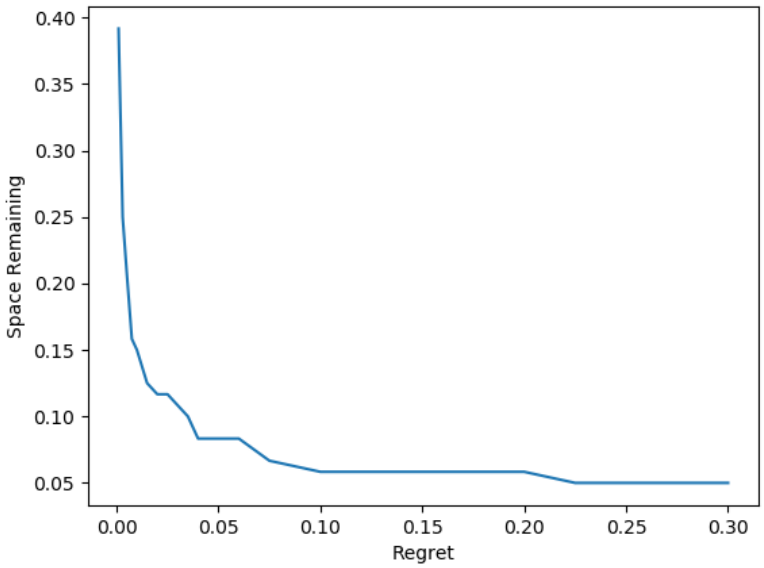

Figure 5.4: Pareto front in the trade space for the first form of the Rosenbrock problem.

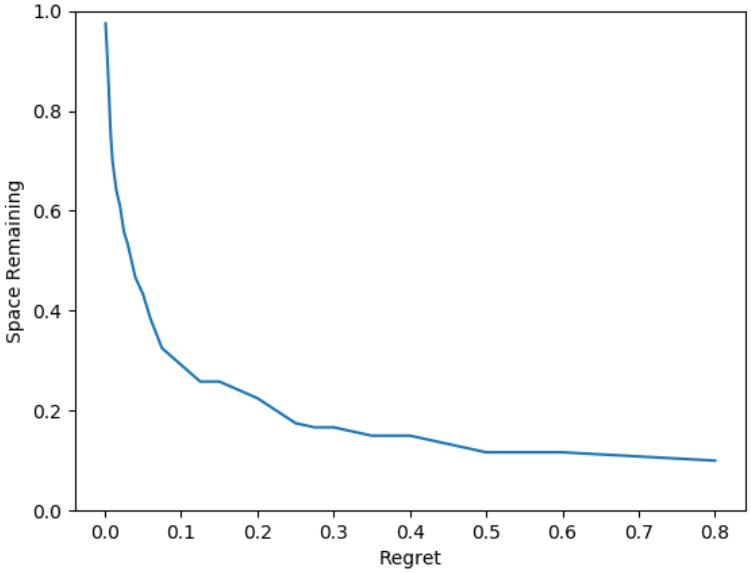

Figure 5.5: Pareto front in the trade space for the second form of the Rosenbrock problem.

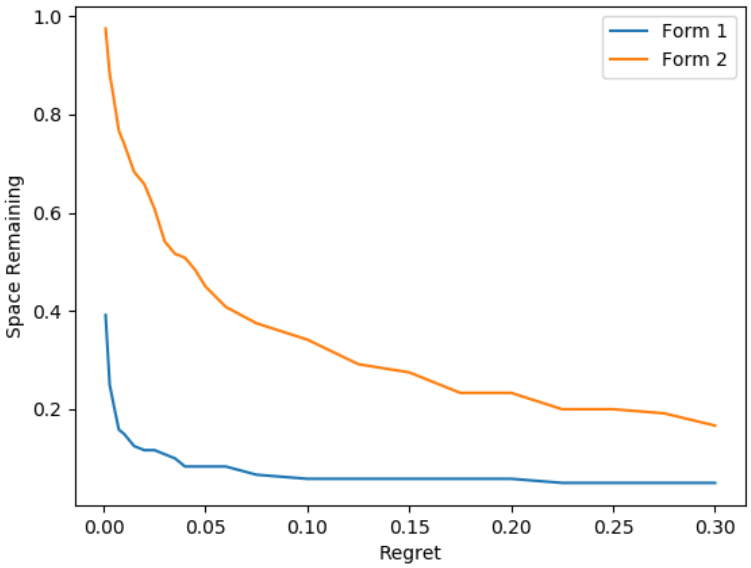

Figure 5.6: Comparison of both Rosenbrock problem solutions

are shown in subplot (d) of Figure 5.7. Subplots (a), (b), and (c) of Figure 5.7 show the ranges used for each variable in the subspaces corresponding to the points on the Pareto front. There are two major behaviors that can be observed for the variables as the design space is reduced. Variables x_1 and x_2 have a smaller range of values used as the design space is reduced. This is in contrast to variables x_4 and x_5 which maintain the same range as the design space is reduced, but uses fewer values within the range. With variable x_3 we see both behaviors; the range is reduced slightly, and the values being used are more spread out in the range.

These observations can provide valuable information to designers when they are making design space reduction decisions. Given this solution, designers can see for which variables it is most important to maintain a the variable range and for which variables it is possible to reduce the range without much regret.

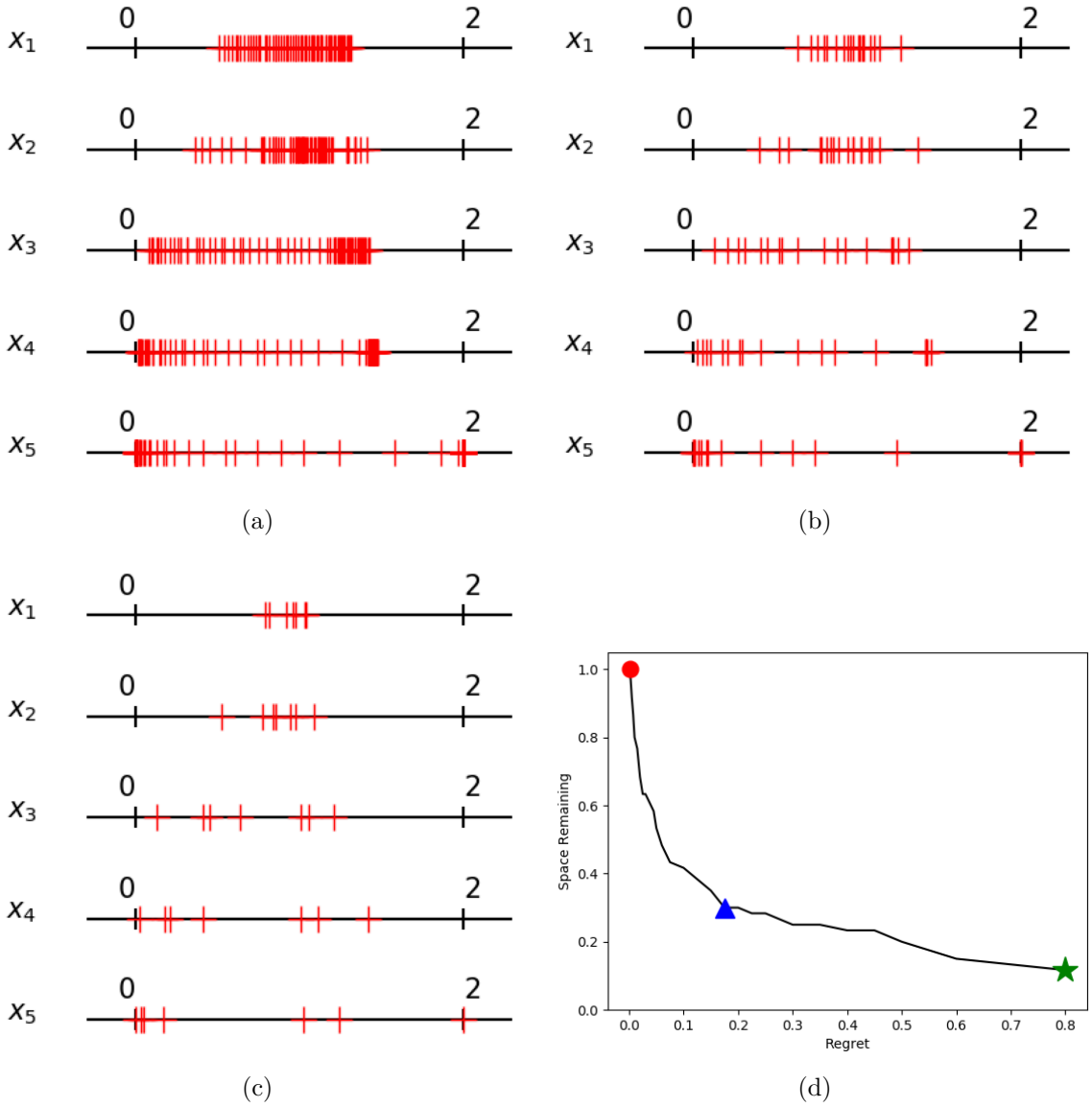

Figure 5.7: Variable values used in the reduced design space for three points along the Pareto front of the second form of the equation shown in subplot (d). Subplot (a) corresponds to the subspace for the red circle, subplot (b) corresponds to the subspace for the blue triangle, and subplot (c) corresponds to the subspace for the green star.

5.4 Midship

To further test the LSC, a larger midship structural design problem was tested. This structural problem has a higher computational cost to perform the design fitness calculation, and more design variables and uncertain parameters.

5.4.1 Problem Description

The framework was applied to the design of a midship section which is shown in figure 5.8. The design problem has 36 design variables and four uncertain parameters. The uncertain parameters introduce uncertainty to the cost calculations and the structural requirements of the section. The cost and structural calculations for this problem are more computationally expensive than previous problems.

The midship section is made up of 33 stiffened panels. These panels were assigned to one of seven functional locations: Bottom Shell, Bilge Plate, Side Shell Below Draft, Side Shell Above Draft, Double Bottom, Weather Deck, and Deck Plates. Figure 5.8 is color coded with the location category for each panel; the location categories are decoded in table 5.3. The above and below draft side shell functional locations were combined to create six groups of plates, where each group is characterized by the same design variables. There are six design variables that define a panel: plate thickness (t_p), web thickness (t_w), web height (h_w), flange thickness (t_f), flange breadth (b_f), number of stiffeners (n_{stiff}); to account for different plate widths, the number

Table 5.3: Location codes for the Midship Section

Code	Description
BSBT	Bottom Shell/Ballast Tank
SBBT	Side Shell Below Draft Line/Ballast Tank
SBIH	Side Shell Below Draft Line/Inner Hull
SAIH	Side Shell Above Draft Line/Inner Hull
DIBT	Double Bottom
WTDK	Weather Deck
IH	Deck Plate

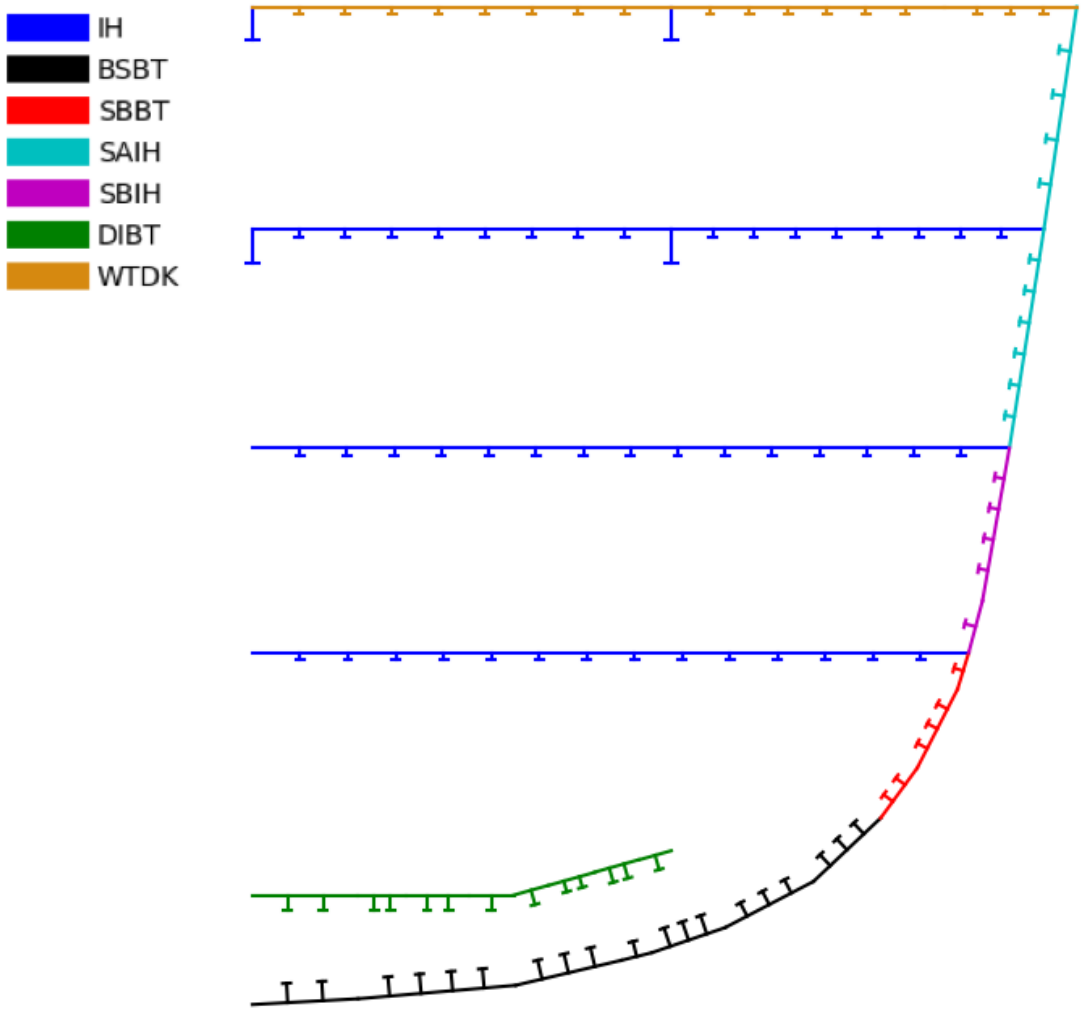

Figure 5.8: Original midship section showing the groups of plates

of stiffeners on a plate is relative to the number in the original design. With 6 design variables for each of the 6 groups of plates there are 36 design variables in total, and four uncertain parameters. The four uncertain parameters were the material cost, the labor cost, the required ultimate compression strength of each panel compared to the original design, and the required section modulus of the section compared to the original design. The design is constrained by a maximum weight and multiple minimum performance metrics; the constraints are in comparison to the original design

where which are designated by a $*$ in the equation. The definition of the problem is below.

$$\begin{aligned} \min \quad & C_p \\ \text{s.t.} \quad & UCS_i \geq f_{UCS} UCS_i^* \quad \forall\ i \in X \\ & SM_v \geq f_{SM} SM_v^* \\ & SM_{pi} \geq SM_{pi}^* \quad \forall\ i \in X \\ & w_s \leq 1.5 w_s^* \\ & ss_i \geq b_{fi} + .006 \quad \forall\ i \in X \\ & x_i \in X_i \quad \forall\ i \in X \end{aligned} \tag{5.5}$$

The production cost, C_p, is used as the design fitness metric for this example. The production cost of the midship section is calculated with the method originally presented in *Rahman and Caldwell* (2012) and adapted in *Temple* (2015) and *Liu* (2016)using equation 5.6. The terms of the cost function are described in table 5.4. The constants for the cost calculation are shown in table 5.5. The cost calculation method includes both the material and labor costs associated with the production of the section.

Table 5.4: Description of cost terms for box girder calculation

Cost term	Description
$C_{i,1}$	material cost for hull plates
$C_{i,2}$	material cost for longitudinal stiffeners
$C_{i,3}$	material cost for longitudinal framers
$C_{i,4}$	welding cost for longitudinal stiffeners
$C_{i,5}$	electricity and electrodes cost

$$C_p = \sum_{i=1}^{n} \sum_{j=1}^{6} C_{i,j} \tag{5.6}$$

There are five constraints to the problem. The first is that the ultimate compressive strength (UCS) of each plate, i, must be no less than a fraction, f_{UCS}, of

the UCS for that plate in the original midship section design, UCS_i^*. The UCS is calculated using the regression equations from *Paik and Duran* (2004). The fraction of the UCS required, f_{USC}, is one of the uncertain parameters. The second is that the section modulus, SM_i, of each plate is constrained to be no less than the SM of that plate in the original design, SM_{pi}^*. In addition, the SM of the vessel, SM_v, is must be no less than a fraction, f_{SM}, of the section modulus of the original midship design, SM_v^*. The fraction of the vessel section modulus required, f_{SM}, is one of the uncertain parameters. Additionally, the weight of the structure must be no more than 150% of the structural weight of the original midship design. Finally, the stiffener spacing must be sufficient such that the stiffener flanges are at least $6mm$ apart.

The upper and lower bounds for the uncertain parameters are shown in table 5.6. These uncertain parameters include two uncertain design requirements, and two uncertain parameters for the cost calculations. The upper and lower bounds for the design variables are shown in table 5.7. The number of stiffeners for each plate is adjusted with the m_s design variable. The number of stiffeners on each plate is multiplied by m_s and then equally spaced on the plate.

Table 5.5: Constant values used for the cost calculation of the midship

Variable	Value	Description
r	7.85	specific weight of the material (ton/m^3)
C_{lm}	1.05	material cost coefficient for longitudinal stiffeners
C_{ls}	1.2	labor hour required per meter welding of stiffeners to plate
C_{ee}	0.9	labor hour equivalent required per meter of stiffeners implementing electricity
C_{fb}	1.5	labor hour required per meter of stiffeners for fabrication

Table 5.6: Ranges for the uncertain parameters of the midship problem

	f_{SM}	f_{UCS}	P_s	P_a
min	0.5	0.5	\$550	\$20
max	1.5	1.0	\$1250	\$55

Table 5.7: Possible Ranges for design variables

Grouping		t_p	t_w	t_f	h_w	b_f	m_s
Bottom Shell/Ballast Tank	x_{min}	0.003	0.003	0.003	0.1	0.07	0.5
	x_{max}	0.012	0.01	0.015	0.3	0.15	1.5
Side Shell	x_{min}	0.004	0.003	0.003	0.1	0.07	0.5
	x_{max}	0.014	0.01	0.015	0.3	0.15	1.5
Side Shell/Ballast tank	x_{min}	0.004	0.003	0.003	0.1	0.07	0.5
	x_{max}	0.014	0.01	0.015	0.3	0.15	1.5
Double Bottom	x_{min}	0.004	0.003	0.003	0.1	0.07	0.5
	x_{max}	0.015	0.01	0.015	0.3	0.15	1.5
Weather Deck	x_{min}	0.006	0.003	0.003	0.07	0.07	0.5
	x_{max}	0.015	0.015	0.03	0.5	0.2	1.5
Inner Hull	x_{min}	0.003	0.003	0.003	0.07	0.07	0.5
	x_{max}	0.01	0.015	0.03	0.5	0.2	1.5

The parameters used for the SOGA to find the optimal solution in the original design space during the preprocessing step are shown in table 5.8. The magnitude of the objective function is significantly higher allowing convergence to occur with higher variable and convergence tolerances relative to the previous example.

Table 5.8: Parameters for the SOGA used in the preprocessing of the midship problem

Parameter	Value
Variable Tolerance	10
Convergence Tolerance	3
Crossover Percent	0.9
Mutation Percent	0.01
Real Parameter	2
Mutation Parameter	1.0
Population Size	200
Constant Generations	25
Max Generations	500
Elitism	1.0

5.4.2 Results

The resulting Pareto front for the problem is shown in figure 5.9. The front for this problem is different than results from the previous problems discussed in that the

regret values associated with the Pareto front are significantly smaller than in other problems.

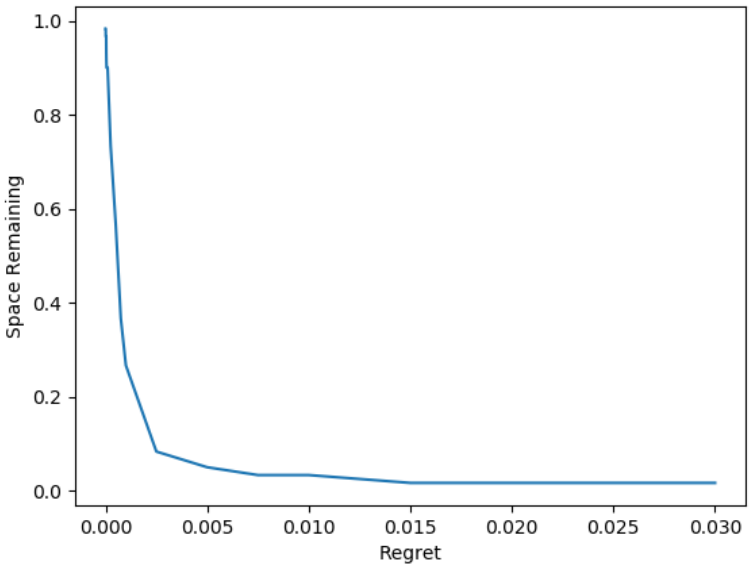

Figure 5.9: Pareto front for the midship test problem

From these results, the designer can conclude that the uncertainties modeled in this problem are not major contributing factors to the optimal design. While the uncertainty of the cost for material and labor will significantly affect the cost of the optimal design, the value of the design variables for the optimal design are not significantly affected by the uncertainty. This does not mean that the uncertainty doesn't affect the cost of the optimal midship design. This differentiation is due to the fact that the regret calculation is comparing the fitness to the optimal solution in the original design space for that specific possible realization of the uncertainty; In this case, the cost parameters are the same in the comparison.

These results tell designers that for the uncertainty and cost objective investigated, the design space can be significantly reduced, even to a single design, with minimal resulting regret. By selecting a single design point, there will be no more than 1.5% regret compared to the optimal solution for each possible realization of the uncertainty

parameters. While the design space is able to be reduced without significant regret, the solution does not provide a reduced range for the cost objective.

5.5 Summary

The LSC algorithm was developed to solve the DSC-U problem, and the framework and results from two test problems were presented. The algorithm solves the DSC-U problem to give the Pareto front in the regret-space remaining trade space. The algorithm utilized level sets and methods to solve the set covering problem (SCP) to solve the DSC-U; the algorithm uses sampling to handle a continuous design space defined by the user. The algorithm is made up of a set of independent optimization problems; the trade space is explored by solving the optimization problem for a number of different regret values. At each regret value of interest a level set of designs, that are within the regret threshold, is created for each sampled point in the uncertainty space; the SCP is then solved using these level sets to find the minimum number of designs required such that there is an acceptable design for each sampled uncertainty point. This algorithm is computationally efficient compared to the algorithm presented in Chapter IV and the computational effort is bound by the number of points sampled in each space.

The results of two problems were presented. The 7-variable Rosenbrock function was used with five design variables and two uncertain parameters. Two forms of this problem were solved where the vector $\mathbf{z}$ was created by combining the $\mathbf{x}$ and $\mathbf{a}$ differently. This resulting in different levels of coupling between the design variables and uncertain parameters. The LSC algorithm was able to capture these differences in the solution trade space. The design of a midship section was also examined. The results of this problem show the algorithm's ability to evaluate the variation of the optimal design over the range of uncertainty and not the variation of the design fitness over the range of uncertainty. In this example the cost associated with labor

and materials had a large uncertainty range, which results in a large cost variation over the range of uncertainty, but the solution showed that the optimal design is fairly independent of the cost uncertainty.

CHAPTER VI

Summary

6.1 Conclusions

The design space covering for uncertainty (DSC-U) problem as defined in Chapter III is a new problem formulation to look at decision making in early stage design. The problem is formulated to solve the trade space between the regret and space remaining metrics. The regret metric is a measure of the expected suboptimality of a final design for a given subspace after a design space reduction. The space remaining metric quantifies the size of a subspace after a design space reduction. The trade space between these metrics is explored and the solution of the problem is the Pareto front. The design space is divided between a design variable space and uncertain parameter space allowing algorithms to handle these parameters more appropriately.

6.1.1 Nested Algorithm

The first algorithm presented in Chapter IV to solve the DSC-U problem is the nested algorithm. The nested algorithm uses a single-objective objective problem within a multi-objective optimization problem to evaluate the trade space defined by the DSC-U problem. The algorithm was used to solve two small design problems: the box girder problem, and the cantilever tube problem.

As the first test problem for the DSC-U problem, the solution to the box girder problem proves that there are interesting findings to be made. The design spaces corresponding to points along the Pareto front were examined to reveal that some of the design variables were more tightly coupled to the uncertain parameters than others. The solution to this problem shows that information regarding design space reductions for specific variables can be produced.

The nested algorithm was also used to solve a cantilever tube problem. The solution to this problem further proved the value of the solution to the DSC-U problem by showing that multi dimension design and uncertainty spaces can be handled.

With further exploration of the nested algorithm, it was found that the problem becomes intractable for larger problems than those presented. The nested structure of the algorithm results in the computational effort exponentially increasing with an increasing design space. Future work was needed to restructure the algorithm for computational efficiency allowing larger problems to be solved.

6.1.2 Level Set Covering Algorithm

The second algorithm presented in Chapter V is the level set covering (LSC) algorithm. This algorithm is capable of solving larger DSC-U problems than the nested algorithm. The LSC algorithm solves an independent optimization problem for a set of regret values of interest. Given the regret value, a level set of acceptable solutions is found for each possible realization of the uncertainty parameters. The set covering problem (SCP) is then solved using these sets to find the smallest set possible to cover the entire uncertainty space. The algorithm was used to solve two larger test problems: the 7-variable Rosenbrock function, and a midship design problem.

Two forms of the Rosenbrock function were used and the comparison of the results between them proved informative. The second form of the problem is designed such that the design variables and uncertain parameters are more tightly coupled than in

the first form of the problem. Not surprisingly, for the same magnitude of design space reduction the more tightly coupled problem showed significantly more regret. The results are capable of capturing the degree that the design variables and uncertain parameters are coupled. The results were also able to capture information on the reduction for each design variable individually for various levels of regret.

The midship design problem is the largest problem solved by the DSC-U problem to date. The problem had 36 design variables that defined the plates and stiffeners of the midship section. The results of the midship design problem provide an example of the insight solutions to the DSC-U problem can provide of the level of dependence between the optimal design and the realized value of the uncertain parameter. The results also show that the algorithm is able to separate the fitness variation and the optimal design variation over the range of uncertainty. The ability to quickly solve a problem of this magnitude shows the computational efficiency compared to the nested algorithm of Chapter IV

6.2 Contributions

A new method to handle uncertainty in early stage design is defined as part of this dissertation. This work also includes two algorithms to solve this new problem definition. The major contributions of this dissertation are summarized here.

The design space covering for uncertainty (DSC-U) problem is a new problem definition to examine early stage design for uncertainty. The problem structure divides the design space of the variables and parameters necessary to evaluate the design fitness into two separate spaces: the X space which is made up of the design variables, and the A space which is made up of the uncertain parameters. This novel decomposition of the design space allows for the algorithm to handle the uncertain parameters separately from the design variables. This problem defines two metric that are competing objectives in early stage design: regret and space remaining. The

regret metric measures how close to optimal is possible after a design space reduction and the space remaining metric measures how large the design space is after reduction. The size of the design space is important because it directly affects the computational effort required to explore the design space. The problem defines a trade space between regret and space remaining to be used as a tool for design space reduction decision making in early stage design.

The nested algorithm was developed to solve the DSC-U problem. This algorithm uses single and multi objective optimization methods to solve for the Pareto front in the regret and space remaining trade space defined by the DSC-U problem. The algorithm has proven successful in solving small problems.

The level set covering (LSC) algorithm was developed to solve the DSC-U problem for larger design problems. The algorithm uses a regret threshold to define level sets that are used to find a subspace by solving the set covering problem. This solves for the Pareto front in the trade space between regret and space remaining.

6.3 Recommendations for future work

The algorithms presented in this work are the first to solve the DSC-U problem. From the results in this work the LSC algorithm shows the most promise to be extended.

Manual post processing analysis of the points in the Pareto front identify that future work on post processing analysis would provide additional insights to the design problem for designers. By analyzing each design variable separately, this post processing analysis could identify how tightly coupled specific design variables are with the uncertainty parameters. Additionally it may be possible to identify relationships between the optimal value of specific design variables and uncertain parameters; this could potentially provide a set of 'if-then' statements for quick decision making as uncertainty is realized. Relationships between design parameters could also be iden-

tified allowing the dimension of the design space to be reduced by having one design variable be dependent of another.

Future work to add the ability to see the design fitness variation is also recommended. The LSC currently only provides information on the maximum regret that will occur and does not provide information on the design fitness to expect. The algorithm should be extended to add the capability of providing the optimal solution for each sampled point in the uncertainty space. Data on the average regret for each subspace in addition to the maximum regret would also be informative to the designer.

Parallel processing has been implemented to improve computational efficiency, and further improvements could be made by introducing a surrogate method for problems that have computationally expensive fitness functions. In the preprocessing step of the algorithm a large number of points are evaluated for each sampled point in the A-space, especially surrounding the global optimum solution. A number of these points and their data could be extracted from the optimizer and used to build the surrogate model; to ensure sufficient coverage of the space a sparse sampling of the space should also be used. This would allow the surrogate model to be created with fewer needed additional design fitness evaluations.

BIBLIOGRAPHY

Altay, E., and B. Alatas (2018), Music based metaheuristic methods for constrained optimization, in *6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding*, vol. 2018-Janua, pp. 1–6.

Álvarez-sánchez, J. R., F. De, P. López, F. J. Toledo-moreo, H. A. Eds, and D. Hutchison (2015), *LNCS 9108 - Bioinspired Computation in Artificial Systems*.

Beasley, J. E., and R. C. Chu (1996), A genetic algorithm for the set covering problem, *94*, 392–404.

Bernstein, J. I. (1998), Design Methods in the Aerospace Industry: Looking for Evidence of Set-Based Practices, Ph.D. thesis, Massachusetts Institute of Technology.

Bertsimas, D., D. B. Brown, and C. Caramanis (2011), Theory and Applications of Robust Optimization, *SIAM Review*, *53*(3), 464–501.

Beyer, H. G., and B. Sendhoff (2007), Robust optimization - A comprehensive survey, *Computer Methods in Applied Mechanics and Engineering*, *196*(33-34), 3190–3218.

Brusco, M. J., L. W. Jacobs, and G. M. Thompson (1999), A morphing procedure to supplement a simulated annealing heuristic for cost-and coverage-correlated set-covering problems, *Tech. rep.*

Byrne, D. M., and S. Taguchi (1986), Taguchi approach to parameter design, pp. 168–177, cited By 78.

Caprara, A., and P. Toth (2000), Algorithms for the Set Covering Problem, pp. 353–371.

Crawford, B., R. Soto, M. Olivares-suárez, and F. Paredes (2014), A Binary Firefly Algorithm for the Set Covering Problem.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002), A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*.

Du, X. (2007), Interval Reliability Analysis, *Volume 6: 33rd Design Automation Conference, Parts A and B*, (3), 1103–1109.

Du, X. (2012), Reliability-based design optimization with dependent interval variables, *International Journal for Numerical Methods in Engineering*, (March), 1885–1891.

Ehrgott, M., J. Ide, and A. Schöbel (2014), Minmax robustness for multi-objective optimization problems, *European Journal of Operational Research*, *239*(1), 17–31.

Evans, J. H. (1959), Basic design concepts, *Journal of the American Society for Naval Engineers*, *71*(4), 671–678.

Faure, A., G. Michailidis, R. Estevez, G. Parry, and G. Allaire (2016), Design of Isotropic Microstructures via a Two-Scale Approach, in *ECCOMAS Congress 2016*, edited by M. Papadrakakis, V. Papadopoulos, G. Stefanou, and V. Plevris, Crete Island, Greece.

Ferson, S., and L. R. Ginzburg (1996), Different methods are needed to propagate ignorance and variability, *Reliability Engineering and System Safety*, *54*, 133–144.

Goerigk, M., and A. Schöbel (2014), Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling, *Computers and Operations Research*, *52*, 1–15.

Hackl, J. (2013), PyRe: Structural Reliability Analysis with Python.

Huang, Z. L., C. Jiang, Y. S. Zhou, J. Zheng, and X. Y. Long (2017), Reliability-based design optimization for problems with interval distribution parameters, *Structural and Multidisciplinary Optimization*, *55*(2), 513–528.

Jacobs, L. W., and M. J. Brusco (1995), Note: A local-search heuristic for large set-covering problems, *Naval Research Logistics (NRL)*, *42*(7), 1129–1140.

Karp, R. (1972), Reducibility among Combinatorial Problems, *Complexity of Computer Computations*, pp. 85–103.

Kok, S., and C. Sandrock (2009), Locating and Characterizing the Stationary Points of the Extended Rosenbrock Function, *Tech. rep.*

Liu, B., Q. Zhang, and G. G. E. Gielen (2014), A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems, *IEEE Transactions on Evolutionary Computation*, *18*(2), 180–192.

Liu, Y. (2016), Surrogate-assisted unified optimization framework for investigating marine structural design under information uncertainty, Ph.D. thesis, University of Michigan.

Liu, Y., and M. Collette (2014), Improving surrogate-assisted variable fidelity multi-objective optimization using a clustering algorithm, *Applied Soft Computing Journal*, *24*, 482–493.

Mckay, M. D., R. J. Beckman, and W. J. Conover (1979), A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code, *TECHNOMETRICS 0*, *21*(2).

Mckenney, T. A. (2013), An Early-Stage Set-Based Design Reduction Decision Support Framework Utilizing Design Space Mapping and a Graph Theoretic Markov Decision Process Formulation, Ph.D. thesis, University of Michigan.

Mckenney, T. A., L. F. Kemink, and D. J. Singer (2011), Adapting to Changes in Design Requirements Using Set-Based Design, *Naval Engineers Journal*.

Mebane, W. L., C. M. Carlson, C. Dowd, D. J. Singer, and M. E. Buckley (2011), Set-Based Design and the Ship to Shore Connector, *Naval Engineers Journal*.

Paik, J. K., and A. Duran (2004), Ultimate Strength of Aluminum Plates and Stiffened Panels for Marine Applications, *Marine Technology*, *41*(3), 108–121.

Qiu, H., Y. Xu, L. Gao, X. Li, and L. Chi (2016), Multi-stage design space reduction and metamodeling optimization method based on self-organizing maps and fuzzy clustering, *Expert Systems with Applications*, *46*, 180–195.

Rahman, M. K., and J. B. Caldwell (2012), Ship structures: improvement by rational design optimisation, *International shipbuilding progress*, *42*(October), 61–102.

Rigterink, D., M. Collette, and D. J. Singer (2013), A method for comparing panel complexity to traditional material and production cost estimating techniques, *Ocean Engineering*, *70*, 61–71.

Rosenbrock, H. (1960), An Automatic Method for Finding the Greatest or Least Value of a Function, *The Computer Journal*, *3*(1), 175–184.

Schmitt, L. M. (2001), Theory of genetic algorithms, *Theoretical Computer Science*, *259*, 1–61.

Shang, Y. W., and Y. H. Qiu (2006), A note on the extended Rosenbrock function, *Evolutionary Computation*, *14*(1), 119–126.

Singer, D. J., N. Doerry, and M. E. Buckley (2009), What is set-based design?, *Naval Engineers Journal*, *121*(4), 31–43.

Temple, D. W. (2015), A Multi-Objective Collaborative Optimization Framework to Understand Trade-offs Between Naval Lifetime Costs Considering Production, Operation, and Maintenance, Ph.D. thesis, University of Michigan.

Tseng, H.-H., S.-W. Wang, J.-Y. Chen, and C.-N. Liu (2014), A novel design space reduction method for efficient simulation-based optimization, pp. 381–384.

Viswanath, A., A. Forrester, and A. Keane (2009), Design space reduction in optimization using generative topographic mapping, *Design*, pp. 1–10.

Wang, G. G., and S. Shan (2004), Design Space Reduction for Multi-Objective Optimization and Robust Design Optimization Problems, *SAE Transactions, Journal of Materials and Manufacturing*, pp. 101–110.

Wang, G. G., and S. Shan (2007), Review of Metamodeling Techniques in Support of Engineering Design Optimization, *Journal of Mechanical Design*, *129*(4), 370.

Yao, W., X. Chen, W. Luo, M. van Tooren, and J. Guo (2011), Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles, *Progress in Aerospace Sciences*, *47*(6), 450–479.

Zhu, G. B. (2018), A new view of classification in astronomy with the archetype technique: an astronomical case of the NP-complete set cover problem, *Tech. rep.*