WHITESPACE EXPLORATION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Jason L. Daniel

November 2017

COMMITTEE MEMBERSHIP

TITLE:	Whitespace Exploration
AUTHOR:	Jason L. Daniel
DATE SUBMITTED:	November 2017
COMMITTEE CHAIR:	David Marshall, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Kira Abercromby, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Rob McDonald, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Colleen Kirk, Ph.D. Professor of Mathematics

ABSTRACT

Whitespace Exploration

Jason L. Daniel

As engineering systems grow in complexity so too must the design tools that we use evolve and allow for decision makers to efficiently ask questions of their model and obtain meaningful answers. The process of whitespace exploration has recently been developed to aid in engineering design and provide insight into a design space where traditional design exploration methods may fail. In an effort to further the research and development of whitespace exploration algorithms, a software package called Thalia has been created to allow for automated data collection and experimentation with the whitespace exploration methodology.

In this work, whitespace exploration is defined and the current state of the art of whitespace exploration algorithms is reviewed. The whitespace exploration library Thalia along with a collection of benchmarking cases are described in detail. A set of experiments on the benchmark cases are run and analyzed to further understand the behavior of the algorithm and outline initial performance results which can later be used for comparison to aid in improving the methodology.

ACKNOWLEDGMENTS

Thanks to:

- Dr. David D. Marshall for never telling me that a problem was too difficult to solve and inspiring me to explore the wonder that is computational science.
- My Parents for all of their love and support.
- Phoenix Integration for showing me how things are done in the real world, and supporting the continued development of whitespace exploration.
- Robert Hawkins for always reminding me to always be world class and the good times we had doing awesome science in the "sad lab".
- Edgar Busovaca for his humor, friendship, and love of science.
- William Keel for teaching me the importance of the user in software development.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

In engineering design, the ability to effectively explore the design space and iterate on concepts can mean the difference in producing a successful system or one fraught with cost and schedule overruns. As technology and design exploration methods have matured, computer models and experiments are being more widely used in engineering design. Often in engineering, the design process is derived to meet a set of system requirements, however requirements often change and it is important to be able to ask 'what if' questions of an engineering model in order to design systems that will be robust and adaptable. There are also cases in design where the requirements are not well known or that one seeks designs that are simply the best possible and one wishes to understand what factors are holding the system back from improved performance. The importance of design space exploration and the ability to do so early on in the project has a direct relation to project costs. Changes in the design are easier to make and much less costly early on as shown in Figure 1.1.

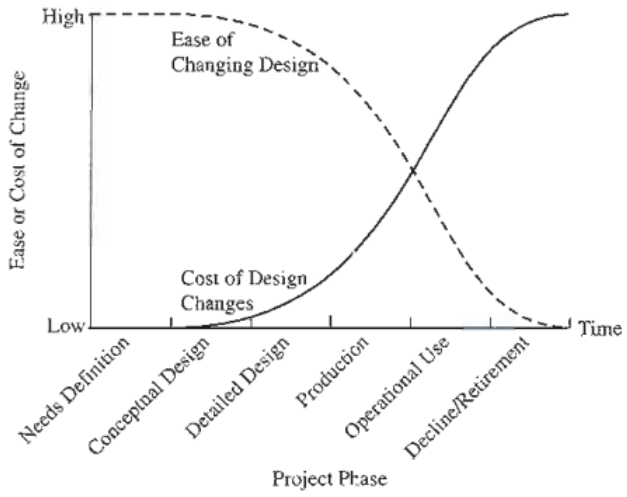

Figure 1.1: Value of design space exploration early on in the design process

Any exploration of the design space could be considered to reside under the umbrella of methods referred to as whitespace exploration; the development of tools and algorithms designed to provide information about previously unexplored regions the the design space. The problem that whitespace exploration is truly trying to solve however, is that of a design space where some exploration has already occurred through a design of experiments, optimization, or similar methods, and yet there still exist regions of the space that contain no information. This is what is referred to as the whitespace as shown in Figure 1.2.

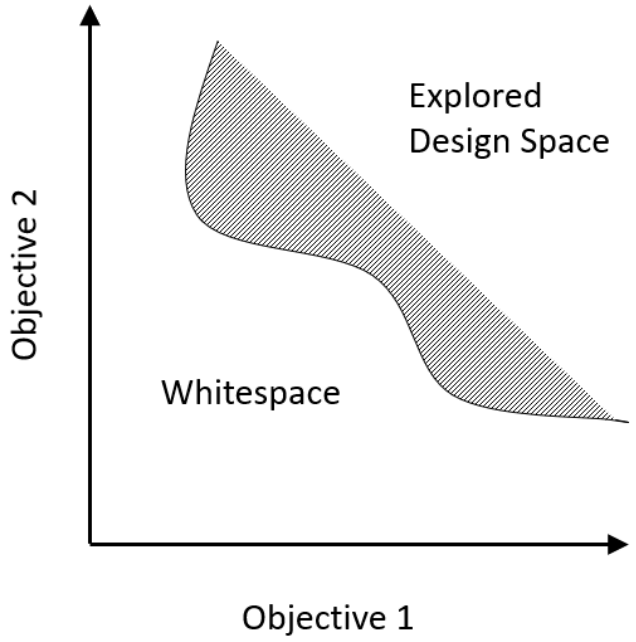

Figure 1.2: Whitespace in a design exploration

There exist a variety of reasons why a design engineer would wish to explore the whitespace of a model. The most straightforward example is that none of the designs found meet system requirements, and that the desired design exist in the whitespace as shown in Figure 1.3. Alternatively one may wish to improve upon an existing system and wishes to know the most effective way to push the design into other regions of the design space. By providing answers to the question of why

the whitespace exist for a model, better decisions can be made as to where additional effort should be placed in the design process, or if the desired region of the whitespace is truly inaccessible, that alternative architectures may need to be explored.

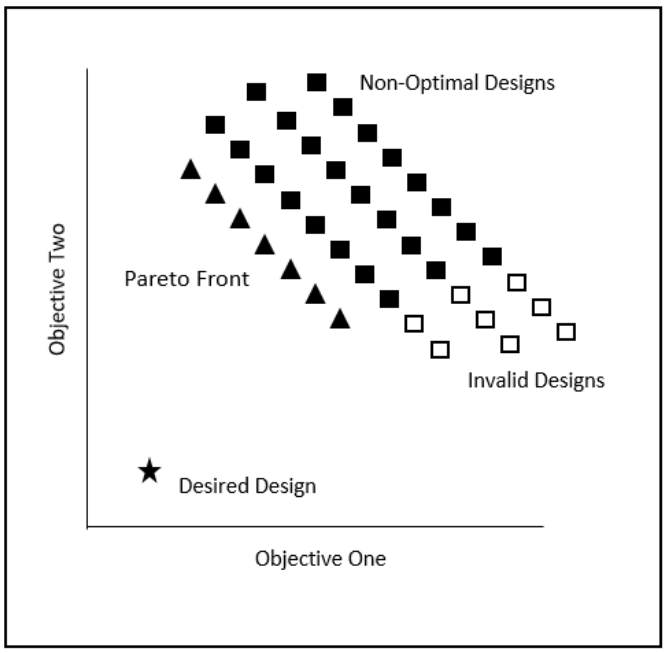

Figure 1.3: Design exploration that does not produce the desired system

1.1 The Whitespace Exploration Algorithm

Whitespace exploration has the primary goal of finding designs in the *whitespace* of a design space that have not yet been reached. The current baseline algorithm proposed by [12] volunteers that in a sufficiently explored design space that regions of whitespace will exist due to the assumptions made about the model. Assumptions for a model can manifest in a few different ways

- **Design Variable Box Constraints**

  The upper and lower bounds specified for input design variables may prevent designs from reaching the whitespace.

- **Model Constraints**

  The specified model output constraints may prevent designs from reaching the whitespace.

- **Non-Design Input Variables**

  Model inputs that were set as constants may prevent designs from populating the whitespace.

The process of whitespace exploration seeks to examine and modify these assumptions in order to produce designs within the whitespace through a mixture of design space exploration techniques. The design variable box bounds are often set based on either some physical constraint, mathematical property of the parameter, or educated guess about what is appropriate. In all cases obtaining insight into which box constraints are active and preventing progression towards a desired design can encourage a deeper look into ways that they could be relaxed.

Model output constraints become an important assumption to examine when the Pareto front, or a portion of it, is active on the constraint boundary. Constraints that are tied to system requirements or preference are the likely candidates to investigate for relaxation to allow for improved system performance. Those constraints that are physical in nature to allow proper function of the system are less likely to allow for modification, but may spur discussion of the system architecture to remove or work around such constraints if they are preventing progress.

The non-design variables are the most unique perspective of whitespace exploration and they represent a paradigm shift in how we look at system inputs. The non-design variables are inputs to the model that are typically considered constants. Examples of these types of variables could be theoretical efficiency values, holdovers from previous design decisions, or based on physical properties. While a parameter like the speed of light can be considered a non-design variable, it is also one that is

unlikely to be changed. A parameter such as the coefficient of drag on a wing, however, is something that can be changed through engineering work. While non-design variables are typically something we do not want to change, otherwise they would be considered design variables, the current model may state that something has to give and the modification of a non-design variable may be the solution.

In all cases where an assumption is shown to be a factor in preventing progression into the design space, whitespace exploration can be used to show the decision maker what they get in return for modifying that assumption to assess if the effort required to change the parameter is worth the potential performance gain.

The inputs to the original whitespace exploration algorithm presented in [12] are as follows

- **Analysis Model**

  The model is treated as a 'black box' which takes input values and produces output values.

- **Design Variables**

  These are the tune-able input parameters for the model and have lower and upper bound constraints associated with them.

- **Constraints**

  These are the output variable constraints on the model which determine if a design is valid or invalid.

- **Non-Design Variables**

  These are the variables that would typically be considered constant inputs for the model but do have variability to them. These variables also have lower and upper bounds associated with them.

- **Baseline Design**

  This is a specified set of inputs that produce a valid output for the model to be used as the starting point for the whitespace exploration process.

- **Desired Point**

  This is a set of objective values in the whitespace that we wish to move towards.

- **Exploration Configuration Parameters**

  These are the set of user specified parameters that are utilized by the sub-processes in the whitespace exploration process.

The whitespace exploration algorithm consist of eleven unique steps.

1. **Generate Pareto Front**

   An initial Pareto front is generated from the model.

2. **Select the *Desired Point* in the Whitespace**

   A point in the objective space is selected by the decision maker that the Pareto front will be driven toward.

3. **Relax Response Constraints**

   If possible, response variable constraints are relaxed.

4. **Identify the *Selected Point* on the Pareto frontier**

   A point on the Pareto front is selected by the decision maker as a representative of the Pareto front.

5. **Identify *Non-Design Variables* as candidates for modification**

   Non-design variables that could be modified are selected by the decision maker for investigation

6. **Perform a sensitivity study on the selected variables**

   A sensitivity analysis is performed about the selected point to determine which of the non-design variables have an affect on the system objectives.

7. **Select variables to modify based on sensitivity results**

   The decision maker selects non-design variables that were shown to have an affect on the system objectives.

8. **Determine optimum combination of values for selected variables**

   A search is performed to determine the values of the selected non-design variables that would best move the selected point in the direction of the desired point.

9. **Determine step size**

   The decision maker determines a step size to apply to the values found by the previous step.

10. **Apply variable changes**

    Update the non-design variable values after the step is applied to the model.

11. **Generate new Pareto front**

    A new Pareto front is generated from the updated model and the process is repeated until sufficient progress of the Pareto front into the whitespace has been achieved.

1.2 Motivation

In general this work seeks to aid in maturing methods for design space exploration. More specifically we are interested in improving the new class of algorithms called

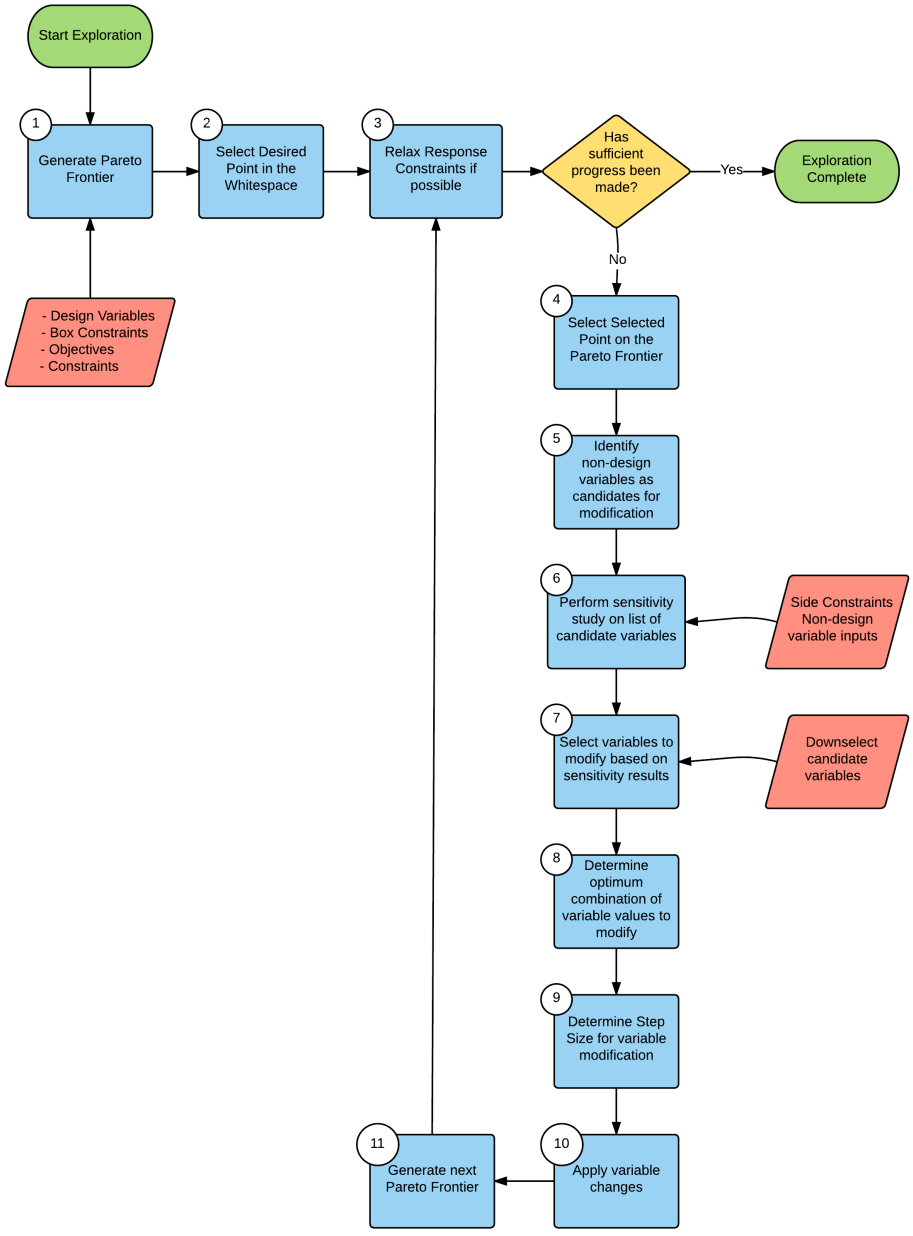

Figure 1.4: Original whitespace exploration methodology [12]

whitespace exploration which have the potential to improve design space exploration and provide an answer to a question faced by design engineers.

Given a set of solutions for a model where no design meets system requirements, what process can be applied to reach the desired design?.

In this work a variety of terminology is also set forth to aid in further discussion about the topics presented. In the optimization community, there exists a variety of well known test problems and published performance metrics on those problems for different algorithms which allows for direct comparison with newly developed algorithms. In a similar fashion to optimization it is necessary to develop a set of test problems and comparable metrics for whitespace exploration algorithms to allow for direct comparison and proof that new algorithms have indeed found an improved procedure. The analyses performed on the benchmark problems are also used to illicit issues with the current algorithm and provide motivation for future research in the area.

This work also serves to promote the Thalia library and associated utilities so that other interested researchers might be able to use it to further this evolving field.

Chapter 2

THALIA: AUTOMATED WHITESPACE EXPLORATION UTILITY

In order to achieve the objectives of this work a software utility was developed to aid in the research and development of whitespace exploration algorithms dubbed Thalia. The initial manual whitespace exploration algorithm along with a prototype for whitespace exploration was developed at Phoenix Integration [12] was built on Phoenix Integration's Model Center Cloud system which provided an example of how whitespace exploration could be implemented and deployed in an enterprise environment. While the cloud system Whitespace Explorer met the needs of that research, for the goals of this work it was desired to have a system that was more lightweight and faster as there is significant overhead incurred by the machinery required for a cloud architecture in both developer time and model evaluation. Another aspect of the original Whitespace Explorer that did not fit well with this research is the required user interaction at each step in the whitespace exploration process. While the human in the loop aspect of whitespace exploration is very important, the goal of this work is to quickly assess bulk behaviors and gain insights into the whitespace exploration process in order to promote further improvement and so a new framework was developed using an automated approach was deemed a to be a more efficient approach to achieve these goals.

The Thalia: Automated Whitespace Exploration software package is a collection of tools developed in Java along with some modifications to the original baseline whitespace exploration algorithm to allow for easy modification of the whitespace exploration methodology and bulk data gathering via automated whitespace explorations. The software package developed for this work contains multiple pieces created with future use and extensibility in mind.

- **Thalia Whitespace Exploration Library**

  A Java library that contains all the necessary components to set up and run a whitespace exploration

- **Thalia CLI**

  A command line interface for interacting with the Thalia Whitespace Exploration Library

- **Thalia SDK**

  A collection of extendable classes and interfaces used to integrate and define new models that can be used in Thalia

- **Benchmark Model Suite**

  A collection of test models suitable for studying whitespace exploration

- **pyThalia**

  A python package developed for post processing the results of explorations produced by Thalia.

2.1 The Automated Whitespace Exploration Algorithm

The algorithm used for whitespace exploration that is implemented in Thalia is a modified version of the original algorithm where each part that involves human interaction has been replaced with an automated procedure. The algorithm is split into five *steps*, each iteration of these steps is called a *round*. The execution of multiple rounds until some stopping criteria is met is called an *exploration*. After each round the model is updated and a new Pareto front can be generated that is expected to be closer to the desired point.

The automated version of the whitespace exploration algorithm that is implemented in Thalia is initialized with a *workflow* and a configuration. The workflow

specifies the analysis model inputs, outputs, and a run method for performing the analysis. The configuration contains all of the information and tuning parameters needed to run an exploration.

For the automated algorithm used in this work some assumptions are made. These assumptions serve to simplify the analysis and scope the use cases that are investigated in this research. The assumptions are as follows:

- **The model is considered a black box**

  The model contains only inputs and outputs and no introspection will be performed based on any internal information from the model during the exploration.

- **All variables are real and continuous**

  Only real valued, continuous input and output variables are considered.

- **Only the progression of the Pareto front is considered**

  While the concepts of whitespace exploration could be applied to a wide variety of cases, in this work only the progression of the Pareto front into the whitespace is considered.

- **The input variables will not change**

  The input variables specified at the beginning of an exploration will remain the same throughout the exploration with no additions or removals of variables. Any changes to the variables that are considered in an exploration would result in a separate exploration for the purposes of this work.

- **The output variables will not change**

  The objective and output constraint variables specified in the exploration will remain the same throughout the exploration. Changing of these variables would be considered a new exploration.

- **The desired point will not change**

  The desired point will remain the same throughout an exploration. A change in the location of the desired point would be considered a new exploration for the purposes of this work/

The five steps utilized for the automated whitespace exploration process are as follows:

1. **Pareto Search**

   A search for the Pareto front is performed.

2. **Constraint Search**

   Active output constraints and design variable box constraints are found.

3. **Sensitivity Analysis**

   A sensitivity analysis is performed to identify non-design variables that may affect progression of the Pareto front into the whitespace.

4. **Direction Search**

   A search to find the optimal set of non-design variable values is sought which defines the direction of progress into the whitespace.

5. **Exploration Search**

   A search is performed to obtain a step size for progressing the selected point into the whitespace and updating the model to a new baseline.

A whitespace exploration problem definition is similar to that of a multi-objective optimization problem and consist of specifying the input variables, output variables, and a convergence criteria. In Thalia the convergence criteria for an exploration is simply a set number of rounds that are run, with each round, hopefully, progressing

the Pareto front further into the whitespace towards the desired point. These inputs are summarized in Table 2.1.

Table 2.1: Whitespace exploration algorithm inputs

Exploration Settings	
Property	Description
Max Rounds	The maximum number of rounds (iterations) to run the exploration
Baseline Design	A set of valid inputs for the model that represents a typical design or starting point
Desired Point	The desired point in the whitespace
Design Variables	The model design variables
Objective Variables	The model objective variables
Constraint Variables	The model constraint variables
Non-Design Variables	The model non-design variables

2.1.1 Pareto Search

The goal of the Pareto search step is to find the model's Pareto front for the current baseline design. The Pareto search step utilizes the Non-Dominated Sorting Genetic Algorithm 2 commonly referred to as NSGA-II [5]. This is a popular and robust multi-objective optimization algorithm that uses genetic operators on an evolving set of solutions known as individuals in it's search. In Thalia a modified version of the jMetal NSGA-II implementation is used [6]. The configuration parameters for this step that are set *a priori* to the exploration primarily contain of the adjustable

parameters to the NSGA-II algorithm which are summarized in Table 2.1.

Table 2.2: The Pareto search step configuration parameters

Pareto Search Configuration	
Property	Description
Population Size	The populations size used for the genetic search algorithm
Max Evaluations	The maximum number of evaluations for the Pareto search
Convergence Tolerance	Pareto search convergence Tolerance
Convergence Generations	Number of convergence generations
Crossover Probability	The genetic search algorithm crossover probability
Crossover Index	The genetic search algorithm crossover index
Mutation Probability	The genetic search algorithm mutation probability
Mutation Index	The genetic search algorithm mutation index

Given that the true Pareto front would not be known in a real engineering scenario, the approximation of the true front that produced by NSGA-II is used. The Pareto front is considered to be sufficiently resolved when a set of convergence metrics have not changed more than a specified tolerance for a set number of generations of the algorithm. The different metrics are utilized to examine the size and shape of the Pareto front and assumes that if the front is seen to be unchanging for a number of generation then further execution of the search algorithm will not yield better results.

These metrics are used in favor of a set number of model evaluations or generations as is also common in order to better compare results between different models as a set number evaluations may produce a sufficiently resolved front for one model while producing one that is unresolved for another model without tweaking this parameter on a model by model bases.

The three convergence metrics used are the expansion metric, density metric, and goodness metric [3]. The expansion metric tracks the 'growth' of the Pareto front by observing the change in the range of objective values. The range is calculated for each objective using

$$\mathcal{R}_i^N = |f_{i_{max}} - f_{i_{min}}| \,. \tag{2.1}$$

where $\mathcal{R}_i^N$ is the range of the i'th objective of a design in the Pareto front for generation N. The expansion metric is then calculated from the maximum normalized difference between the range $\mathcal{R}_i^N$ of the current generation and the previous generation

$$\mathcal{E}^N = max \left\{ \left| \frac{\mathcal{R}_i^N - \mathcal{R}_i^{N-1}}{\mathcal{R}_i^N} \right| \right\}. \tag{2.2}$$

The density metric represents the change in the Pareto front density between generation and is calculated using

$$\mathcal{D} = \frac{\texttt{size}(\mathcal{F}^N)}{\mathcal{V}^N} \tag{2.3}$$

where $\texttt{size}(\mathcal{F}^N)$ is the number of solutions in the Pareto front and $\mathcal{V}^N$ is the volume of the N'th generation's Pareto front calculated using

$$\mathcal{V}^N = \prod_{i=0}^{i=K} |f_{i_{max}} - f_{i_{min}}| \tag{2.4}$$

where K is the number of objectives. The frontier goodness metric also referred to as simply 'goodness' tracks the ratio of solutions in the Pareto front, to the population size.

$$\mathcal{G}^N = \frac{M - \mathcal{F}^N}{M} \tag{2.5}$$

An optimization run is considered converged when the metrics have not changed within a specified tolerance ϵ for a set number of generations G_{conv}. A generation is flagged as approaching convergence when the following criteria is met

$$\texttt{max}(|\mathcal{E}^N - \mathcal{E}^{N-1}|, |\mathcal{D}^N - \mathcal{D}^{N-1}|, |\mathcal{G}^N - \mathcal{G}^{N-1}|) < \epsilon. \tag{2.6}$$

If G_{conv} consecutive generations are flagged as approaching convergence than the optimization is considered converged and it's execution is halted.

Once the NSGA-II algorithm has converged, and a Pareto front found, the entire set of evaluated solutions is then passed into the NSGA-II non-dominated sorting procedure to produce the final Pareto front that is used for calculations in the proceeding steps. This final step is so that the most complete Pareto front is used based on information already obtained by the search process and not constrained by the population size specified for the NSGA-II algorithm.

Along with finding the Pareto front, the Pareto search step is also used to identify the *selected point* on the front. In the manual version of the whitespace exploration algorithm, the selected point is chosen by the user, but since this is an automated algorithm, an automated procedure is used for choosing the selected point. In this case, the Pareto front is normalized on a scale of zero to one with the normalization factor also being applied to the desired point, then the closest point on the Pareto front to the desired point is chosen as the selected point. The normalization process leads to the selected point being chosen on the portion of the Pareto front that has the largest range.

2.1.2 Constraint Search

The constraint search step uses the Pareto front produced by the Pareto search step and seeks to identify active constraints. A constraint, either output constraint or design variable box constraint is determined to be active if a sufficient portion of

designs in the Pareto front are within a specified tolerance from the constraint value. The box constraint activity of each design is checked against the normalized decision variable value using

$$x^i_{\texttt{normalized}} = \frac{x^i - x_{lb}}{x_{ub} - x_{lb}}. \tag{2.7}$$

If the normalized value for a solution is less than the relative tolerance value than it is considered active on the lower box constraint, and if it is greater than one minus the relative tolerance value than it is considered active on the upper box constraint. If the percentage of designs that are active on a box constraint is equal to or greater than the box constraint change tolerance value than that constraint is flagged for modification, and the lower, upper, or both box constraint values will be modified by the box constraint change percentage for the next round in the exploration.

A similar process for the output constraints is used whereby each solution in the Pareto front is checked for activity based on how close it's values are to the constraint boundary, if a sufficient number of solutions are considered active specified by the constraint change tolerance, than that constraint will be modified by the constraint change percentage for the next round.

2.1.3 Sensitivity Analysis

The sensitivity analysis step is used to determine which non-design variables should be used as inputs for the direction search step. The output of the sensitivity analysis step is the relative influence for each of the non-design variables with respect to each of the objective variables. Based on user configuration, those variables that have a sensitivity above a specified tolerance are chosen for use in the direction search step. The sensitivity analysis calculation begins by creating a stencil of designs to be evaluated are used in a finite difference calculation. For each dimension two design

Table 2.3: The constraint search step configuration parameters

Constraint Search Configuration	
Property	Description
Box Constraint Relative Tolerance	The tolerance for indicating that a solution is active on a given box constraint
Box Constraint Change Tolerance	The tolerance for indicating that enough Pareto front solutions are active on the box constraint to trigger a modification
Box Constraint Change Percentage	The percentage amount to modify a flagged box constraint for the next round
Constraint Relative Tolerance	The tolerance for indicating that a solution is active on a given box constraint
Constraint Change Tolerance	The tolerance for indicating that enough Pareto front solutions are active on the constraint to trigger a modification
Constraint Change Percentage	The percentage amount to modify a flagged constraint for the next round

points are specified at

$$x_i^L = x_i(1 - \epsilon_s) \tag{2.8}$$

$$x_i^U = x_i(1 + \epsilon_s) \tag{2.9}$$

where the 'lower point' x_i^L is the value of the selected point on dimension i minus the stencil size parameter ϵ_s and the 'upper point' x_i^U is the value of the selected point plus the stencil size parameter. This results in $2N$ total points for the stencil where N is the number of dimensions in the decision space.A non-dimensionalization parameter is used and is calculated as

$$D_{i,j} = \frac{x_j}{u_i} \tag{2.10}$$

where i ranges the number of output variables and j ranges the number of input variables evaluated at the selected point that is the center of the stencil. The partial derivatives are calculated for each output - input pair

$$\frac{\partial U}{\partial X} = D_{i,j} * \frac{U_{i,j}^+ - U_{i,j}^-}{H_j} \tag{2.11}$$

where $U_{i,j}$ is the output value for output i and input j, with the $+$ and $-$ denoting the high and low values for the stencil. H_j is the spacing for input j and $D_{i,j}$ is the non-dimesionalization parameter. With the partial derivatives known, the relative influence $\Phi_{i,j}$ of each input on each output using the central difference formula

$$\Phi_{i,j} = \frac{\partial U_{i,j}}{\partial X_{i,j}} / \sum_{i=0}^{n} \frac{\partial U_{i,j}}{\partial X_{i,j}}. \tag{2.12}$$

The relative influence value $\Phi_{i,j}$ scales the sensitivity values for each output between 0 and 1 and indicates how influential an input is compared to the others.

2.1.4 Direction Search

The direction search step is used to find the direction vector composed of the non-design variables that will move the selected point into the whitespace in the direction

Table 2.4: Sensitivity analysis step configuration parameters

Sensitivity Analysis Configuration	
Property	Description
Stencil Size	A percentage value applied to each input to create the finite difference stencil
Selection Tolerance	The sensitivity tolerance for a variable to be selected for inclusion in the direction search

of the desired point. The direction search is set up in the form of an optimization problem where the objective is to maximize the dot product between two vectors in the model's objective space. The first vector is constructed from the selected point and the desired point, the second vector is constructed from the selected point and a new point called the *calculated point* which is the point in objective space that is found as a result of changing the non-design variables of the selected point. By optimizing the dot product between these two vectors, we hope to achieve a dot product of one, which results in a calculated point that is directly in line with our desired point.

The optimization algorithm used in Thalia for the direction search is a version of NSGA-II that has been modified to be more efficient for a single objective. In place of the standard non-dominated sorting procedure used in NSGA-II, the population is sorted by objective value and the desired number of best individuals are carried over to the next generation to maintain elitism. Note that because the direction search is a single objective problem and there is no Pareto front, the algorithm does not use the same procedure as the Pareto front search for it's convergence criteria and instead checks to see if the best found objective value has changed for a specified number of generations.

Table 2.5: The direction search step configuration parameters

Direction Search Configuration	
Property	Description
Population Size	The populations size used for the genetic search algorithm
Max Evaluations	The maximum number of evaluations for the direction search
Convergence Tolerance	direction search convergence Tolerance
Convergence Generations	Number of convergence generations
Crossover Probability	The genetic search algorithm crossover probability
Crossover Index	The genetic search algorithm crossover index
Mutation Probability	The genetic search algorithm mutation probability
Mutation Index	The genetic search algorithm mutation index

2.1.5 Exploration Search

The exploration search step uses the direction found by the direction search step and adjusts the non-design variables in order to project the selected point into the whitespace. Once there has been sufficient progress into the whitespace by the exploration search, a new point is defined, the exploration point, which will then become the new baseline design for the next round of exploration. The goal of this step is that if a representative design in the Pareto front, the selected point, can be moved into the whitespace towards the desired point, then the rest of the Pareto front will follow. In

the manual whitespace exploration algorithm the user determines what is sufficient before moving on to the next round.

In Thalia, a Golden Section search [15] is used to find the exploration point that is closest to the desired point. The search iterates on an input exploration parameter η_e which is a multiplier on the vector calculated vector which is the calculated vector from the *selected point* to the resultant calculated point from the direction search. An input value of zero for the exploration parameter results in the selected point itself, while a value of one would give the calculated point. It is desired that other values of the exploration parameter will give a point in line with the direction found by the direction search, however this is model dependent and nonlinearities that exist between the non-design variables and the objective space could cause the exploration point to veer away. By setting a limit on η_e, the amount of progression of the selected point into the whitespace at each round can be controlled.

The objective for the Golden Section search is based on two vectors, the exploration vector V_e and the desired vector V_d which are defined in Equations 2.13 and 2.14.

$$V_e = p_e - p_s \tag{2.13}$$

$$V_d = p_d - p_s \tag{2.14}$$

where p_e is the current exploration point, p_s is the selected point, and p_d is the desired point, which are all defined in the objective space. The objective that we seek to minimize is the negative ratio of norms of these vectors called the exploration ratio given by Equation 2.15.

$$E_r = -\frac{||V_e||}{||V_d||} \tag{2.15}$$

In Thalia, the euclidean norm is used although any norm would be suitable. The more negative the exploration ratio E_r is, the closer the exploration point is to the desired point.

Table 2.6: The exploration search step configuration parameters

Exploration Search Configuration	
Property	Description
Max Evaluations	The maximum number of evaluation that can be performed by the exploration search
Search Tolerance	The golden section search convergence tolerance
Max Distance	The maximum input value of the exploration parameter

2.2 Thalia Whitespace Exploration Library

The primary entry point for to the library for running a whitespace exploration is the Exploration class. The class constructor takes in three arguments, a workflow which defines the model, a configuration which defines all the setup information required to execute an exploration, and an optional status object which is used as a callback mechanism to provide on-line information about the progress of an exploration. The whitespace exploration is initiated by calling the run method which returns a results object containing all of the data pertaining to the run.

The workflow fully defines the model that an exploration is going to be performed on. For the purposes of the exploration, the analysis is treated as a black box analysis where each time the workflow is called for evaluation, all inputs are fed to the analysis and all output values obtained. A workflow is defined by a name, it's inputs, it's outputs and an evaluate method. New workflows can be created by implementing an interface and specifying the necessary model metadata in the concrete class.

Whitespace exploration being composed of a collection of different algorithms contains a variety of tuning parameters that will affect it's search process. These parameters are defined in a configuration object that is passed to the library. The Thalia command line interface allows for this configuration to be defined in a JSON file that is provided as one of the input parameters. A basic configuration would consist of:

- **Design Variables** The names of the model input variables that are the design variables which will be used during the Pareto front search and constraint search steps along with the variable upper and lower box bounds.
- **Objective Variables** The names of the objectives of the model along with the search goal, to either minimize or maximize the objective.

- **Constraint Variables** The names of the output variables that represent model constraints along with values for the upper or lower bound of what is to be considered valid for a design.

- **Non-Design Variables** The names of the input variables that are not considered design variables but can be changed. Lower and upper bounds are also provided to ensure that nonsensical designs are not found.

For any option that is not specified, a default value is used. Part of the output of an exploration is a full representation of the configuration saved in JSON format that can be later used and modified as input for further experiments. This configuration file is also used as an input for the post processing utilities.

2.3 Thalia Command Line Interface

The Thalia command line interface wraps the Thalia whitespace exploration library and provides an easy way to set up and run whitespace explorations. The help output for the interface is shown Figure 2.1 which shows the options to provide the workflow, configuration file, data file output location, and the option to execute a number of batch runs on the same model. In the specified output directory, data files containing all model evaluations and runtime metadata is stored for every step in the exploration which can then be loaded into the pyThalia post processing library for further analysis.

2.4 pyThalia Post Processing Library

The pyThalia post processing library is a collection of utilities for performing analysis on the results of a whitespace exploration. The library loads the results of one or

```
  _____ _   _     _    _     ___    _
 |_   _| | | |  / \  | |   |_ _|  / \
   | | | |_| | / _ \ | |    | |  / _ \
   | | |  _  |/ ___ \| |___ | | / ___ \
   |_| |_| |_/_/   \_\_____|___/_/   \_\
 Automated Whitespace Exploration Utility
****************************************


usage: Thalia [-b <arg>] [-c <arg>] [-h] [-o <arg>] [-w <arg>]
Execute a whitespace exploration
 -b,--batch <arg>      number of batch runs to perform
 -c,--config <arg>     path to the exploration configuration file
 -h,--help             display this message
 -o,--output <arg>     path to the output directory
 -w,--workflow <arg>   path to the jar containing the workflow
Example Usage: thalia -c=/path/to/configuration.json
-w=/path/to/workflow.jar -o=/path/to/output/location -b=30
```

Figure 2.1: Thalia command line interface

more explorations that were run using the Thalia command line tool and serves as an easy way to perform data analysis using the Python programming language.

The pyThalia library also contains utilities for evaluating a set of metrics that can be used to better understand the results of an exploration. One such metric is the hypervolume indicator, also known as the Lebesgue measure or S metric [10] and is a popular metric for tracking the progress of a multi-objective evolutionary algorithm. The metric itself can be thought of as representing the 'size' of the Pareto front in relation to a reference point as shown in Figure 2.2. The specific value of the hypervolume indicator bears little meaning, however the relative change between the value can tell a compelling story about how a Pareto front has evolved over some set of operations.

Using the hypervolume indicator in the context of whitespace exploration, the *desired point* is used as the reference point, and so the hypervolume roughly represents

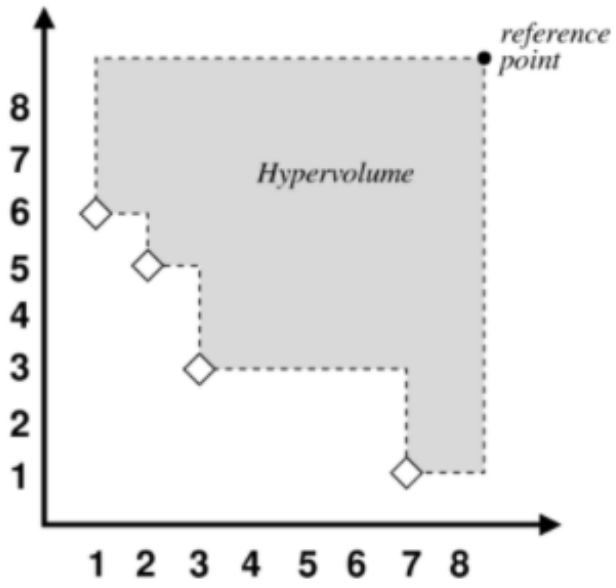

Figure 2.2: The hypervolume indicator in 2D[7]

the volume of whitespace that is left to be explored at each round. The hypervolume can be misleading however in cases where the variable box bounds and constraints are modified, causing the Pareto front to grow in size resulting in a larger hypervolume value that would compete with the amount the volume is reduced due to progression towards the *desired point*. Despite these issues with the hypervolume indicator, it can serve as an additional tool particularly in high dimensional objective spaces where direct visualization of the Pareto front progression may be difficult.

Another metric extracted by pyThalia is the mean function evaluations of the algorithm. The mean function evaluations of an algorithm, or MFE, is the number of times that the analysis model needed to be evaluated for the algorithm to reach a certain state. If the time to evaluate the analysis model is significantly longer than any of the other operations in the algorithm, which is typically true for engineering analyses, then MFE is a good indicator of computational complexity for an algorithm. For an algorithm to be worthwhile in practice, the computational cost of running it

must be low enough to incentivize it's use in practice, making it an important metric to examine and provides a straightforward way to compare the performance of new whitespace exploration implementations.

The last derived metric that we will use to examine an exploration is the mean Pareto distance. The mean Pareto distance metric is the average all pairs distance between the Pareto front generated at consecutive rounds as given in Equation 2.16.

$$\frac{\sum_{i=0}^{N} \sum_{j=0}^{M} ||f_i - f_j||}{K} \tag{2.16}$$

where N is the number of solutions in the current exploration round's Pareto front and M is the number of solutions in the previous round's Pareto front. The metric is useful for giving a relative indication of how much progress the Pareto front has made at each round. While the actual value of the metric does not have much meaning, the trend that it shows is useful for telling if regular progression into the whitespace is being made. The mean Pareto distance metric can be applied to both the objective space and the decision space. Examining the metric gives an indication of how much the design has changed for a resultant step in the objective space for the Pareto front.

Chapter 3

BENCHMARK CASES

In order to evaluate the whitespace exploration algorithm and develop metrics to observe it's behavior, a variety of test cases were developed. Whitespace exploration problems share many similarities with multi-objective optimization problems as both require some set of of input design variables, objectives and constraints along with a model that maps the decision space to the objective space. Also like multi-objective optimization problems, in order for them to be interesting, the objectives must be conflicting, otherwise the solution will not be a Pareto front but simply a single point will be found as the optimal solution. The difference between a multi-objective optimization problem and a whitespace exploration problem, for the purposes of this work, lies in the non-design variables. For a whitespace exploration problem to be interesting, it must also have a set of non-design variables that can be modified which will have an affect on the Pareto front. Another important aspect of these test cases is that they be computationally inexpensive to evaluate. In it's current state whitespace exploration still requires a significant number of model evaluations which would be harmful for promoting rapid development and experimentation. Lastly it is desirable that the models and their parameters make intuitive sense to the researcher. Whitespace exploration seeks to challenge assumptions made in a model and so it is beneficial to have an understanding beforehand what those assumptions are and how they would be expected to change to reach the desired goal. For all of these reasons, coming up with good test cases for whitespace exploration is a difficult task. The four cases that are presented here exemplify the previously outlined criteria for a good whitespace exploration model.

3.1 Circle

Circle is a very simple model that was developed expressly for testing whitespace exploration. The model got it's name because in the objective space, the set of possible designs give the shape of a circle. A constraint was also added which effectively slices off a piece of the circle. The non-design variables represent the centroid of the circle and so it makes intuitive sense that if we want to move the Pareto front towards the *desired point* then the center of the circle should then move in the direction of the of the *desired point*. The circle model does not relate to any engineering system but was instead created to be similar to the Rosenbrock Banana function for optimization; a simple mathematical problem that has just enough complexity to exercise the algorithm. The input and output variables for the model are summarized in Tables 3.1 - 3.4.

Table 3.1: Circle model objective variables

Objective Variables	
Objective	Description
f_1	Objective 1
f_2	Objective 2

Table 3.2: Circle model design variables

Design Variables	
Variable	Description
θ	Angle input
r	Distance input

The objective variables f_1 and f_2 can be calculated using Equations 3.1 and 3.2.

Table 3.3: Circle model constraint variables

Constraint Variables	
Variable	Description
c	Model constraint

Table 3.4: Circle model non-design variables

Non-Design Variables	
Variable	Description
x_c	Circle centroid x position i
y_c	Circle centroid y position

$$f_1 = r\cos(\theta) + x_c \tag{3.1}$$

$$f_2 = r\sin(\theta) + y_c \tag{3.2}$$

The constraint c can be evaluated with

$$c = r^2 - (f_1 - x_\phi)^2 - (f_2 - y_\phi)^2 \tag{3.3}$$

where

$$\phi = \arctan\left(\frac{y_c}{x_c}\right) \tag{3.4}$$

$$x_\phi = x_c - 2.0 r_{\max}\cos(\phi) \tag{3.5}$$

$$y_\phi = y_c - 0.5 r_{\max} \sin(\phi) \tag{3.6}$$

3.2 Disk Brake

The Disk Brake model is a simple engineering design problem involving the mechanical design of a disk brake. The model contains two objectives and was originally developed for testing multi-objective optimization algorithms and presented in [1] while the equations that were used for implementing the model were drawn from [16]. The physical configuration for the problem is shown in Figure 3.1.

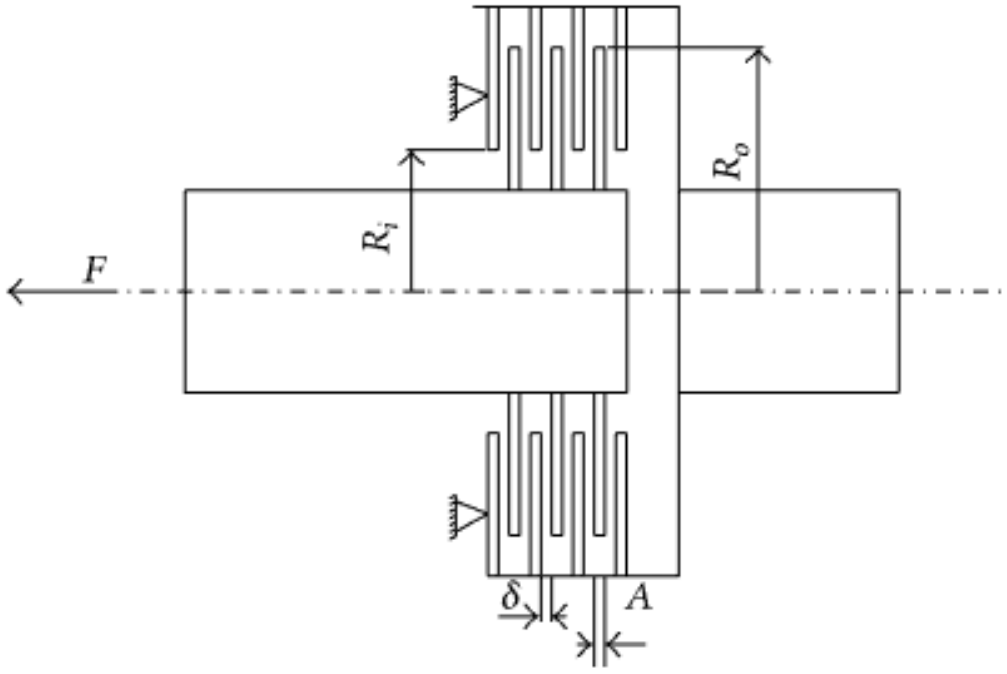

Figure 3.1: Diagram of the Disk Brake model [16]

The input and output variables that comprise the model are summarized in Tables 3.5 - 3.8.

The two objective values can be calculated with Equation 3.7

Table 3.5: Disk Brake model objective variables

Objective Variables	
Variabl	Description
m_b	The total mass of the brake
t_s	The brake stop time

Table 3.6: Disk Brake model design variables

Design Variables	
Variable	Description
R_i	inner radius of the discs
R_o	outer radius of the discs
F	engaging force
n	number of friction surfaces

Table 3.7: Disk Brake model constraint variables

Constraint Variables	
Variable	Description
g_1	minimum distance between radii
g_2	maximum length of the brake
g_3	pressure constraint
g_4	temperature constraint
g_5	generated torque constraint

$$m_b = m_f \left(R_o^2 - R_i^2\right)(n-1) \tag{3.7}$$

for the mass of the brake and Equation 3.8 for the stopping time,

Table 3.8: Disk Brake model non-design variables

Non-Design Variables	
Variable	Description
m_f	static mass factor
t_{sf}	static stop time factor

$$t_s = \frac{t_{sf}\left(R_o^2 - R_i^2\right)}{Fn\left(R_o^3 - R_i^3\right)} \tag{3.8}$$

The problem was modified from it's original form by adding the two non-design variable parameters m_f and t_{sf} which were originally specified as constant values in the equations. It is also of note that the design variable n for the number of friction surfaces is expected to be a discrete integer value but has been modified in it's implementation here to be a continuous input that is rounded to it's nearest integer value before calculation. The five output constraints for the model are defined as follows:

The geometric constraint for distance between radii is given by

$$g_1 = (R_o - R_i) - 20.0 \geq 0 \tag{3.9}$$

and the maximum length of the brake constraint by

$$g_2 = 30 - 2.5(n+1) \geq 0. \tag{3.10}$$

The pressure constraint is given by

$$g_3 = 0.4 - \frac{F}{3.14(R_o^2 - R_i^2)} \geq 0 \tag{3.11}$$

The temperature constraint is given by

$$g_4 = 1 - \frac{2.22 \times 10^{-3} F(R_o^3 - R_i^3)}{(R_o^2 - R_i^2)^2} \geq 0 \tag{3.12}$$

along with the generated torque constraint by

$$g_5 = \frac{2.66 \times 10^{-2} Fn(R_o^3 - R_i^3)}{(R_o^2 - R_i^2)} - 900 \geq 0 \tag{3.13}$$

3.3 Circle Packing

The Circle Packing problem is a straightforward model that represents a collection of five circles in 2D space with the goal of arranging them to minimize the principal moments of inerta about the origin of the system. Circle packing, and the packing problem in general is a longstanding problem in mathematics [17] that also has applications in engineering, related to the efficient layout of components in a design. This is still an active area of research [13]. While the problem is simple to visualize, it's discrete nature makes it difficult for optimization algorithms to solve and produces complex Pareto front shapes which make it an ideal candidate for testing whitespace exploration algorithms.

For this work the classic circle packing problem has been modified with the inclusion of non-design variables that the whitespace exploration algorithm can take advantage of. Each circle contains it's position in the x-y plane along with a radius and area density which are used in the inertia calculations. In order to prevent overlap of the circles an interference constraint is maintained which invalidates any arrangement that has overlapping circles.

The output objective variables are the principle moments inertia for each axis (x and y). In order to minimize the inertia of the system the circles must be arranged

into various "packed" configurations.

Table 3.9: Objective variables for the circle packing problem

Objective Variables	
Objective	Description
I_{xx}	Moment of inertia about the x-axis
I_{yy}	Moment of inertia about the y-axis

Table 3.10: Circle packing model design variables

Design Variables	
Variable	Description
x_i	x position of circle i
y_i	y position of circle i

Table 3.11: Circle packing constraint variables

Constraint Variables	
Variable	Description
I	The system interference between circles to prevent overlap

Table 3.12: Circle packing model non-design variables

Non-Design Variables	
Variable	Description
r_i	Radius of circle i
ρ_i	Area density of circle i

The objective values are calculated by evaluating the inertia about the origin for each circle and then summing up the inertia of each circle to obtain the inertia of the

system. The objectives are evaluated with the following relations give in Equation 3.14.

$$\begin{aligned} I_{xx} &= \sum_{i=1}^{n} \pi \rho_i r_i^2 \left(\frac{r_i^2}{4} + x_i^2 \right) \\ I_{yy} &= \sum_{i=1}^{n} \pi \rho_i r_i^2 \left(\frac{r_i^2}{4} + y_i^2 \right) \end{aligned} \tag{3.14}$$

The interference constraint, I, for the system is calculated by looking at the all pairs distances between each circle and checking if it is less than the sum of radii for the circles in question, if it is then we know that there is some degree of overlap. If there is no overlap than nothing is added to the systems overall interference value.

$$I = d_{ij} - (r_i + r_j) \tag{3.15}$$

where r_i and r_j are the radii of the circles i and j respectively and the distance between the two, d_{ij} is evaluated with

$$d_{ij} = \left[(x_j - x_i)^2 + (y_j - y_i)^2 \right]^{1/2}. \tag{3.16}$$

If the system has no interference than the solution will have an interference value of zero, if there is some interference in the system then it will have some negative real value which increases in magnitude the more interference there is.

3.4 Whipple Shield

The Whipple Shield model is an aerospace engineering design problem involving the mechanical design of the shield coupled with structural design constraints related to the launch environment. A whipple shield is a passive protection system for satellites

against micrometeorites and orbital debris and so the goal of the problem is to maximize the protection capabilities of the shield while keeping mass and volume as small as possible.

Table 3.13: Whipple shield model objective variables

Objective Variables	
Objective	Description
C_d	Critical diameter of the shield
m_s	Shield mass
V_s	Shield Volume

Table 3.14: Whipple shield model design variables

Design Variables	
Variable	Description
s	Spacing between bumper and wall
t_b	Bumper thickness
t_w	Wall thickness
$Æ R$	Support beam aspect ratio

Table 3.15: Whipple shield model constraint variables

Constraints	
Variable	Description
δ_b	Support beam deflection during launch
ω_n	Support beam natural frequency

The critical diameter, C_D, is calculated using empirical equations based on the relative impact velocity [2], for this model the worst case of a direct impact is assumed.

Table 3.16: Whipple shield non-design variables

Non-Design Variables	
Variable	Description
S_w	Shield Width
S_h	Shield Height
a_{max}	Max acceleration during launch
ρ_b	Density of the bumper material
ρ_w	Density of the wall material
σ_{max}	Yield strength of the wall material
v_{rel}	Relative velocity of implact
ρ_p	Density of the projectile material
E	Modulus of elasticity of the posts
ρ_s	Density of the support post material

A different relationship is sued depending on the relative velocity of the impact, for velocities less than 3 km/s

$$C_D = \left(\frac{t_w \sqrt{\sigma_{max}/40} + t_b}{0.6\sqrt{\rho_b + v_{rel}^{2/3}}} \right)^{18/19}, \tag{3.17}$$

for velocities between 3.0 km/s and 7.0 km/s

$$\begin{aligned} C_D = \left[\frac{t_w \sqrt{\frac{\sigma_{max}}{40}} + t_b}{1.248\sqrt{\rho_p}} \right]^{18/19} \times \left(1.75 - \frac{V_{rel}}{4} \right) + \\ \left[1.071 t_w^{2/3} \rho_p^{-1/3} \rho_b^{-1/9} S_{1/3} \left(\frac{\sigma}{70}^{1/3} \right) \right] \times \left(\frac{V_{rel}}{4} - 0.75 \right), \end{aligned} \tag{3.18}$$

for velocities greater than 7.0 km/s

$$C_D = 3.918 t_w^{2/3} \rho_p^{-1/3} \rho_b^{-1/9} v_{rel}^{-2/3} S_{1/3} (\sigma_{max}/70)^{1/3}. \tag{3.19}$$

The volume of the shield is calculated as

$$V_s = S_w S_h (t_b + S + t_w) \tag{3.20}$$

and the total mass of the shield as

$$m_s = \rho_b(S_w S_h t_b) + \rho_w(S_w S_h t_w) + \rho_s S\pi(S/AR)^2. \tag{3.21}$$

The whipple shield model has two constraints related to the launch environment which include the max deflection of the shield during launch and the natural frequency of the system. The deflection constraint δ_b is derived from standard strength of materials relationships and given by

$$\delta_b = a_{max}\left[E\pi\left(\frac{s}{2Æ}\right)^2\left(\frac{s\pi r^2\rho_s}{2} + S_w S_h t_b \rho_b\right)\right]^{-1} \tag{3.22}$$

while the natural frequency constraint ω_n is evaluated with

$$\omega_n = \frac{3E\pi r^4}{s^3\rho_b S_w S_h t_b + s^4\pi r^2\rho_s}. \tag{3.23}$$

Chapter 4

BENCHMARK MODEL RESULTS

For the analysis of the automated whitespace exploration algorithm implemented in the Thalia whitespace exploration framework, we collect the output of thirty runs of the algorithm where each run consists of ten *rounds* of exploration. The data from each case is then post processed to calculate various metrics and visualizations to examine how the exploration progressed. For each of the models, an in depth look at a single exploration run is reviewed along with the results of the thirty batch runs.

In the results we are looking to see that they make sense in the context of the model, e.g. the correct variables are chosen for the direction search. Through the post processed output we are also looking to see what the results of the exploration can tell us about the problem; were there aspects of it that made it difficult for the exploration to progress? was there a high variance in the results over the thirty runs?

4.1 Circle Model Results

The variable configuration specified for the exploration is summarized in Tables 4.1 - 4.4 and gives the initial bounds that were set where appropriate. For this case the design variable θ has it's bounds fixed as modifying them does not make sense in the context of the model. The variable r is free to have it's bounds modified by the algorithm should it be found to be an active box constraint. The individual design variable box bound, constraint variable bound, and non-design variable change histories are given in Appendix A.1 for this case.

The configuration values for each of the exploration steps is summarized in tables 4.5 - 4.9.

Table 4.1: Circle model exploration objective variables

Objective Variable Configuration	
Variable	Goal
f_1	Minimize
f_2	Minimize

Table 4.2: Circle model exploration design variables

Design Variable Configuration		
Variable	Lower Bound	Upper Bound
θ	0.0	2π
r	0.0	1.0

Table 4.3: Circle model exploration constraint variables

Constraint Variable Configuration		
Variable	Lower Bound	Upper Bound
c	N/A	0.0

Table 4.4: Circle model exploration non-design variables

Non-Design Variable Configuration		
Variable	Lower Bound	Upper Bound
x_c	1.0	6.0
y_c	1.0	6.0

Table 4.5: Pareto search configuration for the Circle model

Pareto Search Configuration	
Population Size	40
Max Evaluations	12000
Convergence Tolerance	0.01
Convergence Generations	10
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.5
Mutation Index	20.0

Table 4.6: Constraint search configuration for the Circle model

Constraint Search Configuration	
Box Constraint Relative Tolerance	0.05
Box Constraint Change Tolerance	0.05
Box Constraint Change Percentage	0.05
Constraint Relative Tolerance	0.05
Constraint Change Tolerance	0.05
Constraint Change Percentage	0.05

Table 4.7: Sensitivity analysis configuration for the Circle model

Sensitivity Analysis Configuration	
Stencil Size	0.01
Selection Tolerance	0.1

Table 4.8: Direction search configuration for the Circle case

Direction Search Configuration	
Population Size	20
Max Evaluations	2000
Convergence Tolerance	0.01
Convergence Generations	5
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.5
Mutation Index	20.0

Table 4.9: Exploration search configuration for the Circle model

Exploration Search Configuration	
Max Evaluations	100
Search Tolerance	0.01
Exploration Distance	1.0

The Pareto search results showing the design space at the start of the exploration for the Circle model are given in Figure 4.1 with the corresponding input decision space shown in Figure 4.2. In Figure 4.1 we can see the circular design space where the model gets it's name from, with the upper portion of the Pareto front active on the constraint and the lower portion active on the box constraints. The convergence history for the Pareto search shown in Figure 4.3 we see that a bulk of the Pareto front was found very quickly however the constraint caused some issues for the algorithm to get proper resolution around the constrained area as shown by the gaps in the Pareto front in Figure 4.2. The selected point shown on the Pareto front in Figure

4.1 was found using the automated procedure which sought the closest point on the Pareto front to the desired point which happens to be on the 'knee' of the front.

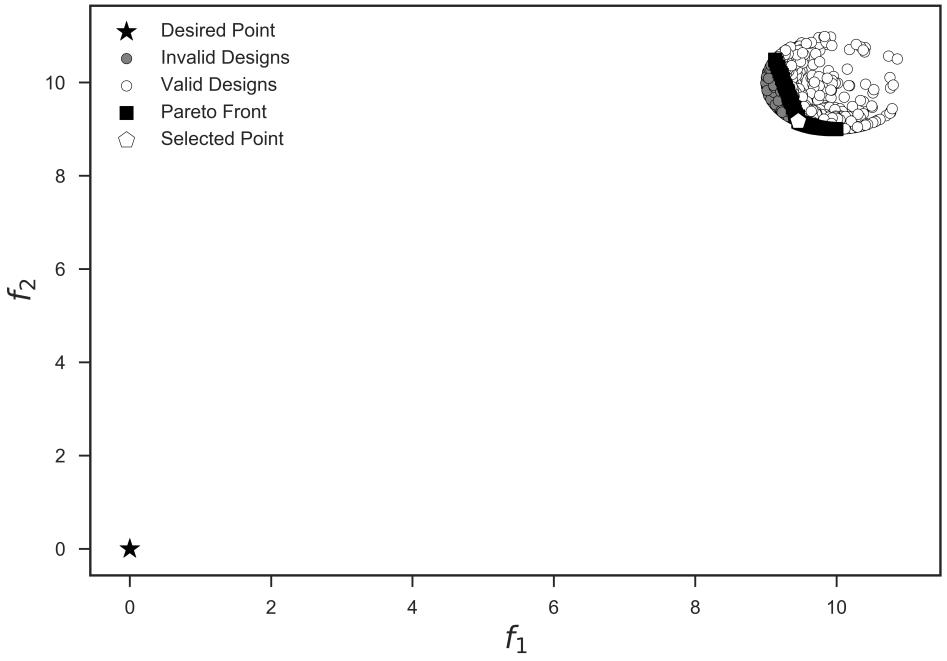

Figure 4.1: Initial design space for the Circle model

In the first round, we find that the box constraint upper bound for the variable r is found to be active. The upper bound of the output constraint is also active. Visual inspection of the Pareto front leads us to expect a near even split on the percentage of Pareto front points active on either constraint, which is reported by the algorithm. The upper box constraint has 64.03 percent of its solutions on the active boundary and 43.17 percent on the active output constraint boundary. Both are of a significant enough percentage to be flagged by the algorithm for relaxation in the next round.

The sensitivity analysis results are summarized in Table 4.12 and show what was expected for this model, the first objective f_1 only has dependence on the non-design input x_c and is correctly found to have a sensitivity of one, the same is true for objective f_2 and the non-design input x_y.

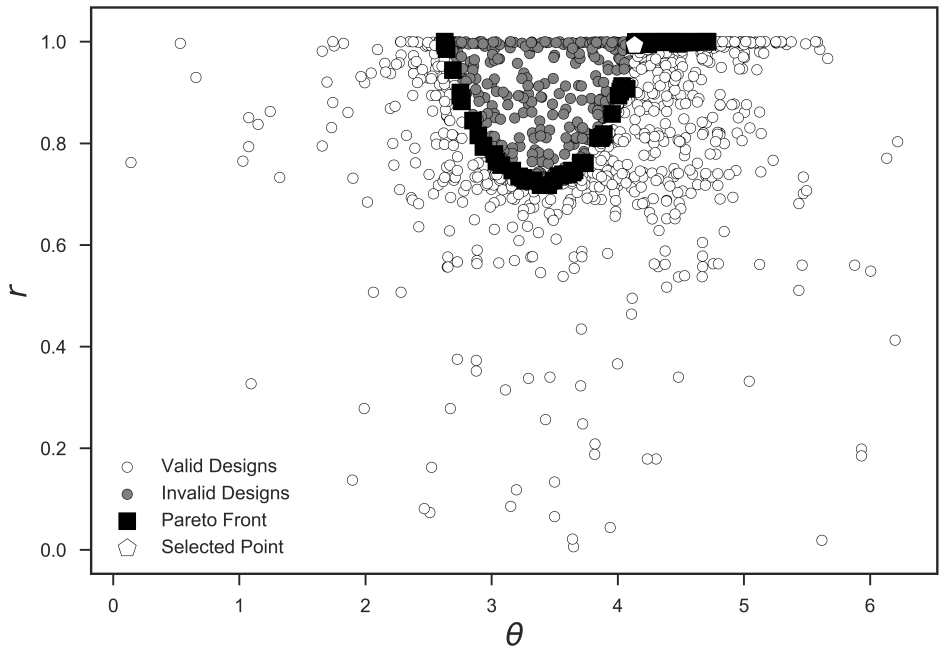

Figure 4.2: Initial decision space for the Circle model

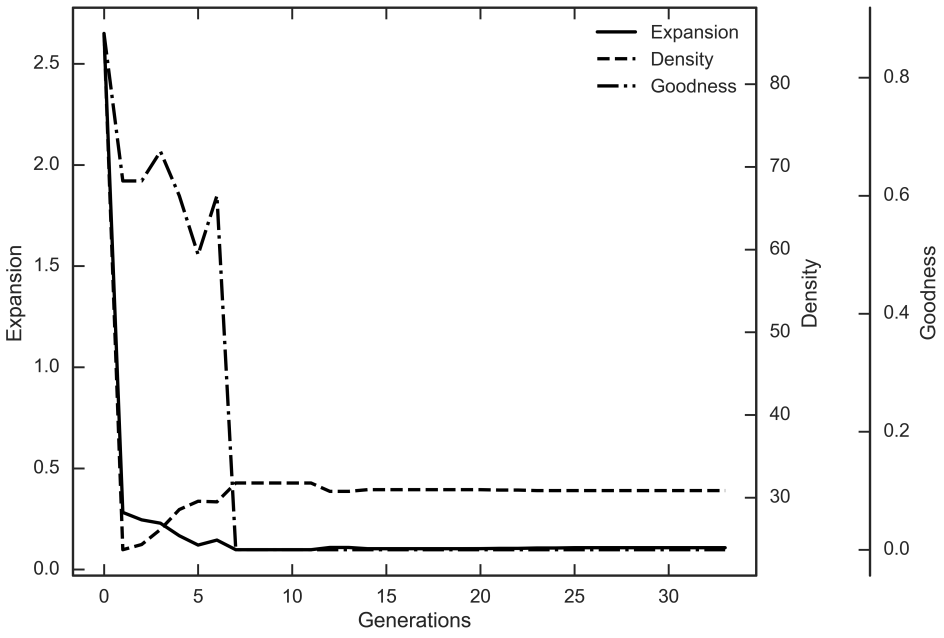

Figure 4.3: Initial Pareto search convergence for the Circle case

Table 4.10: First round box constraint activity for the Circle model

Box Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
r	0.0	INACTIVE	64.03	ACTIVE

Table 4.11: First round constraint activity for the Circle model

Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
c	0.0	INACTIVE	43.17	ACTIVE

Table 4.12: The initial sensitivities for the Circle model

Sensitivities		
Output	Input	Sensitivity
f_1	x_c	1.0
f_1	y_c	0.0
f_2	x_c	0.0
f_2	y_c	1.0

The direction search step for the first round is shown in Figure 4.4 in the objective space. The search is centered at the selected point and visual inspection shows that a variety of points are found that are in line with the desired point.

The exploration search step history for the first round is shown in Figure 4.5 and shows the objective continues to decrease out to the set maximum exploration parameter value of one.

Figure 4.6 shows the results of the Pareto search step on the tenth and final round.

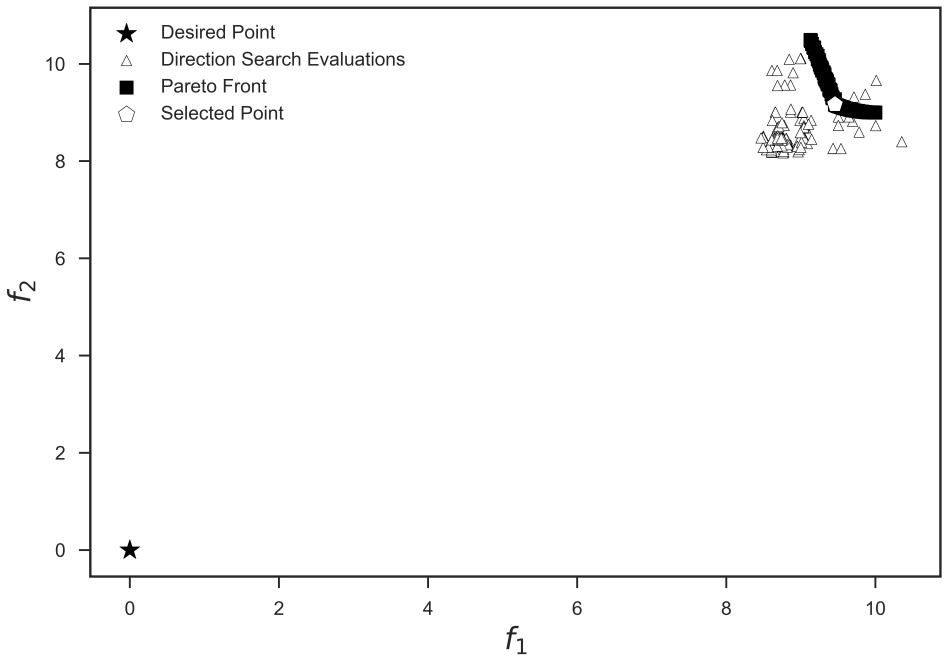

Figure 4.4: Initial direction search for the Circle model

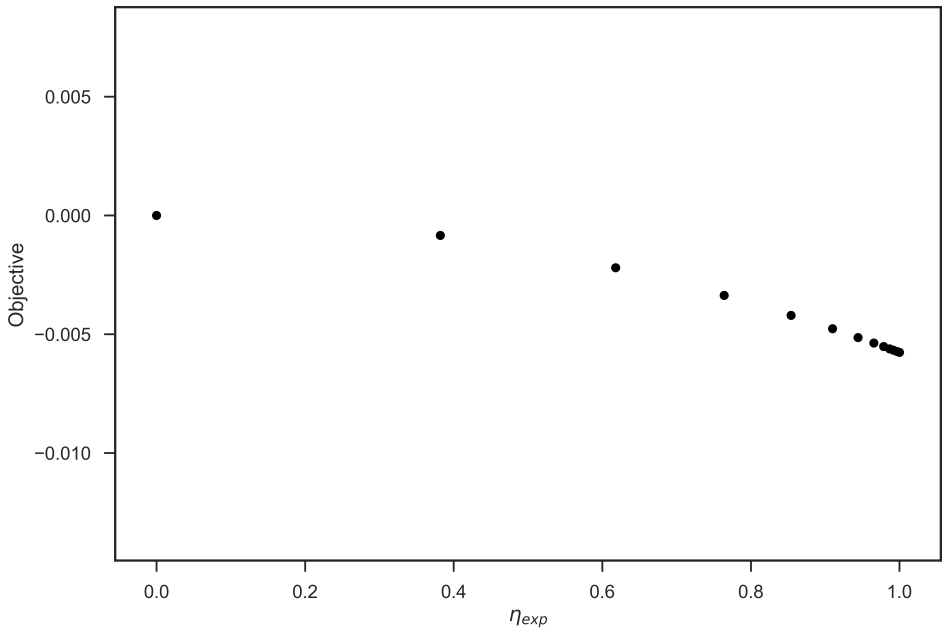

Figure 4.5: Initial exploration search convergence for the Circle model

We can see that the Pareto front is now much closer to the desired point and has grown in size due to the upper bound on the design variable r being relaxed. The convergence history for the Pareto search shown in Figure 4.8 shows that in the final round had more difficulty time converging on a front, and that there still exist some gaps in the front likely due to the larger search area.

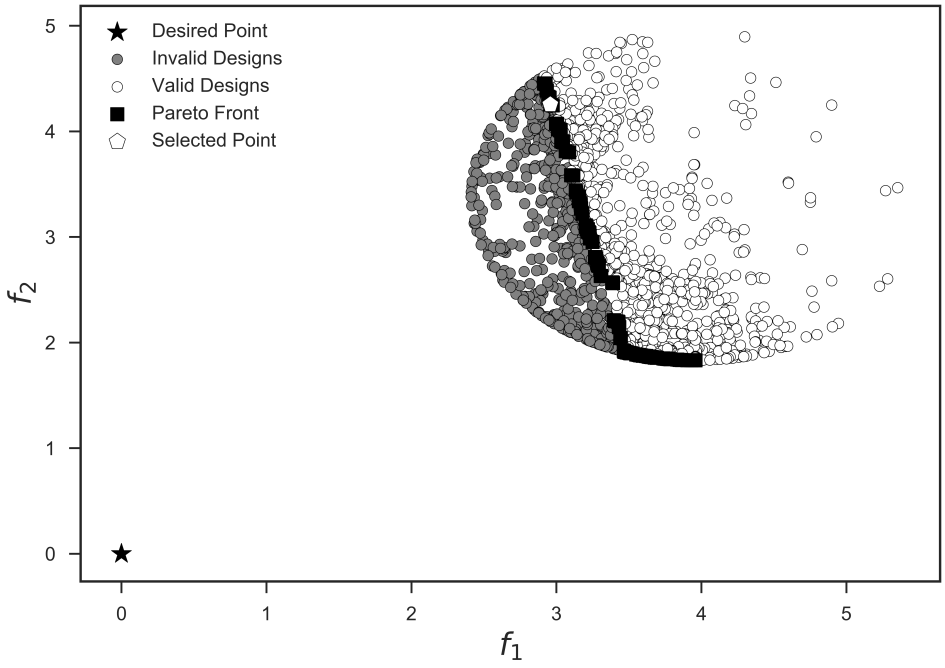

Figure 4.6: Final exploration search convergence for the Circle model

Table 4.13 shows the box constraint activity for the final round, and we see that the upper bound on the design variable r is still active for a significant portion of the Pareto front. The output constraint, c, has decreased in activity significantly from the first round as shown in Table 4.14.

The sensitivities at the final round are just the same as they were in the first round as shown in Table 4.15, which is expected since nothing in the model would cause the effect of the non-design variables to change with respect to the objectives.

In the direction search, we notice that the selected point has moved up the Pareto

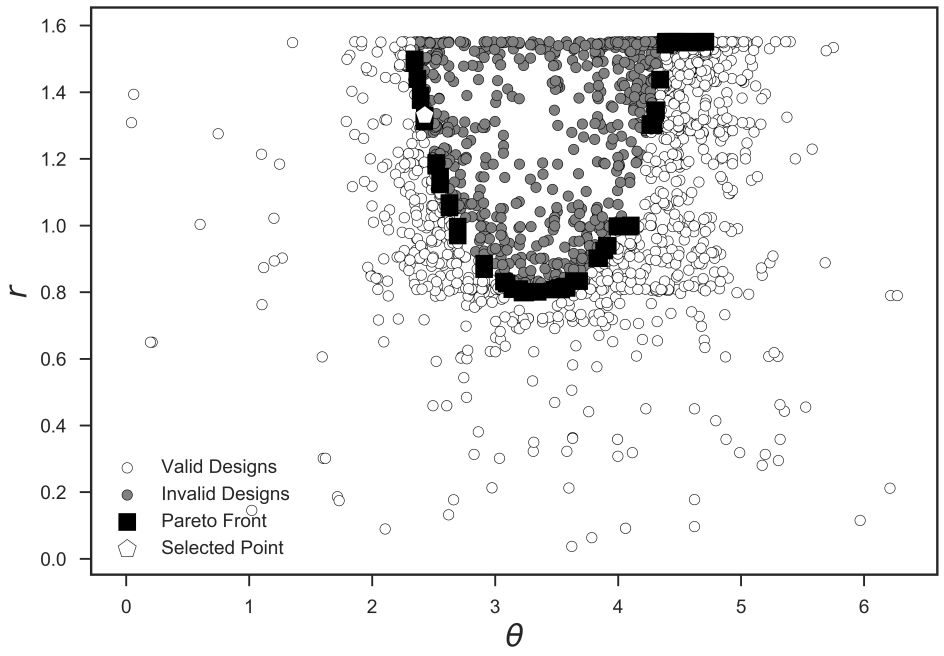

Figure 4.7: Final decision space for the Circle model

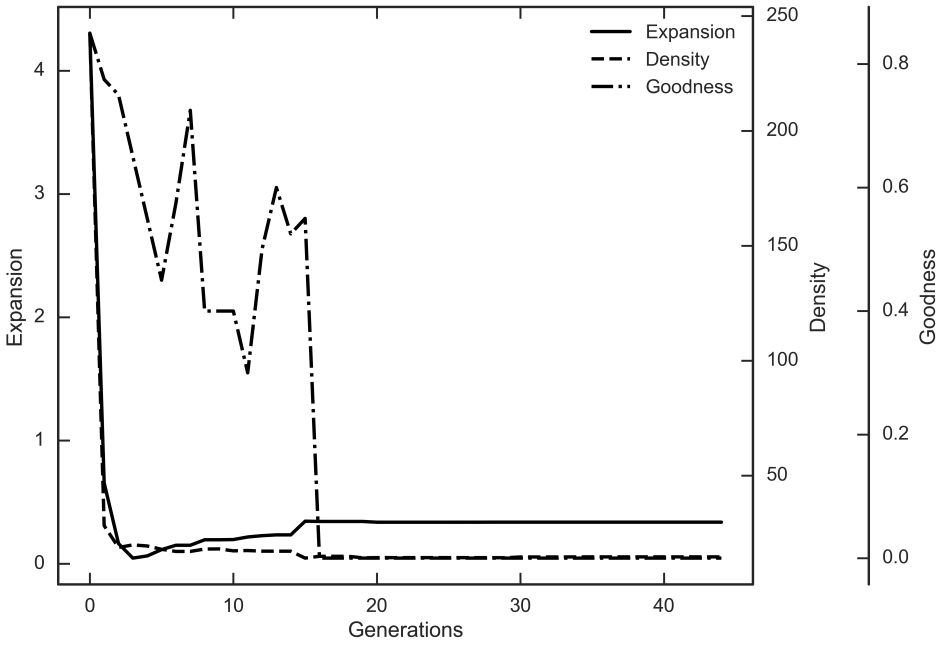

Figure 4.8: Final Pareto search convergence for the Circle model

Table 4.13: Final box constraint activity for the Circle model

Box Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
r	0.0	INACTIVE	44.33	ACTIVE

Table 4.14: Initial constraint activity for the Circle model

Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
c	0.0	INACTIVE	7.5	ACTIVE

Table 4.15: Final sensitivities for the Circle model

Sensitivities		
Output	Input	Sensitivity
f_1	x_c	1.0
f_1	y_c	0.0
f_2	x_c	0.0
f_2	y_c	1.0

front due to the Front size changing. The direction search is still able to find a dot product of one and has a direct line towards the desired point as shown in Figure 4.9.

Looking at the exploration as a whole, Figure 4.11 shows the Pareto front progression for each round the exploration. The figure shows a very regular progression of the Pareto front towards the desired point with the front increasing in size along the way due to constraint and box-constraint relaxation.

The hypervolume progression of the Pareto front as shown in Figure 4.12 tells

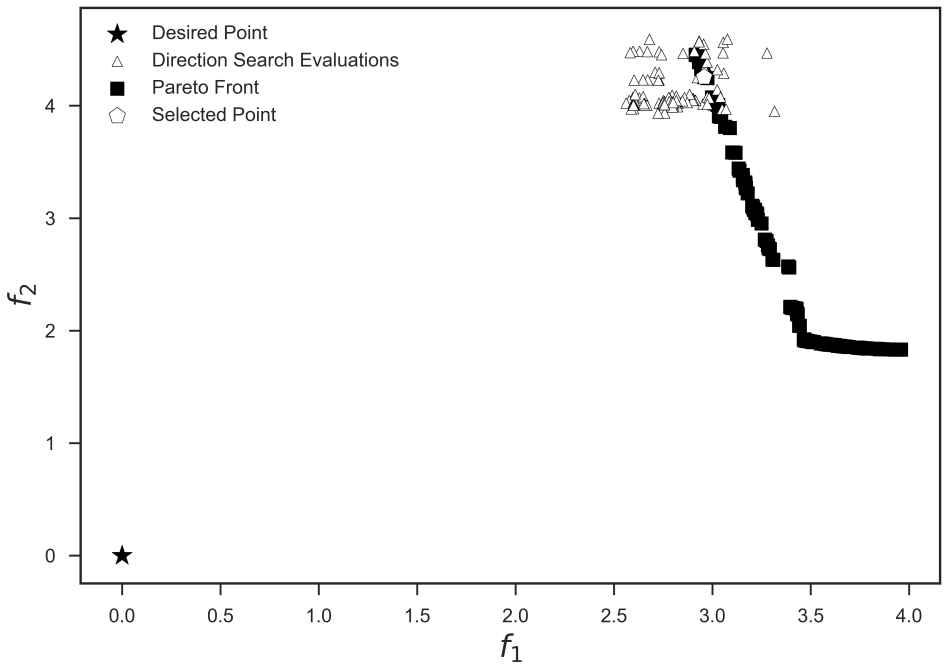

Figure 4.9: Final direction search for the Circle model

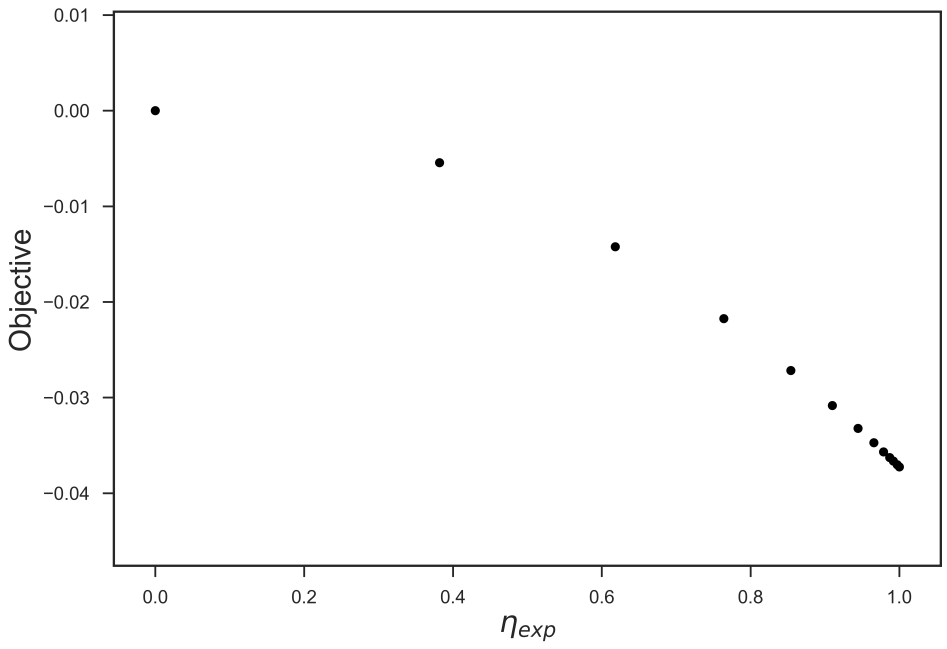

Figure 4.10: Final exploration search convergence for the Circle model

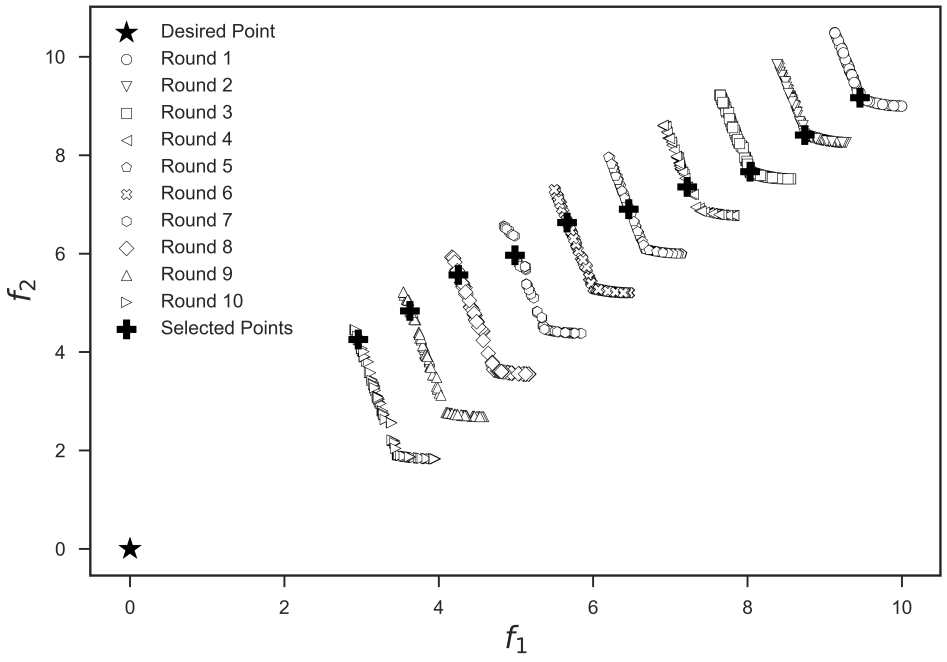

Figure 4.11: Pareto front progression for the Circle model exploration

a similar tale to the Pareto front progression plot in Figure 4.11 where we have a regular march towards the desired point, thus decreasing the hypervolume. The slope of the curve deceases slightly as the rounds progress due to the Pareto front size itself increasing because of constraint and box constraint relaxation.

The hypervolume progression for the batch runs are shown in Figure 4.13. The min and max values for the hypervolume at each round are nearly indistinguishable from the mean line which gives high confidence that any further explorations on the model with this configuration will result in nearly identical behavior. For the simple circle model this is to be expected as there are no other pathways that could be found through the random processes in the algorithm. Any variance over the set of batch runs is likely due to the Pareto search step not fully resolving the endpoints of the Pareto front.

The mean Pareto front distance progression graphs for both the objective space

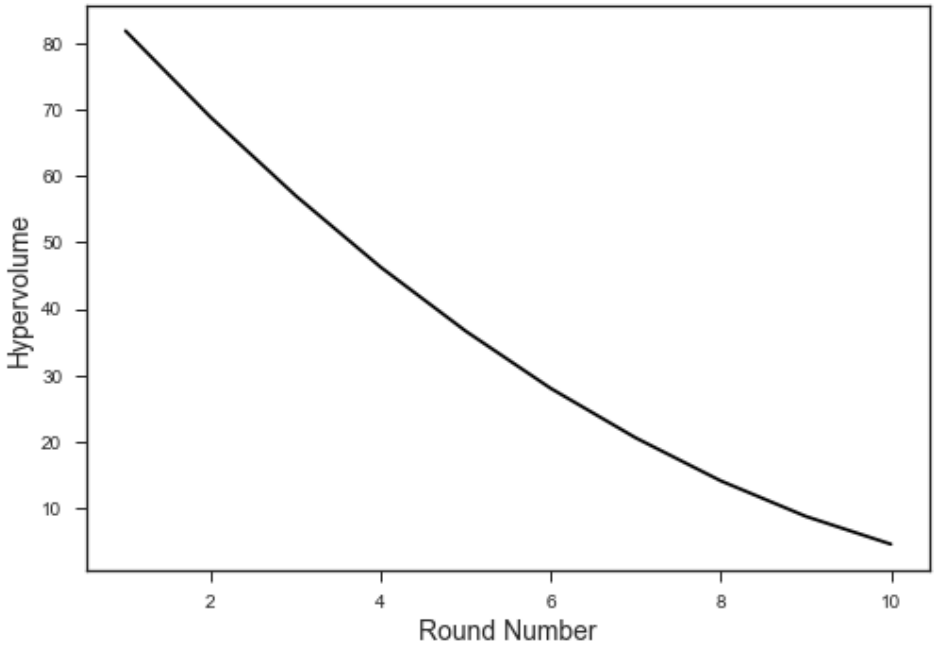

Figure 4.12: Hypervolume progression for the Circle model exploration

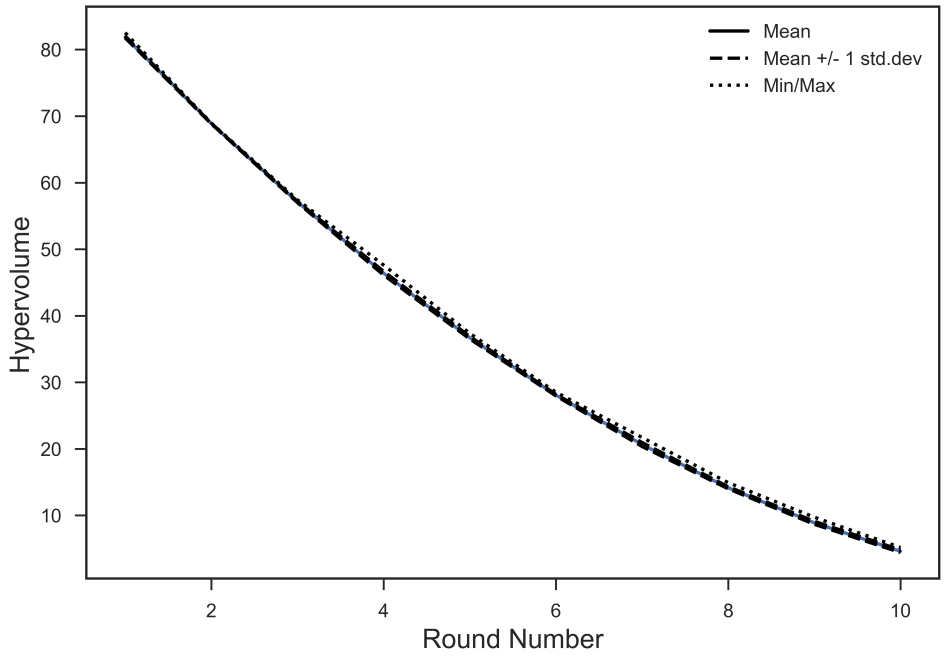

Figure 4.13: Hypervolume progression for the Circle model batch runs

and the decision space are shown in Figure 4.14 and Figure 4.15 respectively. For the Pareto front in the objective space, it is clear that as the exploration progressed the distance between successive Pareto fronts decreased slightly. In the decision space, the Pareto front distances remained relatively constant The reason for the Pareto front not making the same progress each round could be due to the constraint relaxation not 'keeping up' with the non-design variable changes.

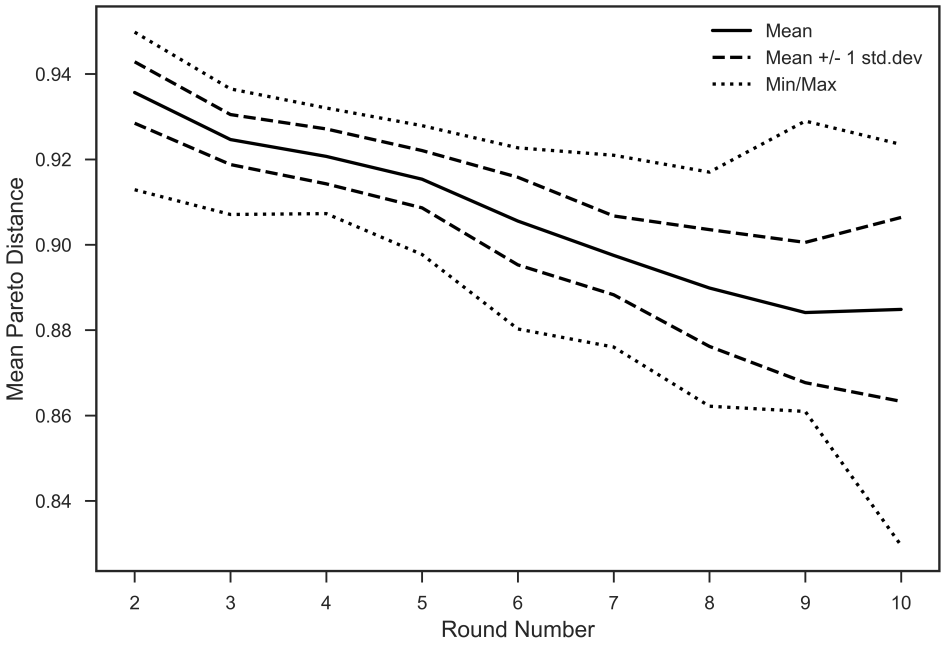

Figure 4.14: Mean Pareto front distance progression for the Circle model batch run in objective space

The mean function evaluations for the batch runs shown in Figure 4.16 had an average of 2000 evaluations per round but varied between 1000 and 3000 evaluations per round which is significant and could be too computationally expensive to run for all but the fastest models.

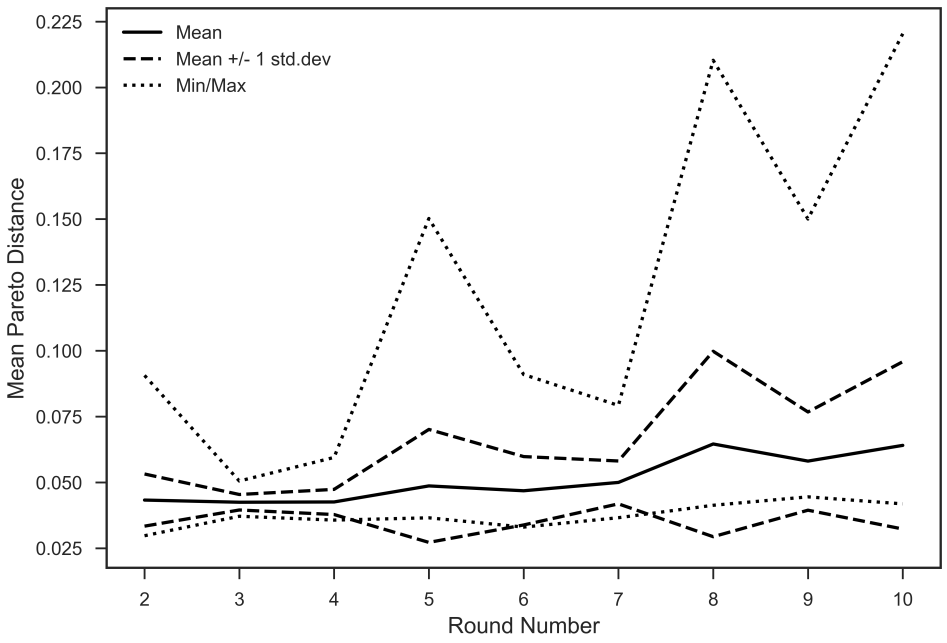

Figure 4.15: Mean Pareto front distance progression for the Circle model batch run in decision space

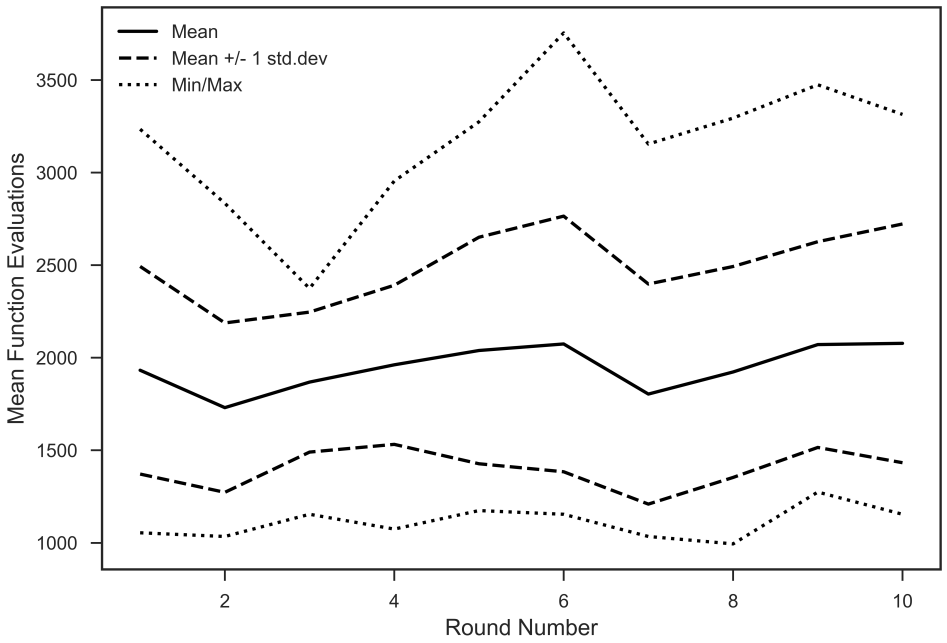

Figure 4.16: Mean function evaluations for the Circle model batch runs

4.2 Disk Brake Model Results

The results for the disk brake model have many similarities to the Circle model discussed previously. The exploration is set up to minimize the two objectives, brake mass and stop time of the brake. The disk brake exploration has four design variables, three of which were allowed to have their bounds modified by the whitespace algorithm; inner radius, outer radius, and engaging force. The number of friction surfaces design variable, n, was not allowed to vary. The individual design variable box bound, constraint variable bound, and non-design variable change histories are given in Appendix A.2 for this case.

Table 4.16: Disk brake model objective variable setup

Objective Variable Configuration	
Variable	Goal
m_b	MINIMIZE
t_s	MINIMIZE

Table 4.17: Disk brake model design variable setup

Design Variable Configuration		
Variable	Lower Bound	Upper Bound
R_i	55.0	80.0
R_o	75.0	110.0
F	1000.0	3000.0
n	2.0	20.0

Of the five model constraints, the three that related to material properties were allowed to be modified, g_3, g_4, and g_5, while the constraints related to the physical

form of the system remained constant.

Table 4.18: Disk brake model constraint variable setup

Constraint Variable Configuration		
Variable	Lower Bound	Upper Bound
g_1	0.0	N/A
g_2	0.0	N/A
g_3	0.0	N/A
g_4	0.0	N/A
g_5	0.0	N/A

Table 4.19: Disk Brake model exploration variable configuration

Non-Design Variable Configuration		
Variable	Lower Bound	Upper Bound
m_f	1e-5	1e-3
t_{sf}	1e6	1e8

The Pareto search configuration was set to suggested values in Thalia, with a population size set to twenty times the number of design variables and a maximum number of evaluations which would allow for three-hundred generation of the search algorithm which should be sufficient for proper resolution of the Pareto front.

The results of the first round Pareto search for the single run analysis is shown in Figure 4.17. We can see from the figure that a very regular Pareto front is found with many of the solutions found being near the Pareto region. The constraint space also appears to lie 'in front' of the Pareto front.

The constraint search results summarized in Table 4.25 and Table 4.26 show that nearly the entire Pareto front is active on the upper bound box constraint of the

Table 4.20: Pareto search configuration for the Disk Brake model

Pareto Search Configuration	
Population Size	80
Max Evaluations	24000
Convergence Tolerance	0.01
Convergence Generations	10
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.25
Mutation Index	20.0

Table 4.21: Constraint search configuration for the Disk Brake model

Constraint Search Configuration	
Box Constraint Relative Tolerance	0.05
Box Constraint Change Tolerance	0.05
Box Constraint Change Percentage	0.05
Constraint Relative Tolerance	0.05
Constraint Change Tolerance	0.05
Constraint Change Percentage	0.05

Table 4.22: Sensitivity analysis configuration for the Disk Brake model

Sensitivity Analysis Configuration	
Stencil Size	0.01
Selection Tolerance	0.1

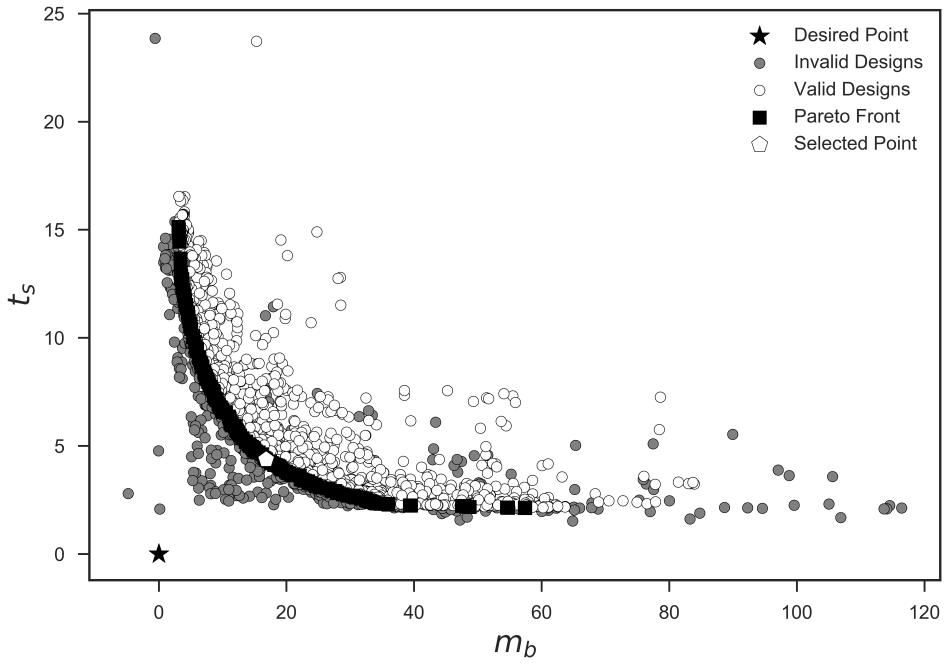

Figure 4.17: Initial design space for the Disk Brake model

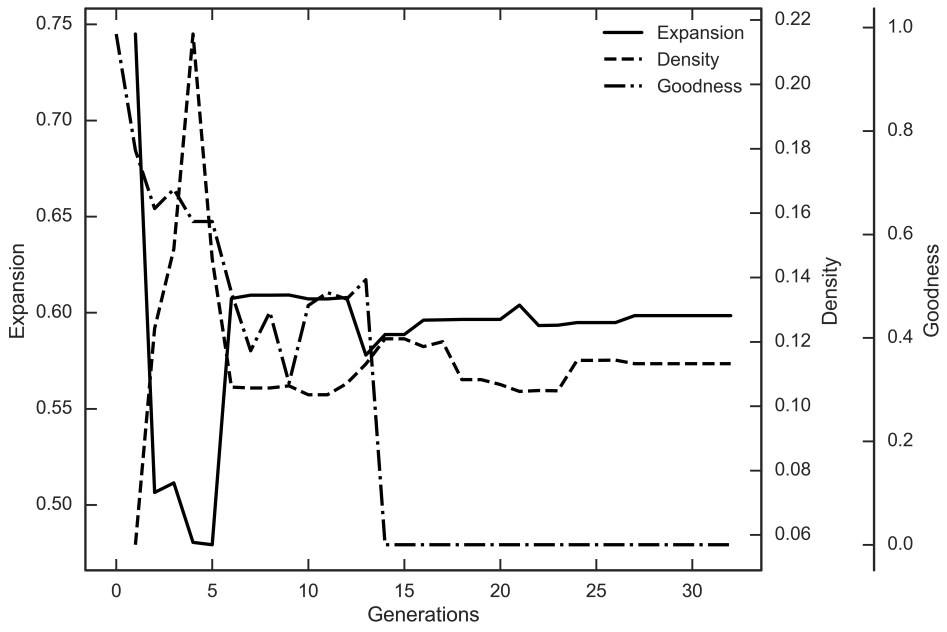

Figure 4.18: Initial Pareto search convergence for the Disk Brake model

Table 4.23: Direction search configuration for the Disk Brake model

Direction Search Configuration	
Population Size	20
Max Evaluations	2000
Convergence Tolerance	0.01
Convergence Generations	5
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.5
Mutation Index	20.0

Table 4.24: Exploration search configuration for the Disk brake model

Exploration Search Configuration	
Max Evaluations	100
Search Tolerance	0.01
Search Distance	10.0

engaging force design variable.

Table 4.25: Initial box constraint activity for the Disk Brake model

Box Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
R_i	0.0	INACTIVE	6.47	ACTIVE
R_o	0.0	INACTIVE	1.99	INACTIVE
F	0.0	INACTIVE	99.50	ACTIVE

Like the Circle model, each non-design variable has a direct relationship to their

Table 4.26: Initial constraint activity for the Disk Brake model

Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
g_3	0.0	INACTIVE	0.0	INACTIVE
g_4	0.0	INACTIVE	0.0	INACTIVE
g_5	0.0	INACTIVE	0.0	INACTIVE

respective objective variables which was found during the sensitivity analysis step and summarized in Table 4.27.

Table 4.27: Initial sensitivities for the Disk Brake model

Sensitivities		
Output	Input	Sensitivity
m_b	t_{sf}	0.0
m_b	m_f	1.0
t_s	t_{sf}	1.0
t_s	m_f	0.0

Given the direct relationship of the non-design variables on the objectives, the direction search is able to find a direction vector in line with the desired point.

In the final round, the Pareto search does indeed find a Pareto front that is closer to the desired point than the one found initially and interestingly many invalid solutions that are non-physical with zero or negative mass for the system as shown in Figures 4.21 and 4.22.

At the end of the exploration, the engaging force upper bound box constraint is still the primary constraint interacting with the Pareto front, however the other design variables have also increased significantly in activity.

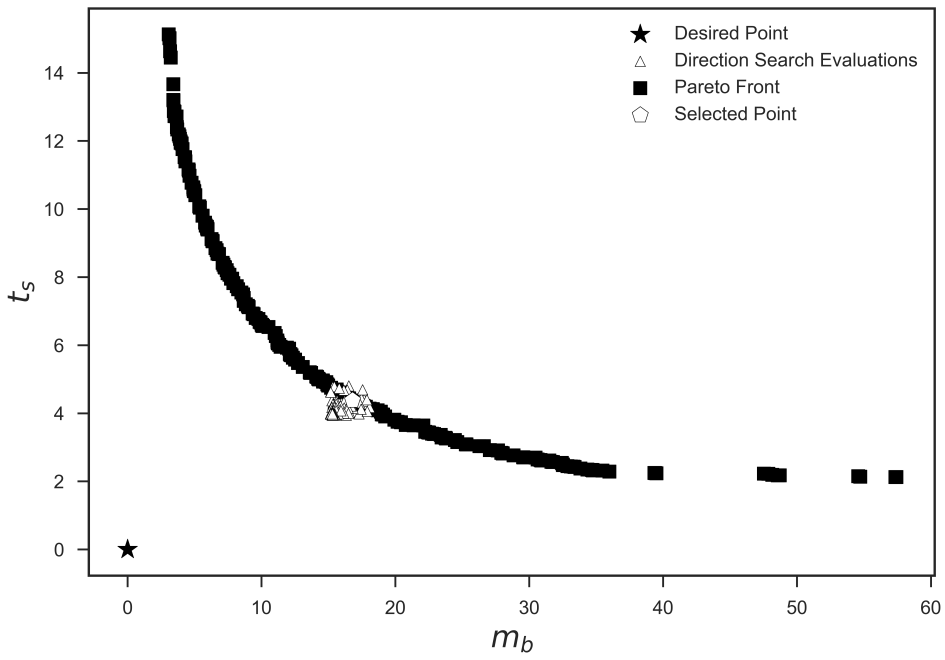

Figure 4.19: Initial direction search for the Disk Brake model

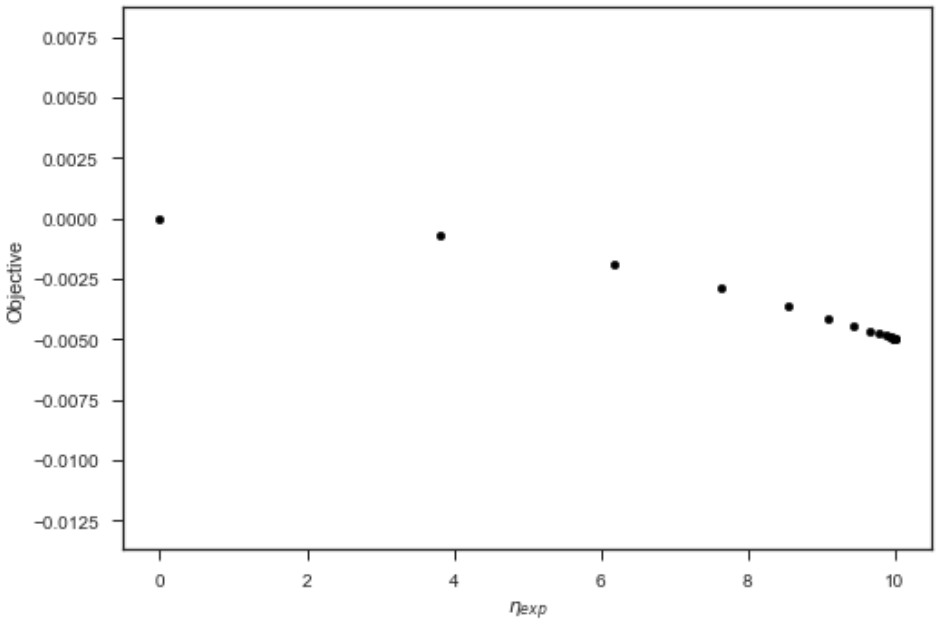

Figure 4.20: Initial exploration search convergence for the Disk Brake model

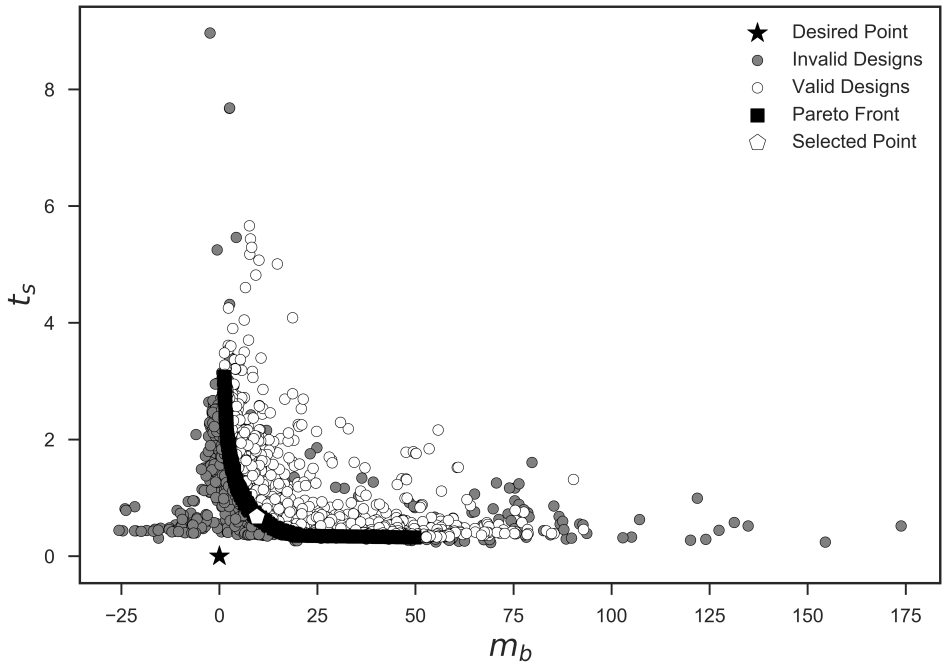

Figure 4.21: Final design space for the Disk Brake model

Table 4.28: Final box constraint activity for the Disk Brake model

Box Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
R_i	0.0	INACTIVE	19.79	ACTIVE
R_o	0.0	INACTIVE	3.74	ACTIVE
F	0.0	INACTIVE	98.93	ACTIVE

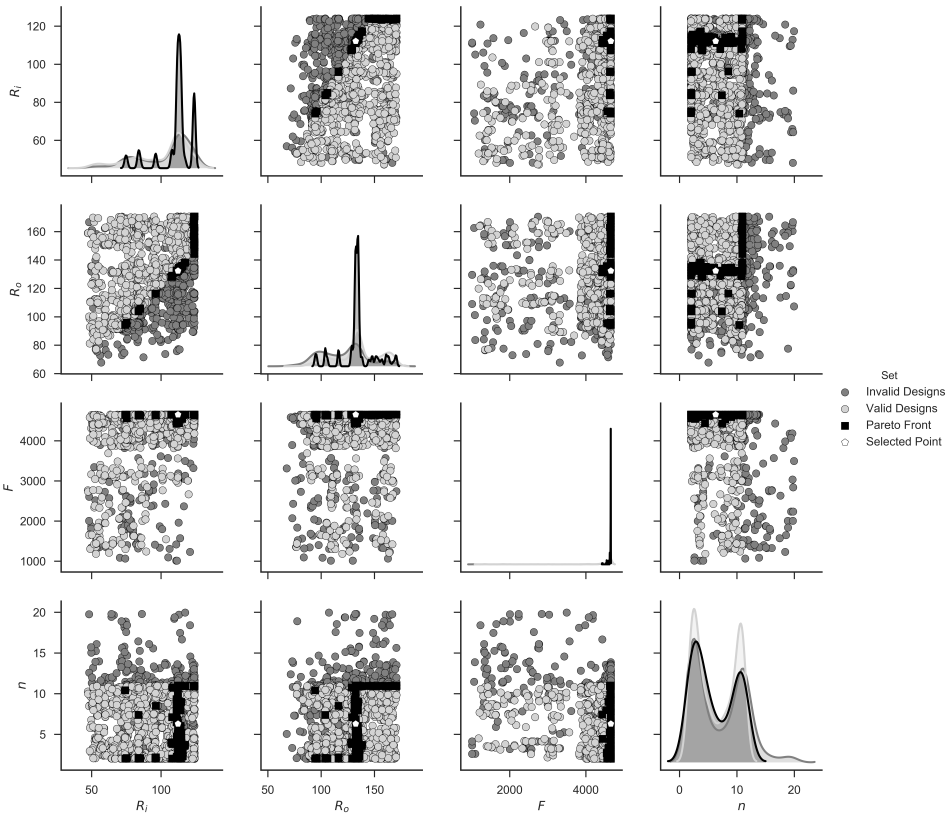

Figure 4.22: Final decision space for the Disk Brake model

The output constraint activity in the last round is zero for all of the specified constraints as shown in Table 4.29 however if we look at the lower bound history for the output constraint g_3 shown in Table 4.30 we can see that during rounds 2, 7, and 9 that the constraint was active and relaxed. Although g_3 did experience some activity, g_4 and g_5 were not active during the duration of the exploration.

In the final round, the sensitivity values in Table 4.31 are the same as those found in the first round which is as expected

Figure 4.26 shows the Pareto front progression for the entire exploration. Similar to the circle model, we have a regular progression of the Pareto front with some change in size to to constraint relaxation.

Figure 4.27 shows how the hypervolume of the Pareto front evolved during the

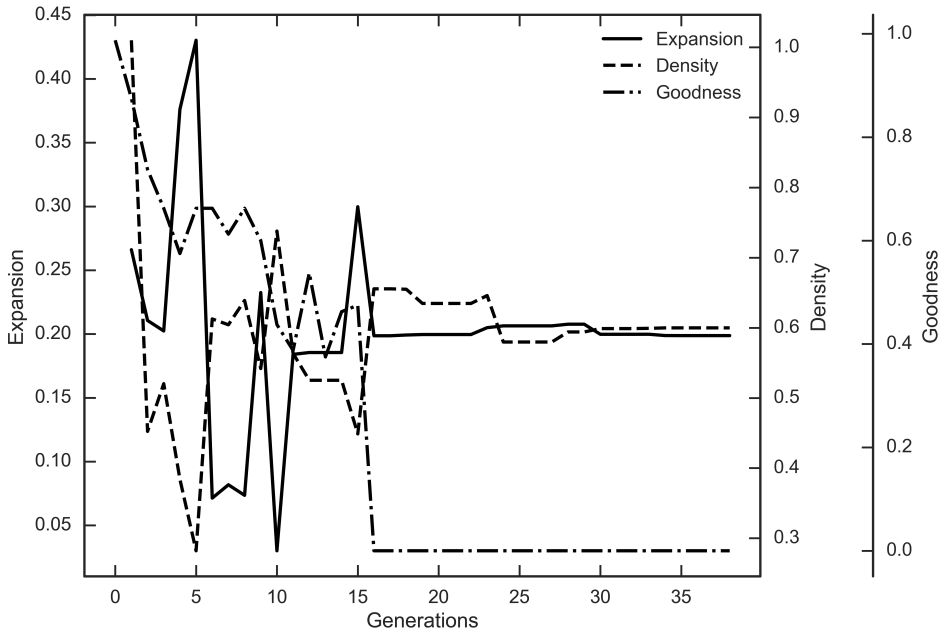

Figure 4.23: Final Pareto search convergence for the Disk Brake model

Table 4.29: Final constraint activity for the Disk Brake model

Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
g_3	0.0	INACTIVE	0.0	INACTIVE
g_4	0.0	INACTIVE	0.0	INACTIVE
g_5	0.0	INACTIVE	0.0	INACTIVE

Table 4.30: Lower bound history for g_3 constraint

g_3 Lower Bound History	
Round	Lower Bound
1	0.0
2	-0.05
3	-0.05
4	-0.05
5	-0.05
6	-0.05
7	-0.0525
8	-0.0525
9	-0.0551
10	-0.0551

Table 4.31: Final sensitivities for the Disk Brake model

Sensitivities		
Output	Input	Sensitivity
m_b	t_{sf}	0.0
m_b	m_f	1.0
t_s	t_{sf}	1.0
t_s	m_f	0.0

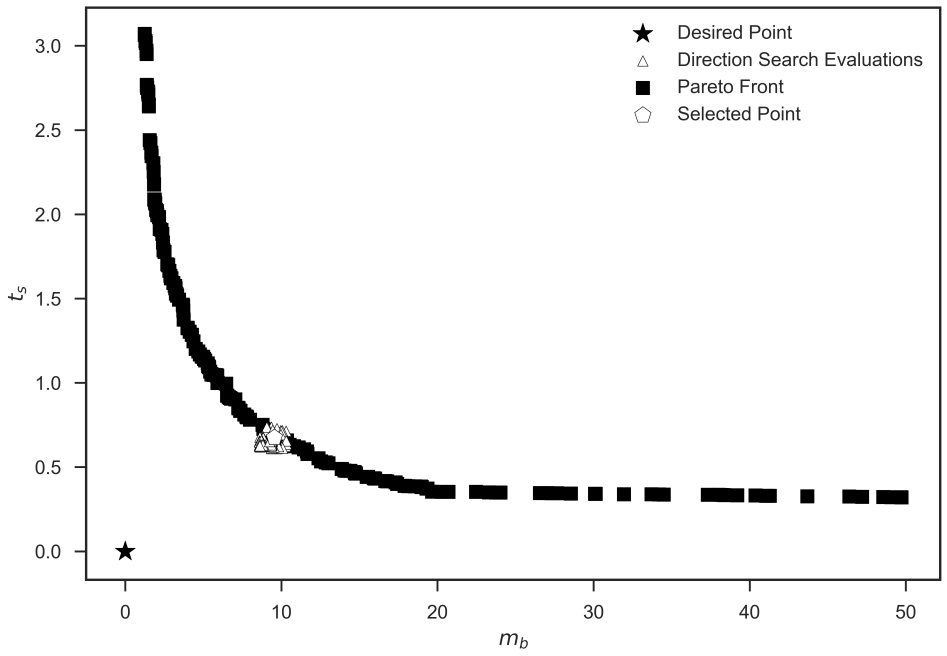

Figure 4.24: Final direction search for the Disk Brake model

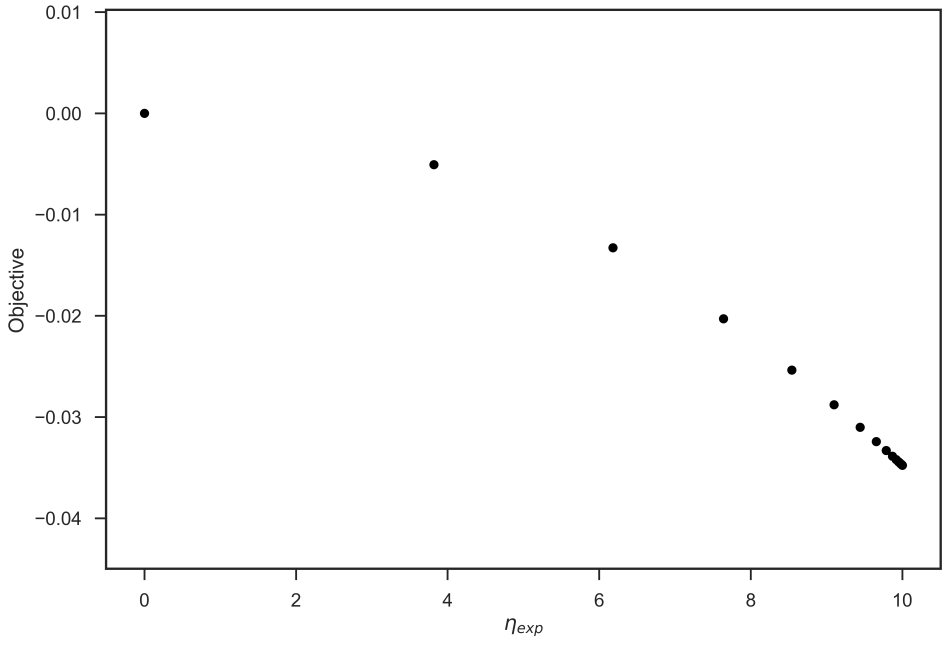

Figure 4.25: Final exploration search convergence for the Disk Brake model

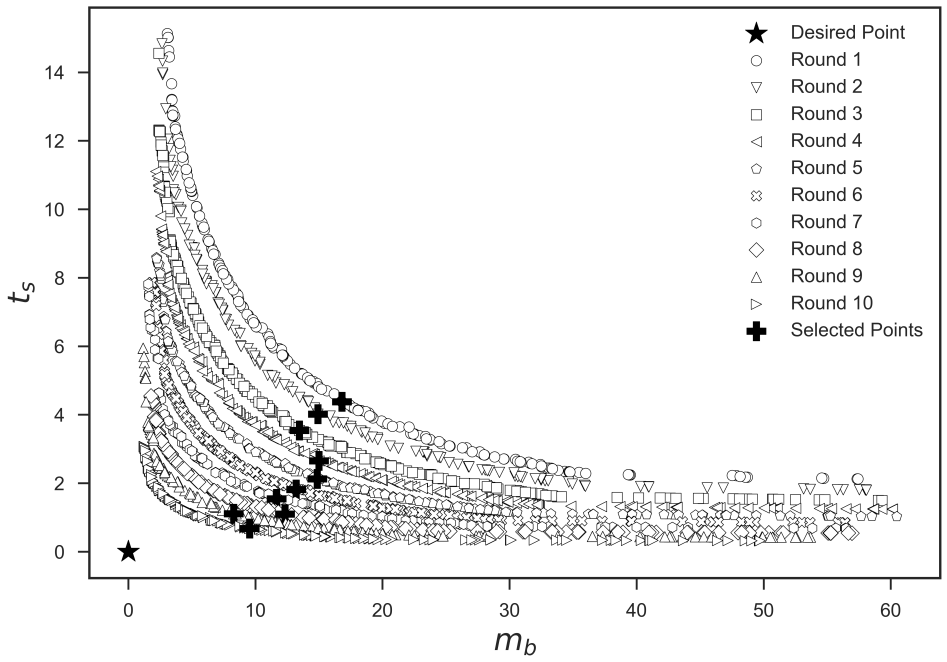

Figure 4.26: Pareto front progression for the Disk Brake model exploration

exploration. Similar to the Circle model, the hypervolume shows a regular trend towards a reduced volume of whitespace.

The batch run hypervolume progression begins to show some variability across the thirty runs, this is likey due to the endpoints of the Pareto front being difficult to resolve by the Pareto search algorithm.

The Pareto progression results given in Figures 4.29 and 4.30 show that as the exploration progresses the distances between Pareto fronts in the objective space tend to get larger, while in the decision space the tend to get smaller. In this case the data appears to be misleading as looking at the Pareto front progression in Figure 4.26 shows the gap between fronts decreases at each round. The increasing Pareto distance metric is likely due to the Pareto front 'shrinking' slightly faster than it progresses causing the distance between fronts to increase.

The mean function evaluations for the disk brake model shown in Figure 4.31

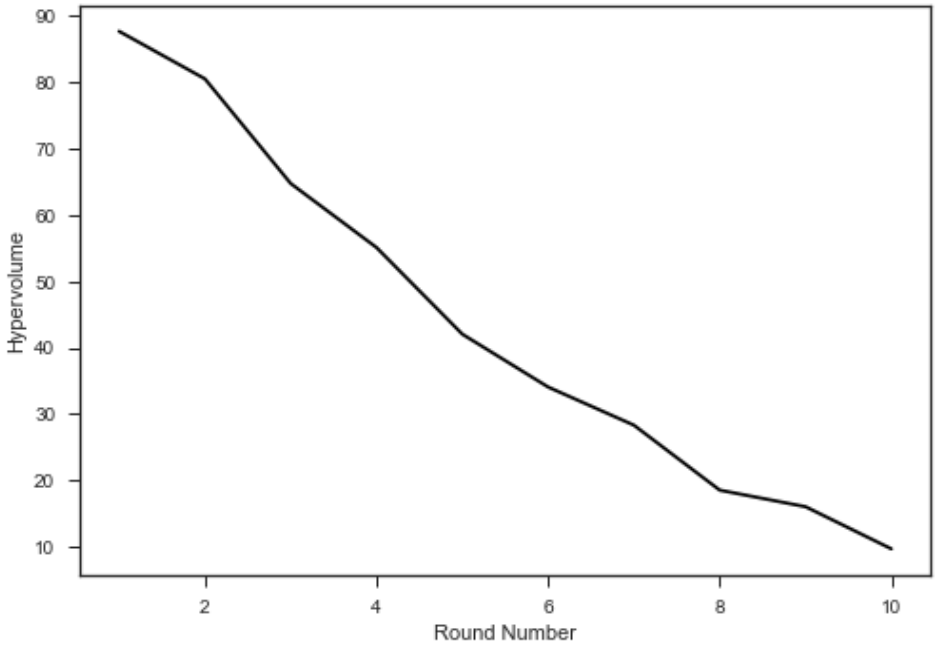

Figure 4.27: Hypervolume progression for the Disk Brake model exploration

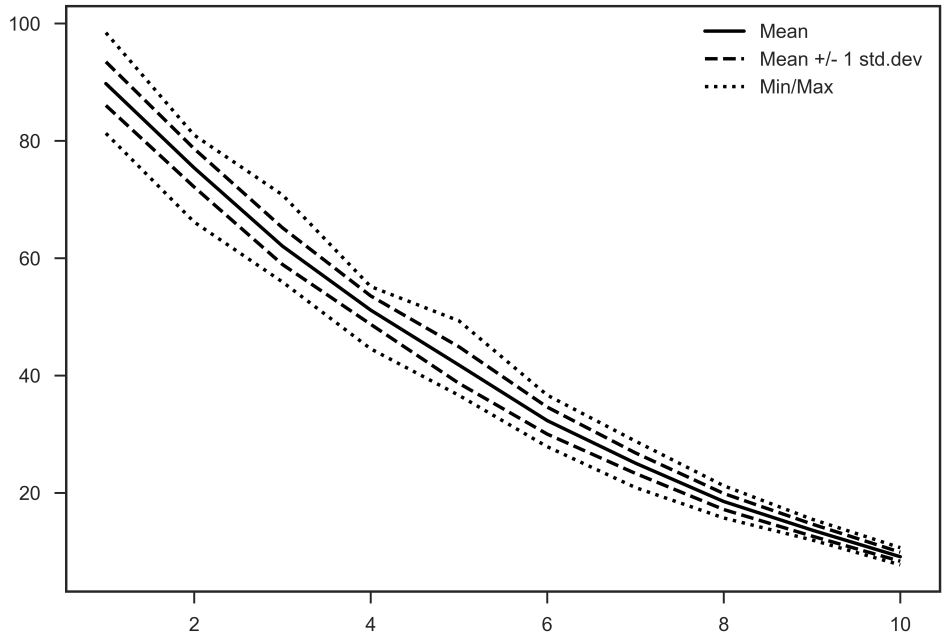

Figure 4.28: The hypervolume progression for the Disk Brake model batch runs

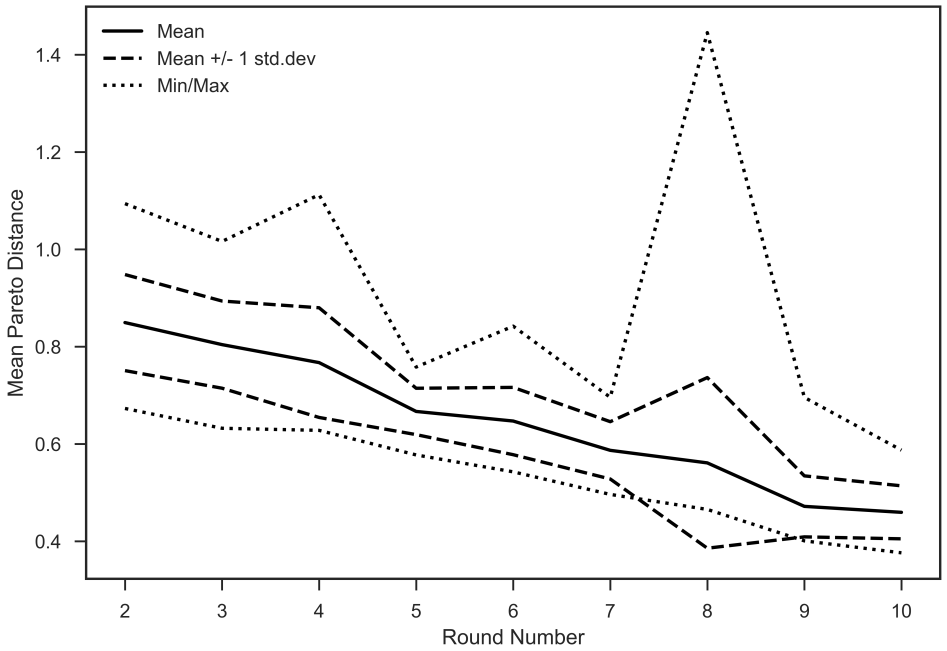

Figure 4.29: Mean Pareto front distance progression for the Disk Brake model batch run in objective space

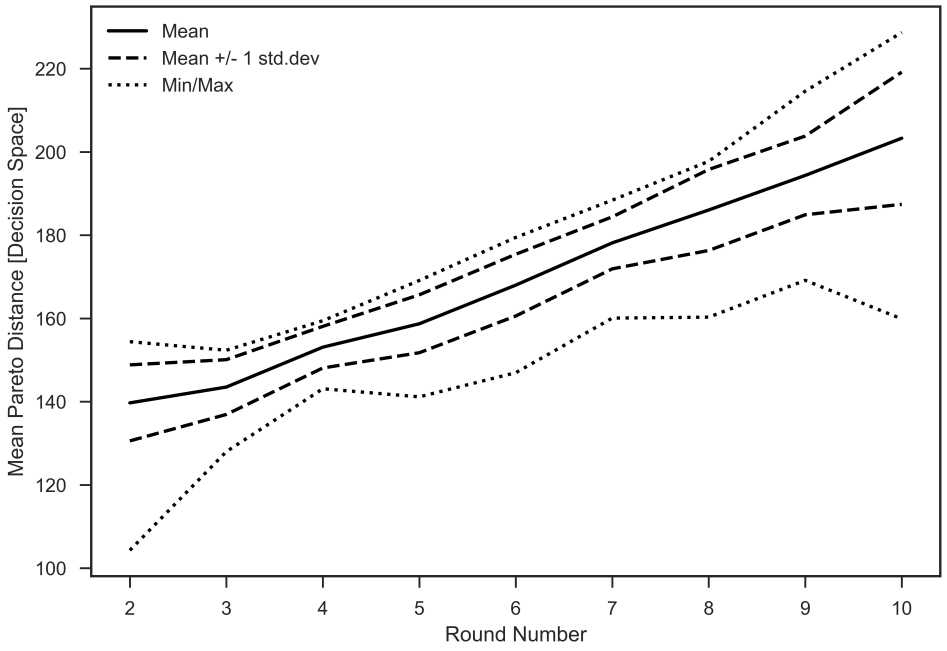

Figure 4.30: Mean Pareto front distance progression for the Disk Brake model batch run in decision space

shows a similar trend to the circle model with a slight upward trend towards the end of the exploration which could be caused by the increased load on the Pareto search algorithm due to the widened search space from relaxing output variable and box constraints.

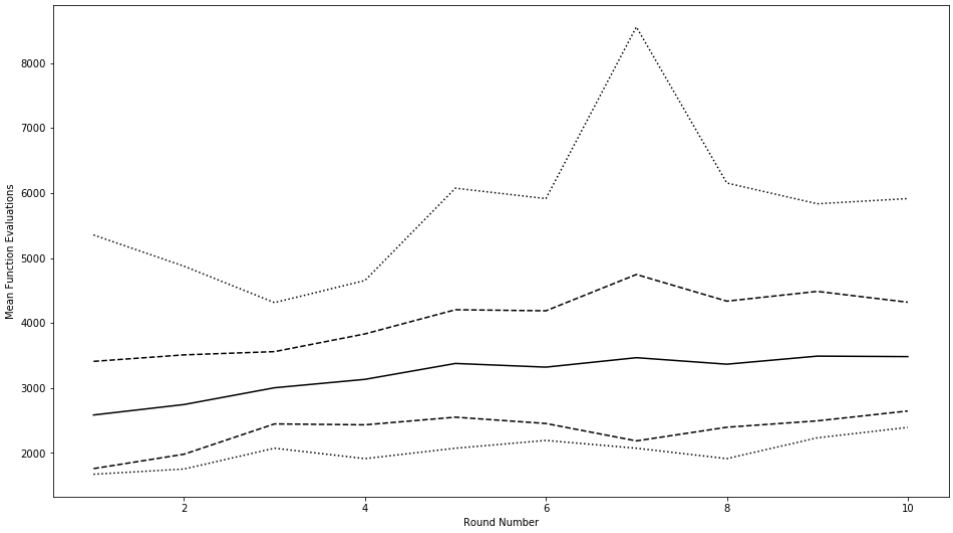

Figure 4.31: Mean function evaluations for the Disk Brake model batch run

4.3 Circle Packing Model Results

The circle packing model represents the most dynamic case presented in this work due to the complexity of the Pareto front and high dimensionality of the search space. Logically we expect that the whitespace exploration algorithm should drive the values of the radius and area density down as this will reduce the inertia of the individual circles and thus the system. The variable configuration for the exploration is given in Tables 4.32 - 4.35. The model setup confines the spheres in to a square area and the variable box bounds are not allowed to be modified during the exploration to maintain this confinement. The interference output constraint is also not allowed to be modified because allowing the circles to overlap would go against the purpose of this model. The individual design variable box bound, constraint variable bound, and non-design variable change histories are given in Appendix A.3 for this case.

Table 4.32: Circle packing model design variable configuration

Design Variable Configuration		
Variable	Lower Bound	Upper Bound
x_i	0.0	5.0
y_i	0.0	5.0

Table 4.33: Circle packing model objective variable configuration

Objective Variable Configuration	
Variable	Goal
I_{xx}	MINIMIZE
I_{yy}	MINIMIZE

The tuning parameters for the exploration are given in Tables 4.36 through 4.40.

Table 4.34: Circle packing model constraint variable configuration

Constraint Variable Configuration		
Variable	Lower Bound	Upper Bound
I	0.0	N/A

Table 4.35: Circle packing model non-design variable configuration

Non-Design Variable Configuration		
Variable	Lower Bound	Upper Bound
r_i	2.0	2.5
ρ_i	6.0	10.0

Notable differences in the circle packing configuration compared to the other models is in the Pareto search configuration, Table 4.36 where the optimization algorithm has been tuned to handle the large number of design variables.

Table 4.36: Pareto search configuration for the circle packing model

Pareto Search Configuration	
Population Size	300
Max Evaluations	200000
Convergence Tolerance	0.01
Convergence Generations	40
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.1
Mutation Index	20.0

The initial design space for the circle packing model in Figure 4.32 shows the

Table 4.37: Constraint search configuration for the Sphere Packing model

Constraint Search Configuration	
Box Constraint Relative Tolerance	0.05
Box Constraint Change Tolerance	0.05
Box Constraint Change Percentage	0.05
Constraint Relative Tolerance	0.05
Constraint Change Tolerance	0.05
Constraint Change Percentage	0.05

Table 4.38: Sensitivity analysis configuration for the Sphere Packing model

Sensitivity Analysis Configuration	
Stencil Size	0.01
Selection Tolerance	0.1

Table 4.39: Direction search configuration for the circle packing model

Direction Search Configuration	
Population Size	100
Max Evaluations	10000
Convergence Tolerance	0.01
Convergence Generations	5
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.1
Mutation Index	20.0

Table 4.40: Exploration search configuration for the Sphere Packing model

Exploration Search Configuration	
Max Evaluations	100
Search Tolerance	0.01
Exploration Distance	0.5

large area of constrained space and the complex shape that makes up the valid design space. The Pareto search convergence, Figure 4.33 highlights the difficulty of the problem, taking nearly 500 generations for the algorithm to converge.

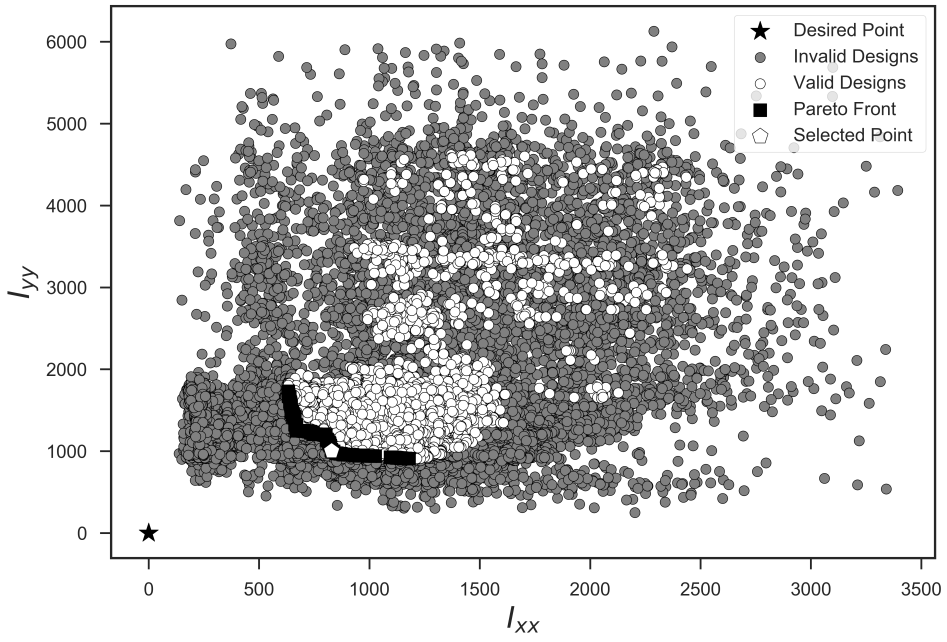

Figure 4.32: Initial design space for the circle packing model

The sensitivity analysis results given in Table 4.41 appear to be random and are highly dependent on the orientation of the circles for the selected point. The sensitivity values emphasize an issue with the current implementation of the sensitivity analysis as some of the non-design variables that would have an affect on the Pareto

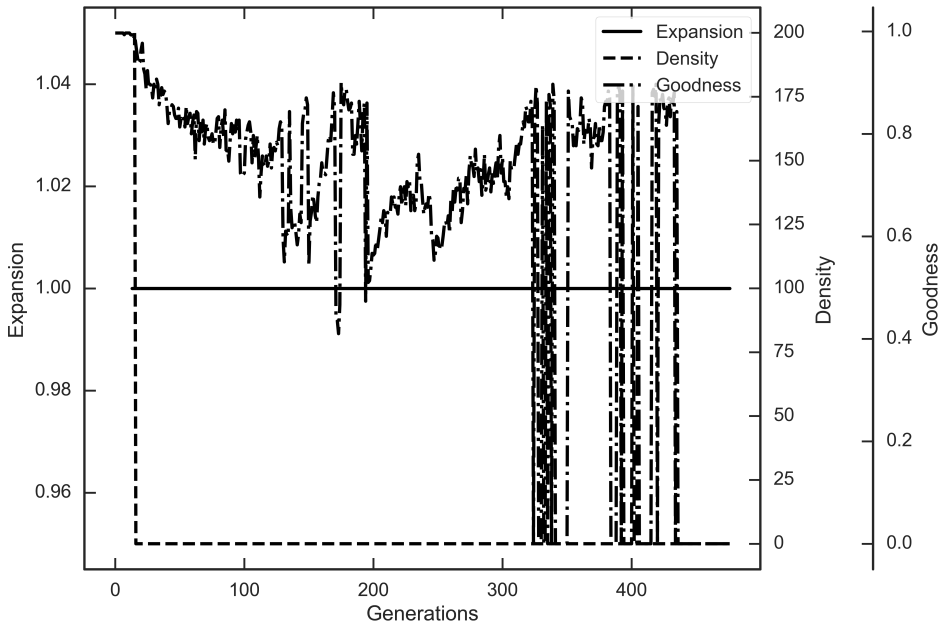

Figure 4.33: Initial Pareto search convergence for the circle packing model

front are not selected for use in the direction search.

The direction search evaluations in Figure 4.34 show that with the selected non-design variables, solutions are found that are in line with the desired point.

The design space found in the last round of the exploration is given in Figure 4.35, shows the drastic improvement in Pareto front progress to the desired point. Figure 4.36 continues to show the difficulty in finding the Pareto front for this model.

The Pareto front progression for the entire exploration is given in Figure 4.37 which shows that continued improvement in the Pareto front that was achieved early on in the exploration, while later on as the front drew close to the desired point progress slowed down. The Pareto front is shown to vary in shape throughout the exploration and does show the algorithms resilience to a dynamic front.

The hypervolume progression for the single run is given in Figure 4.38 and for the batch runs in Figure 4.39 which shows a large degree of variance early on in

Table 4.41: Initial sensitivities for the circle packing case

Sensitivities					
Output	Input	Sensitivity	Output	Input	Sensitivity
I_{xx}	r_1	0.054	I_{yy}	r_1	0.408
I_{xx}	r_2	0.0783	I_{yy}	r_2	0.072
I_{xx}	r_3	0.0286	I_{yy}	r_3	0.046
I_{xx}	r_4	0.521	I_{yy}	r_4	0.168
I_{xx}	r_5	0.010	I_{yy}	r_5	0.016
I_{xx}	ρ_1	0.014	I_{yy}	ρ_1	0.290
I_{xx}	ρ_2	0.038	I_{yy}	ρ_2	0.000
I_{xx}	ρ_3	0.007	I_{yy}	ρ_3	0.000
I_{xx}	ρ_4	0.243	I_{yy}	ρ_4	0.000
I_{xx}	ρ_5	0.005	I_{yy}	ρ_5	0.000

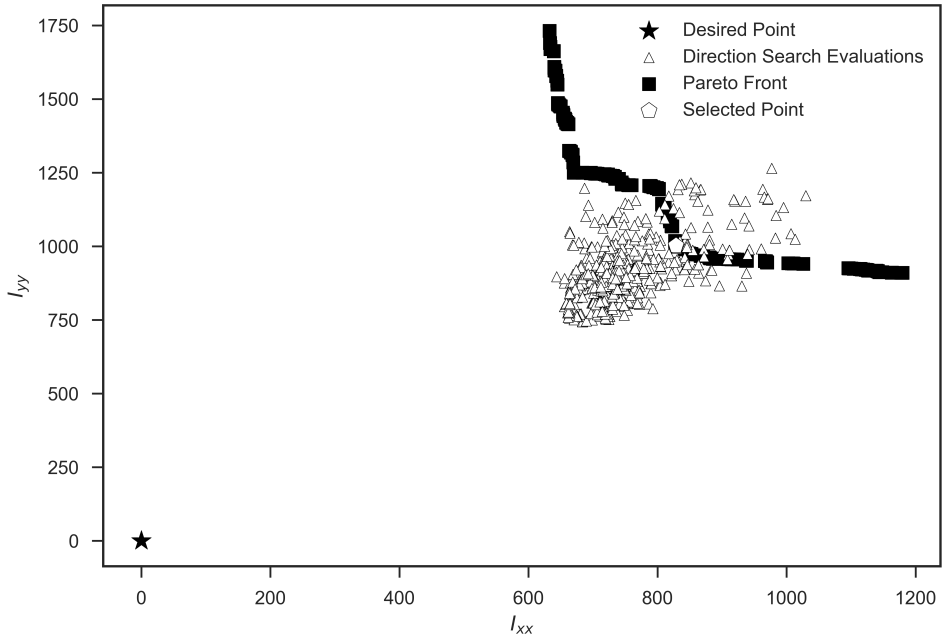

Figure 4.34: Initial direction search for the circle packing model

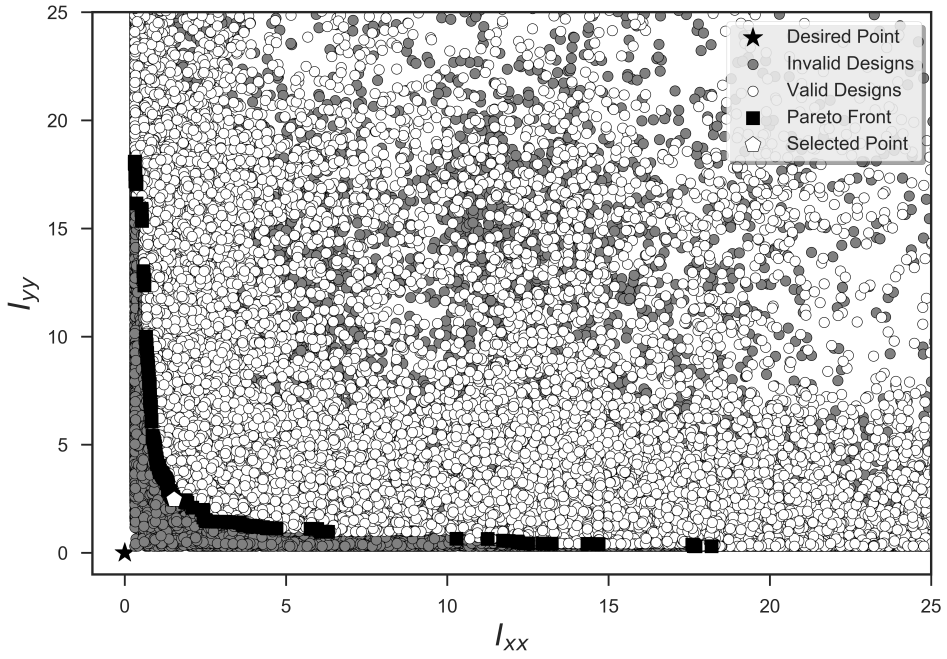

Figure 4.35: Final design space for the circle packing model

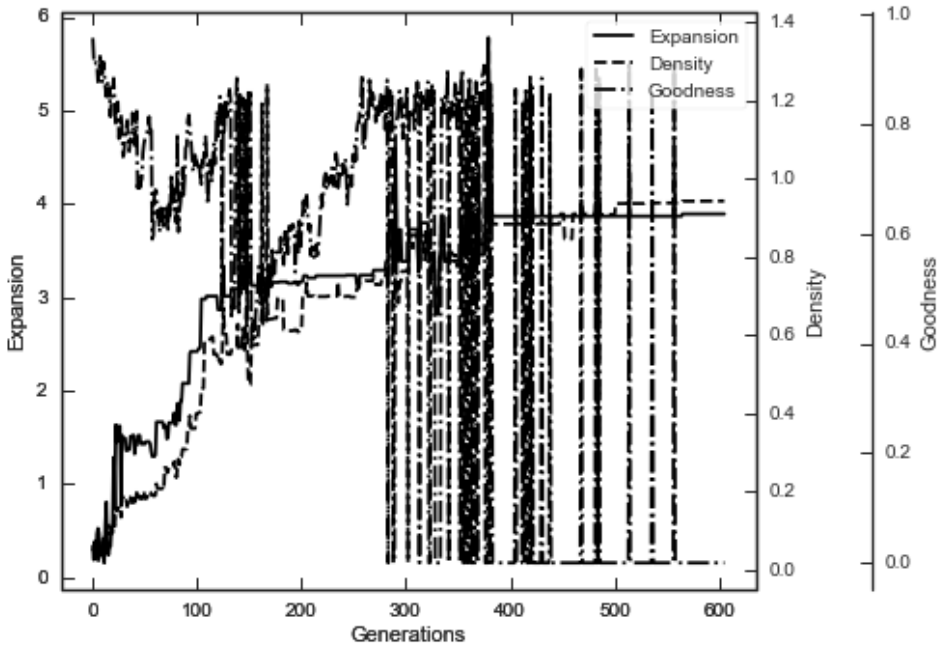

Figure 4.36: Final Pareto search convergence for the circle packing model

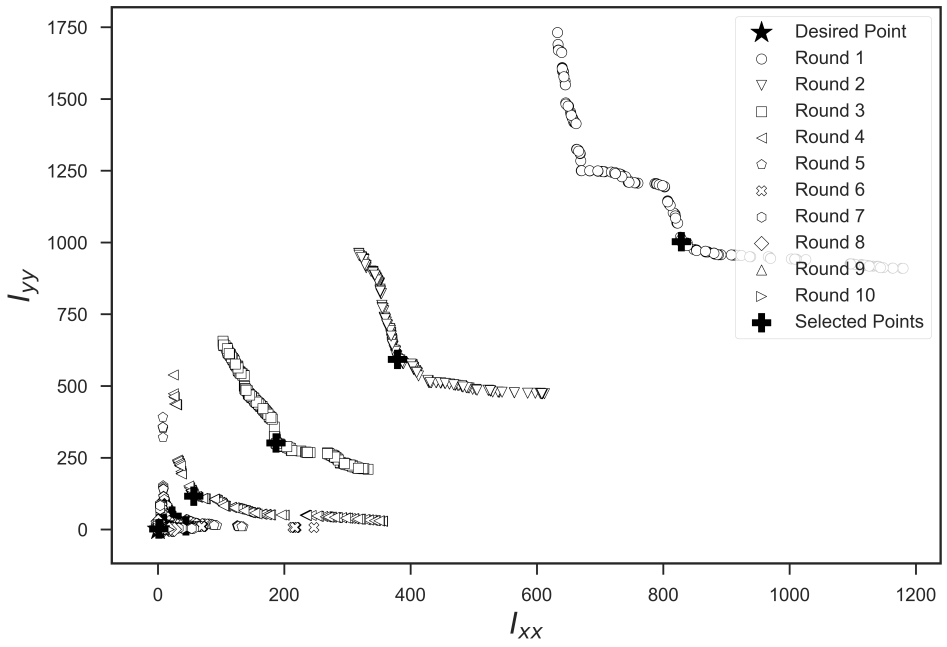

Figure 4.37: Pareto front progression for the circle packing model exploration

the exploration but converged to nearly zero for all thirty runs. Due to the discreet nature of the problem, the initial population has a large affect on the resulting Pareto front found by the Pareto search algorithm. The large exploration step size used in this case causes the Pareto front to become very close to the desired point as the non-design variables are all driven towards zero.

The mean Pareto distance given in Figure 4.40 also shows a similar trend to the hypervolume progression as the magnitude of the objective values decreases drastically through the exploration.

This problem took significantly more functional evaluations than the other models, requiring nearly two orders of magnitude more than the circle model.

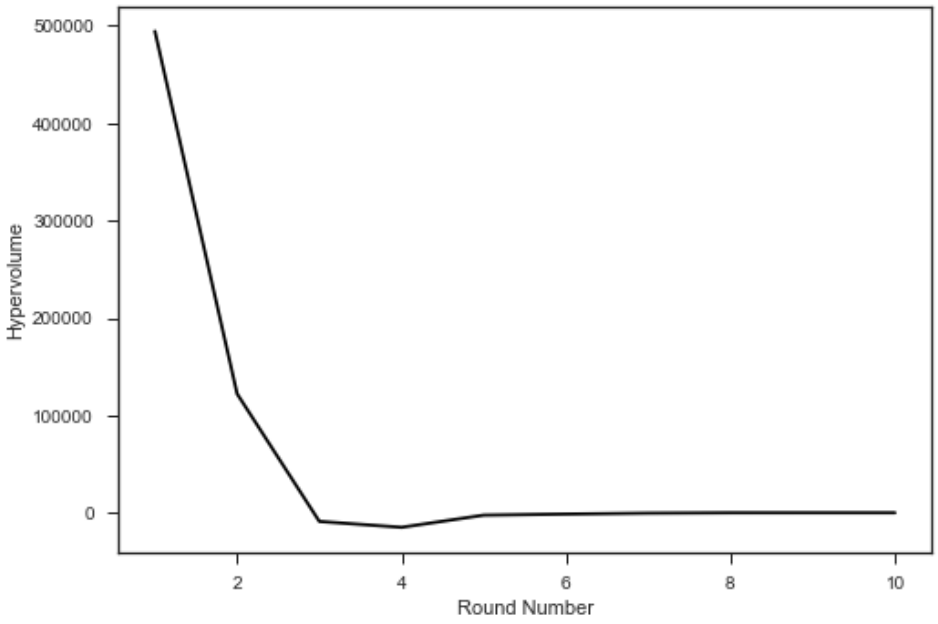

Figure 4.38: Hypervolume progression for the circle packing model exploration

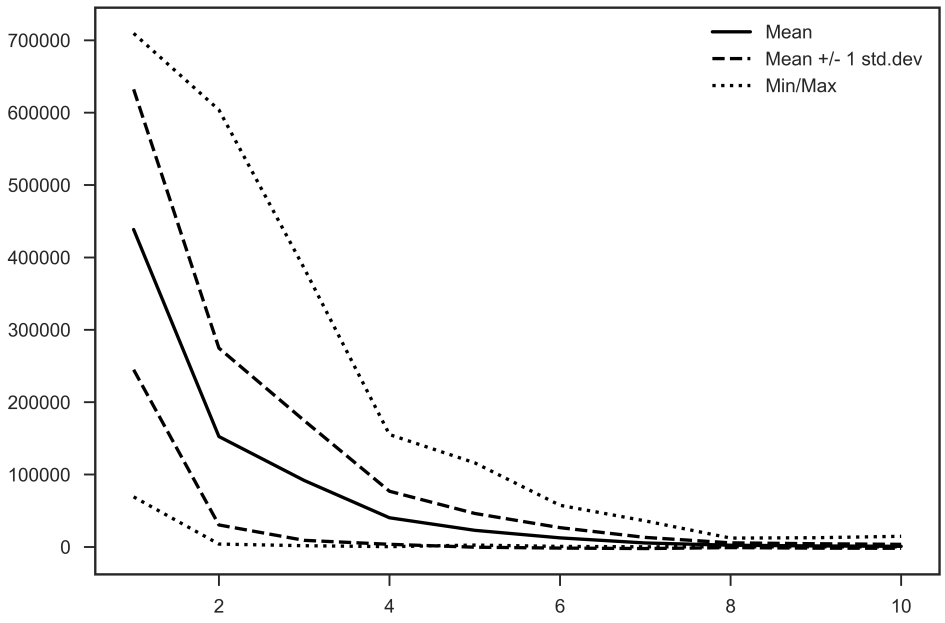

Figure 4.39: Hypervolume progression for the circle packing model batch runs

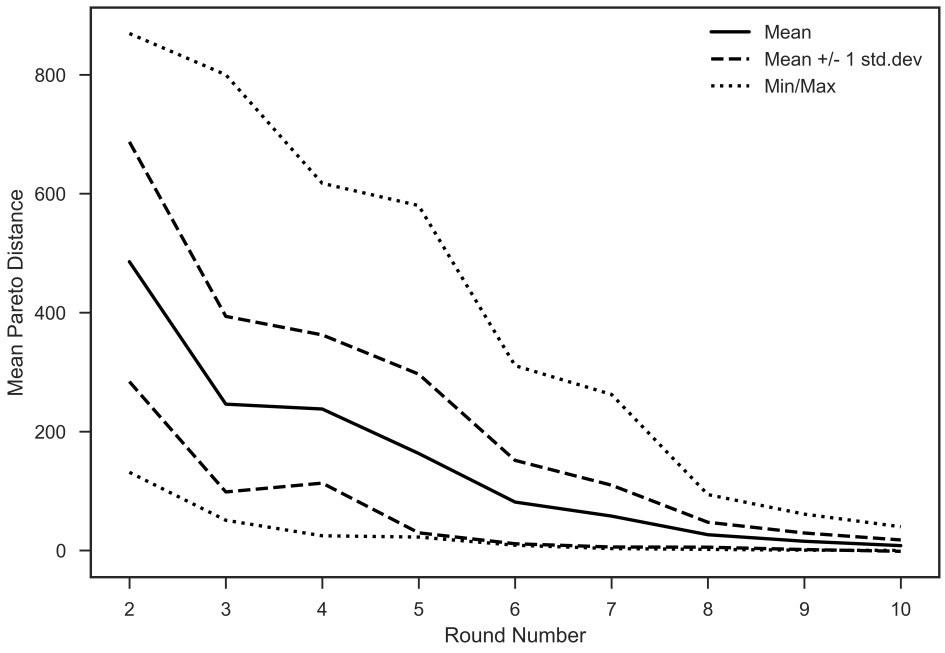

Figure 4.40: Mean Pareto front distance progression for the circle packing model batch run in objective space

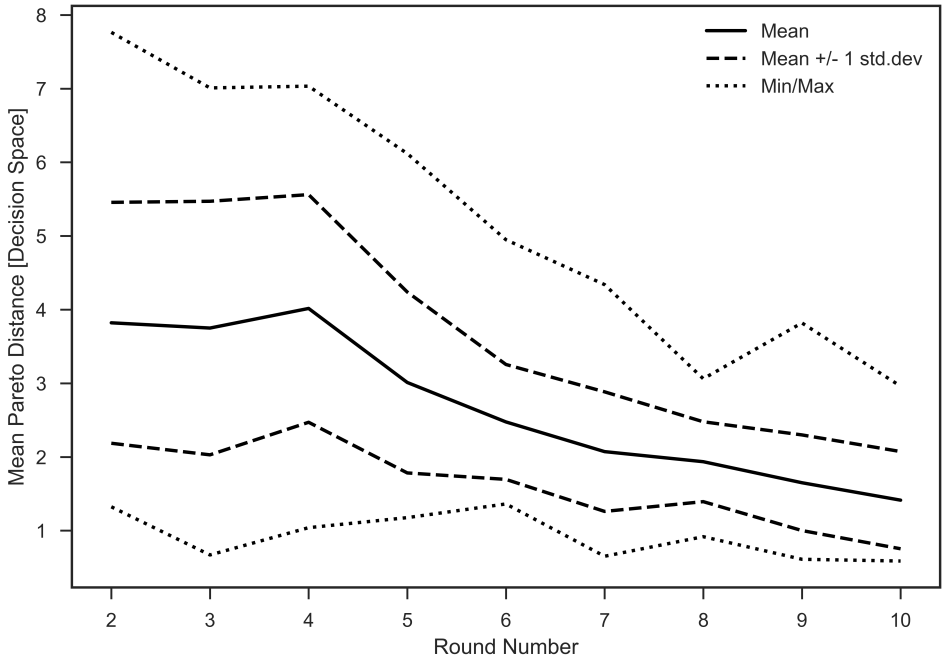

Figure 4.41: Mean Pareto front distance progression for the circle packing model batch run in decision space

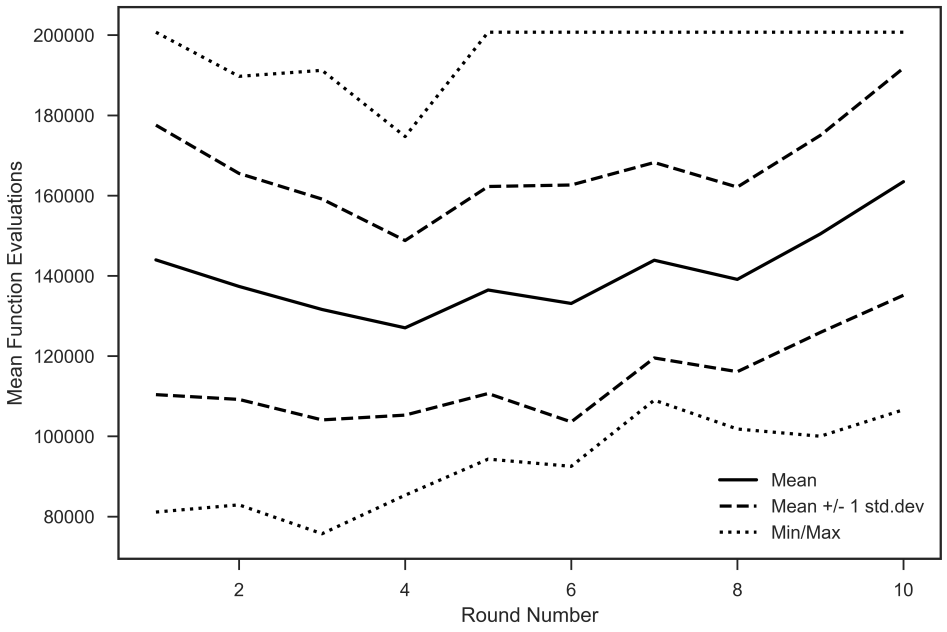

Figure 4.42: Mean function evaluations for the circle packing model batch run

4.4 Whipple Shield Model Results

The whipple shield model, unlike the previous three presented here, contains three objectives and serves as an example of how the algorithm extends to three dimensions. The configuration for the variables is given in Tables 4.42 - 4.45 along with the exploration configuration parameters in Tables 4.46 - 4.50. The individual design variable box bound, constraint variable bound, and non-design variable change histories are given in Appendix A.4 for this case.

Table 4.42: Whipple shield objective variable configuration

Objective Variable Configuration	
Variable	Goal
C_D	MAXIMIZE
m_s	MINIMIZE
V_s	MINIMIZE

Table 4.43: Whipple shield design variable configuration

Design Variable Configuration		
Name	Lower Bound	Upper Bound
s	8.0	25.0
t_b	0.05	0.8
t_w	0.5	1.5
$\text{Æ}R$	8.0	30.0

The first round Pareto search results are shown in Figure 4.43 which takes on the appearance of a curved plate in three dimensions and the optimization algorithm has done a good job at getting even coverage over the front.

Table 4.44: Whipple shield constraint variable configuration

Constraint Variable Configuration		
Name	Lower Bound	Upper Bound
δ_b	N/A	10.0
ω_n	N/A	100.0

Table 4.45: Whipple shield non-design variable configuration

Non-Design Variable Configuration		
Name	Lower Bound	Upper Bound
S_w	1.0	1000.0
S_h	1.0	1000.0
a_{max}	1.0	100.0
ρ_b	1.0	5.0
ρ_w	1.0	5.0
σ_{max}	10.0	100.0
v_{rel}	1.0	40.0
ρ_p	1.0	5.0
E	10e6	10e8
ρ_s	1.0	5.0

Table 4.46: Pareto search configuration for the Whipple Shield model

Pareto Search Configuration	
Population Size	80
Max Evaluations	24000
Convergence Tolerance	0.01
Convergence Generations	10
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.25
Mutation Index	20.0

Table 4.47: Constraint search configuration for the Whipple Shield model

Constraint Search Configuration	
Box Constraint Relative Tolerance	0.05
Box Constraint Change Tolerance	0.05
Box Constraint Change Percentage	0.05
Constraint Relative Tolerance	0.05
Constraint Change Tolerance	0.05
Constraint Change Percentage	0.05

Table 4.48: Sensitivity analysis configuration for the Whipple Shield model

Sensitivity Analysis Configuration	
Stencil Size	0.01
Selection Tolerance	0.1

Table 4.49: Direction search configuration for the Whipple Shield model

Direction Search Configuration	
Population Size	100
Max Evaluations	10000
Convergence Tolerance	0.01
Convergence Generations	5
Crossover Probability	0.9
Crossover Index	20.0
Mutation Probability	0.1
Mutation Index	20.0

Table 4.50: Exploration search configuration for the Whipple Shield model

Exploration Search Configuration	
Max Evaluations	100
Search Tolerance	0.01
Exploration Search Distance	50.0

The decision space for the first round in Figure 4.44 shows that many of the initial box constraints are active as summarized in Table 4.51. None of the output constraints are active with only the deflection constraint δ_b seeing a small number of solutions on the constraint boundary shown in Table 4.52.

The sensitivity analysis results in Table 4.53, which do not include sensitivities of zero for brevity, correctly identifies the correct relationships for the model's non-design variables

The final design space given in Figure 4.45 shows how the Pareto front shape has

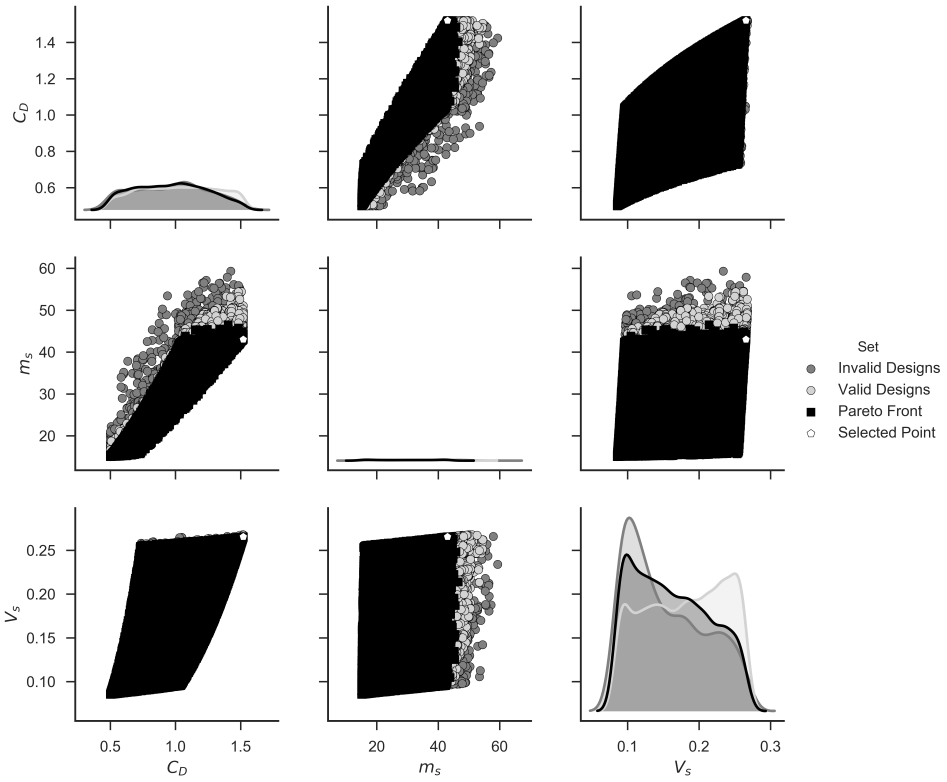

Figure 4.43: Initial design space for the Whipple Shield model

Table 4.51: Initial box constraint activity for the Whipple Shield model

Box Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
s	13.17	ACTIVE	4.80	INACTIVE
t_b	99.68	ACTIVE	0.0	INACTIVE
t_w	12.17	ACTIVE	14.56	ACTIVE
$\mathcal{R}$	29.29	ACTIVE	0.0	INACTIVE

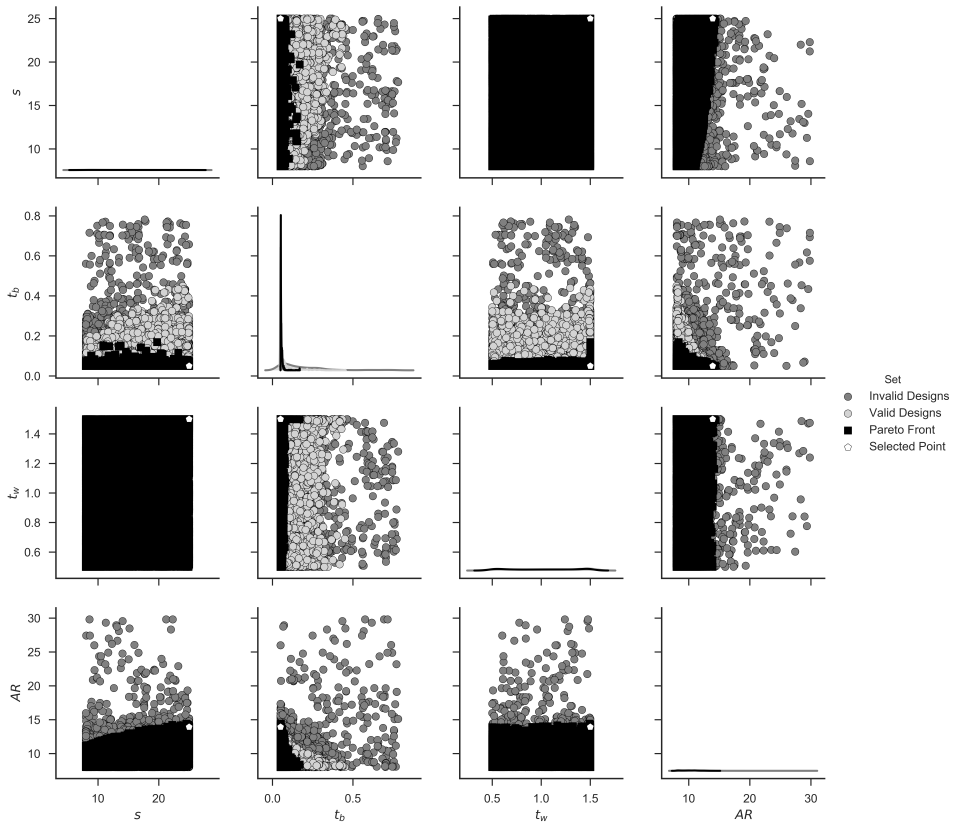

Figure 4.44: Initial decision space for the Whipple Shield model

Table 4.52: Initial constraint activity for the Whipple Shield model

Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
δ_b	0.0	INACTIVE	1.59	INACTIVE
ω_n	N/A	N/A	0.0	INACTIVE

Table 4.53: Initial sensitivities for the Whipple Shield model

Sensitivities		
Output	Input	Sensitivity
C_D	ρ_p	-0.231
C_D	σ_{max}	0.231
C_D	ρ_w	-0.076
C_D	v_{rel}	-0.462
m_s	ρ_b	0.011
m_s	ρ_s	0.005
m_s	ρ_w	0.321
m_s	S_w	0.332
m_s	S_h	0.332
V_s	S_w	0.5
V_s	S_h	0.5

evolved. Inspection of the bounds also highlights the improvement that was made, particularly in the critical diameter, C_D, and volume, V_s, objectives. The final decision space, Table 4.46 shows that many of the design variables still have active boundaries and that we now have an active deflection constraint, summarized in Tables 4.54 and 4.55 respectively.

The Pareto front progression for the exploration is given in Figure 4.47 and shows a steady progression towards the desired point along with an expansion of the front in all directions.

The hypervolume progression for the batch runs is given in Figure 4.48 and shows a familiar trend towards a smaller volume as the exploration proceeds. At least one run appears to have made drastic progress in the first round.

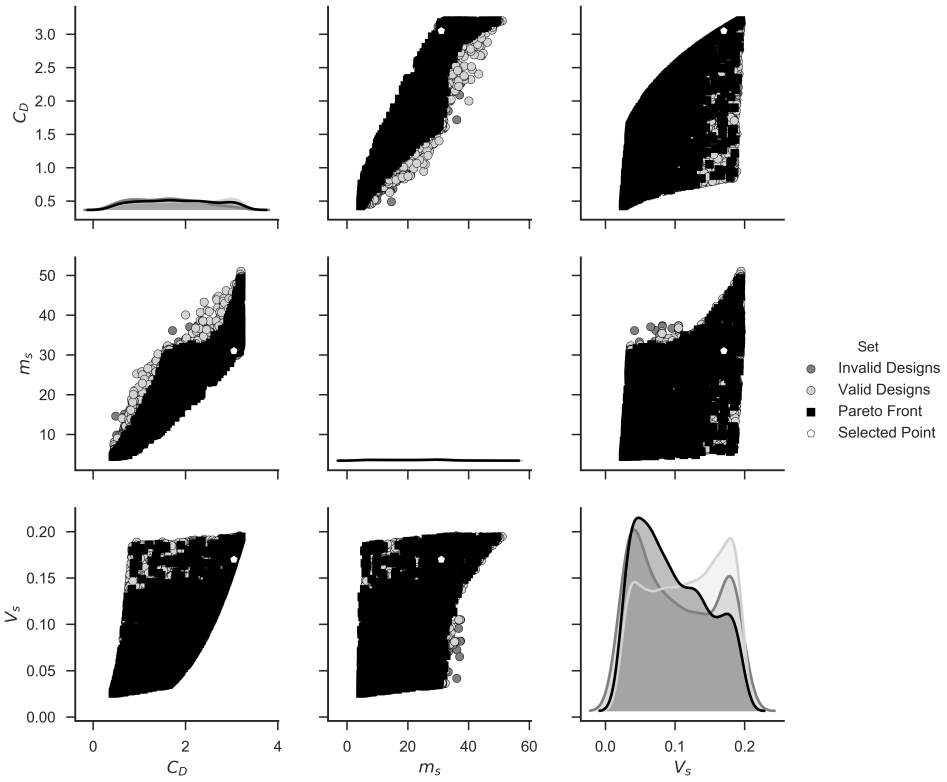

Figure 4.45: Final design space for the Whipple Shield model

Table 4.54: Final box constraint activity for the Whipple Shield model

Box Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
s	12.81	ACTIVE	5.49	INACTIVE
t_b	89.79	ACTIVE	0.0	INACTIVE
t_w	8.10	ACTIVE	29.52	ACTIVE
$\mathcal{R}$	31.82	ACTIVE	0.0	INACTIVE

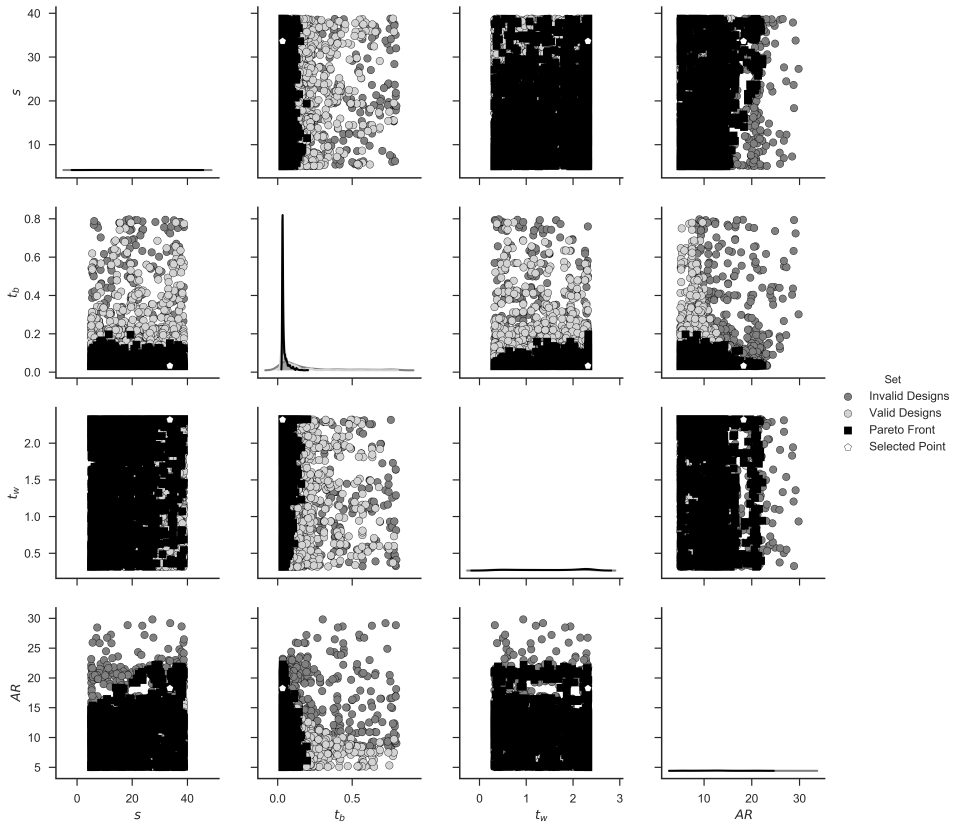

Figure 4.46: Final decision space for the Whipple Shield model

Table 4.55: Final constraint activity for the Whipple Shield model

Constraint Activity				
Variable	Lower %	Activity	Upper %	Activity
δ_b	0.0	INACTIVE	8.69	ACTIVE
ω_n	N/A	N/A	0.0	INACTIVE

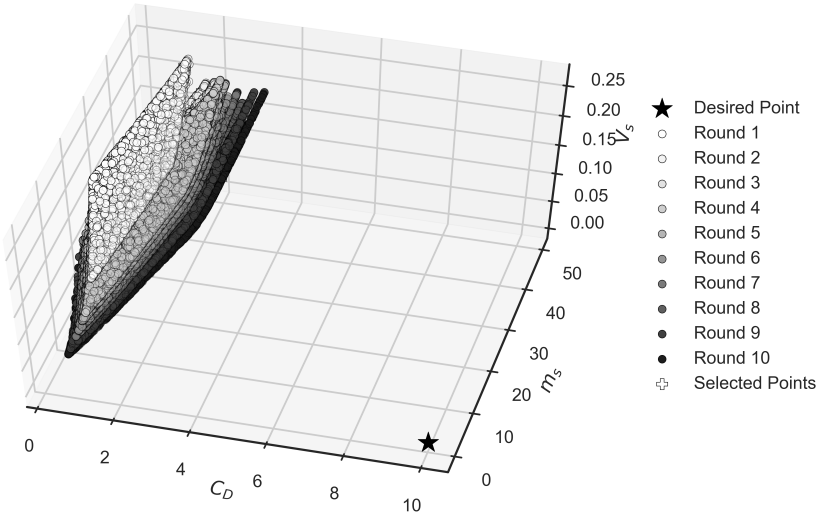

Figure 4.47: Pareto front progression for the Whipple Shield model exploration

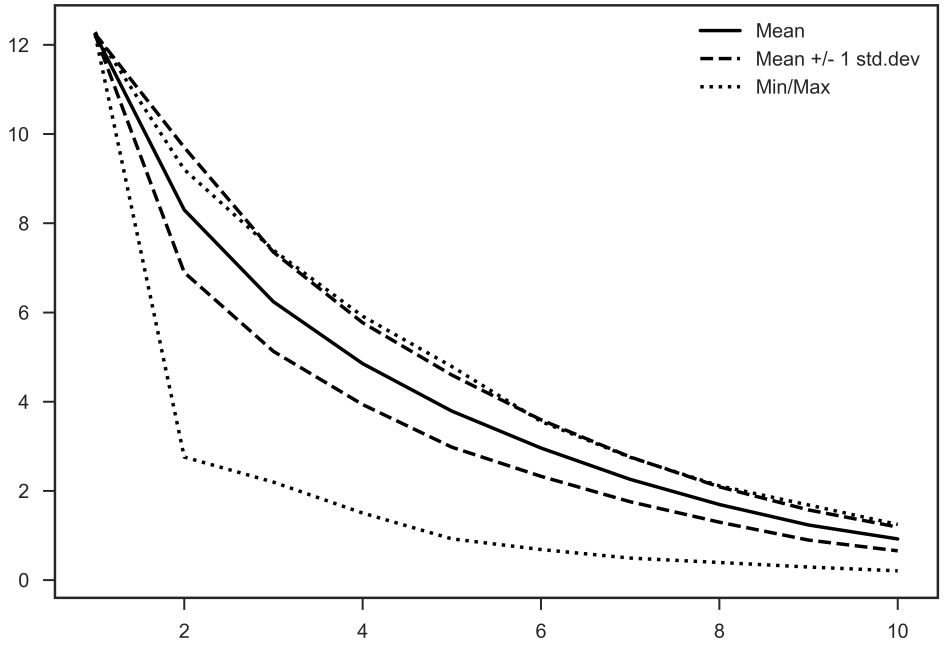

Figure 4.48: Hypervolume progression for the Whipple Shield model batch runs

The Pareto front distance progression in Figures 4.49 and 4.50 show that after the first round, progress in the objective space dropped significantly while the movement in the decision space steadily increased.

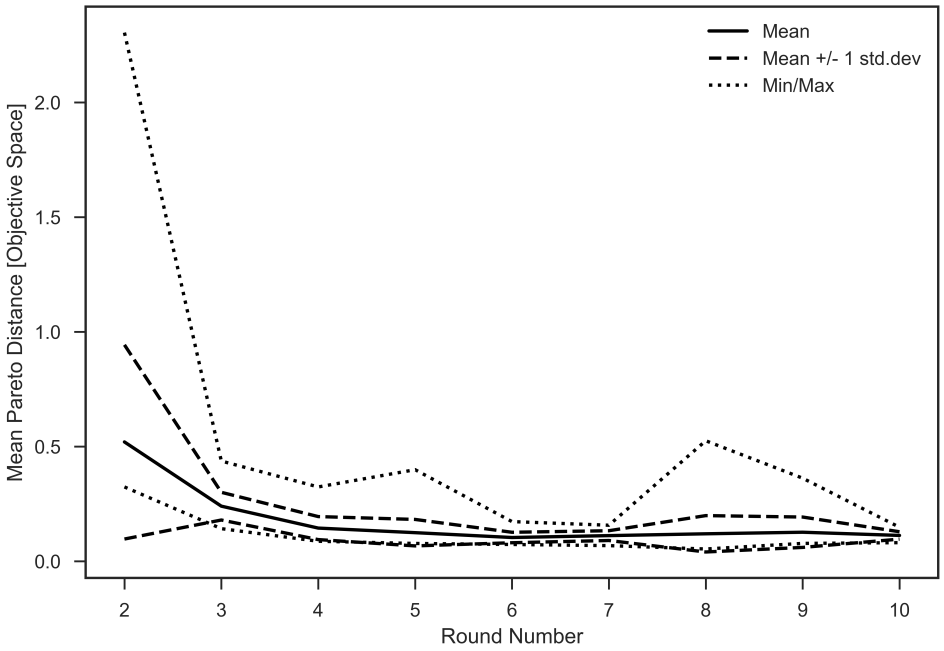

Figure 4.49: The mean Pareto front distance progression for the Whipple Shield model batch run in objective space

The mean function evaluations in Figure 4.51 for the model again highlights the issue of the large number of function evaluations currently required by the algorithm.

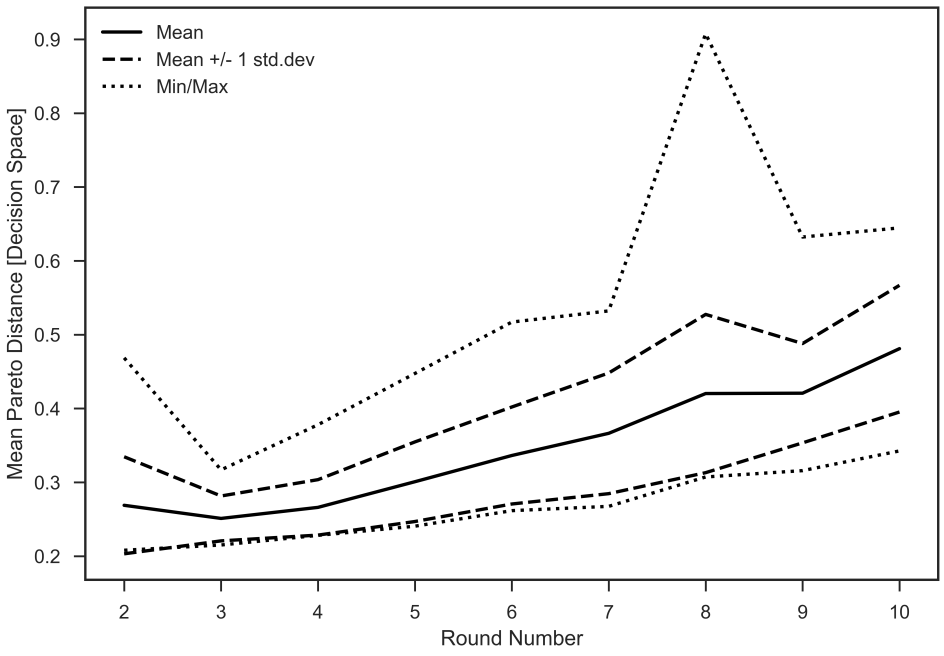

Figure 4.50: The mean Pareto front distance progression for the Whipple Shield model batch run in decision space

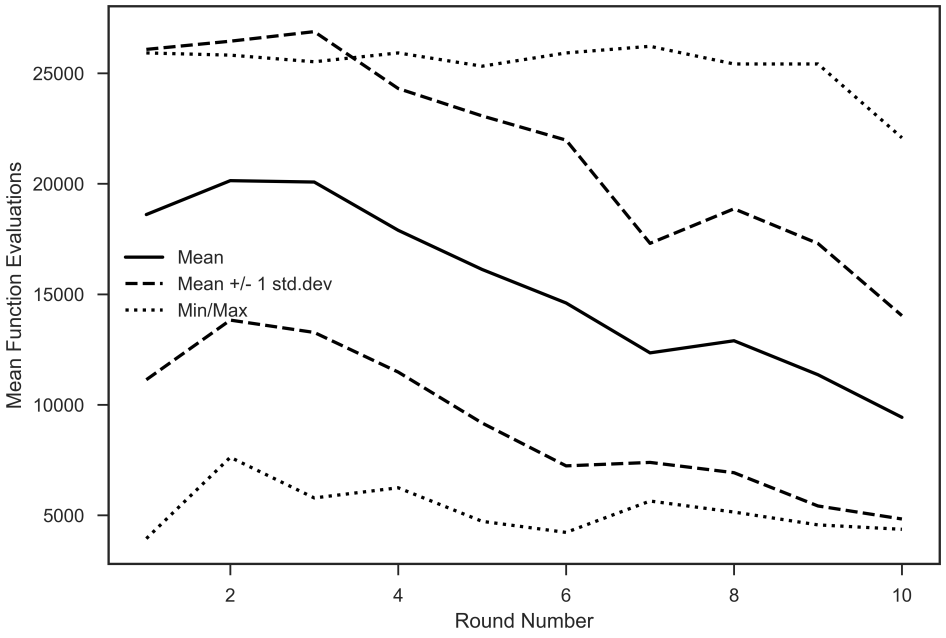

Figure 4.51: The mean function evaluations for the Whipple Shield model batch run

Chapter 5

CONCLUSION

In this work, whitespace exploration has been explained to be a new class of algorithms focused on searching previously unexplored regions of the design space. The software package Thalia was developed to aid in further development and research of these algorithms in an automated fashion. Four different benchmark models were developed and a set of tests were run on them to examine the efficacy of the current whitespace exploration algorithm and discover any issues.

One of the primary issues facing whitespace exploration is the sheer number of model evaluations that are required. Using the circle packing model as an example of the type of complexity that could be expected in real world engineering models, required on average around 15000 model evaluations each round. If we assume that an average engineering model takes a minute to run this would require around ten days of CPU time. The large amount of time needed to run a whitespace exploration could be a serious limiting factor to it's use in an engineering design environment. Luckily, some gains could be had by performing the model evaluations in parallel, however this requires a high performance computing environment and the ability to run multiple copies of the model, which is often limited by commercial software licensing.

The exploration parameter, η_e, is another aspect of the current algorithm that leaves a lot to be desired. In it's current state it is an arbitrary input parameter to make the algorithm work, but does not have a direct physical meaning to aid the designer in selecting an appropriate value.

The use of the selected point proved to be effective through the benchmark cases, however for cases like the circle packing model, it can have a large affect on how the

exploration progresses. The selected point is also built on an assumption that a single point on the Pareto front is representative of the front itself which may not always be true. The development of methods that view the Pareto front as a whole would be a great boon for whitespace exploration.

Future work on improving whitespace exploration should focus first and foremost on reducing the large number of function evaluations the algorithm currently requires. A bulk of the evaluations lie in the Pareto search step which would be the first place to look for improving the overall algorithmic efficiency. The next steps for whitespace exploration should focus on developing input parameters and output metrics that would make sense to design engineers that don't have a dedicated background in design space exploration. Improving the general usability and understanding of whitespace exploration results would go a long way towards bringing these methods into a mainstream engineering environment.

Additionally comparisons between the manual algorithm and the automated algorithm developed in this work should be compared to determine if one approach is inherently better than the other along with assessing possible use cases for either implementation.

BIBLIOGRAPHY

[1] S. K. Andrzej Osyczka. A modified distance method for multicriteria optimization using genetic algorithms. *Computers and Industrial Engineering*, 30:871–882, 1996.

[2] E. L. Christiansen. Handbook for designing mmod protection. Technical Report 214785, NASA Johnson Space Center, 6 2009.

[3] J. Daniel. Modelcenter opttool nsga-ii convergence criteria. Technical report, Phoenix Integration, 1715 Pratt Drive Suite 200 Blacksburg, Virgina 24060, 2015.

[4] P. Datseris. Weight minimization of a speed reducer by heuristic and decomposition techniques. *Mechanism and Machine Theory*, 17:255–262, 1982.

[5] K. Deb, S. Agawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *KanGAL*, (200001), 2002.

[6] J. J. Durillo and A. J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[7] C. M. Fonseca, L. Paquete, and M. López-Ibáñez. An improved dimension-sweep algorithm for the hypervolume indicator. *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'06*, pages 1157–1163, 2006.

[8] J. Golinski. Optimal synthesis problems solved by means of nonlinear programming and random methods. *Journal of Mechanisms*, 5:287–309, 1970.

[9] Z. He. Performance metrics ensemble for multiobjective evolutionary

algorithms. Master's thesis, University of Science and Technology Beijing, may 2008.

[10] S. Jiang, Y.-S. Ong, J. Zhang, and L. Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, 44:2391–2404, 2014.

[11] G. C. Kiam Heong Ang and Y. Li. Visualization technique for analyzing non-dominated set comparison. 2002.

[12] A. Ko, J. Daniel, W. Keel, and A. Baines. Whitespace exploration: The next step in searching the design space. AIAA, 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference AIAA SciTech Forum, 2017.

[13] M. Mukhayad, I. Hermadi, and S. Hardhienata. Layout optimization of microsatellite components using genetic algorithms. *Telkomnika*, 15:341–350.

[14] D. G. Newnan, T. G. Eschenabach, and J. P. Lavelle. *Engineering Economic Analysis*. Oxford University Press, 198 Madison Avenue, New York, New York 10016, 9 edition, 2004.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies: The Art of Scientific Computing*. Cambridge University Press, 32 Avenue of the Ameriacs, New York, NY 10013-2473 USA, 3 edition, 2007.

[16] P. Sabarinath and et al. Multiobjective optimization method based on adaptive parameter harmony search algorithm. *Jounral of Applied Mathematics*, 2015:12, 2015.

[17] K. Stephenson. Circle packing: a mathematical tale. *Notices of the AMS*, 50:1376–1388, 2003.

APPENDICES

Appendix A

A.1 Circle model variable change histories

These are the variable change histories for design variable box bounds, constraints, and non-design variables at each round in the single exploration analysis of the Circle model.

Table A.1: Circle model r box bound change history

Round	Lower Bound	Upper Bound
0	0.0	1.0
1	0.0	1.05
2	0.0	1.1025
3	0.0	1.157625
4	0.0	1.215506
5	0.0	1.276281
6	0.0	1.340095
7	0.0	1.407100
8	0.0	1.477455
9	0.0	1.551328

Table A.2: Circle model constraint c change history

Round	Lower Bound	Upper Bound
0		0.0
1		0.05
2		0.0525
3		0.055125
4		0.057881
5		0.060775
6		0.063814
7		0.067004
8		0.070355
9		0.073872

Table A.3: Circle model non-design variable change history

x_c	y_c
10.0	10.0
9.283665	9.306700
8.560448	8.620585
7.840174	7.931379
7.152403	7.209736
6.472502	6.480673
5.826383	5.721509
5.184855	4.958463
4.578005	4.167558
3.963205	3.382816

A.2 Disk Brake model variable change histories

These are the variable change histories for design variable box bounds, constraints, and non-design variables at each round in the single exploration analysis of the Disk Brake model.

Table A.4: Disk Brake model design variable R_i box bound change history

Round	Lower Bound	Upper Bound
0	55.0	80.0
1	55.0	84.0
2	55.0	88.2
3	52.25	92.61
4	52.25	97.2405
5	52.25	102.102525
6	49.6375	107.207651
7	49.6375	112.568033
8	47.155625	118.196435
9	47.155625	124.106257

A.3 Circle Packing model variable change histories

These are the variable change histories non-design variables at each round in the single exploration analysis of the Disk Brake model. Note that for the circle packing model design variable box bounds and constraint bounds were not allowed to change during the exploration.

Table A.5: Disk Brake model design variable R_o box bound change history

Round	Lower Bound	Upper Bound
0	75.0	110.0
1	75.0	115.5
2	75.0	121.275
3	71.25	127.33875
4	71.25	133.705687
5	71.25	140.390971
6	67.6875	147.410520
7	67.6875	154.781046
8	67.6875	162.520098
9	67.6875	170.646103

Table A.6: Disk Brake model design variable F box bound change history

Round	Lower Bound	Upper Bound
0	1000.0	3000.0
1	1000.0	3150.0
2	1000.0	3307.5
3	1000.0	3472.875
4	1000.0	3646.51875
5	1000.0	3828.8446875
6	1000.0	4020.2869218
7	1000.0	4221.3012679
8	1000.0	4432.3663313
9	1000.0	4653.9846479

Table A.7: Disk Brake model constraint g_3 bound change history

Round	Lower Bound	Upper Bound
0	0.0	
1	-0.05	
2	-0.05	
3	-0.05	
4	-0.05	
5	-0.05	
6	-0.0525	
7	-0.0525	
8	-0.055125	
9	-0.055125	

Table A.8: Disk Brake model constraint g_4 bound change history

Round	Lower Bound	Upper Bound
0	0.0	
1	0.0	
2	0.0	
3	0.0	
4	0.0	
5	0.0	
6	0.0	
7	0.0	
8	0.0	
9	0.0	

Table A.9: Disk Brake model constraint g_5 bound change history

Round	Lower Bound	Upper Bound
0	0.0	
1	0.0	
2	0.0	
3	0.0	
4	0.0	
5	0.0	
6	0.0	
7	0.0	
8	0.0	
9	0.0	

Table A.10: Disk Brake model non-design variable change history

m_s	t_{sf}
100.0	100.0
92.951799	92.916477
85.857459	85.879164
78.914335	78.692617
71.748538	71.728080
64.678274	64.666580
57.612724	57.600362
50.527956	50.553413
43.476315	43.473315
36.413635	36.404229

Table A.11: Circle Packing model non-design variable change history

r_1	ρ_1	r_2	rho_2	r_3	ρ_3	r_4	ρ_4	r_5	ρ_5
1.605	7.036	0.813	8.252	1.636	3.423	2.264	2.416	0.640	4.188
1.207	7.691	0.813	8.252	1.636	3.423	1.678	2.667	0.640	4.188
0.927	6.915	0.813	8.252	1.636	3.423	1.306	2.263	0.536	4.188
0.664	6.316	0.852	8.252	1.188	3.423	0.931	1.800	0.440	4.188
0.648	5.956	0.580	8.252	1.092	3.423	0.717	1.0	0.274	4.188
0.551	4.970	0.536	8.252	1.081	3.423	0.613	1.0	0.274	4.188
0.641	4.021	0.377	8.088	0.899	3.423	0.635	1.0	0.274	4.188
0.355	4.832	0.258	8.088	0.409	3.423	0.618	1.0	0.274	4.188
0.25	3.950	0.258	8.088	0.409	3.423	0.559	1.0	0.25	4.188
0.25	3.037	0.400	8.088	0.409	3.423	0.507	1.0	0.25	4.188

A.4 Whipple Shield model variable change histories

These are the variable change histories for design variable box bounds, constraints, and non-design variables at each round in the single exploration analysis of the Disk Brake model.

Table A.12: Whipple Shield model design variable s box-bound change history

Round	Lower Bound	Upper Bound
0	8.0	25.0
1	7.6	26.25
2	7.22	27.5625
3	6.859	28.940625
4	6.51605	30.387656
5	6.190247	31.907039
6	5.880735	33.502391
7	5.586698	35.177510
8	5.307363	36.936386
9	5.041995	38.783205

Table A.13: Whipple Shield model design variable t_b box-bound change history

Round	Lower Bound	Upper Bound
0	0.05	0.8
1	0.0475	0.8
2	0.045125	0.8
3	0.042868	0.8
4	0.040725	0.8
5	0.038689	0.8
6	0.036754	0.8
7	0.034916	0.8
8	0.033171	0.8
9	0.031512	0.8

Table A.14: Whipple Shield model variable t_w box-bound change history

Round	Lower Bound	Upper Bound
0	0.5	1.5
1	0.475	1.575
2	0.45125	1.65375
3	0.428687	1.736437
4	0.407253	1.823259
5	0.386890	1.914422
6	0.367545	2.010143
7	0.349168	2.110650
8	0.331710	2.216183
9	0.315124	2.326992

Table A.15: Whipple Shield model variable AR box-bound change history

Round	Lower Bound	Upper Bound
0	8.0	30.0
1	7.6	30.0
2	7.22	30.0
3	6.859	30.0
4	6.51605	30.0
5	6.190247	30.0
6	5.880735	30.0
7	5.586698	30.0
8	5.307363	30.0
9	5.041995	30.0

Table A.16: Whipple Shield model constraint δ_b bound change history

Round	Lower Bound	Upper Bound
0		10.0
1		10.5
2		11.025
3		11.57625
4		12.155062
5		12.762815
6		13.400956
7		14.071004
8		14.774554
9		15.513282

Table A.17: Whipple Shield model constraint ω_n bound change history

Round	Lower Bound	Upper Bound
0	0.0	100.0
1	0.0	100.0
2	0.0	100.0
3	0.0	100.0
4	0.0	100.0
5	0.0	100.0
6	0.0	100.0
7	0.0	100.0
8	0.0	100.0
9	0.0	100.0

Table A.18: Whipple Shield model non-design variable change history

S_w	S_h	a_max	ρ_b	ρ_w	σ_{max}	v_{rel}	ρ_p	E	ρ_s
100.0	100.0	15.0	2.73	2.73	40.0	12.0	2.73	680000000.0	2.73
90.98	96.34	15.0	2.73	2.81	89.04	12.0	2.73	680000000.0	2.73
94.17	88.68	15.0	2.73	2.79	100.0	12.0	2.73	680000000.0	2.732
87.31	91.08	15.0	2.73	2.79	100.0	12.0	2.73	680000000.0	2.732
99.93	73.97	15.0	2.73	2.80	100.0	12.0	2.73	680000000.0	2.732
128.64	51.85	15.0	2.73	2.80	100.0	12.0	2.73	680000000.0	2.732
107.41	57.87	15.0	2.73	2.80	100.0	12.0	2.73	680000000.0	2.732
137.13	40.40	15.0	2.73	2.78	100.0	12.0	2.73	680000000.0	2.732
120.55	43.80	15.0	2.73	2.71	100.0	12.0	2.73	680000000.0	2.732
149.08	31.69	15.0	2.73	2.70	100.0	12.0	2.73	680000000.0	2.732