A Novel Analysis Framework for Evaluating Predisposition of Design Solutions through the Creation of Hereditary-Amelioration Networks Derived from the Dynamics within an Evolutionary Optimizer

by

Michael J. Sypniewski

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

Doctoral Committee:

Associate Professor David J. Singer, Chair
Associate Professor Matthew D. Collette
Associate Professor Kevin J. Maki
Professor Romesh Saigal

Michael J. Sypniewski

mjsyp@umich.edu

ORCID iD: 0000-0003-1437-9341

For my family

TABLE OF CONTENTS

DEDICATION ii

LIST OF FIGURES v

LIST OF TABLES vii

LIST OF ABBREVIATIONS viii

ABSTRACT ix

CHAPTER

I. Introduction 1

1.1 Background and Motivation 1
1.1.1 Current Design Environment in the Navy 1
1.1.2 Introducing Quality and Bias 3
1.1.3 Relevant Issues 5
1.2 Research Scope 6
1.3 Dissertation Structure 7

II. Discussing Quality and Bias 8

2.1 Quality and Optimality 8
2.2 Defining Two Views of Quality 10
2.2.1 Solution-centric Quality 11
2.2.2 Generative Quality 12
2.3 The Necessity of Comparison 12
2.4 Capturing Solution Generation Dynamics 13
2.5 Biasing a Solution Generation Process 14
2.6 Summary 14

III. Developing the Framework 16

3.1 Background . . . 17
3.1.1 Objective-based Models . . . 17
3.1.2 Genetic Algorithms . . . 19
3.2 Capturing Solution Dynamics . . . 25
3.2.1 Creating Hereditary-Amelioration Networks . . . 26
3.2.2 Deriving the Parent-Parent Network . . . 36
3.3 Establishing a Comparative Context . . . 38
3.3.1 Alternative Ranking Functions . . . 39
3.3.2 Reference and Biasing Cases . . . 39
3.3.3 Temporal Network Subsets . . . 41
3.3.4 Effects of Increasing Tournament Size . . . 43
3.4 Summary . . . 44

IV. Analyzing the Framework . . . 46

4.1 Cardinality . . . 46
4.2 Coverage . . . 51
4.3 Superfront Relative Distance . . . 53
4.4 Generational Distance . . . 60
4.5 Out-degree . . . 61
4.6 Betweenness Centrality . . . 67
4.7 Summary . . . 71

V. Case Study . . . 73

5.1 Case Study Setup . . . 73
5.2 Cardinality Results . . . 76
5.3 Coverage Results . . . 81
5.4 Superfront Relative Distance Results . . . 83
5.5 Generational Distance Results . . . 87
5.6 Out-degree Results . . . 88
5.7 Betweenness Centrality Results . . . 92
5.8 Effects of Increasing Tournament Size . . . 95
5.9 Conclusions . . . 96

VI. Conclusions . . . 98

6.1 Contributions . . . 98
6.2 Future Topics of Interest . . . 100

APPENDIX . . . 103

BIBLIOGRAPHY . . . 108

LIST OF FIGURES

Figure

2.1 Nondominated and dominated solutions defined by a bi-objective evaluation of Pareto dominance. 9
3.1 The optimization procedure of a GA. 20
3.2 Two different types of networks. The edges in an *undirected* network (*right*) do not signify directional relationships, whereas those in a *directed* network (*left*) do. 26
3.3 HANs after initializing a population containing six individuals. Individuals in the GA are represented by nodes in the PCN and PPN. . 27
3.4 Development of HANs during crossover at $t = 1$. Nodes are added to the networks for each newly created individual. Blue nodes indicate parents that were chosen to engage in crossover and create children. In the PCN, edges are added *to* child nodes *from* parent nodes. In the PPN, edges are added *between* parent nodes. 29
3.5 Development of HANs during selection at $t = 1$. Green nodes indicate those that were selected to remain in the population. 31
3.6 Development of HANs during crossover at $t = 2$. Red nodes indicate those that were removed from the population at $t = 1$ and therefore cannot be chosen to engage in crossover. Blue nodes indicate parents that were chosen to engage in crossover and create children at $t = 2$. 33
3.7 Development of HANs during selection at $t = 2$. Red nodes represent those that were removed from the population at $t = 1$. Green nodes indicate those that were selected to remain in the population at $t = 2$. 34
3.8 An aggregate Parent-Child Network. 35
3.9 An aggregate Parent-Parent Network. 35
4.1 Cardinality of the noncumulative Pareto-set, $|P(t)|$. 47
4.2 Cardinality of the cumulative Pareto-set, $|P_c(t)|$. 48
4.3 Cardinality of the noncumulative feasible-set, $|F(t)|$. 50
4.4 Cardinality of the cumulative feasible-set, $|F_c(t)|$. 51
4.5 The mechanics of the superfront-relative-distance metrics. 56
4.6 Distance to superfront of the Pareto-set, $D_T(P(t))$. 57
4.7 Distance from superfront of the Pareto-set, $D_F(P(t))$. 58
4.8 Distance to superfront of the feasible-set, $D_T(F(t))$. 59

4.9 Distance from superfront of the feasible-set, $D_F(F(t))$. 60
4.10 Generational Distance as a convergence metric, $GD(P(t-1), P(t))$. 61
4.11 Aggregate PCNs for each case. A node's size illustrates its relative out-degree, and colored nodes are contained in $P_c(E)$. 63
4.12 Probability of a node residing in $P_c(E)$ given its out-degree. 65
4.13 Probability of a node residing in $P_c(E)$ given its out-degree. Evaluating out-degree as a leading indicator of bias. 66
4.14 Aggregate PPNs for each case. A node's size illustrates its relative betweenness, and colored nodes are contained in $P_c(E)$. 69
4.15 Probability of a node residing in $P_c(E)$ given its betweenness. . . . 70
4.16 Probability of a node residing in $P_c(E)$ given its betweenness. Considering only nodes in the largest component of the PPN. 71
5.1 Cardinality of the noncumulative Pareto-set, $|P(t)|$. 76
5.2 Cardinality of the cumulative Pareto-set, $|P_c(t)|$. 78
5.3 Cardinality of the noncumulative feasible-set, $|F(t)|$. 79
5.4 Cardinality of the cumulative feasible-set, $|F_c(t)|$. 80
5.5 Distance to superfront of the Pareto-set, $D_T(P(t))$. 83
5.6 Distance from superfront of the Pareto-set, $D_F(P(t))$. 84
5.7 Distance to superfront of the feasible-set, $D_T(F(t))$. 86
5.8 Distance from superfront of the feasible-set, $D_F(F(t))$. 87
5.9 Generational Distance as a convergence metric, $GD(P(t-1), P(t))$. 88
5.10 Aggregate PCNs for each case. A node's size illustrates its relative out-degree, and colored nodes are contained in $P_c(E)$. 89
5.11 Probability of a node residing in $P_c(E)$ given its out-degree. 90
5.12 Probability of a node residing in $P_c(E)$ given its out-degree. Evaluating out-degree as a leading indicator of bias. 91
5.13 Aggregate PPNs for each case. A node's size illustrates its relative betweenness, and colored nodes are contained in $P_c(E)$. 93
5.14 Probability of a node residing in $P_c(E)$ given its betweenness. . . . 94
5.15 Probability of a node residing in $P_c(E)$ given its betweenness. Considering only nodes in the largest component of the PPN. 95
5.16 Cardinality of the noncumulative feasible-set, $|F(t)|$. Comparison of tournament sizes $k = 2$ (*top*), $k = 3$ (*bottom left*), and $k = 5$ (*bottom right*). 96

LIST OF TABLES

Table

3.1	A simple multi-objective model.	17
3.2	Common hyperparameters of a GA.	20
3.3	A population's first three individuals with their corresponding input, constraint, penalty, and objective values.	21
3.4	Temporal nodal attributes after initialization at $t = 0$.	28
3.5	Temporal nodal attributes after crossover at $t = 1$.	28
3.6	Temporal nodal attributes after the Pareto-set is updated at $t = 1$.	30
3.7	Temporal nodal attributes after selection at $t = 1$.	31
3.8	Summary of temporal network subsets.	43
4.1	Coverage results $C(P^A(E), P^B(E))$ assessing the percentage of the Pareto-set of B that is dominated by the Pareto-set of A.	52
4.2	Interpretation of $D_T(A)$ in terms of mean μ and standard deviation σ.	56
4.3	Interpretation of $D_F(A)$ in terms of mean μ and standard deviation σ.	56
5.1	Hyperparameters of the GA used to generate HANs for the case studies.	75
5.2	Coverage results $C(P^A(E), P^B(E))$ assessing the percentage of the Pareto-set of B that is dominated by the Pareto-set of A.	81

LIST OF ABBREVIATIONS

ONR Office of Naval Research

GA Genetic Algorithm

HAN Hereditary-Amelioration Network

PCN Parent-Child Network

PPN Parent-Parent Network

ABSTRACT

In early-stage design, critical decisions are made within a limited information environment. Designers conduct and interpret analyses, while taking into account the associated risks, so that meaningful design trade-offs can be investigated. To aid in this, design tools are utilized to generate solutions in the hopes of characterizing a design space. However while it is known that solutions are prescribed by the tools used to generate them, there seems to be little concern toward how these predisposed biases affect the quality of solutions relative to the desired outcome. Without the ability to determine a tool's biases, one cannot understand their effects on decision-making. This inability can promote inaccurate perspectives of the desired design space, can negatively impact the ultimate success of the design, and poses a currently unquantified risk within the design process. To make truly informed decisions, designers must be able to assess a tool's inherent biases, its intended applications, and its contextual appropriateness to the design questions it is being used to answer.

To provide these capabilities, this thesis presents a framework for evaluating a model's underlying biases. Within this thesis, new and novel aspects of quality have been developed, namely *solution-centric* quality and *generative* quality. Novel quality metrics have been created and are used to evaluate a model as it is subjected to biases. A modified Genetic Algorithm (GA) has been developed so that an ensemble of biased solutions can be generated over time. Thereby, this GA provides a dynamic environment of the solution generation process associated with a model. Additionally, newly created and novel Hereditary-Amelioration Networks (HANs) are derived from the dynamics within this modified GA. The HANs capture the implied-causality

behind solution dynamics and represent temporal, causal relationships between solutions. This newly developed approach is used to establish the comparative context necessary for a model's biases to be analyzed and understood by creating reference and biasing cases.

Utilizing the developed framework, a variety of solution-centric and generative analyses are developed to evaluate a model's inherent tendencies. A comprehensive case study is conducted to demonstrate how the framework and the various analyses are implemented and interpreted. The case study's results demonstrate that the framework can successfully identify a model's biases, providing designers with the contextual information necessary to make truly informed decisions.

CHAPTER I

Introduction

1.1 Background and Motivation

The design of complex products, such as naval vessels, is difficult and has been classified as a "wicked problem" (*Andrews*, 2012). Due to the nature of wicked problems, decisions made during the conceptual design phase have the largest impacts on cost, performance, and schedule. In addition to being the most impactful toward the successful creation of a complex product, early-stage design is uniquely difficult. Within early-stage design, decisions are made with limited information, and thus the designer is tasked with not only making a decision but also with intuitively evaluating their associated risk given a limited information environment. To aid in this, design tools are utilized to explore and evaluate potential design solutions and characterize the design space. Designers are responsible for generating and interpreting these results to understand their significance in the context of a given design problem, with the hopes of providing meaningful decision trade-offs so that future emergent design failure risks can be mitigated.

1.1.1 Current Design Environment in the Navy

Over the past four decades, the U.S. Navy and the Office of Naval Research (ONR) have made significant investments toward developing engineering analysis tools, early-

stage design tools, and computational models to facilitate information generation and decision-making throughout an acquisition's lifecycle. The cultural effect of this investment is apparent when considering the current state of design within the U.S. Navy. Modeling and synthesis dominate all design activities, and design itself is tracked and managed through product models (*Naval Sea Systems Command*, 2012). Early-stage design activities are characterized by a strong cultural bias toward design space exploration, with an emphasis on solution generation capabilities (*Mackenna*, 2011). The perceived limitations of this current approach are seen to primarily reside in computing power or in graphical representations (*Chalfant*, 2015). The Navy's vision for analysis tools comprise massively integrated synthesis codes and simulation environments, with the goal of creating automated tools to rapidly produce a full range of feasible solutions, evaluate them, and thereby make the designer instantly aware of design implications:

> " *Because of the limited amount of tool integration, and a manual ship design definition process, the Navy enterprise is usually driven to select one design alternative early in the design process. The vision for Navy design tools is to move to an automated high-end toolset that integrates many information dense design definition tools with high fidelity physics-based analysis tools. A system such as this could be used to explore the design space to ensure that the correct design is selected before signing a contract to build a ship. For example, an automated tool could rapidly produce a full range of feasible ship arrangements from a basic shell of a ship, and then a vulnerability assessment could be performed on each of these many design variations and the resultant range of achievable levels of vulnerability can be fed back to the designer. Thus, the designer is instantly aware of the vulnerability implications of the sizing and arrangement of the ship.* "
>
> - *Kassel et al.* (2010)

From these sources, the current state of design in the U.S. Navy can be summarized by the belief that automated tools and additional computational resources will provide novel knowledge for decision-making and thereby eliminate decision errors in early-stage design.

1.1.2 Introducing Quality and Bias

While useful, it is important to understand the limitations of synthesis models and optimization techniques within a design process (*McKenney*, 2013). The blind, unfounded faith in automated tools exemplifies how design is perceived by many in academics and industry to simply be the act of developing and utilizing tools or the act of running models and simulations. An alternative view is that design is the act of generating knowledge for decision-making through time (*Shields*, 2017); it is about understanding design drivers and interdependencies. If one views design as the act of generating knowledge for decision-making, then evaluating "what the analysis results are" is equally as important as assessing "how" or "why" those are the results.

An underlining issue is that more results do not necessarily equate to additional novel knowledge for decision-making. The belief that generating a massive number of solutions will provide novel knowledge for decision-making is grounded on the presumption that novel knowledge is simply hidden and can be uncovered after addressing the limiting factor of how much can be processed. The reality is that the creation of a model is based on the subjective judgment of the modeler, and different methods can produce different results (*Papalambros and Wilde*, 2000). Solution attributes and solution generation strategies are embedded within a model during its development (*Dosi*, 1982). Knowledge development is implicit in the synthesis process, and therefore previous knowledge, regardless of applicability to the design problem under consideration, is embedded in design tools. Thus rather than uncovering any new knowledge, the act of generating more solutions merely enumerates the

hard-coded knowledge, applicable or not, embedded within the tool. The presumption that increasing solution generation capabilities will generate novel knowledge is fundamentally untrue, and thus a shift away from blinding generating solutions is needed. Since it is known that generated solutions are driven by the modeler's judgment and the tool's embedded artifacts, the most important considerations should be toward identifying these biases and understanding their effects on decision-making.

One of the most significant implications from the stated realization is the existence of solution predication. Solutions are prescribed by the equations used, interdependencies defined, and methods used to synthesize them (*Gillespie*, 2012; *Parker*, 2014; *Shields*, 2017). There are multiple examples in the literature that demonstrate how design solutions are predicated by the tools used to generate them. Regarding the Intelligent Ship Arrangements tool, *Gillespie* (2012) shows how solutions are preordained by the set of global and local preference constraints. Given initial parameters to a general arrangements problem, *Shields* (2017) demonstrates how the most probable solutions matched those resulting from an intensive optimization procedure. By decomposing a model into its constituent parts, *Parker* (2014) identifies internal design drivers that are indicative of the model's primary tendencies.

Published limitations on the current state of the art are perceived as either a lack of tool integration or computational resources and not as the applicability of a tool to a specific design problem. However, embedded knowledge development and solution predication raise the question of quality. Given that a tool may be predisposed to generating particular solutions, how does this affect the tool's quality in terms of generating knowledge for decision-making, and how is this quality evaluated? Within optimization, simulation, and analysis literature, there are countless metrics to evaluate the quality of a model through its execution (*Sargent*, 2011). Traditional verification and validation (V&V) is focused on (1) ensuring that a model and its operation are "correct" and (2) confirming that a model is satisfactorily accurate

when used within its domain of applicability. The assumption is that if a model is verified and validated then the solutions that it produces have high quality and thus the tool has high quality. The issue lies in the fact that a tool can pass all the "tests" but, due to reasons discussed and demonstrated throughout this thesis, may not be appropriate for the early-stage design questions that it is being used to answer. There seems to be little concern for (1) understanding how the quality of a tool's results is affected by its appropriateness or (2) evaluating if the generated solutions go against its intended use.

Regardless of the environment, it is still the designer's responsibility to accurately inform decision-making through meaningful interpretations of analyses. To do this, designers must be able to assess a tool's intended applications, its inherent biases, and its appropriateness in the context of the given design problem. Currently, designers do not have the capability to assess the quality of prospective design solutions toward a specific intent in this manner. Without this type of information, decision-makers must operate under the erroneous notion that all solutions contain an equivalent quality of applicable information, which leaves designers and decision-makers unknowingly vulnerable to only considering solutions that may be predicated by the tools themselves. The solutions that are generated, and thus used, can provide inaccurate perspectives of the desired design space, which impacts the ultimate success of the design and poses a currently unquantified risk to the design process.

1.1.3 Relevant Issues

The consequences of not accounting for solution predication are significant given the simple fact that solutions generated using early-stage design tools provide the basis for critical early-stage design decisions. The importance of addressing these issues is increased when considering that design is path dependent with respect to use of information and decision-making (*Page*, 2006). Decisions made during the design

process affect the availability of future decisions and solutions. Decisions based on synthesis impact future developments and may predicate future outcomes.

Increases in tool automation, opacity, integration, and complexity affect a designer's ability to determine a tool's original purpose or the assumptions that went into its development. As toolsets continue to grow, designers will become increasingly limited in their ability to manually diagnose how the tools affect solution quality (*Savoie and Frey*, 2012), necessitating an automated evaluation method to fill this gap. The integration of disparate tools enables the propagation of a single source's inferior quality, creating a product with a "lowest common denominator" degree of quality. Furthermore, this integration increases the complexity of the combined product and can lead to emergent failures (*Leveson*, 2004).

Given that tool development costs are high, tool reuse becomes a very attractive alternative within early-stage design. Tools are often utilized to address questions that they were not originally developed to answer. The push toward purely physics-based analysis tools should reduce some of their biases, however these methods often require more expertise and design information to execute. In contrast, early-stage ship design must operate with very limited information and generally relies on reduced-order models that have been developed with assumptions concerning physics or with regressions using data that may not be applicable to all reuse cases the tool will be applied to. While verification of analysis tools is straightforward, a truly honest validation is not.

1.2 Research Scope

The presented thesis is focused on developing methods to answer the fundamental question, "How can an analysis tool be evaluated to determine if it is presenting biased information?" The existence of biased information is not necessarily a concern, but rather the issues occur when the tool's inherent biases are misaligned with the design

space under consideration. To answer this question, one must determine if the tool is appropriate and applicable for reuse to answer questions outside its original reasons for development. More specifically, does the quality of result when asking question A match the quality of result when asking question B? How can designers be made aware of when a tool's embedded knowledge affects design outcomes? How can this be identified and measured? The following section outlines how these questions are approached and answered in this thesis.

1.3 Dissertation Structure

This dissertation is divided into the five following chapters that are organized as follows:

- Chapter II presents a discussion of quality and proposes that, in design, a holistic assessment of quality requires multiple contextual views. The chapter discusses the considerations necessary to evaluate quality and how these evaluations can be used to understand the biases of a model.

- Chapter III details the mechanics of the developed framework and how aspects of the framework address the considerations necessary for evaluating bias.

- Chapter IV details a variety of analyses that, utilizing the developed framework, are used to evaluate a model's inherent tendencies.

- Chapter V presents a comprehensive case study that was conducted to demonstrate how a model's biases are evaluated using the framework proposed in Chapter III and the analyses detailed in Chapter IV.

- Chapter VI details the contributions of this thesis and future topics of interest.

CHAPTER II

Discussing Quality and Bias

This chapter discusses quality, defines two types of quality and their intents, and discusses considerations that are needed to evaluate quality and thereby assess and understand a model's inherent biases. This provides answers to the questions: what does quality mean within the context of early stage design tools, how can it be measured, what does bias mean, how do quality and bias relate to each other, and how can evaluations of quality be used to understand the biases of a model?

2.1 Quality and Optimality

When discussing solution quality, a term that is bound to come up is *optimality*. A solution's optimality is determined by the objectives that are defined and the other solutions that it is compared to. When only one objective is specified, determining the optimal solution is trivial: it is the best solution. When more than one objectives are specified, the optimality of a solution is often determined using the concept of Pareto dominance. Consider two solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with objective evaluations $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively. Solution $\boldsymbol{x}_1$ is said to *dominate* solution $\boldsymbol{x}_2$ when two conditions are met:

1. No element in $\boldsymbol{z}_1$ is worse than the corresponding element in $\boldsymbol{z}_2$.

2. At least one element in $\boldsymbol{z}_1$ is better than the corresponding element in $\boldsymbol{z}_2$.

The expression $\boldsymbol{x}_1 \succ \boldsymbol{x}_2$ is used to denote that solution $\boldsymbol{x}_1$ dominates solution $\boldsymbol{x}_2$[1]. Given a solution set $G = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$, solution $\boldsymbol{x}_i \in G$ is considered optimal (or *nondominated*) if there does not exist another solution $\boldsymbol{x}_j \in G$ that dominates $\boldsymbol{x}_i$. Thereby, the Pareto-set P of G is defined as the set containing all optimal solutions in G relative to the given objectives.

$$P = \{\boldsymbol{x}_i \in G \mid (\nexists \boldsymbol{x}_j \in G)[\boldsymbol{x}_j \succ \boldsymbol{x}_i]\} \tag{2.1}$$

Figure 2.1 depicts an example that demonstrates the concept of Pareto dominance, where the Pareto-set has been defined in terms of minimizing both objectives.

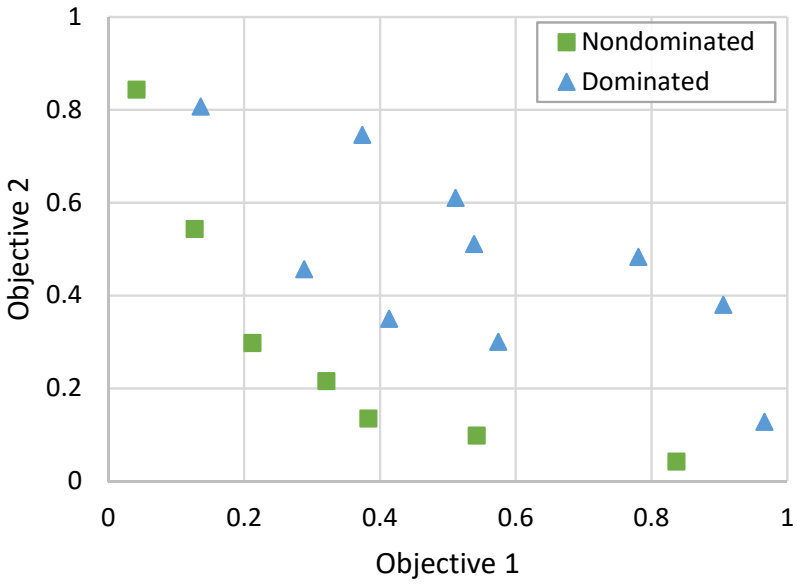


Figure 2.1: Nondominated and dominated solutions defined by a bi-objective evaluation of Pareto dominance.

In design, optimality is often erroneously perceived as synonymous with quality. By its definition, optimality is solely a product of the objective functions that are specified and the solution set considered. Thus given objectives and any solution set

[1]The notation on dominance varies widely in literature, such that both $\boldsymbol{x}_1 \succ \boldsymbol{x}_2$ and $\boldsymbol{x}_1 \prec \boldsymbol{x}_2$ have been used to denote that $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_1$ (*Audet et al.*, 2018). In this document, the former notation is used, as specified in the text.

G, optimal solutions within G can be determined. But therein lies the rub; solutions are deemed optimal relative to the other solutions in G. If all the solutions are bad, the optimal solutions will also be bad, just better than the other bad solutions that they have been compared against.

Therefore without a sufficient context, the assessment that a solution is optimal does not presuppose or imply any evaluation of its quality relative to solutions that were not considered during this assessment. Additionally, the act of finding optimal solutions does not provide an evaluation of the quality of the process used to generate those solutions. Without assessing the quality of the solution generation process, the overall quality of solutions cannot be determined.

Designers are responsible interpreting analyses by looking at "what the results are" and from these, attempting to understand "what they mean" in the context of a given design problem. However, "what the results are" does not explain "how or why they are." By looking at results alone, one cannot determine if the results are predisposed by the processes used to generate them. Without such knowledge, interpreting the results and evaluating their quality, from a design decision perspective, becomes a fool's errand. Both aspects ("what" and "why" the results are) must be considered to inform the interpretation of what results mean; either alone is insufficient. With that in mind, the following section presents the definitions of quality used in this research.

2.2 Defining Two Views of Quality

The discussion of optimality brings to light the fact that an honest assessment of quality requires multiple contextual views. A determination of quality requires assessing both the quality of the solutions in terms of the questions being asked and the quality of the solution generation process so that one can determine if optimal solutions are honestly generated or simply random blind luck. To accomplish this goal,

two different views of quality are defined: *solution-centric* quality (representing the quality of solutions) and *generative* quality (representing the quality of the solution generation process).

2.2.1 Solution-centric Quality

Solution-centric quality refers to how well solutions characterize the *desired* design space in terms of thoroughness, optimality, and diversity. Solution-centric evaluations of quality describe "what results are" and operate primarily within the solution-space and objective-space of a model. They are used to determine if the solutions generated by the tool provide an accurate and sufficient representation of the information required to make design decisions.

In the field of optimization, many metrics have been developed to evaluate and compare different Pareto-sets (*Audet et al.*, 2018; *Li et al.*, 2014, 2015; *Wu and Azarm*, 2001; *Zitzler et al.*, 2003). These metrics are primarily used by optimization practitioners as performance indicators to determine superiority between different optimizers (*Zitzler and Thiele*, 1998; *Wang et al.*, 2016). In optimization, the model is never the focus of the research. While the interplay between models and optimizers is recognized, the model is viewed as an input, and the focus is on developing more superior optimizers to solve them. In contrast, designers are more concerned with the models than with the optimizers used to solve them. Designers need a method for evaluating the predispositions of a model to understand how the generated solutions affect the characterization of the design space.

While solution-centric evaluations of quality may utilize similar metrics to those developed in optimization, the intent behind solution-centric quality is focused on providing a more thorough understanding of the model. In optimization, performance indicators are primarily concerned with optimal solutions. Even in design, much of the current focus is on optimal solutions. However, there are other aspects of

solution-centric quality aside from optimality. For example in early stage design activities, solution feasibility has significant importance to designers. How the model's biases affect the creation of feasible solutions will affect the characterization and understanding of the feasible design space. Therefore, analyzing aspects other than solely optimality may provide additional indicators of quality and a better understanding of the design space; ignoring them would be naive and could cause one to make false conclusions.

2.2.2 Generative Quality

Generative quality refers to how effectively the solution generation process utilizes previous information to inform its progression over time. Generative assessments of quality attempt to explain "why results exist" and are critical to understanding if a tool is producing solutions against its designed intent. Until now, generative metrics have not been considered as a means to evaluate solution quality. This is due to the fact that the quality of solutions is currently not seen as dependent on the quality of the process used to generate them. In the field of optimization, this may not be a concern. However in design, the quality of solution generation can have significant implications on the quality of resultant solutions.

While related, generative evaluations of quality are distinct from temporal solution-centric ones: solution generation implies a temporal process, but evaluating a temporal process does not necessitate an understanding of the process's driving causes. For example, the convergence of an optimization procedure assesses a temporal attribute, but it does not provide an understanding of why the results exist.

2.3 The Necessity of Comparison

Optimality, quality, and bias share a commonality in design: each requires comparison in order to provide any meaningful value. A designer should want to know if

a model's biases are predisposing solutions against its intended application. Bias cannot be understood without a reference; the very idea of bias implies some comparison. Therefore, a comparative context is necessary to analyze, assess, and understand a model's inherent biases. When establishing a context to provide comparisons for evaluating the biases of a model, at least one reference is needed: a baseline. Within this thesis, the baseline contains the execution dynamics and solutions generated when the tool is operated with its original intent and set-points. Given this context, the optimality of the baseline's solutions can be used to provide a measure of quality.

2.4 Capturing Solution Generation Dynamics

In traditional optimization, people often only look at the final set of optimal solutions or the convergence of optimal solutions over time. As generative quality is defined, it evaluates how efficiently the optimizer utilizes previous information to inform its future actions. Therefore to evaluate generative quality, a method is needed to capture the utilization of information within the solution generation process.

Though the methods discussed in Section 1.1.2 present analyses that demonstrate solution predication, these methods cannot be used to evaluate generative quality. They are based on interrogating the structure of a model and draw conclusions from the connectivity within the model: the relationships between its inputs, outputs, constraints, external and internal functions, initialization parameters, etc. Analyzing a tool's structure may provide additional insight to understand how information propagates within it, and may identify potential leading indicators of the tool's tendencies. However, these analyses do not involve the process of generating solutions, and therefore cannot be used to evaluate how solutions are produced over time. The structure of a tool does not provide the information necessary to evaluate the solution generation dynamics within the tool, which is critical to enable an understanding of how and why solutions are created through time. Additionally, these analyses cannot

be utilized when the model's internal structure is unknown, as is the case with many "black-box" design tools.

2.5 Biasing a Solution Generation Process

As stated above, evaluating the biases of a model requires at least one reference: a baseline. However, if only the baseline is known, no conclusions can be made concerning the model's biases. Therefore to evaluate bias, additional references are needed to provide comparisons against the baseline. Specifically, references are needed that enable a designer to draw conclusions regarding the model. Therefore, a method is needed to generate references in such a way that their results reflect a given bias so that the biases of the model can be determined by comparison.

In addition, the evaluation of a model's bias should assess whether that bias is causal or simply correlated. Causation is used to indicate that one event is the result of another event, such that a cause and effect relationship exists between them. On the other hand, correlation is used to measure the probabilistic relationship between the values of In the determination of a model's biased toward producing something requires the ability to exercise a model in ways counter to the baseline so that causation or correlation can be determined. In order to look at this, a dynamic environment is needed where the solution generation process can be subjected to a bias over time. By "pushing" the generation process toward a particular bias, the resulting quality of solutions is used to evaluate if the model is predisposed toward producing solutions corresponding to the given bias.

2.6 Summary

This chapter discussed an alternative view of quality that is critically needed within early stage design. Two types of quality, solution-centric and generative, were

defined and discussed. Within the discussion, the intent and considerations of bias and quality were presented so that an assessment of a model's inherent biases can be evaluated. This chapter outlined what quality means within the context of early stage design tools and provided the justification for the development of several execution views of a model so that a comparative analysis between the model's baseline and alternative execution strategies can be achieved.

CHAPTER III

Developing the Framework

This chapter details the mechanics of the developed framework and how different aspects of this framework address the considerations for evaluating and understanding bias.

1. Evaluating generative aspects of quality requires a solution generation process. While operating on a given model, a Genetic Algorithm (GA) is used generate solutions over time and thereby provide a dynamic environment of the solution generation process associated with the model. Section 3.1 presents foundational information concerning GAs and the types of models used.

2. Evaluating the utilization of information throughout the solution generation process requires a method for capturing and representing how that information is being used over time. Section 3.2 discusses how Hereditary-Amelioration Networks (HANs) are derived from the dynamics within a GA and capture the implied-causality behind solution dynamics by representing temporal, causal relationships between solutions throughout the GA's optimization procedure.

3. Evaluating a model's biases requires the ability to make meaningful comparisons. Section 3.3 presents how the GA is modified to establish the comparative context necessary for a model's biases to be analyzed and understood.

3.1 Background

3.1.1 Objective-based Models

An objective-based model is defined by its inputs $\boldsymbol{x}$, functions $\boldsymbol{f}(\boldsymbol{x})$, objectives $\boldsymbol{\Omega}(\boldsymbol{x})$, and constraints $\boldsymbol{c}(\boldsymbol{x})$. A simple example of a multi-objective model is given in Table 3.1. A set of instantiated inputs (called a *solution*) represents a single point in the model's *solution-space* (also called the *decision-space* (*Zitzler et al.*, 2000)). The model's functions define the mapping from a solution's inputs to its corresponding point in the model's *objective-space*.

Table 3.1: A simple multi-objective model.

Component	**Example**
Inputs	$x_1 \in \mathcal{R}$
	$x_2 \in \mathcal{R}$
	$x_3 \in \mathcal{R}$
	$\boldsymbol{x} = [x_1, x_2, x_3]$
Functions	$f_1(\boldsymbol{x}) = x_1^2 - x_2 x_3^2$
	$f_2(\boldsymbol{x}) = x_1 x_2 x_3 + f_3(\boldsymbol{x})$
	$f_3(\boldsymbol{x}) = x_2^2 - x_1$
	$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x})]$
Objectives	$\Omega_1(\boldsymbol{x}) = \min f_1(\boldsymbol{x})$
	$\Omega_2^*(\boldsymbol{x}) = \max f_2(\boldsymbol{x})$
	$\boldsymbol{\Omega(x)} = [\Omega_1(\boldsymbol{x}), \Omega_2(\boldsymbol{x})]$
Constraints	$c_1^*(\boldsymbol{x}) = x_1 \leq x_2 - x_3$
	$c_2^*(\boldsymbol{x}) = x_3 \geq 1.4$
	$\boldsymbol{c}(\boldsymbol{x}) = [c_1(\boldsymbol{x}), c_2(\boldsymbol{x})]$

The model's objectives define how to assess if a solution is optimal, as discussed in Section 2.1. Though objectives can be specified as minimization or maximization problems, common practice is to express objectives solely as minimization problems. To comply with this standard, a maximization problem, $\max f(\boldsymbol{x})$, is expressed as its corresponding minimization problem, $\min \bar{f}(\boldsymbol{x})$, where $\bar{f}(\boldsymbol{x}) = -f(\boldsymbol{x})$. In the example provided, $\Omega_2^*(\boldsymbol{x})$ would be formatted to be expressed as $\Omega_2(\boldsymbol{x}) = \min \bar{f}_2(\boldsymbol{x})$.

The model's constraints define boundaries that assess a solution's feasibility: a

solution is *infeasible* if it violates a constraint; otherwise, it is *feasible*. Common practice is to express inequality constraints in the form $c(\boldsymbol{x}) \leq 0$. Using algebraic manipulation, the example's constraints $c_1^*(\boldsymbol{x})$ and $c_2^*(\boldsymbol{x})$ would be formatted to be expressed as $c_1(\boldsymbol{x}) = x_1 - x_2 + x_3$ and $c_2(\boldsymbol{x}) = 1 - \frac{x_3}{1.4}$.

To account for the model's constraints, the objective values of infeasible solutions are generally modified by a penalty function (*Goos et al.*, 2007). For example, to ensure that feasible solutions are never dominated by infeasible solutions, an infinitely large penalty can be added to the objective values of infeasible solutions. This represents an extreme version of a penalty function. However, large, discrete penalties can greatly affect an optimizer's ability to find optimal solutions along constraint boundaries. To remedy this, an external penalty function is utilized that penalizes solutions linearly for each violated constraint. The penalty associated with solution $\boldsymbol{x}$ from constraint $c(\boldsymbol{x})$ is given by:

$$g(c(\boldsymbol{x})) = \max(c(\boldsymbol{x}), 0) \tag{3.1}$$

If $\boldsymbol{x}$ is feasible ($c(\boldsymbol{x}) \leq 0$), no penalty is applied to $\boldsymbol{x}$ ($g(c(\boldsymbol{x})) = 0$); if $\boldsymbol{x}$ is infeasible ($c(\boldsymbol{x}) > 0$), a positive penalty is applied to $\boldsymbol{x}$ ($g(c(\boldsymbol{x})) = c(\boldsymbol{x})$). Multiple constraints are accounted for by summing each of their associated penalties.

$$\Phi(\boldsymbol{c}, \boldsymbol{x}) = \sum_{c_i \in \boldsymbol{c}} g(c_i(\mathbf{x})) \tag{3.2}$$

whereby the penalized objective values of solution $\boldsymbol{x}$ are given by:

$$\boldsymbol{R}(\boldsymbol{\Omega}, \boldsymbol{c}, \boldsymbol{x}, \xi) = \boldsymbol{\Omega}(\boldsymbol{x}) + \xi\, \Phi(\boldsymbol{c}, \boldsymbol{x}) \tag{3.3}$$

where ξ is a positive linear scaling factor. Since ξ is positive and $g(c_i(\boldsymbol{x}))$ is greater than or equal to zero, the penalty term $\xi\, \Phi(\boldsymbol{c}, \boldsymbol{x})$ is subsequently also greater than

or equal to zero. When solution $\boldsymbol{x}$ is infeasible, its associated penalty is, by design, a positive number. As $\boldsymbol{\Omega}(\boldsymbol{x})$ represents minimization problems, the addition of a positive penalty means that infeasible solutions receive a worse evaluation.

3.1.2 Genetic Algorithms

Genetic Algorithms (GAs) are iterative, population-based, evolutionary search algorithms commonly used in optimization (*Holland*, 1975, 1992; *Gen et al.*, 2008). Figure 3.1 depicts the GA optimization procedure that is used in this research. Before discussing each of these steps in detail, an overview of the procedure is given.

The goal of a GA is to find optimal solutions to a given optimization problem (in the field of optimization, an objective-base model, such as the one defined in Table 3.1, is often referred to as the optimization problem). In a GA, the model's solution-space is represented by a set (or *population*) of *individuals*, each of which represent a single solution by encoding a set of instantiated inputs to the model. Until its stopping condition is fulfilled, a GA searches the model's objective-space for optimal solutions by iteratively manipulating its population using *genetic operators*. As shown in Figure 3.1, the GA used in this research utilizes two types of genetic operators: crossover and selection. GAs do not have any well-established convergence criteria; the user commonly specifies the stopping condition as a number of iterations to complete. Once its stopping condition is fulfilled, a GA returns the set of optimal solutions that it has found throughout its progression.

Running a GA requires specifying an optimization problem to solve and setting the algorithm's hyperparameters. Hyperparameters are configuration parameters whose values cannot be estimated from data and are instead specified by the user (*Coello Coello et al.*, 2007). The GA hyperparameters utilized in this research are presented in Table 3.2 and each will be discussed as they become relevant. The remainder of this section details each step of the optimization procedure shown in Figure 3.1.

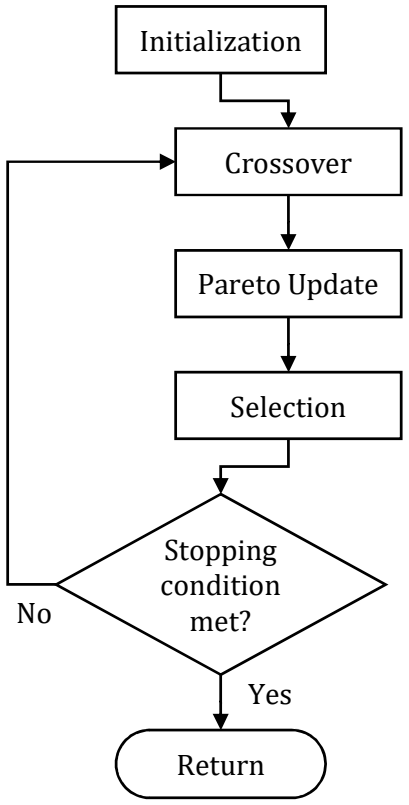


Figure 3.1: The optimization procedure of a GA.

Table 3.2: Common hyperparameters of a GA.

Hyperparameter	Description
Population size (N)	The number of individuals that comprise the GA's population.
Probability of crossover (p_{cx})	The probability that a selected individual will engage in crossover.
Blend parameter (α)	A parameter used in blend crossover.
Tournament size (k)	A parameter used in tournament selection.
Number of epochs (E)	The total number of iterations that the GA completes before finalizing.

Step 1: Initialization

Before the population can be used to search the model's objective-space, it must be initialized. This entails filling the population with N individuals, where hyperparameter N is specified as an integer value. As stated, each individual represents a solution in the model's solution-space by encoding a set of input values. The method by which these values are determined varies; they may be assigned randomly, dictated by the user, or chosen from a set of provided inputs. In this research, the initial input values of each individual were chosen from a large set of feasible inputs, which was generated by randomly sampling the model's solution-space. After assigning input values for each of the individuals, their corresponding constraint, penalty, and objective values are calculated using the given model, as discussed in Section 3.1.1. Table 3.3 presents an example of a population's first three individuals, utilizing the model defined in Table 3.1.

Table 3.3: A population's first three individuals with their corresponding input, constraint, penalty, and objective values.

Individual	Inputs, $\boldsymbol{x}$			Contraints, $\boldsymbol{c}(\boldsymbol{x})$		Penalty	Objectives, $\boldsymbol{\Omega}(\boldsymbol{x})$	
	x_1	x_2	x_3	$c_1(\boldsymbol{x})$	$c_2(\boldsymbol{x})$	$\Phi(\boldsymbol{c}, \boldsymbol{x})$	$\Omega_1(\boldsymbol{x})$	$\Omega_2(\boldsymbol{x})$
0	1.5	2.5	-1.0	-2.00	1.71	1.71	-0.25	-1.00
1	-0.7	1.3	1.5	-0.50	-0.07	0.00	-2.44	-1.03
2	2.4	1.5	2.3	3.20	-0.64	3.20	-2.17	-8.13
⋮		⋮			⋮	⋮		⋮

After the population has been created, the GA establishes its initial Pareto-set using Equation 2.1 and the objectives $\boldsymbol{\Omega}(\boldsymbol{x})$ specified by the model. Additionally, the Pareto-set is created using only feasible ($\Phi(\boldsymbol{x}) = 0$) individuals from the population, to guarantee that optimal solutions are never infeasible.

Step 2: Crossover

After initialization, the GA begins its core iterative cycle. At the start of this cycle, the GA's current epoch is incremented: $t = t + 1$ (the initialization step is considered as $t = 0$). Then as shown in Figure 3.1, each epoch begins with the crossover operator, which is responsible for creating new individuals. Since individuals represent points in the model's solution-space, crossover is the mechanism through which new search points for the model are determined.

In *blend crossover*, new individuals are created by the following process. Each individual in the current population $Q(t)$ is selected in turn, and has a probability p_{cx} to engage in crossover. When an individual engages in crossover, the selected individual $\boldsymbol{p}_1$ is paired with a different individual $\boldsymbol{p}_2$, chosen randomly from the population. This pair of individuals (called *parents*) are used to create a new individual $\boldsymbol{c}_1$ (called a *child*) by assigning input values to $\boldsymbol{c}_1$ as a combination its parents' input values. For each of i inputs, the child's input value $\boldsymbol{c}_i$ is determined by:

$$\boldsymbol{c}_i = (1 - \xi)\boldsymbol{p}_{1i} + \xi\boldsymbol{p}_{2i} \tag{3.4}$$

where ξ is given by:

$$\xi = (1 + 2\alpha)\eta - \alpha \tag{3.5}$$

where η is random number in the range $[0, 1]$, and α is the blend parameter. The parameter ξ dictates how much of the child's input value $\boldsymbol{c}_i$ comes from its first parent $\boldsymbol{p}_{1i}$ and how much comes from its second parent $\boldsymbol{p}_{2i}$. For example when $\xi = 0$, the child's input value $\boldsymbol{c}_i$ is inherited completely from $\boldsymbol{p}_{1i}$. The positive blend parameter α scales the random number η about 0.5, such that ξ effectively becomes a random number in the range $[-\alpha, 1 + \alpha]$. When $\alpha > 0$, the input value assigned to the

child can extend beyond the ranges of its parents input values, enabling the GA to more effectively search the breadth and extents of the model's solution-space. After assigning a child's inputs in this manner, its corresponding constraint, penalty, and objective values are calculated using the given model, similarly to the evaluation discussed in the initialization step. After each individual is given a chance to engage in crossover, the newly created children are added to the GA's current population $Q(t)$, resulting in population growth. The expected amount of population growth is a function of the population size N and the probability of crossover p_{cx}. Since p_{cx} is the probability that an individual engages in crossover and creates a child, the expected number of children created across N individuals is equal to Np_{cx}.

In this way, the GA creates new search points each epoch; it decides where to search based on where it's been. As a final note before discussing the next step, most GAs utilize a mutation operator that is applied directly after the inputs of a child are assigned. The mutation operator it is not utilized in the current research and therefore is not discussed here.

Step 3: Pareto Update

After crossover, the GA's Pareto-set P is updated using the feasible children that were created during crossover this iteration. Using Equation 2.1, the optimality of each child is assessed by comparing it to individuals in P, which is updated accordingly. If the child is dominated by one or more individuals in P, it is not optimal and is not added to P. If the child is nondominated by individuals in P, it is optimal and is added to P. Furthermore, if a nondominated child dominates one or more individuals in P, the dominated individuals are subsequently removed from P. Given this implementation, the size of the Pareto-set is not constrained and can continue to grow throughout the progression of the GA.

Step 4: Selection

After updating the Pareto-set, the selection operator is used to reduce the inflated population in order to maintain a constant population size at each iteration of the GA. With population size N and tournament size k specified as hyperparameters, *tournament selection* operates as follows. The next epoch's population, $Q(t+1)$, is initialized as an empty set ($Q(t+1) = \{\}$). Then until $Q(t+1)$ contains N individuals, $Q(t+1)$ is filled as follows:

1. A subset T (called a *tournament*) is filled with k individuals, chosen randomly without replacement from the current population $Q(t)$.

2. The individuals in T are evaluated by a penalized ranking function $\boldsymbol{f_r}(\boldsymbol{x})$, where $\boldsymbol{f_r}$ is traditionally defined as the problem's objective functions $\boldsymbol{\Omega}$. Using Equation 3.3, this penalized ranking function can be expressed as:

$$\boldsymbol{R}(\boldsymbol{f_r}, \boldsymbol{c}, \boldsymbol{x}, t) = \boldsymbol{f_r}(\boldsymbol{x}) + t\, \Phi(\boldsymbol{c}, \boldsymbol{x}) \tag{3.6}$$

   where $\Phi(\boldsymbol{c}, \boldsymbol{x})$ is given by Equation 3.2. Here, the GA's current epoch t is used as the scaling factor, so that infeasible individuals receive higher penalties as the GA progresses.

3. The nondominated individual in T with respect to $\boldsymbol{R}(\boldsymbol{f_r}, \boldsymbol{c}, \boldsymbol{x}, t)$ is selected. If T contains multiple nondominated individuals, one of them is selected at random.

4. The selected individual is removed from $Q(t)$ and added to $Q(t+1)$. Since the selected individual no longer resides in $Q(t)$, subsequent tournaments cannot contain individuals that have already been selected. In other words, this process represents selection without replacement, whereby $Q(t+1)$ does not contain any duplicate individuals.

While the selection operator does not create additional search points, it directly affects solution generation process by determining which individuals remain in the population for the next iteration of the GA. It is the mechanism that enforces "natural selection" with the purpose of finding additional optimal solutions.

Step 5: Iteration and Finalization

As shown in Figure 3.1, steps 2 through 4 are repeated until the GA's stopping condition has been satisfied. As stated, GAs do not have any well-established convergence criteria, and instead the user commonly specifies the stopping condition as a total number of epochs E to complete. Therefore until $t = E$, the GA continues to create new solutions, update its optimal solutions, and remove solutions from its population at every epoch. Once its stopping condition is satisfied, the GA returns its Pareto-set, containing the optimal solutions that it found throughout its progression.

3.2 Capturing Solution Dynamics

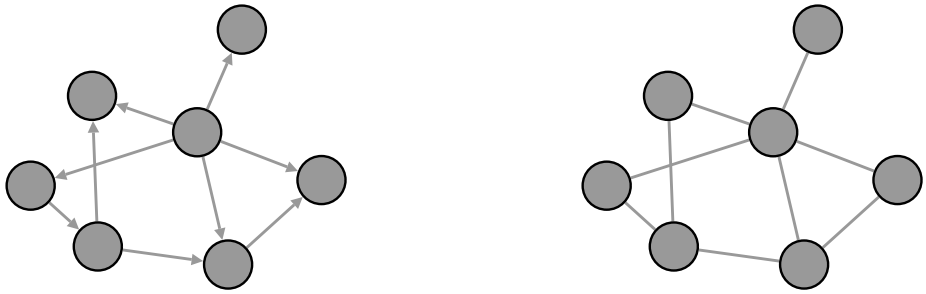
As stated in Chapter II, effectively establishing the impact of bias or predication of an early stage design tool requires a method for creating and capturing the evolutionary dynamics of the solution generation process. The GA's process of creating and removing solutions at each epoch embodies how the GA represents an iterative evolution of solutions over time. To model this generative process, HANs are derived from the dynamics within the GA as it progresses through time.

To understand the effects of bias, it is not only the final solutions that must be investigated, but the generative process as well. Rather than assuming that intermediate solutions are solely a means to the final Pareto-set, different types of relationships between intermediate solutions are represented using networks. HANs capture the implied-causality behind the evolution of solutions by representing temporal, causal relationships throughout the progression of the GA. The networks are termed *heredi-*

tary, in reference to the relationship of information between individuals (parents and children), and *amelioration*, in reference to the progressive improvement of solutions; both terms denote the temporal nature of the iterative solution generation process. The development of HANs is a novel and unique contribution of this thesis.

3.2.1 Creating Hereditary-Amelioration Networks

This section details how two types of HANs are created during a GA's optimization procedure, namely the Parent-Child Network (PCN) and the Parent-Parent Network (PPN). As stated, HANs represent relationships within the solution generation process as networks. A network is simply a collection of points (called *nodes*) that are connected by lines (called *edges*) *Newman* (2003, 2010). As demonstrated in Figure 3.2, a network can be directed or undirected, as determined by its edges. In directed networks, edges specify a direction, pointing from one node and to another. In undirected networks, edges do not specify a direction and represent bi-directional relationships between nodes.



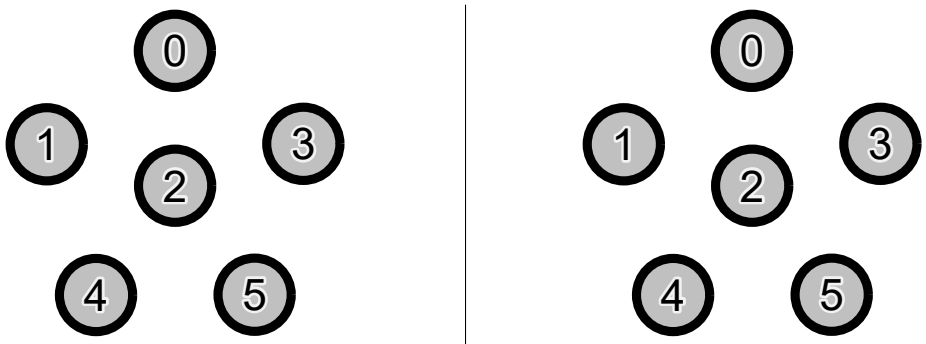
Directed network Undirected network

Figure 3.2: Two different types of networks. The edges in an *undirected* network (*right*) do not signify directional relationships, whereas those in a *directed* network (*left*) do.

Mirroring the steps discussed in Section 3.1.2, the following sections detail how HANs are created during the GA's optimization procedure.

Step 1: Initialization

Prior to initializing the population, both HANs are initialized as empty networks, having no nodes or edges. As individuals are created during the GA's initialization step, a node is added to the HANs for each. Thereby, individuals in the GA are represented by nodes in the two HANs. Figure 3.3 depicts the PCN and PPN after initializing a population of six individuals.



Parent-Child Network | Parent-Parent Network

Figure 3.3: HANs after initializing a population containing six individuals. Individuals in the GA are represented by nodes in the PCN and PPN.

Each node contains the attributes of its corresponding individual: its input, constraint, penalty, and objective values (e.g., as shown in Table 3.3). In addition, each node encodes temporal attributes that indicate when it is created (t_{start}), when it is dominated ($t_{dominated}$), and when it leaves the population (t_{end}). In this case when the population is initialized ($t = 0$), all nodes in the network set $t_{start} = 0$. After the GA establishes its initial Pareto-set, any dominated nodes set $t_{dominated} = 0$. Table 3.4 presents the temporal attributes of the six nodes from Figure 3.3. As indicated by $t_{dominated}$ in the table, nodes 1 and 2 are either dominated by other nodes or infeasible ($\Phi(\boldsymbol{c}, \boldsymbol{x}) > 0$) and therefore cannot be contained in the Pareto-set.

Table 3.4: Temporal nodal attributes after initialization at $t = 0$.

Node	t_{start}	$t_{dominated}$	t_{end}
0	0	-	-
1	0	0	-
2	0	0	-
3	0	-	-
4	0	-	-
5	0	-	-

Step 2: Crossover

When individuals are created during crossover, nodes are added to the HANs for each of them, and these nodes set t_{start} equal to the GA's current epoch t. Continuing the example, Table 3.5 contains the temporal attributes of three new nodes created during crossover at $t = 1$. Though individuals are similarly represented as nodes in both networks, the networks' edges capture and represent different relationships. When a node is created during crossover, the PCN creates an edge *to* it *from* each of its parents, while the PPN creates an edge *between* the child's parents. Thereby, the PCN is a directed network that represents the mapping from parents to children, and the PPN is an undirected network that represents the mapping between parents.

Table 3.5: Temporal nodal attributes after crossover at $t = 1$.

Node	t_{start}	$t_{dominated}$	t_{end}
⋮		⋮	
6	1	-	-
7	1	-	-
8	1	-	-

Continuing the example, Figure 3.4 depicts the development of both HANs during crossover at $t = 1$. In the figure, blue nodes indicate parents that were chosen to engage in crossover and create children. In this example, node parent pair (2,3) created node 6, (0,3) created 7, and (2,5) created 8. Representing these three children, nodes 6, 7, and 8 are created and added to each of the networks. Since an edge in the PCN points *to* a child *from* its parent, edges are created to child 6 from parents

2 and 3, to 7 from 0 and 3, and to 8 from 2 and 5. Since an edge in the PPN points *between* parents, edges are created between node pairs (2,3), (0,3), and (2,5).

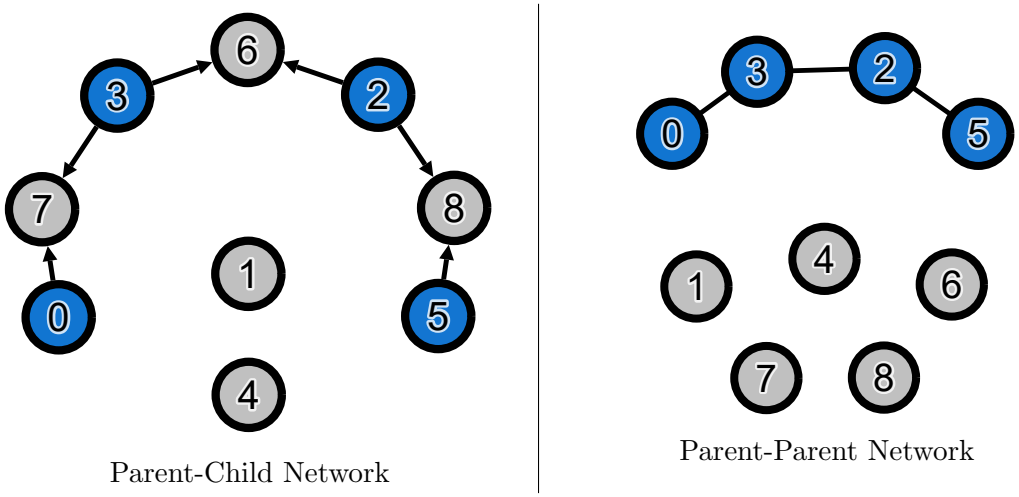


Figure 3.4: Development of HANs during crossover at $t = 1$. Nodes are added to the networks for each newly created individual. Blue nodes indicate parents that were chosen to engage in crossover and create children. In the PCN, edges are added *to* child nodes *from* parent nodes. In the PPN, edges are added *between* parent nodes.

This describes how edges and nodes are created in the HANs, but does not immediately convey their significance in terms of the GA's solution generation process. When parents engage in crossover and create a child, the child's input values are assigned as a function of its parents' inputs values. Thereby, a child inherits information contained in both of its parents, and parents influence their children through this inheritance of information. Through its edges, the Parent-Child Network captures this directional influence of information and represents the genealogy of solutions created throughout the GA. The Parent-Parent Network captures the connectivity between parents and represents the community of information that is utilized to generate new solutions throughout the GA. Both HANs provide a structured representation of the temporal solution generation process, though each identifies different aspects.

Step 3: Pareto Update

When the Pareto-set P is updated, any newly dominated nodes set $t_{dominated}$ to the current epoch t. This includes children that were created during crossover this epoch as well as any nodes in P that are dominated by these children this epoch. Table 3.6 presents the updated temporal attributes of the nine nodes from Figure 3.4. As indicated by $t_{dominated}$ in the table, nodes 4 and 7 are either infeasible or dominated by other nodes at $t = 1$. Since node 7 was also created at $t = 1$, it represents a solution that was never optimal. In contrast, node 4 was nondominated at $t = 0$ and therefore represents a previously optimal solution.

Table 3.6: Temporal nodal attributes after the Pareto-set is updated at $t = 1$.

Node	t_{start}	$t_{dominated}$	t_{end}
0	0	-	-
1	0	0	-
2	0	0	-
3	0	-	-
4	0	1	-
5	0	-	-
6	1	-	-
7	1	1	-
8	1	-	-

Step 4: Selection

Though selection reduces the GA's population by effectively removing individuals, nodes are not removed from the HANs. Therefore, these networks continue to grow each iteration of the GA. However when an individual is removed from the population, its corresponding node sets t_{end} to the current epoch t. Continuing the previous example, Figure 3.5 depicts both HANs at $t = 1$ after the next population's nodes have been selected. In the figure, green nodes indicate those that were selected to remain in the population at $t = 1$. Therefore, nodes 2, 4, and 7 are not chosen during selection and their respective individuals are subsequently removed from the

population. The updated temporal attributes of these nodes are reflected in Table 3.6.

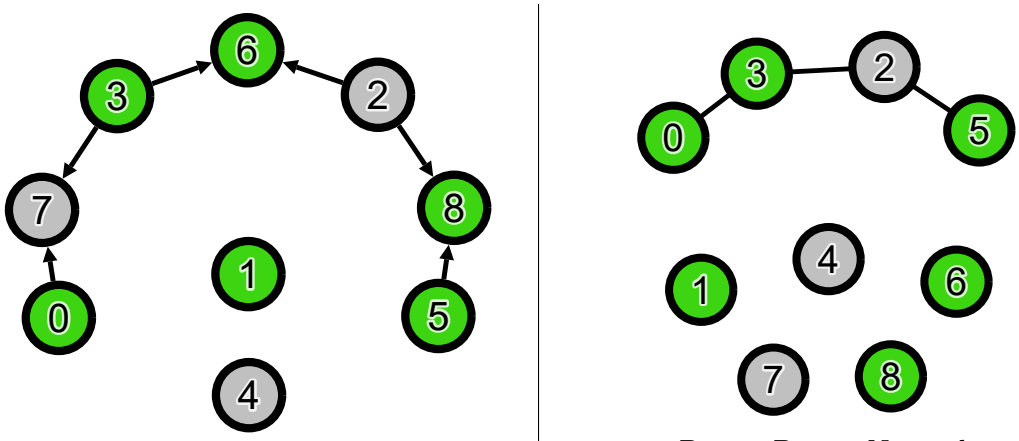


Figure 3.5: Development of HANs during selection at $t = 1$. Green nodes indicate those that were selected to remain in the population.

Table 3.7: Temporal nodal attributes after selection at $t = 1$.

Node	t_{start}	$t_{dominated}$	t_{end}
0	0	-	-
1	0	0	-
2	0	0	1
3	0	-	-
4	0	1	1
5	0	-	-
6	1	-	-
7	1	1	1
8	1	-	-

The selection step does not alter the HANs' structures through the addition or removal of nodes or edges. However, it does influence which individuals remain in the GA's population, and therefore has significant implications on the solution dynamics within the procedure. In the example, nodes 4 and 7 did not create children, and since they are removed from the population at $t = 1$, they can never be chosen to create children in future epochs. Therefore, the information contained in these nodes is never utilized, and they have no influence on the process. In Figure 3.5, this effect

is captured by the PPN, where no edges are connected to nodes 4 and 7. Although node 2 is also removed from the population, it parented two children: nodes 6 and 8, as shown in the PCN. While node 2 cannot be chosen to create children in future epochs, information contained in it is inherited by its children (and subsequently by its "grand-children", etc.). Therefore, a node continues to influence the process for as long as one of its descendants exists within the population.

Step 5: Iteration and Finalization

As discussed in Section 3.1.2, steps 2 through 4 are repeated until iterating for the specified number of epochs E. Therefore, the HANs continue to grow throughout the duration of the GA. To demonstrate this growth and reiterate the main steps in creating HANs, the presented example is continued into epoch 2.

After selection at $t = 1$, the stopping condition is not met, and epoch 2 begins with crossover. Figure 3.6 depicts the development of both HANs during crossover at $t = 2$. As shown previously in Figure 3.5, nodes 2, 4, and 7 were not chosen during selection at $t = 1$ and were removed from the population. While these nodes still exist in the HANs, they cannot be chosen to engage in crossover. To indicate this, these nodes are shaded red in Figure 3.6. Again, blue nodes indicate parents that were chosen to engage in crossover and create children. In the example at $t = 2$, node parent pairs (3,5), (3,0), and (5,0) created child nodes 9, 10, and 11, respectively. To represent these children, nodes 9, 10, and 11 are created and added to each HAN. Since an edge in the PCN points *to* a child *from* its parent, edges are created to child 9 from parents 3 and 5, to 10 from 3 and 0, and to 11 from 5 and 0. Since an edge in the PPN points *between* parents, edges are created between node pairs (3,5), (3,0), and (5,0).

After crossover and updating the Pareto-set, Figure 3.7 depicts selection at $t = 2$. Again, red nodes indicate those that were removed from the population at $t = 1$ and

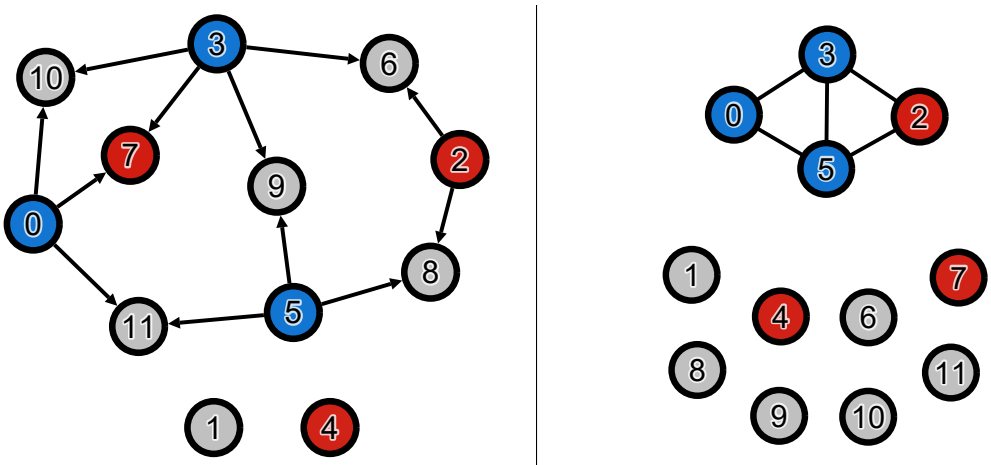
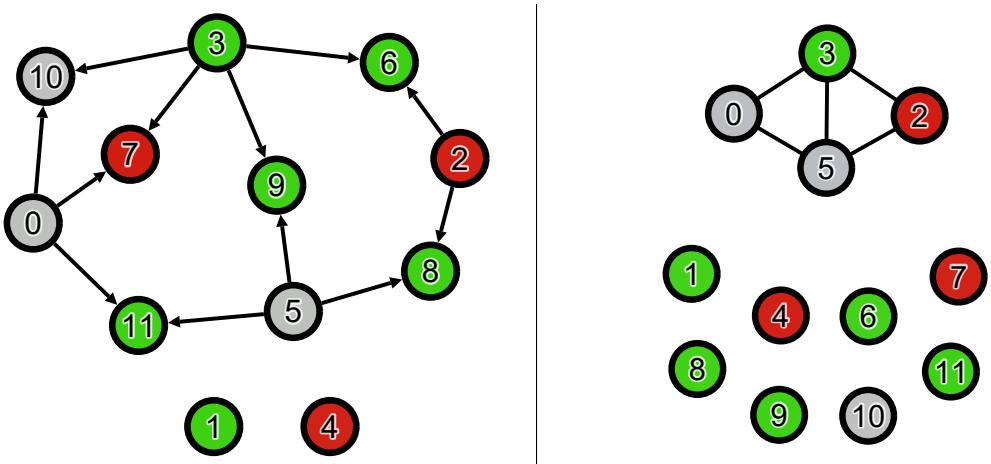


Figure 3.6: Development of HANs during crossover at $t = 2$. Red nodes indicate those that were removed from the population at $t = 1$ and therefore cannot be chosen to engage in crossover. Blue nodes indicate parents that were chosen to engage in crossover and create children at $t = 2$.

therefore cannot be selected this epoch; green nodes indicate those that were selected to remain in the population at $t = 2$. Therefore, nodes 0, 5, and 10 are not chosen during selection and their respective individuals are subsequently removed from the population at $t = 2$. Since node 10 is removed from the population and did not create any children, its information is never utilized, and it has no influence on the process. Although nodes 0 and 5 are removed from the population, their influence on the process continues through their children. However while node 5 has three existing descendants, node 0 only has one. In order for node 0 to continue influencing the process, node 11 (or one of its children) will need to be selected at $t = 3$.

Once the GA has completed E epochs, the GA returns the two resulting HANs that were created during the process. To provide an intuitive visualization of these networks, a network layout was developed. Respectively, Figures 3.8 and 3.9 utilize this layout to depict the aggregate PCN and PPN after completing 40 iterations of the continued example. One of the layout's aspects is that the progression of the GA is represented by connected nodes as roughly following the passage of time on



Parent-Child Network

Parent-Parent Network

Figure 3.7: Development of HANs during selection at $t = 2$. Red nodes represent those that were removed from the population at $t = 1$. Green nodes indicate those that were selected to remain in the population at $t = 2$.

a clock-face. Thereby, the GA is initialized at 1:00 and iterates clockwise before finalizing at 11:00. In the figures, this effect is demonstrated by the roughly linear increase of connected node labels while moving clock-wise along the networks. Any unconnected nodes are positioned in the center of the network; these do not indicate time. The layout's representation of time is approximate since other aspects of the layout necessarily detract from this goal; these are discussed in Chapter IV as they become relevant.

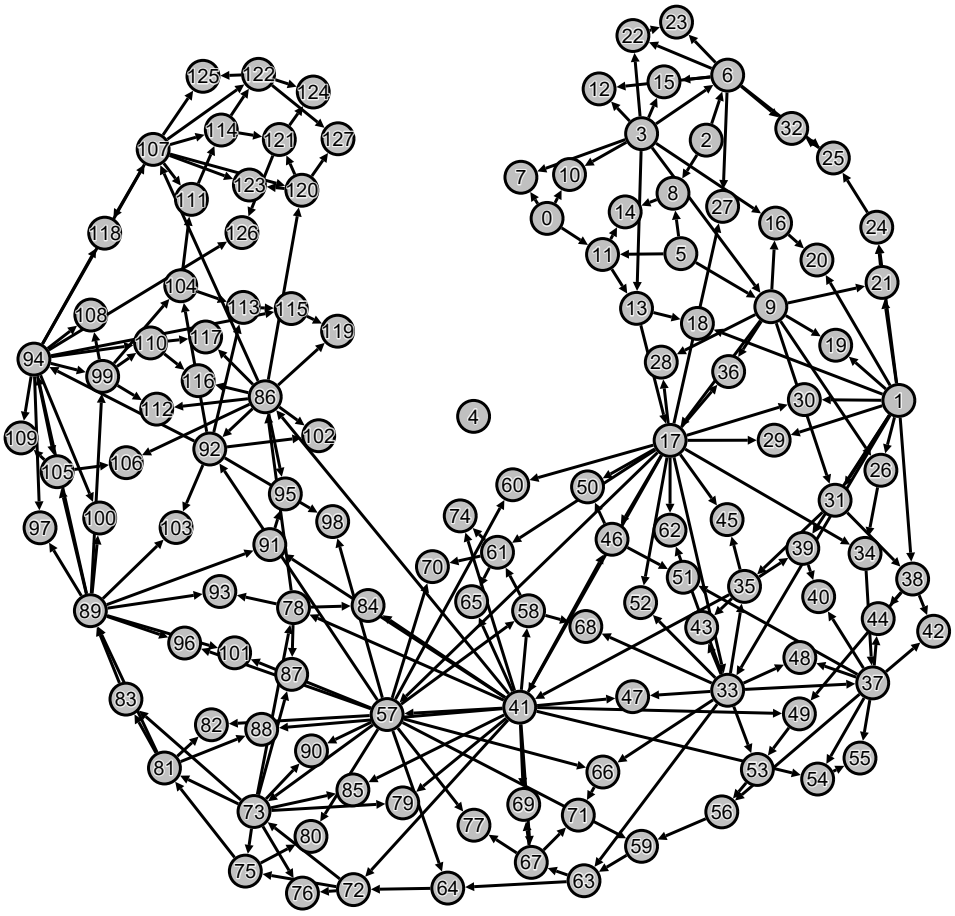


Figure 3.8: An aggregate Parent-Child Network.

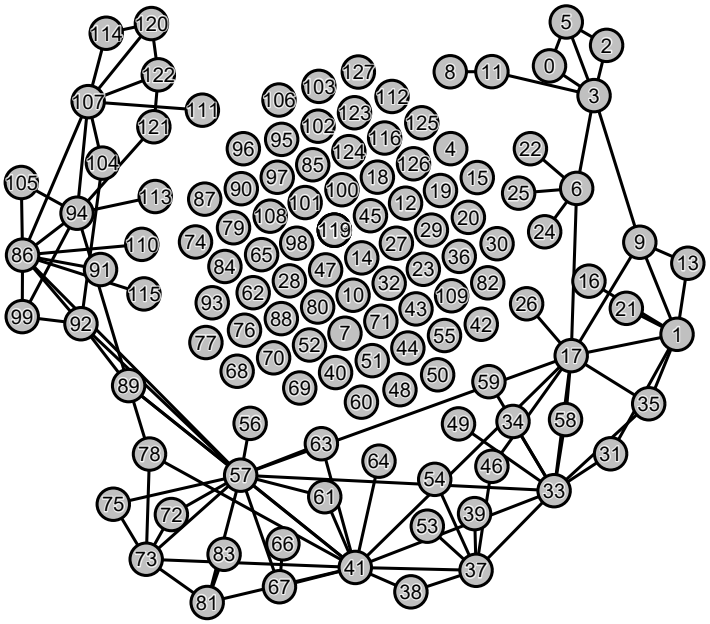


Figure 3.9: An aggregate Parent-Parent Network.

3.2.2 Deriving the Parent-Parent Network

While the creation of the Parent-Parent Network was detailed to provide an intuitive understanding for what it represents, it does not need to be manually created in this fashion. Instead, it can be derived directly using the *adjacency matrix* of the Parent-Child Network. In network theory, the adjacency matrix of a network is defined as a square matrix where rows and columns represent nodes within the corresponding network. For example, a network with n nodes has an $n \times n$ adjacency matrix where row i refers to node i and column j refers to node j. Entries within the adjacency matrix denote the existence of edges between nodes. As shown below, A_{ij} equals one if an edge exists from i to j, and zero otherwise. By this definition, the adjacency matrix of an undirected network is necessarily symmetric.

$$A_{ij} = \begin{cases} 1 & \text{if there exists an edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The derivation of the PPN is most easily demonstrated using an example. For the PCN depicted in Figure 3.4, its adjacency matrix can be constructed using Equation 3.7 as:

$$\boldsymbol{A}^{PCN} = \begin{bmatrix} 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{3.8}$$

As seen in the figure, the only edge that exists from node 0 is to node 7. This is captured in the adjacency matrix above where $A_{07}^{PCN} = 1$. The adjacency matrix of

the PCN can be expressed in terms of its rows as:

$$\boldsymbol{A}^{PCN} = \begin{bmatrix} \boldsymbol{r}_0 \\ \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \\ \boldsymbol{r}_4 \\ \boldsymbol{r}_5 \\ \boldsymbol{r}_6 \\ \boldsymbol{r}_7 \\ \boldsymbol{r}_8 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.9}$$

where row vector $\boldsymbol{r}_i$ indicates nodes that were parented by node i. For example, $\boldsymbol{r}_0$ indicates that node 7 was parented by node 0. The adjacency matrix of the PCN can also be expressed in terms of its columns as:

$$\boldsymbol{A}^{PCN} = \begin{bmatrix} \boldsymbol{c}_0 & \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_3 & \boldsymbol{c}_4 & \boldsymbol{c}_5 & \boldsymbol{c}_6 & \boldsymbol{c}_7 & \boldsymbol{c}_8 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.10}$$

where the column vector $\boldsymbol{c}_i$ indicates the parents of node i. For example, $\boldsymbol{c}_6$ indicates the parents of node 6 are nodes 2 and 3. However, this is precisely the information needed to create an edge in the PPN. Therefore given this representation, the adjacency matrix of the Parent-Parent Network $\boldsymbol{A}^{PPN}$ can be derived. First, $\boldsymbol{A}^{PPN}$ is initialized as an $n \times n$ matrix where $A_{ij}^{PPN} = 0$ for all i and j. Then for each column vector $\boldsymbol{c}_i$ of $\boldsymbol{A}^{PCN}$, the non-zero elements of $\boldsymbol{c}_i$ which indicate inter-parental relationships are used to define non-zero elements of $\boldsymbol{A}^{PPN}$. Continuing the example from Equation 3.10, the non-zero elements of column vectors $\boldsymbol{c}_6$, $\boldsymbol{c}_7$, and $\boldsymbol{c}_8$ define the following non-zero elements of $\boldsymbol{A}^{PPN}$:

$$\boldsymbol{c}_6 \Rightarrow A_{23}^{PPN} = A_{32}^{PPN} = 1 \tag{3.11}$$

$$\boldsymbol{c}_7 \Rightarrow A_{03}^{PPN} = A_{30}^{PPN} = 1 \tag{3.12}$$

$$\boldsymbol{c}_8 \Rightarrow A_{25}^{PPN} = A_{52}^{PPN} = 1 \tag{3.13}$$

which produces the following adjacency matrix for the PPN:

$$\boldsymbol{A}^{PPN} = \begin{bmatrix} 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&1&0&0&0 \\ 1&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{3.14}$$

Inspecting Figure 3.4, Equation 3.14 indeed reproduces the depicted PPN, with undirected edges between node pairs (2,3), (3,0), and (5,2). This demonstrates that the PPN can be derived from the PCN; however, it should be noted that this operation is not invertible, i.e. $\boldsymbol{A}^{PCN} \Rightarrow \boldsymbol{A}^{PPN}$ but $\boldsymbol{A}^{PPN} \not\Rightarrow \boldsymbol{A}^{PCN}$.

3.3 Establishing a Comparative Context

As discussed in Section 2.3, evaluating bias requires comparisons; it cannot be understood without a reference. This section details the how a comparative context is established so that a model's inherent biases may be analyzed, assessed, and understood. Section 3.3.1 details how a GA is modified by implementing alternative ranking functions that can intentionally bias the solution generation process within the GA. Section 3.3.2 details how these ranking functions are used to define reference and biasing cases. For each of these cases, HANs are created using the modified GA; these results are then comparatively analyzed to evaluate a model's innate tendencies. However, the ranking functions do not affect the macro behavior of the GA, as will be discussed. Section 3.3.3 defines temporal network subsets that are used in Chapter IV to differentiate results and demonstrate the model's biases. Lastly, Section 3.3.4 discusses how the GA's tournament size affects the influence imposed by ranking functions.

3.3.1 Alternative Ranking Functions

As discussed in step 4 of Section 3.1.2, tournament selection utilizes a ranking function $\boldsymbol{f_r}$ such that for each tournament T, the nondominated individual with respect to $\boldsymbol{R}(\boldsymbol{f_r}, \boldsymbol{c}, \boldsymbol{x}, t)$ (given by Equation 3.6) is selected to remain in the population. Thereby, $\boldsymbol{f_r}$ is the mechanism through which the GA influences the evolution of its population over time. As previously stated, $\boldsymbol{f_r}$ is traditionally defined as the model's objectives $\boldsymbol{\Omega}$. In this case for each tournament T, the nondominated individual in T with respect to model's penalized objectives is selected to remain in the population. Over time, this promotes the GA's search of the model's objective-space toward more optimal solutions.

Since $\boldsymbol{f_r}$ directly influences the population's evolution over time, it also provides a means to intentionally bias the solution generation process. Rather than enforcing that $\boldsymbol{f_r}$ be defined as $\boldsymbol{\Omega}$, the GA is modified so that $\boldsymbol{f_r}$ can be defined as an alternative ranking function. If an alternative ranking function is used, it must be specified as a hyperparameter before running the GA. Aside from this modification, the GA's tournament selection operates in the same way discussed in Section 3.1.2. The penalized ranking function $\boldsymbol{R}(\boldsymbol{f_r}, \boldsymbol{c}, \boldsymbol{x}, t)$ is still used to rank individuals in each tournament. However rather that being influenced by the model's objectives, the population's temporal evolution is influenced by the alternative ranking function that has been defined.

3.3.2 Reference and Biasing Cases

Modifying a GA to utilize an alternative ranking function is not done for reasons of optimality. On the contrary, it may be the case that the alternative ranking function finds fewer optimal solutions, does not progress the Pareto-set, or creates a entire population of infeasible individuals. However, this is precisely its purpose: to impose a bias on the model and analyze the response of the solution generation process. By

itself, implementing a single alternative ranking function is not particularly useful or informative. This is due to the fact that analyzing a model's inherent biases requires a comparative context.

To establish these comparisons, multiple alternative ranking functions are used to create reference and biasing cases. Each case implements a different ranking function that influences the solution generation process within the GA. Reference cases are created to provide a base context for interpreting results of the biasing cases. This research defines and utilizes two distinct reference cases: the baseline case and the random case.

- The baseline case provides an "ideal" reference by representing the traditional application of the model. The baseline case directly favors solutions that are optimal with respect to the model in question. Therefore, the ranking function of the baseline case $\boldsymbol{f_r^B}$ is defined as the model's objectives $\boldsymbol{\Omega}$.

$$\boldsymbol{f_r^B}(\boldsymbol{x}) = \boldsymbol{\Omega}(\boldsymbol{x}) \tag{3.15}$$

  Thereby for each tournament T chosen during the GA's selection procedure, the dominating individual with respect to $\boldsymbol{R}(\boldsymbol{f_r^B}, \boldsymbol{c}, \boldsymbol{x}, t)$ is selected to remain in the population. If T contains multiple non-dominated individuals, one of them is selected at random.

- The random case provides a reference by representing a completely unguided search. The random case favors no particular solution, and the ranking function of the random case, $\boldsymbol{f_r^R}$, does not actually rank individuals. Instead, for each tournament T chosen during the GA's selection procedure, a random individual in T is selected to remain in the population.

These reference cases can be constructed for any objective-based model, and they do not require any additional input from the designer: the baseline case is defined by

the model and the random case is self-contained. In contrast, a biasing case require the designer to define its ranking function. Since this ranking function will bias the solution generation process within the GA, it should be defined to represent a particular bias that is being investigated. To define this ranking function, a designer may leverage their intuition, knowledge of relevant physical phenomena, or other analyses. Future research will investigate methods for implementing biasing cases that do not require such manual definitions by the designer.

After defining reference and biasing cases, HANs are generated for each case using the modified GA and its corresponding ranking function. In addition, this process is performed a number of times for each, to obtain statistical quantification of the results. In this research, the number of trials completed for each case was specified by the user; however, this number could also be assessed dynamically based on results (e.g., desired 95% confidence interval).

3.3.3 Temporal Network Subsets

The ranking functions do not affect the macro behavior of the GA; on average, the HANs create the same number of nodes and edges at each epoch, regardless of ranking function. However the goal is not to assess this macro behavior but instead to understand how an imposed bias affects the solution generation process within the GA. The analyses presented in Chapter IV demonstrate the effects of these biases by differentiating results using temporal network subsets, which are defined here. These subsets contain nodes from a given HAN based on conditions defined by the attributes of each node.

The noncumulative super-set, $S(t)$, contains all solutions that exist in the population at time t. For a node in G to reside in $S(t)$, it must have been created

$(t_{start}(x) \leq t)$ and must be currently active $(t \leq t_{end}(x))$.

$$S(t) = \{x \in G \mid t_{start}(x) \leq t \leq t_{end}(x)\} \tag{3.16}$$

The cumulative super-set $S_c(t)$ contains all solutions that have been created by time t. For a node in G to reside in $S_c(t)$, it must have been created $(t_{start}(x) \leq t)$. In terms of $S(t)$, $S_c(t)$ can be expressed as the union of all $S(\tau)$ for $0 \leq \tau \leq t$.

$$S_c(t) = \{x \in G \mid t_{start}(x) \leq t\} \tag{3.17}$$

$$= \bigcup_{\tau=0}^{t} S(\tau) \tag{3.18}$$

The noncumulative Pareto-set, $P(t)$, contains all optimal solutions at time t. For a node in G to reside in $P(t)$, it must have been created $(t_{start}(x) \leq t)$ and must be currently non-dominated $(t < t_{dominated}(x))$.

$$P(t) = \{x \in G \mid t_{start}(x) \leq t < t_{dominated}(x)\} \tag{3.19}$$

The cumulative Pareto-set $P_c(t)$ contains all solutions that are or have been optimal at time t. For a node in G to reside in $P_c(t)$, it must have been created $(t_{start}(x) \leq t)$ and must have been non-dominated at that time $(t_{start}(x) \neq t_{dominated}(x))$. In terms of $P(t)$, $P_c(t)$ can be expressed as the union of all $P(\tau)$ for $0 \leq \tau \leq t$.

$$P_c(t) = \{x \in G \mid (t_{start}(x) \leq t) \wedge (t_{start}(x) \neq t_{dominated}(x))\} \tag{3.20}$$

$$= \bigcup_{\tau=0}^{t} P(\tau) \tag{3.21}$$

The noncumulative feasible-set $F(t)$ contains all feasible solutions that exist in the population at time t. For a node in G to reside in $F(t)$, it must have been created

$(t_{start}(x) \leq t)$, must be currently active $(t \leq t_{end}(x))$, and must be feasible $(\Phi(x) = 0)$.

$$F(t) = \{x \in G \mid (t_{start}(x) \leq t \leq t_{end}(x)) \wedge (\Phi(x) = 0)\} \tag{3.22}$$

The cumulative feasible-set $F_c(t)$ contains all feasible solutions that have been created by time t. For a node in G to reside in $F_c(t)$, it must have been created $(t_{start}(x) \leq t)$ and must be feasible $(\Phi(x) = 0)$. In terms of $F(t)$, $F_c(t)$ can be expressed as the union of all $F(\tau)$ for $0 \leq \tau \leq t$.

$$F_c(t) = \{x \in G \mid (t_{start}(x) \leq t) \wedge (\Phi(x) = 0)\} \tag{3.23}$$

$$= \bigcup_{\tau=0}^{t} F(\tau) \tag{3.24}$$

These temporal network subsets are summarized below in Table 3.8.

Table 3.8: Summary of temporal network subsets.

Subset	Type	Symbol	Contains
Super-set	Noncumulative	$S(t)$	Nodes that exist within the population at t.
	Cumulative	$S_c(t)$	Nodes that were created at or before t.
Pareto-set	Noncumulative	$P(t)$	Nodes that are optimal at t.
	Cumulative	$P_c(t)$	Nodes that are or have been optimal at t.
Feasible-set	Noncumulative	$F(t)$	Feasible nodes that exist within the population at t.
	Cumulative	$F_c(t)$	Feasible nodes that were created at or before t.

3.3.4 Effects of Increasing Tournament Size

In using alternative ranking functions, it is important to understand how tournament size affects the potency of the imposed bias. As discussed in step 4 of Section 3.1.2, the tournament size k is the number of randomly chosen individuals in each tournament T created during the GA's selection process. It controls how much influ-

ence the ranking function $\boldsymbol{f_r}$ has on the selection process. When $k = 1$, each tournament contains a single randomly chosen individual, and that individual is added to the next epoch's population; $\boldsymbol{f_r}$ exerts no influence, and the selection is effectively random. The amount of influence imposed by $\boldsymbol{f_r}$ increases with k. Therefore when $k = 2$, the selection is influenced by $\boldsymbol{f_r}$, but as minimally as possible. By using multiple values of $k \geq 2$, the increasing influence of $\boldsymbol{f_r}$ can be compared and analyzed to determine its impact on the solution generation process.

3.4 Summary

This chapter presented details associated with development and execution of the proposed framework. Within the framework, a GA was used generate solutions over time and thereby provide a dynamic environment of the solution generation process. Derived from these dynamics, HANs were developed to capture the implied-causality behind solution dynamics and represent the utilization of information throughout the GA's optimization procedure. These networks provide a means for understanding bias within the execution of an optimization procedure. Lastly, the GA was modified to establish the comparative context necessary for a model's biases to be analyzed and understood. The combination of the GA, HANs, and the context make up this novel framework.

To summarize the methodology, the following text describes how a designer would implement the framework in practice:

1. First, the designer decides what model they choose to investigate. This model defines inputs, constraints, and objectives. Then, the form of these constraints and objectives are assessed and manipulated appropriately, as discussed in Section 3.1.1.

2. The reference and biasing cases are defined. As discussed in Section 3.3.2,

the baseline and random reference cases are defined by the model. Then, the designer chooses the biasing cases that they'd like to consider. This entails the designer defining a ranking function for each biasing case that embodies what they are trying to investigate. Chapter V provides a detailed example.

3. Finally HANs for each case are created by running the modified GA with the corresponding ranking function. Each case runs a number of trials to establish an understanding of the variance in results and produce the appropriate statistics needed to draw a conclusion.

The resulting HANs provide a comparative context and can be analyzed to understand the model's tendencies regarding the biasing cases. The next chapter discusses how these are analyzed through the execution of a test case.

CHAPTER IV

Analyzing the Framework

While the previous chapter provided the procedural details concerning the networks and solution generation aspects of the method, this chapter presents the various novel analyses created to enable the evaluation of a model's inherent tendencies relative to the comparative context established by reference and biasing cases. Sections 4.1 through 4.4 present solution-centric analyses while sections 4.5 and 4.6 present generative analyses.

4.1 Cardinality

As a first step in evaluating the different cases, this section discusses analyses that assess the cardinality of the temporal network subsets defined in Section 3.3.3. The cardinality of a set A is defined as the number of elements contained in A. Mathematically, the cardinality of A, denoted as $|A|$, can be expressed as:

$$|A| = \sum_{a \in A} 1 \tag{4.1}$$

In this analysis, cardinality is used to evaluate the dynamics of four temporal subsets: the noncumulative and cumulative Pareto-sets as well as the noncumulative and cumulative feasible-sets. The analysis of these subsets are discussed in turn.

Pareto-set

The cardinality of the noncumulative Pareto-set, $|P(t)|$, assesses the number of optimal solutions at time t. An increase in $|P(t)|$ indicates that more simultaneously optimal solutions are being found. Consequently, a decrease in $|P(t)|$ indicates that an optimal solution is found that dominates more than one of the previously nondominated solutions; for continuous objectives, this situation is rarely observed. When $|P(t)|$ reaches an equilibrium, either (1) the process has either converged, or (2) additional optimal solutions are found that improve on previous ones. If an optimal solution is found that dominates a single previously nondominated solution, the net change in $|P(t)|$ is zero; this is interpreted as solution improvement.

When evaluated and compared across cases, Figure 4.1 gives an example of these results. As discussed in Section 3.3, multiple trials should be completed for each case to provide an understanding of the results' variability. For each case in the figure, the dark line represents the average across trials, and the lighter shading represents a 68% confidence interval off this average. Unless otherwise specified, this format applies to all such figures presented in this document.

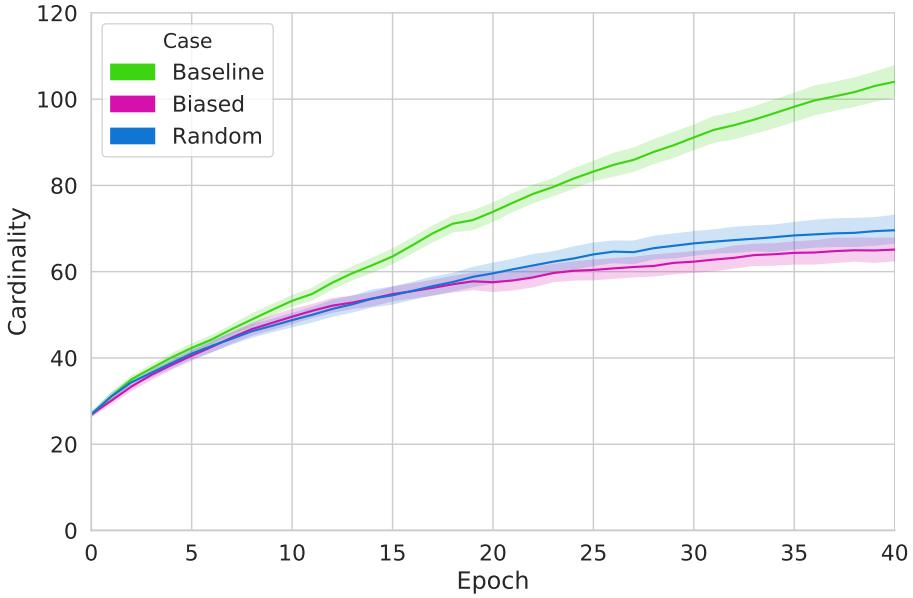


Figure 4.1: Cardinality of the noncumulative Pareto-set, $|P(t)|$.

The results in Figure 4.1 demonstrate the necessity of comparisons. If the results of only one case is shown, there is not sufficient context to interpret the quality of the results or the tendencies of the model. Using the BASELINE (ideal) and RANDOM (unguided) cases as references, results of the BIASED case can be interpreted. In discussing these results, the terms $P^B(t)$, $P^R(t)$, and $P^I(t)$ are used to denote the noncumulative Pareto-sets of the BASELINE (B), RANDOM (R), and BIASED (I) cases, respectively. In the example, $P^I(t)$ is slightly less than $P^R(t)$, meaning that BIASED finds fewer simultaneously optimal solutions than RANDOM.

The cardinality of the cumulative Pareto-set, $|P_c(t)|$, assesses the number of optimal solutions that have been found at or before time t. An increase in $|P_c(t)|$ indicates that additional optimal solutions are being found; by its definition, $|P_c(t)|$ cannot experience a decrease. When $|P_c(t)|$ reaches an equilibrium, the process is not finding additional optimal solutions and has, for practical purposes, converged.

Once evaluated and compared across cases, Figure 4.2 presents an example of these results. The reference cases exhibit similar trends as in the noncumulative case: both BASELINE and RANDOM increase over time, but BASELINE increases

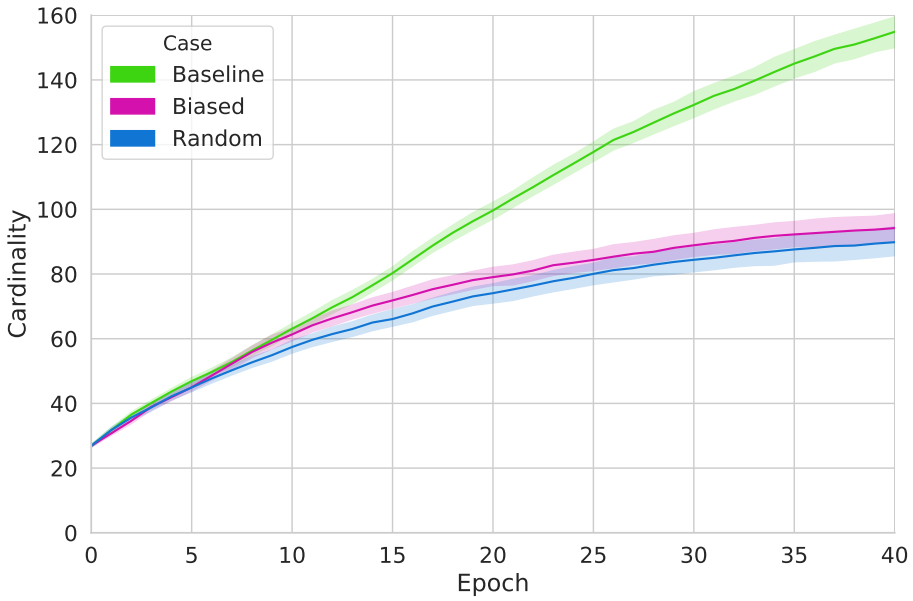


Figure 4.2: Cardinality of the cumulative Pareto-set, $|P_c(t)|$.

considerably more than RANDOM. In the biasing case, BIASED is slightly more than RANDOM; BIASED has found more total optimal solutions than RANDOM.

Since $|P_c(t)| - |P(t)|$ is the number of previously optimal solutions found at time t, the difference between the cumulative and noncumulative Pareto-sets provides a measure of how well the process *improves* on optimal solutions that it has already found. When assessed at $t = E$, this measures the total amount of improvement throughout the process. In the examples, BIASED finds fewer optimal solutions than RANDOM ($|P^I(E)| < |P^R(E)|$), but finds more of them throughout its progression ($|P_c^I(t)| > |P_c^R(t)|$). This implies that the biased case improves on previous solutions more than the random case does.

Feasible-set

The cardinality of the noncumulative feasible-set, $|F(t)|$, assesses the number of feasible solutions that exist in the population at time t. This provides an understanding of the state of the search at t and the search dynamics over time. Furthermore since the expected number of individuals each epoch equals $N(1 + p_{cx})$, this measures of the fraction of feasible individuals in the population at time t, which is used to determine if the solution generation process is promoting the creation of feasible individuals.

An example of these results are shown in Figure 4.3. For the reference cases, both BASELINE and RANDOM decrease initially and then continue to increase. BASELINE does not decrease as much initially, and increases significantly higher than RANDOM. The increase of feasible solutions over time makes sense given that the ranking functions are penalized using a linear scale factor equal to the current epoch: infeasible solutions get worse evaluations as time progresses. BASELINE has the most feasible solutions by the end; since optimal solutions must be feasible, BASELINE has a higher opportunity of creating optimal solutions due to its number of feasible

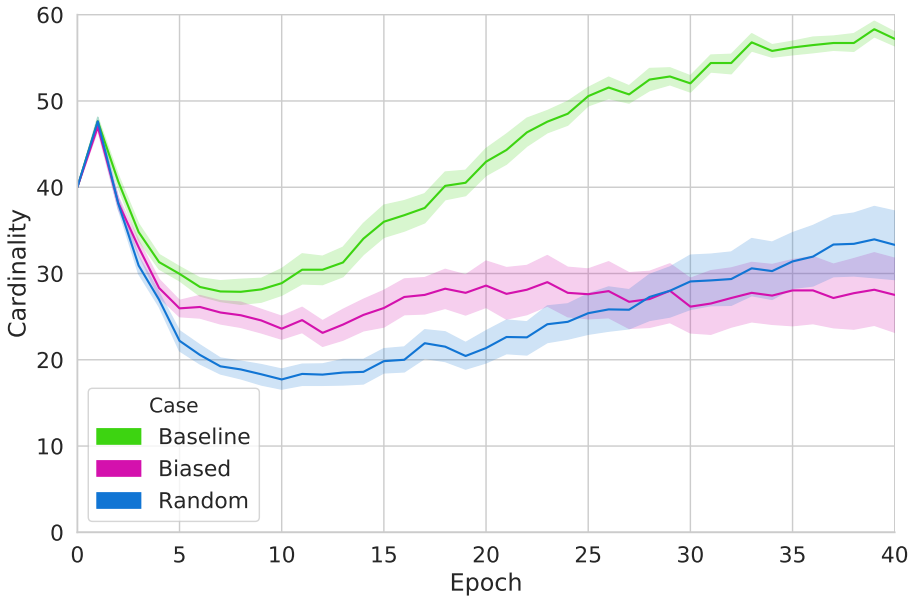


Figure 4.3: Cardinality of the noncumulative feasible-set, $|F(t)|$.

solutions. Another difference between the reference cases is that BASELINE has much less variability than RANDOM. BIASED decreases initially like the reference cases, however rather than increasing after this, BIASED instead reaches an equilibrium. Since for the number of feasible solutions does not grow or shrink over time, the solution-space searched by BIASED must not be very close to constraint boundaries: if it was, feasible solutions would be promoted more.

Cardinality of the cumulative feasible-set, $|F_c(t)|$, assesses the number of feasible solutions that have been created at or before time t. This differentiates whether solutions are sticking around or if additional feasible solutions are being generated. Additionally, it provides an understanding of how many feasible solutions are created for a given bias, which can be used to assess the likelihood of feasible options in an early stage design activity. When $|F_c(t)|$ reaches an equilibrium, the process has ceased finding feasible solutions.

An example of these results are shown in Figure 4.4. For the reference cases, both BASELINE and RANDOM share the same increasing trend over time, with BASELINE producing more than RANDOM; this makes sense given the noncumulative

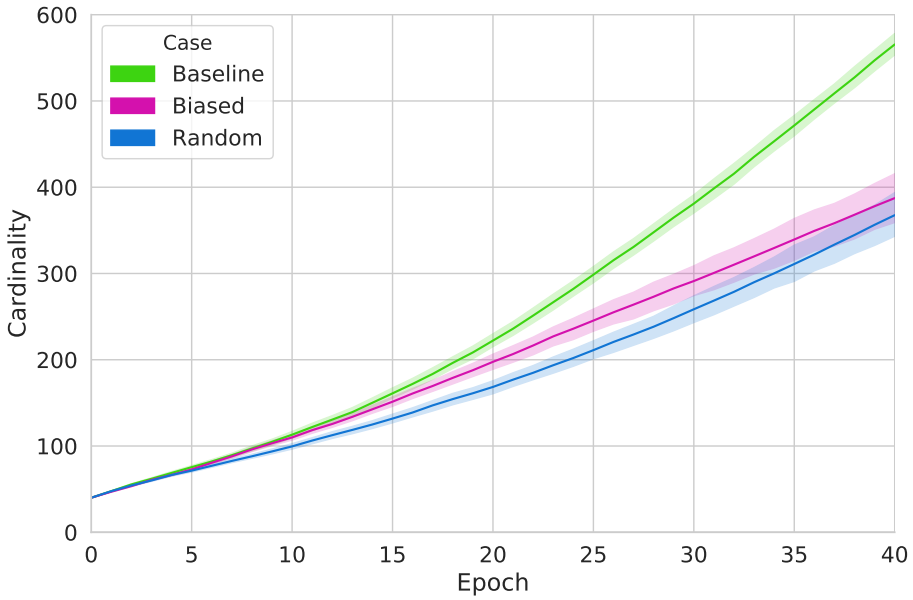


Figure 4.4: Cardinality of the cumulative feasible-set, $|F_c(t)|$.

results already shown. For BIASED, although $|F^I(t)|$ reaches an equilibrium, $|F_c^I(t)|$ continues to grow. BIASED is still finding feasible solutions, and the 30 solutions in $|F^I(t)|$ are not the same 30 solutions each epoch.

4.2 Coverage

While the cardinality analyses assess how each case affects the generation of optimal solutions, those solutions are only deemed optimal relative to the other solutions found by that particular case's optimization procedure. To evaluate the relative optimality of Pareto-sets across cases, a different analysis is needed. Here, the coverage metric presented by *Zitzler* (1999) is used to assess the optimality of the cases' final Pareto-sets $P(t = E)$ relative to each other.

The coverage metric compares two different Pareto-set approximations, A and B, by evaluating how much of B is dominated by A. It is defined as:

$$C(A, B) = \frac{|\{b \in B | (\exists a \in A)[a \succ b]\}|}{|B|} \tag{4.2}$$

where $a \succ b$ indicates that solution a dominates solution b. The numerator in Equation 4.2 is the number of solutions in B that are dominated by a solution in A. By dividing the number of dominated solutions by the total number of solutions in B, the coverage metric gives the fraction of B that is dominated by A and is therefore bounded between zero and one: $0 \leq C(A, B) \leq 1$. When $C(A, B) = 1$, all solutions in B are dominated by solutions in A, and when $C(A, B) = 0$, no solutions in B are dominated by solutions in A. By its definition, $C(A, B)$ does not imply $C(B, A)$; therefore, both orderings have to be computed.

The coverage metric is used to compare the final Pareto-sets found by each case, and thereby evaluate how a case's ranking function affects the relative optimality of its solutions. An example of these results is presented in Table 4.1, where A is shown in rows, B in columns, and coverage results are expressed as percentages. Thereby, values presented in the table represent the percentage of B's final Pareto-set $P^B(E)$ that is dominated by A's final Pareto-set $P^A(E)$. The mean μ and standard deviation σ of these values are calculated across the trials performed for each case.

Table 4.1: Coverage results $C(P^A(E), P^B(E))$ assessing the percentage of the Pareto-set of B that is dominated by the Pareto-set of A.

A \ B	Baseline		Biased		Random	
	μ	σ	μ	σ	μ	σ
Baseline	-	-	26.0%	14.2%	38.2%	17.1%
Biased	18.7%	12.5%	-	-	29.5%	13.7%
Random	18.6%	13.0%	20.2%	12.1%	-	-

To provide some intuition for interpreting these results, the reference cases are compared. In the example presented in Table 4.1, BASELINE dominates 38.2% of RANDOM, meaning that 38.2% of RANDOM's optimal solutions are dominated by BASELINE's optimal solutions. On the other hand, RANDOM dominates 18.6% of BASELINE. Since BASELINE dominates RANDOM more than RANDOM dominates BASELINE, the coverage results demonstrate that the BASELINE case produces a more superior Pareto-set than the RANDOM case, as is expected.

For a given row, higher values indicate a more optimal solution set. The BASELINE row shows BASELINE dominates BIASED less than RANDOM. The BIASED row shows BIASED dominates BASELINE less than RANDOM. The RANDOM row shows RANDOM dominates BASELINE less than BIASED. In Table 4.1, the first row shows that the baseline Pareto-set dominates 26.0% and 38.2% of the biased and random Pareto-sets, respectively. Similarly, the last row shows that the random Pareto-set dominates 18.6% and 20.2% of the baseline and biased Pareto-sets, respectively. When the model is biased toward a given case, that case will dominated more of the other cases. For biasing cases, higher row values indicate a bias that the model is predisposed toward.

For a given column, lower values indicate a more optimal solution set. The BASELINE column shows BASELINE is dominated by BIASED the same as by RANDOM. The BIASED column shows BIASED is dominated by BASELINE more than by RANDOM. The RANDOM column shows RANDOM is dominated by BASELINE much more than by BIASED. In Table 4.1, the first column shows that 18.7% and 18.6% of the baseline Pareto-set is dominated by the biased and random Pareto-sets, respectively. The last column shows that 38.2% and 29.5% of the random Pareto-set is dominated by the baseline and biased Pareto-sets, respectively. When the model is biased toward a given case, that case will be less dominated by the other cases. For biasing cases, lower column values indicate a bias that the model is predisposed toward.

4.3 Superfront Relative Distance

The coverage analysis considers the number of solutions in Pareto-set B that are dominated by solutions in Pareto-set A, but it does not evaluate the degree to which these solutions are dominated; that is, solutions' objective values are used to assess dominance, but the objective-space distance between solutions is not assessed.

Furthermore, the optimality of Pareto-set B is assessed relative to Pareto-set A and not relative to the model's "true" Pareto front. This section defines two metrics assess the objective-space distance of a given solution set A relative to the model's "true" Pareto front. These metrics are then used to evaluate the dynamics of the Pareto-set and feasible-set as they develop over time.

In this analysis, the model's "true" Pareto front is represented by a solution set called the *superfront*. To construct the superfront, solutions from the final Pareto-sets across all cases and trials are combined into a single set, denoted as $\bar{P}$. Mathematically, $\bar{P}$ can be expressed as:

$$\bar{P} = \bigcup_i \bigcup_j P^{ij}(E) \tag{4.3}$$

where $P^{ij}(E)$ denotes the final Pareto-set from trial j of case i. The superfront SF is defined as the Pareto-set of $\bar{P}$, excluding any duplicate solutions.

$$SF = \{x_i \in \bar{P} | (\nexists x_j \in \bar{P})[x_j \succ x_i]\} \tag{4.4}$$

Thereby, the superfront contains each nondominated solution from the set of all nondominated solutions that were generated; it is the Pareto-set of Pareto-sets. Thereby, the superfront provides a representation the "true" Pareto front in the model's objective-space and is used as a standard basis from which to measure the relative distance of a given solution.

To calculate the objective-space distance between two discrete solution sets A and B, a variation of the M_1^*-metric presented by *Zitzler et al.* (2000) is utilized:

$$M_1^*(A, B) = \frac{1}{|A|} \sum_{i \in A} \min_{j \in B} d_{ij} \tag{4.5}$$

where d_{ij} is the Euclidean (or L^2) distance between the objective values of solutions

i and j:

$$d_{ij} = \sqrt{\sum_{\Omega \in \mathbf{\Omega}} (\Omega(i) - \Omega(j))^2} \tag{4.6}$$

The term $\min_{j \in B} d_{ij}$ calculates the distance from solution $i \in A$ to its closest solution in B. The sum over A assesses this distance for each solution in A, and normalizing by $|A|$ equates to $M_1^*(A, B)$ representing the average minimum distance of solutions in A to solutions in B.

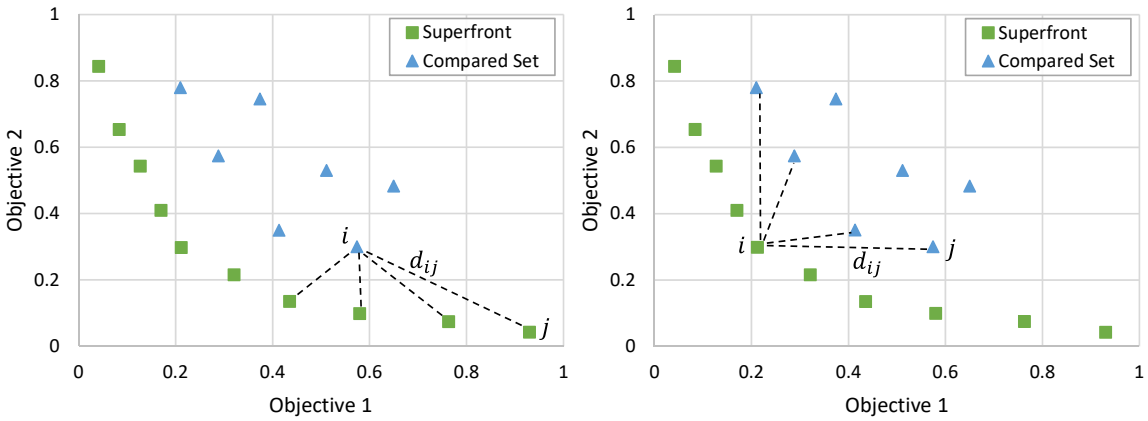
Before any distances are evaluated, the objective-space is normalized so as to not skew results toward any single objective. For example if values of Ω_1 are in $[10, 11]$ and values of Ω_2 are in $[1, 100]$, then the distance between solutions will only effectively reflect the difference between their Ω_2 values. To account for this, solutions in $\bar{P}$ are normalized to the interval $[0, 1]$ by linearly scaling each of their objective values based on its corresponding minimum and maximum values in $\bar{P}$.

Using the M_1^*-metric, two new metrics are defined to measure the objective-space distance of solution set A relative to the superfront. *Distance to superfront* $D_T(A)$ measures the distances of solutions in A to solutions in the superfront, and *distance from superfront* $D_F(A)$ measures the distances of solutions in the superfront to solutions in A. These metrics can be expressed in terms of the M_1^*-metric as:

$$D_T(A) = M_1^*(A, SF) = \frac{1}{|A|} \sum_{i \in A} \min_{j \in SF} d_{ij} \tag{4.7}$$

$$D_F(A) = M_1^*(SF, A) = \frac{1}{|SF|} \sum_{i \in SF} \min_{j \in A} d_{ij} \tag{4.8}$$

These metrics are depicted in Figure 4.5. $D_T(A)$ evaluates the proximity of solutions in A to the superfront. Table 4.2 summarizes how to interpret $D_T(A)$ in terms of its mean μ and standard deviation σ. $D_F(A)$ evaluates the proximity of solutions in the superfront to A. Table 4.3 summarizes how to interpret $D_F(A)$ in terms of



(a) $D_T(A)$ (b) $D_F(A)$

Figure 4.5: The mechanics of the superfront-relative-distance metrics.

its mean μ and standard deviation σ. The difference between $D_T(A)$ and $D_F(A)$ is subtle, but important.

The combination of $D_T(A)$ and $D_F(A)$ provides a visual idea of how solutions in A are situated in objective-space. For example if $D_T(A)$ is low and $D_F(A)$ is high, then solutions in A are close to optimal, although many solutions in the superfront are still far distances away. This effect demonstrates that A is clustered in a localized

Table 4.2: Interpretation of $D_T(A)$ in terms of mean μ and standard deviation σ.

Term	Value	Interpretation
μ	Low	Solutions in A are close to solutions in the superfront.
	High	Solutions in A are not close to solutions in the superfront.
σ	Low	Solutions in A have similar minimum distances to solutions in the superfront.
	High	Solutions in A have dissimilar minimum distances to solutions in the superfront.

Table 4.3: Interpretation of $D_F(A)$ in terms of mean μ and standard deviation σ.

Term	Value	Interpretation
μ	Low	Solutions in the superfront are close to solutions in A.
	High	Solutions in the superfront are not close to solutions in A.
σ	Low	Solutions in the superfront have similar minimum distances to solutions in A.
	High	Solutions in the superfront have dissimilar minimum distances to solutions in A.

area of the superfront and is not well-distributed across it. Thereby the solutions in A lack diversity, which is a very important quality to designers, especially in early stage ship design.

To evaluate the objective-space progression of solutions over time, these metrics are used to investigate the Pareto-sets and feasible-sets of the reference and biasing cases. To assess the state of solutions specifically at time t, only the noncumulative subsets are considered, since accounting for previously optimal solutions or all feasible solutions would inhibit an understanding of the search's progression. The results of each subset is discussed in turn.

Pareto-set

The distance to superfront of the noncumulative Pareto-set, $D_T(P(t))$, calculates the average minimum distance at time t from optimal solutions in $P(t)$ to solutions in the superfront. An example of these results is shown in Figure 4.6. Agreeing with expectation, BASELINE is always closer than RANDOM. In addition, BASELINE has lower variability than RANDOM; BASELINE optimal solutions have more similar

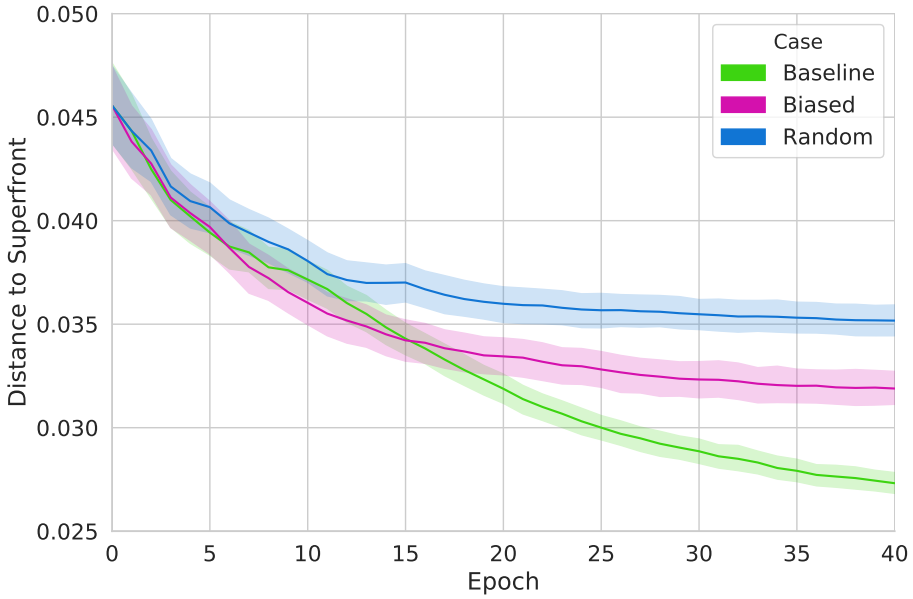


Figure 4.6: Distance to superfront of the Pareto-set, $D_T(P(t))$.

minimum distances than random. BIASED decreases at a similar rate as RANDOM, but is less than RANDOM; BIASED finds optimal solutions closer to the superfront.

The distance from superfront of the noncumulative Pareto-set, $D_F(P(t))$, calculates the average minimum distance at time t from solutions in the superfront to optimal solutions in $P(t)$. An example of these results are shown in Figure 4.7. Similarly for these results, the superfront is always closer to BASELINE than RANDOM. BIASED is farther from the superfront than RANDOM. Comparing $D_T(P(t))$ and $D_F(P(t))$, BIASED is lower than RANDOM for D_T but higher for D_F: BIASED optimal solutions are focusing on a localized area; they are closer to optimal, but are less distributed across objective-space.

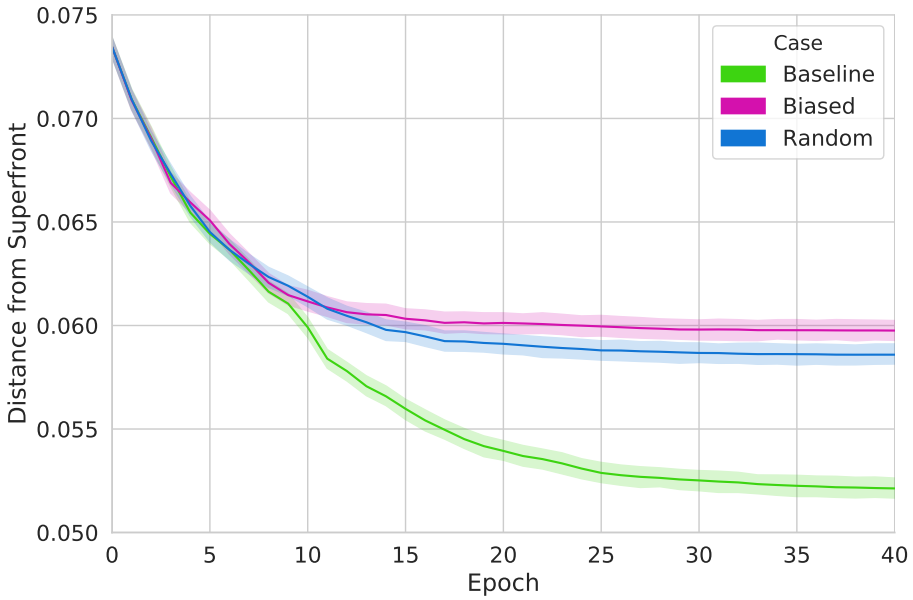


Figure 4.7: Distance from superfront of the Pareto-set, $D_F(P(t))$.

Feasible-set

Beyond inspecting the Pareto-set, solution dynamics may also be understood by investigating where the optimizer is searching for potential solutions. The distance to superfront of the noncumulative feasible-set, $D_T(F(t))$, calculates the average minimum distance from feasible solutions at time t to solutions in the superfront. An

example of these results are shown in Figure 4.8. For the reference cases, BASELINE steadily decreases, while RANDOM decreases initially, increases slightly, and then seems to reach an equibilbrium. Feasible solutions found by RANDOM are not getting closer to superfront over time, continuing the search with little solution improvement. BIASED has similar dynamics as RANDOM, and BIASED is always closer than RANDOM. In fact before reaching an equilibrium at $t = 10$, BIASED is initially closer than BASELINE. BIASED finds a feasible region of solutions faster than BASELINE but then ceases to improve. These results indicate that BIASED is capturing some, but not all, of the intent in BASELINE.

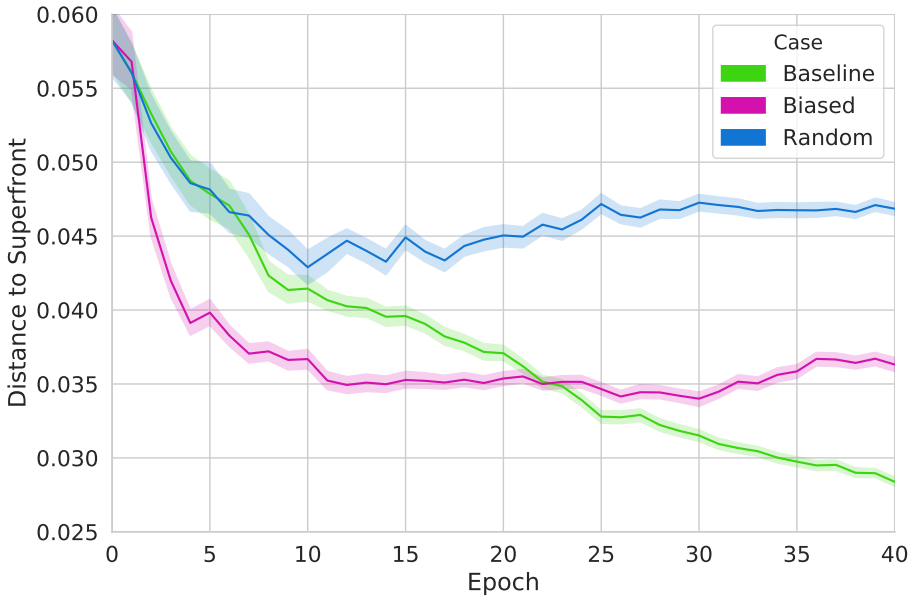


Figure 4.8: Distance to superfront of the feasible-set, $D_T(F(t))$.

The distance from superfront of the noncumulative feasible-set, $D_F(F(t))$, calculates the average minimum distance at time t from solutions in the superfront to feasible solutions in $F(t)$. An example of these results are shown in Figure 4.9. For the reference cases, BASELINE and RANDOM increase steadily, with BASELINE being less than RANDOM. D_F increasing means solutions are getting farther from SF, while D_T decreasing means solutions are getting closer to SF, and the low variability for both metrics means than solutions are all similar distances. Therefore, feasible

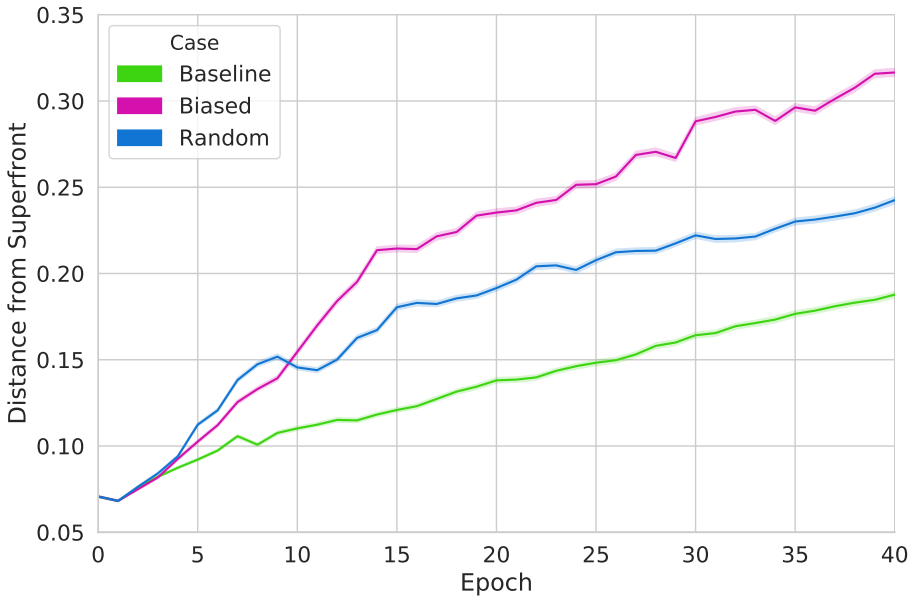


Figure 4.9: Distance from superfront of the feasible-set, $D_F(F(t))$.

solutions are getting closer to optimal, but they are concentrated in an area and are not well-distributed across the superfront. This type of behaviour demonstrates that feasible solutions are becoming less diverse over time. BIASED being farther than RANDOM shows that BIASED creates feasible solutions with less diversity than RANDOM.

4.4 Generational Distance

As stated in Section 3.1.2, GAs do not have any well-established stopping criteria, and the data presented in this work is created by iterating for a specified number of epochs E. However, assessing the convergence of the Pareto-set is important for at least two reasons: (1) it would be incorrect to make claims concerning a model's biases based on the comparison of solutions sets that have not converged, and (2) differences in convergence dynamics may aid an understanding of the model's tendencies, the quality of resultant solutions, and the quality of the solution generation process.

Pareto-set convergence is assessed using the Generational Distance metric pre-

sented by *Van Veldhuizen* (1999). This indicator is given by the following formula:

$$GD(A, B) = \frac{1}{|A|}\left(\sum_{i \in A} \min_{j \in B} ||\mathbf{\Omega}(i) - \mathbf{\Omega}(j)||^p\right)^{\frac{1}{p}} \tag{4.9}$$

where A is a Pareto-set approximation and B a discrete representation of the Pareto-set. When $p = 2$, Generational Distance is equivalent to the M_1^*-metric discussed in Section 4.3. To assess convergence, *Collette and Siarry* (2011) present a variation of Generational Distance where $GD(P(t-1), P(t))$ evaluates successive iterations of the noncumulative Pareto-set. Figure 4.10 shows an example of results using this variant.

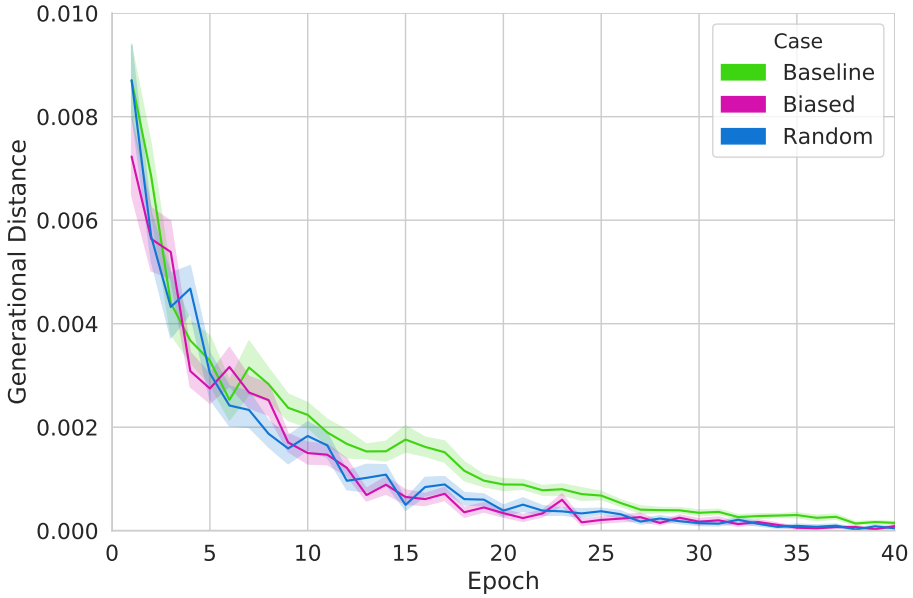


Figure 4.10: Generational Distance as a convergence metric, $GD(P(t-1), P(t))$.

4.5 Out-degree

This section details an analysis of the PCN that utilizes the out-degree centrality to evaluate the relationships between solutions. In network theory, out-degree centrality (often simply referred to as out-degree) is one of the simplest measures of the importance or influence of nodes within a network (*Newman*, 2010). In directed

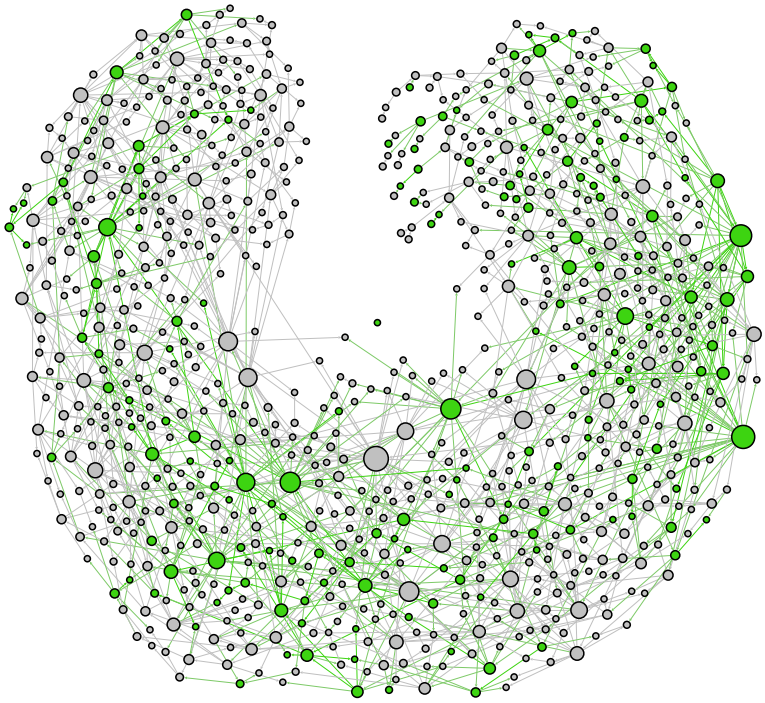
networks such as the PCN, the out-degree of a node is defined as the number of edges that point away from it. Out-degree represents a node's ability to influence other nodes within the network, such that nodes with a high out-degree are perceived as being highly influential. Using the adjacency matrix as defined in Section 3.2.2, the out-degree of a node i can be written as:

$$x_i = \sum_{j=1}^{n} A_{ij} \tag{4.10}$$

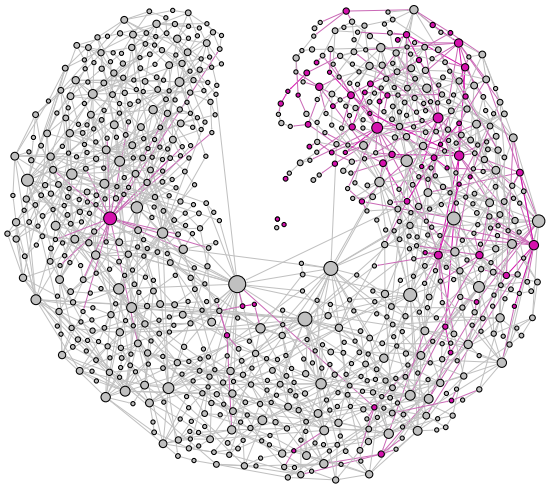
As discussed in Section 3.2, edges in the PCN represent the relationships from parents to children in the solution generation process. When parents create a child in the GA, the child inherits information contained in both of its parents, and parents influence the child through this transference of information; this is captured by the PCN. In the PCN, a node's out-degree corresponds to the number of children it created and thereby to the number of times that its input values are used to define a new search point in the model's solution-space. For a node to have a high out-degree, it must be chosen to engage in crossover multiple times. Naturally, the longer an individual remains in the population, the more opportunity it has to create children; to remain in the population, it must be chosen during the selection process. Therefore in the PCN, out-degree indicates a node's popularity as an information source; a node with a high out-degree is more influential to the solution generation process.

Since a node's out-degree corresponds to its influence within the solution generation process, optimal nodes are expected to have higher out-degree scores. Using this basic tenant, the PCN is analyzed by comparing the out-degree of its optimal nodes to that of its dominated nodes. To account for previously optimal nodes, this analysis considers the network's cumulative Pareto-set $P_c(t)$ when evaluating the status of a node; i.e., a node are considered optimal if it is in $P_c(t)$ and dominated otherwise. Then, PCNs are compared across cases to assess their relative quality. An example of

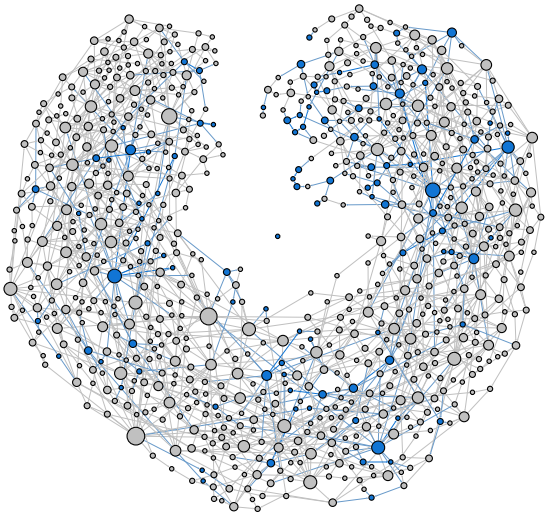
this is shown in Figure 4.11, where the aggregate PCNs from each case are visualized using the layout detailed in step 5 of Section 3.2. In the figures, a node's size illustrates its relative out-degree, and colored nodes are contained in $P_c(E)$. The progression of time is roughly represented by a clockwise rotation starting at 12 o'clock.



(a) Baseline



(b) Biased



(c) Random

Figure 4.11: Aggregate PCNs for each case. A node's size illustrates its relative out-degree, and colored nodes are contained in $P_c(E)$.

In BASELINE, optimal nodes are generated continually throughout the process. Additionally, optimal nodes have more influence than dominated nodes, in general. There are some large nodes that are not optimal; these solutions may be close to optimal, but that isn't assessed in this analysis. In RANDOM, the out-degree of optimal nodes seems similar to dominated nodes; they are not perceived as being better or worse by the solution generation process. In BIASED, very few optimal nodes are found in the second half (6:00 to 11:00), and the most influential (highest out-degree) nodes are not optimal nodes.

The network visualization provides intuition for understanding the generative process in a snapshot. However, Figure 4.11 depicts the cases' PCNs of only a single trial; and no trivial method is apparent for compiling these results across multiple trials. Instead, multiple trials are considered by using logistic regression to evaluate how a node's out-degree corresponds to its optimality. Specifically, binary logistic regression is used to determine the log-odds (logarithm of odds) that a node resides in $P_c(E)$ given its out-degree (the odds is defined as the probability p that a node resides in $P_c(E)$ divided by the probability that it does is not). The log-odds ℓ that node i resides in $P_c(E)$ is calculated as:

$$\ell = \ln \frac{p}{1-p} = \beta_0 + \beta_1 x_i \tag{4.11}$$

where x_i is the out-degree of node i, and β_j are parameters of the model. Parameters β_j are calculated from a given set of observations using iteratively reweighted least squares. The odds are calculated by exponentiating the log-odds:

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x} \tag{4.12}$$

Then by algebraic manipulation, the probability p that node i resides in $P_c(E)$ can

be expressed in terms of the logistic function as:

$$p(i \in P_c(E)|x_i) = \frac{e^{\beta_0+\beta_1 x_i}}{e^{\beta_0+\beta_1 x_i}+1} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}} \tag{4.13}$$

where p can vary between zero (certainly dominated) and one (certainly optimal).

To consider all trials, logistic regression is used to model the probability that a node resides in $P_c(E)$ given its out-degree. This answers the question, "Do optimal nodes influence the process more?" An example of these results is shown in Figure 4.12. For a high quality results, optimal nodes should have higher out-degree scores; the probability should increase with out-degree. This is seen in BASELINE, where the most influential nodes have an 80% probability of being optimal; out-degree is a good indicator of a node's optimality. RANDOM is less than BASELINE, as expected; the most influential nodes only have an 35% probability of being optimal. However, RANDOM also has a positive trend; this makes sense given that many optimal nodes are found in the initial population, which are more likely to have higher out-degree. Again, BIASED cases are compared against the reference cases to evaluate the model's

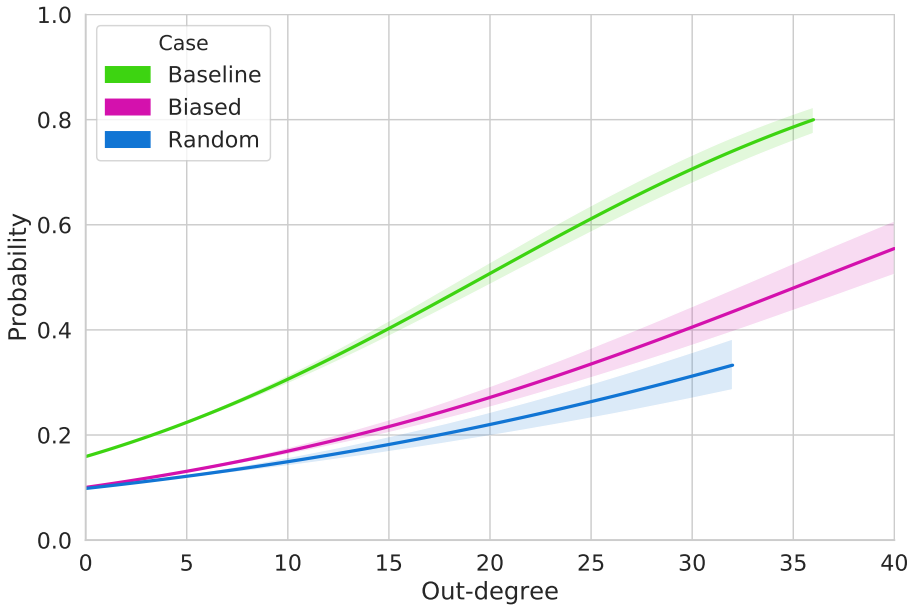
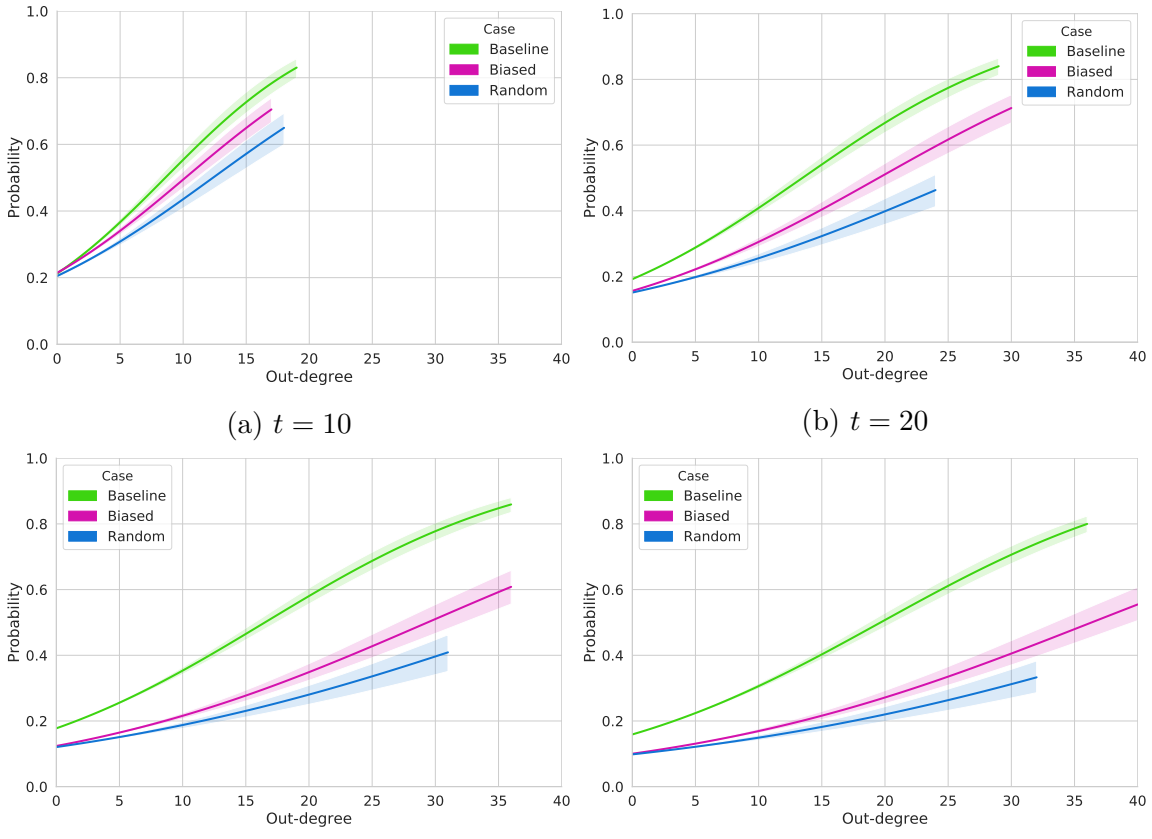


Figure 4.12: Probability of a node residing in $P_c(E)$ given its out-degree.

tendencies. BIASED is greater than RANDOM and less than BASELINE. Although BIASED trends similarly to RANDOM, its endpoint is situated between BASELINE and RANDOM; the most influential nodes have a 55% probability of being optimal.

The example in Figure 4.12 considers the aggregate PCNs at the end of the process. The same process can be conducted at different time steps to understand the progression and to look for leading indicators of quality. Figure 4.13 shows an example of these results at epochs 10, 20, 30, and 40. All cases have high values at $t = 10$, due to the fact that optimal solutions are easier to find initially. However as t increases, BASELINE remains high while RANDOM decreases. In BASELINE, the influence of optimal nodes continues to be heavily utilized; whereas in RANDOM, this is not the case.



(a) $t = 10$ (b) $t = 20$

(c) $t = 30$ (d) $t = 40$

Figure 4.13: Probability of a node residing in $P_c(E)$ given its out-degree. Evaluating out-degree as a leading indicator of bias.

4.6 Betweenness Centrality

Out-degree analysis considers the relationships in the PCN. This section presents an analysis of the PPN to assess how information is used throughout the process and to evaluate if optimal nodes are central to the process. This is done by using betweenness centrality to evaluate the PPN. Betweenness centrality (often simply referred to as betweenness) measures the extent to which a node lies on paths between other nodes (*Newman*, 2010). A *path* between two nodes is a route across the network that runs along the edges of the network. Thereby, *path length* is determined by the number of edges that are traversed along a given path. A geodesic path is a path between two nodes such that no shorter path exists. A number of ways exist to calculate this, such as (*Dijkstra*, 1959). Using the adjacency matrix $\boldsymbol{A}$, the total number of paths with length r between nodes i and j is given by:

$$N_{ij}^{r} = \sum_{k=1}^{n} A_{ik} A_{kj} = [\boldsymbol{A}^{r}]_{ij} \tag{4.14}$$

Using this, the geodesic distance from i to j is the smallest value of r such that $[\boldsymbol{A}^{r}]_{ij} > 0$. The betweenness of node i is defined as:

$$x_i = \frac{1}{n^2} \sum_{st} \frac{n_{st}^{i}}{g_{st}} \tag{4.15}$$

where n_{st}^{i} is the number of geodesic paths from s to t that pass through i, and g_{st} is the total number of geodesic paths from s to t. Therefore, the term n_{st}^{i}/g_{st} represents the fraction of geodesic paths from s to t that pass through i. We adopt the convention that $n_{st}^{i}/g_{st} = 0$ if both n_{st}^{i} and g_{st} are zero. This is summed over all nodes s and t such that a path st exists which passes through i. The entire sum is normalized by the factor $1/n^2$ (where n is the number of nodes in the network) which means that betweenness values will always lie between 0 and 1. Betweenness is an approximate

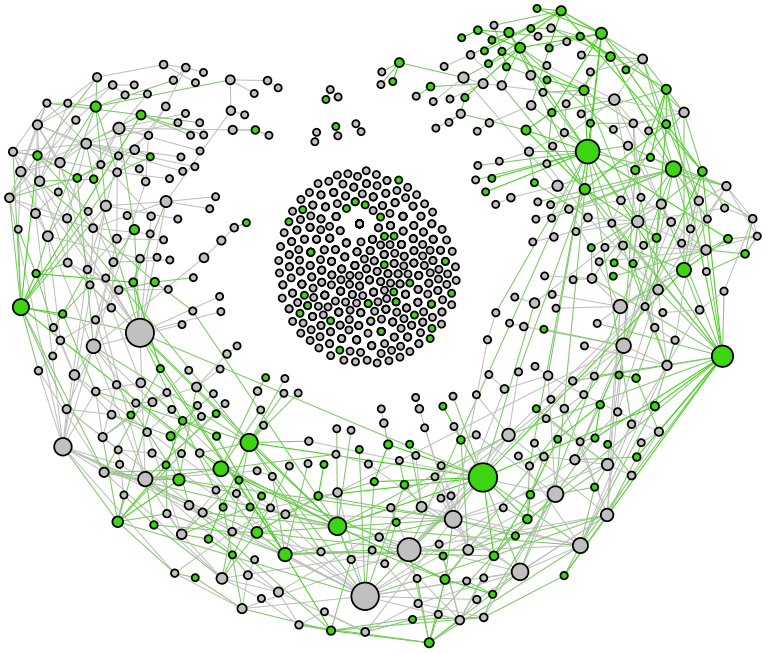
guide to evaluating the influence that nodes have on the flow of information within the network.

As discussed in Section 3.2, edges in the PPN represent the relationships between parents in the solution generation process. When parents create a child in the GA, information contained in the parents is utilized to define a new search point in the model's solution-space; this utilization of information is captured by the PPN. In the PPN, betweenness indicates how central a solution's information is within the solution generation process. A node with a high betweenness has more influence in terms of the importance of its information.

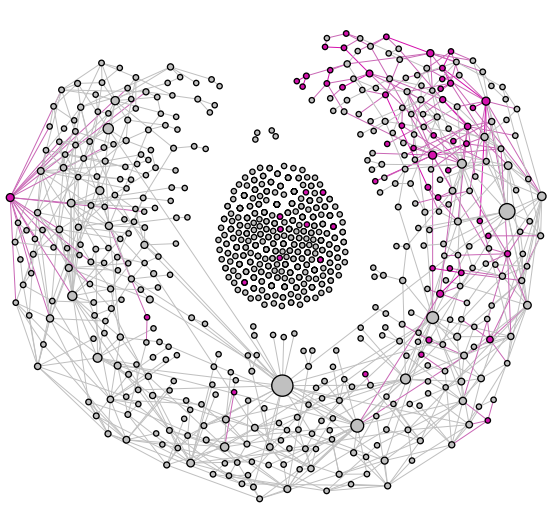
For these reasons, optimal nodes are expected to have higher betweenness scores. Similarly to out-degree on the PCN, the PPN is analyzed by comparing the betweenness of its optimal nodes to that of its dominated nodes; the network's cumulative Pareto-set $P_c(t)$ is considered to account for previously optimal nodes. Then, PPNs are compared across cases to assess their relative quality. An example of this is shown in Figure 4.14, where the aggregate PPNs from each case are visualized using the layout detailed in step 5 of Section 3.2. In the figures, a node's size illustrates its relative betweenness, and colored nodes are contained in $P_c(E)$.

In BASELINE, optimal nodes have hightest betweenness; optimal nodes are most central in the network. In RANDOM, optimal nodes have low betweenness, especially later in the process (early nodes are predisposed to being optimal). The information contained in optimal nodes is not perceived as being better by the solution generation process. In BIASED, optimal nodes have low betweenness, similarly to RANDOM; the most central nodes are not optimal nodes. However, BIASED also has the least diversity of betweenness; some nodes are highly central while most are not central at all.

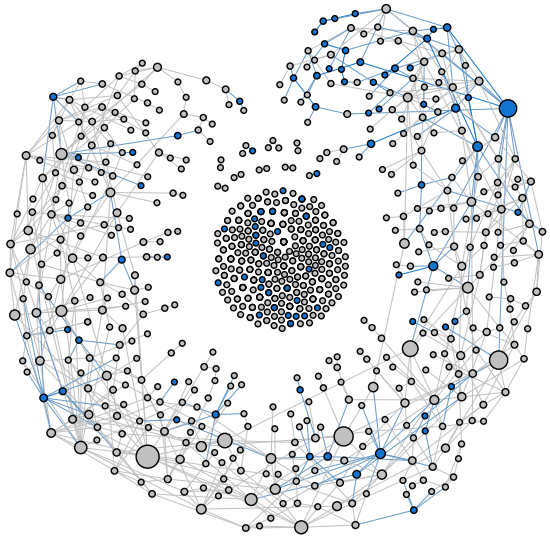
To consider all trials, similarly to out-degree, logistic regression is used to calculate the probability that node resides in $P_c(E)$ given its betweenness. This answers the



(a) Baseline



(b) Biased



(c) Random

Figure 4.14: Aggregate PPNs for each case. A node's size illustrates its relative betweenness, and colored nodes are contained in $P_c(E)$.

question, "Are optimal nodes central to the utilization of information in the process?" An example of these results are shown in Figure 4.15. In BASELINE, the most central nodes have more than a 95% probability of being optimal; betweenness is a good indicator of a node's optimality. The information contained in BASELINE's

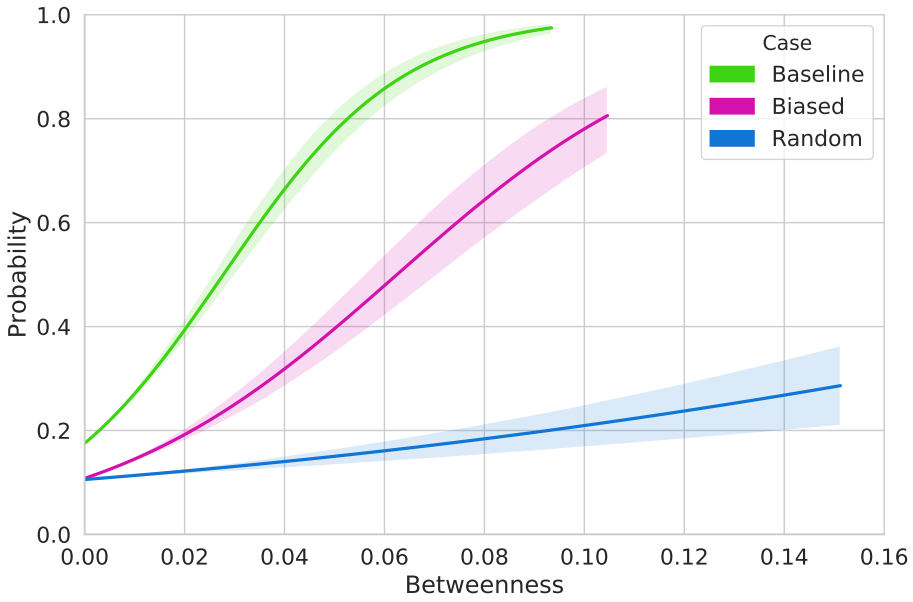


Figure 4.15: Probability of a node residing in $P_c(E)$ given its betweenness.

optimal nodes is very central to the solution generation process. RANDOM is much less than BASELINE, but still has a positive trend; the most central nodes only have a 30% probability of being optimal. BIASED is greater than RANDOM but still less than BASELINE; the most central nodes have a 80% probability of being optimal. Optimal nodes are central and their information is utilized by the process.

The results shown in Figure 4.15 consider all the nodes in the PPN. However, the intent of this analysis is more concerned with evaluating nodes that actually contribute to the process. In Figure 4.14, there are a large number of unconnected nodes in the middle of the networks. These are nodes that were created but didn't create any children; their information is never utilized by the solution generation process. The connect nodes (those that do contribute to the process) can be analyzed by only considering the largest *component* in the network (a *component* is a subset of nodes within a network such that there exists at least one path from each member of that subset to each other member of that subset). This is done to evaluate how central a node is within the community of utilized information.

The betweenness of nodes in the largest component are analyzed using logistic

regression. An example of these results is shown in Figure 4.16. Again, BASELINE has a strong positive trend with betweenness, and the most central nodes have an 90% probability of being optimal. RANDOM no longer has a positive trend: being optimal doesn't affect how its information is utilized. This makes sense given RANDOM doesn't discriminate between solutions in selection; it just picks a random solution. Therefore *in the community of nodes that are utilized*, RANDOM is expected to be relatively flat; the most central nodes only have a 15% probability of being optimal. BIASED is less than BASELINE but better than RANDOM; the most central nodes have a 60% probability of being optimal.

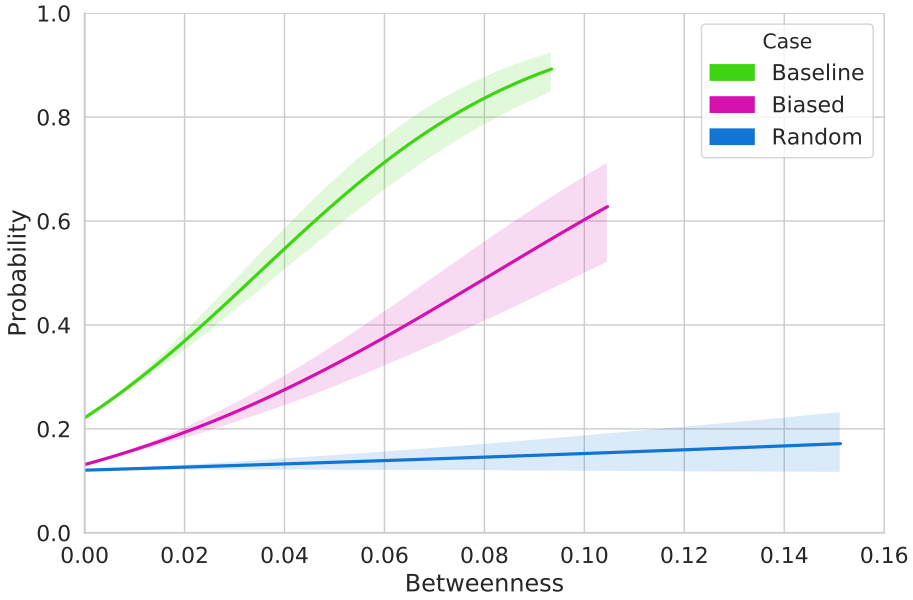


Figure 4.16: Probability of a node residing in $P_c(E)$ given its betweenness. Considering only nodes in the largest component of the PPN.

4.7 Summary

Chapter 4 details six analysis methods that were developed to determine the bias and predication of a early stage design tool. The first four methods provide solution-centric evaluations, while the last two methods provide generative evaluations. It is important to consider both types of evaluation since they complement one another

by providing different types of information. In the next chapter, a case study is conducted to demonstrate the conclusions that can be made using these novel analysis techniques.

CHAPTER V

Case Study

This chapter presents a case study that demonstrates how the innate tendencies of a model are evaluated using the framework proposed in Chapter III and the analyses detailed in Chapter IV. Section 5.1 details the model that is investigated and how the framework was implemented for the case studies. Sections 5.2 through 5.5 present results of the solution-centric analyses, and Sections 5.6 through 5.7 present results of the generative analyses. Finally, Section 5.8 presents results demonstrating the effects of increasing tournament size, as discussed in Section 3.3.4.

5.1 Case Study Setup

The model used to demonstrate the contributions of this thesis is the bulk carrier synthesis model developed by *Sen and Yang* (1998). Outlined in Appendix A, this model defines the inputs $\boldsymbol{x}$, objectives $\boldsymbol{\Omega}$, and constraints $\boldsymbol{c}$ that are used by the modified GA to generate and evaluate solutions over time. The Sen bulker model's inputs are length (L), beam (B), draft (T), depth (D), speed (V), and block coefficient (C_B); these inputs are expressed by the input vector $\boldsymbol{x} = [L, B, T, D, V, C_B]$. Though the model defines many intermediate functions, they are solely dependent on its inputs. Thereby, constraint and objective evaluations follow directly and deterministically for a given set of inputs (i.e., $\boldsymbol{c}$ and $\boldsymbol{\Omega}$ can be expressed as deterministic

functions of $\boldsymbol{x}$). As discussed in Section 3.1.1, the model's constraints and objectives were appropriately formatted, and constraint handling was accomplished using an external penalty function.

As discussed in Section 3.3, a comparative context is established by implementing alternative ranking functions to define reference and biasing cases. The two reference cases described in Section 3.3.2 were created: the baseline case and the random case.

- The baseline case provides an "ideal" reference by representing the traditional application of the model as described in the referenced paper. The ranking function of the baseline case, $\boldsymbol{f_r^B}$, is defined by the model's objectives $\boldsymbol{\Omega}$, given in Equations A.24 - A.26.

- The random case provides a reference by representing a completely unguided search. The random case favors no particular solution, and the ranking function of the random case, $\boldsymbol{f_r^R}$, does not actually rank solutions, but instead chooses one randomly.

As stated in Section 3.3.2, biasing cases are specified by the designer. In his research, *Parker* (2014) investigated the same synthesis model and determined that deadweight was a significant design driver and resistance criteria were not. These findings agree with the institutional knowledge concerning bulk carriers, which are designed to traverse slowly and carry a lot of cargo. Leveraging this information, two biasing cases were specified: the length-over-beam (L/B) case and the deadweight case.

- The L/B case favors design solutions that have less resistance due to their more slender profiles. The ranking function of the L/B case, $\boldsymbol{f_r^L}$, is defined by maximizing a solution's length-to-beam ratio.

$$\max \boldsymbol{f_r^L}(\boldsymbol{x}) = \frac{\text{length}(\boldsymbol{x})}{\text{beam}(\boldsymbol{x})} \tag{5.1}$$

In general, resistance criteria are not primary concerns associated with the design of a bulk carrier. This case was developed to bias the solution generation process in a *dissimilar* way to the model's tendencies.

- The deadweight case favors design solutions that have higher capacities. The ranking function of the deadweight case, $\boldsymbol{f_r^D}$, is defined by maximizing a solution's deadweight.

$$\max \boldsymbol{f_r^D}(\boldsymbol{x}) = \text{deadweight}(\boldsymbol{x}) \tag{5.2}$$

where deadweight($\boldsymbol{x}$) is calculated by Equation A.10. For bulk carriers, cargo capacity is definitely a primary concern. This case was developed to bias the solution generation process in a *similar* way to the model's tendencies.

These four ranking functions define the four cases that are used to demonstrate the analyses presented in Chapter IV. For each of the cases, HANs were created using the modified GA discussed in Chapter III. Aside from each case utilizing its associated ranking function, the GA's other hyperparameters were the same across cases; these are presented in Table 5.1. For each case, twenty-five separate trials were completed. To control for randomness and guarantee reproducibility, the computer's random number generator was seeded explicitly, and a trial's zero-based index was used as the seed value s (e.g., for trial i, $s = i$). Thereby for a given trial, the same

Table 5.1: Hyperparameters of the GA used to generate HANs for the case studies.

Hyperparameter	Value
Population size (N)	40
Probability of crossover (p_{cx})	0.5
Blend parameter (α)	0.15
Tournament size (k)	2
Number of epochs (E)	40
Ranking function ($\boldsymbol{f_r}$)	per case basis
Random number generator seed (s)	per trial basis

seed value is used for each case. Subsequently, this implies that the cases initialize their populations using the same individuals.

5.2 Cardinality Results

As presented in Section 4.1, cardinality analyses were conducted on the Pareto-set and feasible-set. These results are presented and discussed in turn.

Pareto-set

Cardinality of the noncumulative Pareto-set, $|P(t)|$, assesses the number of solutions that are simultaneously optimal at time t. These results are shown in Figure 5.1. For the reference cases, both BASELINE and RANDOM increase over time, though BASELINE increases more than RANDOM. This behavior agrees with our expectation: BASELINE finds simultaneously optimal solutions at a faster rate than RANDOM. When the GA is initialized, each case shares the same initial population, and therefore initial Pareto-sets are established using the same individuals. This is

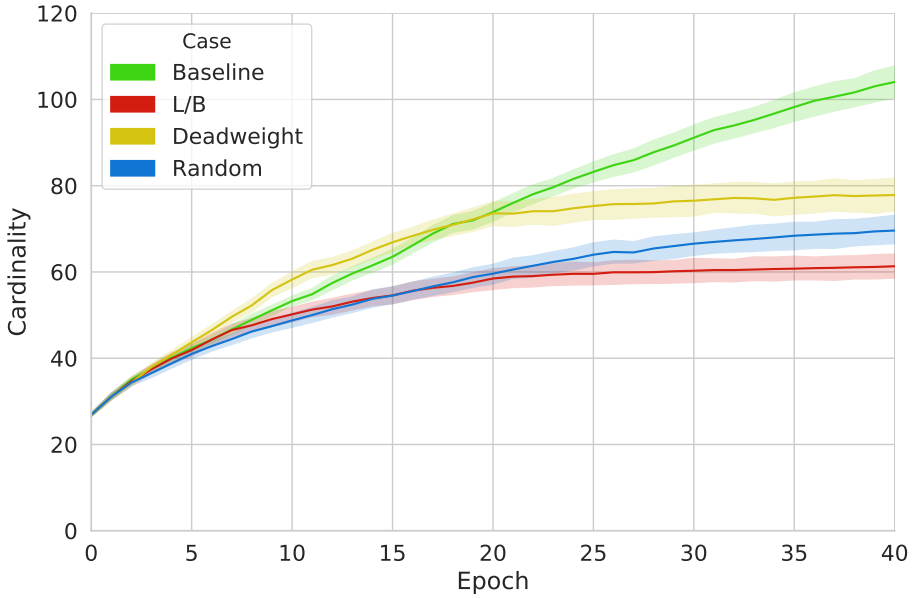


Figure 5.1: Cardinality of the noncumulative Pareto-set, $|P(t)|$.

the reason that the cases all contain the same number of optimal solutions at $t = 0$.

L/B has fewer simultaneously optimal solutions than RANDOM, and L/B has only minimal increase past $t = 20$. Compared to RANDOM, L/B performs worse. Initially, DEADWEIGHT has more than BASELINE, suggesting that the model may be significantly predisposed toward generating DEADWEIGHT-based solutions. However, DEADWEIGHT decreases rapidly while BASELINE continues to increases. This may be due to the fact that DEADWEIGHT is concentrating around a local minimum and ceases to progress outside this, but other analyses will have to be used to explain why this is. By the end, DEADWEIGHT still has more than RANDOM, but RANDOM is increasing faster than DEADWEIGHT at that point: if more iterations were completed, RANDOM would likely find more than DEADWEIGHT.

Cardinality of the cumulative Pareto-set, $|P_c(t)|$, assesses the number of optimal solutions that have been found at or before time t. This equates to the total number of optimal solutions that have been found by time t. When $|P_c(t)|$ reaches an equilibrium, the process has ceased finding additional optimal solutions and has, for practical purposes, converged. These results are shown in Figure 5.2. The reference cases exhibit the same trends that they do in the noncumulative case: both BASELINE and RANDOM increase over time, and BASELINE increases considerably faster than RANDOM. For the biasing cases, the results are similar to those of the noncumulative ($|P(t)|$ and $|P_c(t)|$ have similar dynamics). However, the differences across cases are more pronounced: L/B appears worse, and DEADWEIGHT appears better.

Comparing results between the noncumulative and cumulative Pareto-sets provides a measure of solution improvement over time, since $|P_c(t)| - |P(t)|$ is the number of previously optimal solutions found at time t. When $t = E$, this measures the total improvement on past solutions. BASELINE has good improvement ($|P_c^B(40)| - |P^B(40)| \approx 50$); since $|P^B(40)| \approx 105$, BASELINE improves on roughly

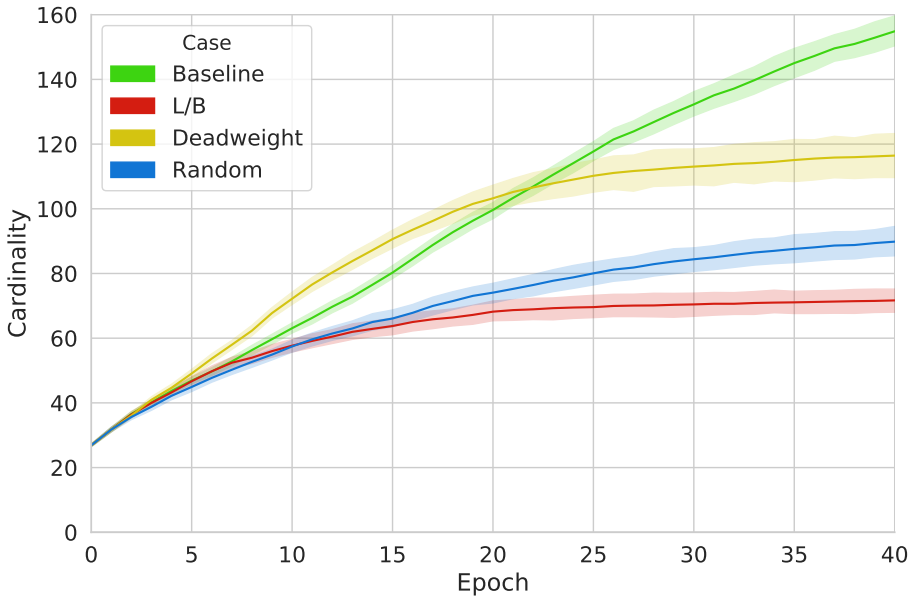


Figure 5.2: Cardinality of the cumulative Pareto-set, $|P_c(t)|$.

48% of its solutions. RANDOM has less improvement ($|P_c^R(40)| - |P^R(40)| \approx 20$); since $|P^R(40)| \approx 70$, RANDOM improves on roughly 28% of its solutions. L/B has less improvement than RANDOM ($|P_c^L(40)| - |P^L(40)| \approx 10$); by the end, only 10 solutions are found that are better than other previously optimal solutions; since $|P^L(40)| \approx 60$, L/B improves on roughly 17% of its solutions. DEADWEIGHT has twice as much improvement as RANDOM ($|P_c^D(40)| - |P^D(40)| \approx 40$); since $|P^D(40)| \approx 75$, DEADWEIGHT improves on roughly 53% of its solutions; in terms of percentage, DEADWEIGHT has more solution improvement than BASELINE.

Feasible-set

Cardinality of the noncumulative feasible-set, $|F(t)|$, assesses the number of feasible solutions that exist in the population at time t. This provides an understanding of the state of the search at t and the search dynamics over time. Since $N = 40$ and $p_{cx} = 0.5$, the expected number of individuals each epoch is equal to $N(1 + p_{cx}) = 40(1.5) = 60$. These results are shown in Figure 5.3. For the reference cases, both BASELINE and RANDOM decrease initially and then continue to

increase. BASELINE does not decrease as much initially, and increases significantly more than RANDOM. The increase of feasible solutions over time makes sense since the ranking functions are penalized using a linear scale factor equal to the current epoch: infeasible solutions are getting worse evaluations as time progresses Since an average of 60 individuals exist each epoch, BASELINE has almost completely feasible solutions by the end. Since optimal solutions must be feasible, BASELINE has a higher opportunity of creating optimal solutions due to its number of feasible solutions.

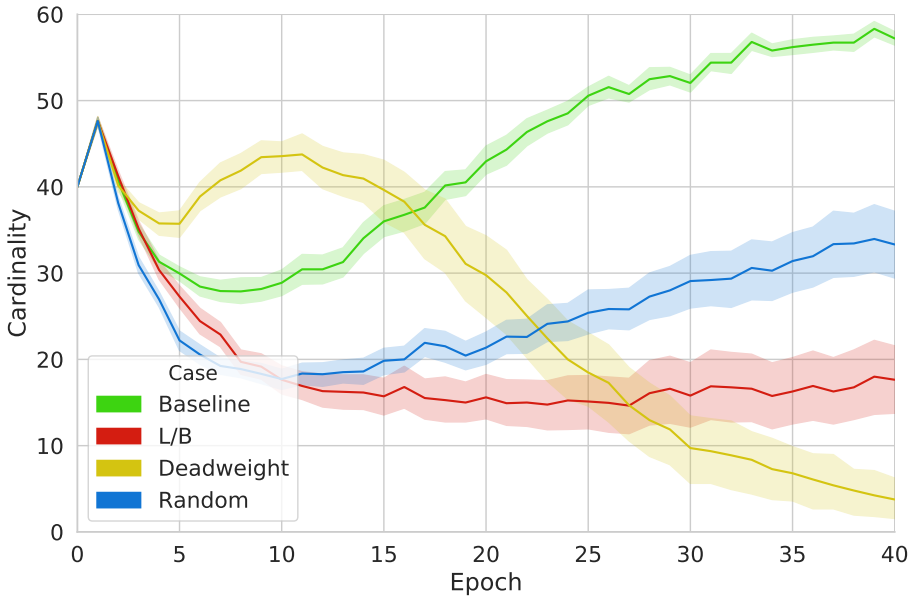


Figure 5.3: Cardinality of the noncumulative feasible-set, $|F(t)|$.

L/B decreases initially like the others, but unlike the others it never increases and instead reaches an equilibrium. Even though L/B has 10-20 feasible solutions each epoch, L/B is not finding more optimal solutions: either (1) these are the same feasible solutions each epoch, or (2) the feasible solution-space motivated by L/B is far from optimal such that they do not dominate the currently optimal solutions. Since L/B does not increase over time, the solution-space searched by L/B must not be very close to constraint boundaries; if it was, feasible solutions would be promoted more. DEADWEIGHT decreases initially, but increases faster than any of the others.

DEADWEIGHT has more than BASELINE in $t = [3, 16]$; however, DEADWEIGHT decreases again starting at $t = 11$. Since the rise of feasible solutions implies that solutions are being penalized, this behaviour suggests that DEADWEIGHT hits a constraint boundary, adapts to it, but then progressively fails to find feasible solutions around that boundary. This also provides an explanation for why DEADWEIGHT has more optimal solutions than BASELINE initially but then decreases: since optimal solution must be feasible, the lack of feasible solutions in DEADWEIGHT represents a lack of opportunity. By the end, DEADWEIGHT has almost no feasible solutions.

Cardinality of the cumulative feasible-set, $|F_c(t)|$, assesses the number of feasible solutions that have been created at or before time t. This differentiates whether feasible solutions are sticking around or if additional feasible solutions are being generated. Additionally, it provides an understanding of how many feasible solutions are created for a given bias, which can be used to assess the likelihood of feasible options in an early stage design activity. When $|F_c(t)|$ reaches an equilibrium, the process has ceased finding feasible solutions. These results are shown in Figure 5.4.

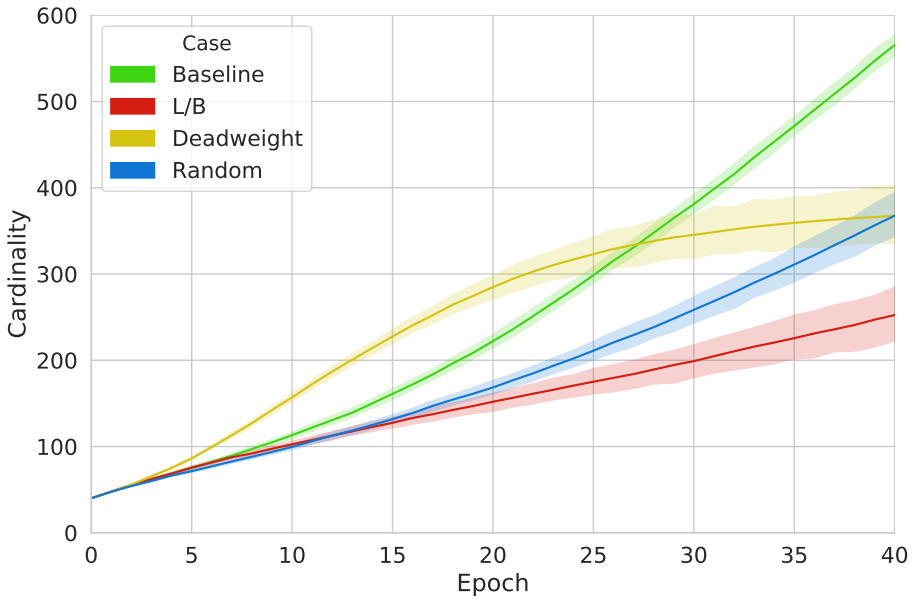


Figure 5.4: Cardinality of the cumulative feasible-set, $|F_c(t)|$.

For the reference cases, both BASELINE and RANDOM have an increasing trend over time, with BASELINE producing more than RANDOM. This makes sense given the noncumulative results already shown. Again, L/B finds fewer than any other case. L/B does not embody a bias that generates feasible options; the model is not biased toward L/B. DEADWEIGHT finds more than any other case initially, but this falls off and by the end, DEADWEIGHT and RANDOM have generated the same number of feasbile solutions. However given their trends, RANDOM would continue to surpass DEADWEIGHT in this regard.

5.3 Coverage Results

To compare the relative optimality of the cases' final Pareto-sets $P(E)$, the coverage analysis discussed in Section 4.2 was conducted. Given the final Pareto-sets $P^A(E)$ and $P^B(E)$ from cases A and B, the coverage metric $C(P^A(E), P^B(E))$ was used to calculate the percentage of solutions in $P^B(E)$ that were dominated by solutions in $P^A(E)$. These results of the reference and biasing cases are presented in Table 5.2, where A is shown in rows, B in columns. The mean μ and standard deviation σ are calculated across the trials performed for each case.

Table 5.2: Coverage results $C(P^A(E), P^B(E))$ assessing the percentage of the Pareto-set of B that is dominated by the Pareto-set of A.

A \ B	Baseline		L/B		Deadweight		Random	
	μ	σ	μ	σ	μ	σ	μ	σ
Baseline	-	-	43.3%	16.4%	22.6%	11.3%	38.2%	17.1%
L/B	13.6%	9.91%	-	-	16.8%	9.91%	22.9%	12.3%
Deadweight	16.2%	10.4%	30.3%	14.4%	-	-	25.7%	13.2%
Random	18.6%	13.0%	34.1%	15.2%	19.4%	11.5%	-	-

For the reference cases, BASELINE dominates 38.2% of RANDOM, meaning that 38.2% of RANDOM's optimal solutions are dominated by BASELINE's optimal solutions. On the other hand, RANDOM dominates 18.6% of BASELINE. Since BASELINE dominates RANDOM more than RANDOM dominates BASELINE, the

BASELINE case produces a more superior Pareto-set than the RANDOM case, as is expected.

Results show that the model is not biased toward the L/B case. L/B dominates BASELINE (13.6%) much less than BASELINE dominates L/B (43.3%). In fact, these are the lowest and highest values in the table. BASELINE generates a much more superior Pareto-set than L/B. Additionally, BASELINE dominates L/B (43.3%) even more than BASELINE dominates RANDOM (38.2%). From BASELINE's perspective, RANDOM generates a better Pareto-set than L/B, indicating that the model is not predisposed toward producing L/B solutions. L/B dominates RANDOM (22.9%) less than RANDOM dominates L/B (34.1%); therefore, RANDOM generates a more superior Pareto-set than L/B. When biased toward L/B, the model generates less optimal solutions than when performing an unguided search. Additionally, RANDOM dominates L/B (34.1%) much more than RANDOM dominates BASELINE (18.6%).

Results show that the model is biased toward the DEADWEIGHT case. DEADWEIGHT dominates BASELINE (16.2%) less than BASELINE dominates DEADWEIGHT (22.6%). However although BASELINE generates a more superior Pareto-set than DEADWEIGHT, the difference is closest. BASELINE dominates DEADWEIGHT (22.6%) much less than BASELINE dominates RANDOM (38.2%). DEADWEIGHT dominates RANDOM (25.9%) more than RANDOM dominates DEADWEIGHT (19.4%). RANDOM dominates DEADWEIGHT (19.4%) only slightly more than RANDOM dominates BASELINE (18.6%).

DEADWEIGHT is producing better solutions than RANDOM; however, the results also suggest that DEADWEIGHT is focusing on an specific area of the objective-space while RANDOM is more well-distributed. RANDOM dominates BASELINE (18.6%) more than DEADWEIGHT dominates BASELINE (16.2%); given the other results for DEADWEIGHT, this makes sense if RANDOM has more solution diver-

sity. DEADWEIGHT dominates RANDOM (25.7%) only slightly more than L/B dominates RANDOM (22.9%). Aside from these results, DEADWEIGHT clearly has better solutions than L/B, further suggesting that DEADWEIGHT is clustered and RANDOM is more diverse.

5.4 Superfront Relative Distance Results

To evaluate the cases' objective-space dynamics relative to the model's "true" Pareto front, the analysis discussed in Section 4.3 was conducted. The superfront-relative-distance metrics were calculated for the Pareto-set and feasible-set at each time step. Results from each of these temporal subsets are discussed in turn.

Pareto-set

The distance to superfront of the noncumulative Pareto-set, $D_T(P(t))$, calculates the average minimum distance from optimal solutions at time t to solutions in the superfront. These results are depicted in Figure 5.5. Agreeing with expectation,

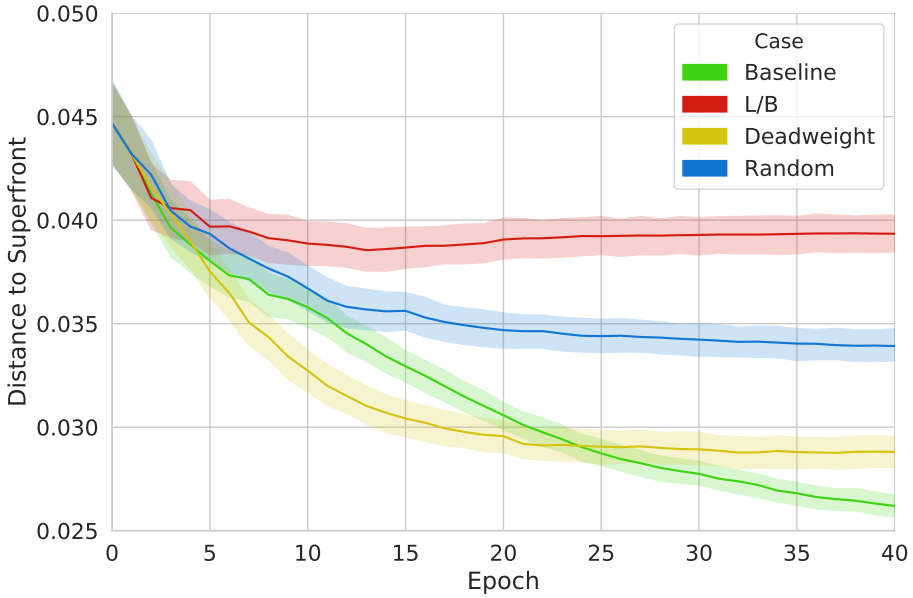


Figure 5.5: Distance to superfront of the Pareto-set, $D_T(P(t))$.

BASELINE is always closer than RANDOM. In addition, the BASELINE has lower variability than RANDOM, indicating that BASELINE's optimal solutions have more similar minimum distances than RANDOM's.

L/B demonstrates poor relative quality: it is far from superfront relative to reference cases and does not show much improvement over time. L/B is always worse than RANDOM, and always much worse than BASELINE; even RANDOM find better solutions than L/B. Aside from a slight initial decrease, L/B does not move closer to the superfront; any optimal solutions it is finding are the same distance away. From the cardinality analysis, L/B does not find additional optimal solutions for $t > 20$. DEADWEIGHT demonstrates high relative quality. DEADWEIGHT is always closer than RANDOM. DEADWEIGHT is even closer than BASELINE for $t < 20$, at which point it reaches an equilibrium while BASELINE continues to improve.

The distance from superfront of the noncumulative Pareto-set, $D_F(P(t))$, calculates the average minimum distance from solutions in the superfront to optimal solutions at time t. These results are depicted in Figure 5.6. Similarly for these results, the superfront is always closer to BASELINE than RANDOM.

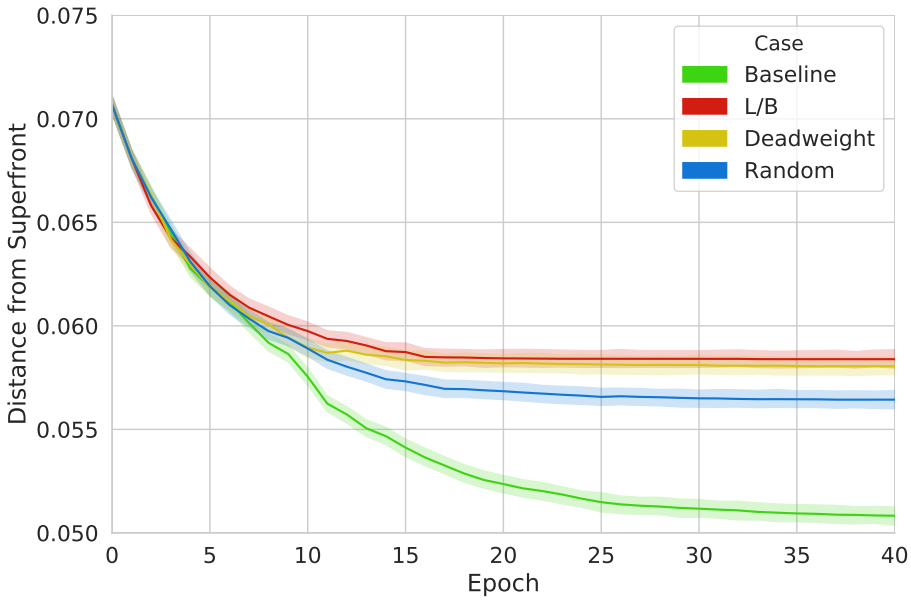


Figure 5.6: Distance from superfront of the Pareto-set, $D_F(P(t))$.

L/B and DEADWEIGHT are both farther from the superfront than RANDOM. L/B is higher than RANDOM for both D_T and D_F; it has worse quality all around. DEADWEIGHT is lower than RANDOM for D_T but higher than RANDOM for D_F; this indicates that DEADWEIGHT is focusing on a localized area in objective-space. This also provided an explanation for the coverage results in Table 5.2. Despite DEADWEIGHT having good results otherwise, BASELINE was dominated by RANDOM more than by DEADWEIGHT, and RANDOM was dominated by DEADWEIGHT only slightly more than by L/B. This scenario makes sense if DEADWEIGHT is generating good solutions that are not well-distributed across the model's objective-space.

Feasible-set

Beyond inspecting the Pareto-set, solution dynamics may also be understood by investigating where the optimizer is searching for potential solutions. The distance to superfront of the noncumulative feasible-set, $D_T(F(t))$, calculates the average minimum distance from feasible solutions in $F(t)$ to solutions in the superfront. These results are depicted in Figure 5.7. For the reference cases, BASELINE steadily decreases, while RANDOM decreases initially, increases slightly, and then seems to reach an equilibrium. Feasible solutions found by RANDOM are not getting closer to superfront over time, continuing the search with little solution improvement.

L/B is always farther than RANDOM, although it has similar dynamics as RANDOM. In L/B, variability decreases over time. In order for variability to decrease, the distance of the feasible solutions to superfront has to be becoming more similar over time: in L/B, the feasible solution-space is moving away from the superfront. DEADWEIGHT is always closer than RANDOM, and closer than BASELINE up until the end. DEADWEIGHT's trend is sporadic and has the most variability at $t = E$. From the cardinality results in Figure 5.3, the number of feasible solutions

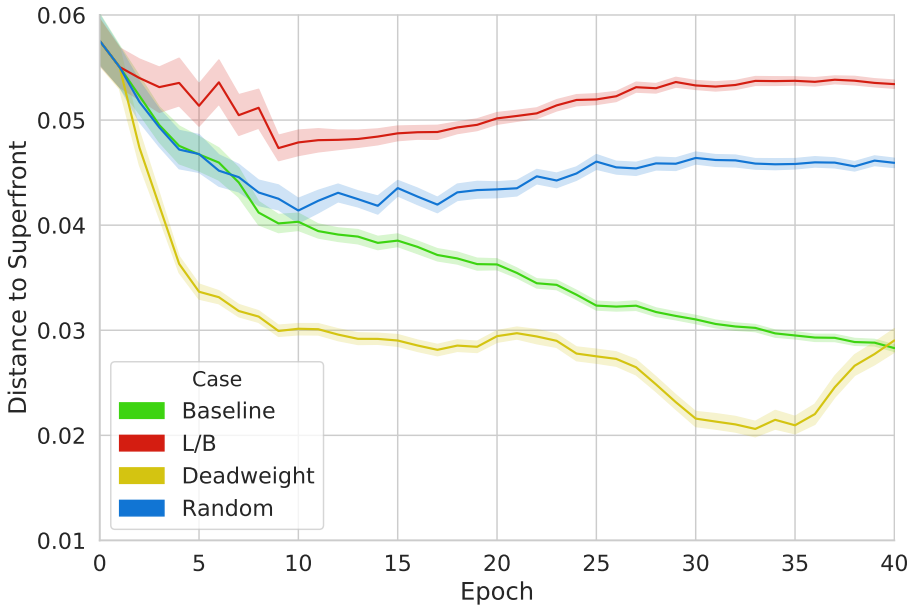


Figure 5.7: Distance to superfront of the feasible-set, $D_T(F(t))$.

for DEADWEIGHT decreases for $t > 10$. This suggests that DEADWEIGHT is very close to optimal but is pushing into a constraint boundary in objective-space. Although DEADWEIGHT finds fewer feasible solutions for $t > 10$, the feasible solutions that it does find are all very close to optimal, relatively.

The distance from superfront of the noncumulative feasible-set, $D_F(F(t))$, calculates the average minimum distance from solutions in the superfront to solutions in $F(t)$. These results are depicted in Figure 5.8. For the reference cases, BASELINE and RANDOM increase steadily, with BASELINE being less than RANDOM. D_F increasing means solutions are getting farther from SF, while D_T decreasing means solutions are getting closer to SF, and the low variability of both metrics means than solutions are all similar distances. This behavior shows that solutions are getting closer to optimal but are concentrated in an particular area and not well-distributed across the superfront; the diversity of feasible solutions decreases over time.

L/B is farther than RANDOM, similarly as with D_T. L/B is worse than RANDOM in all regards since L/B is farther *to* and *from* the superfront. DEADWEIGHT increases more rapidly than any other case; feasible solutions in DEADWEIGHT are

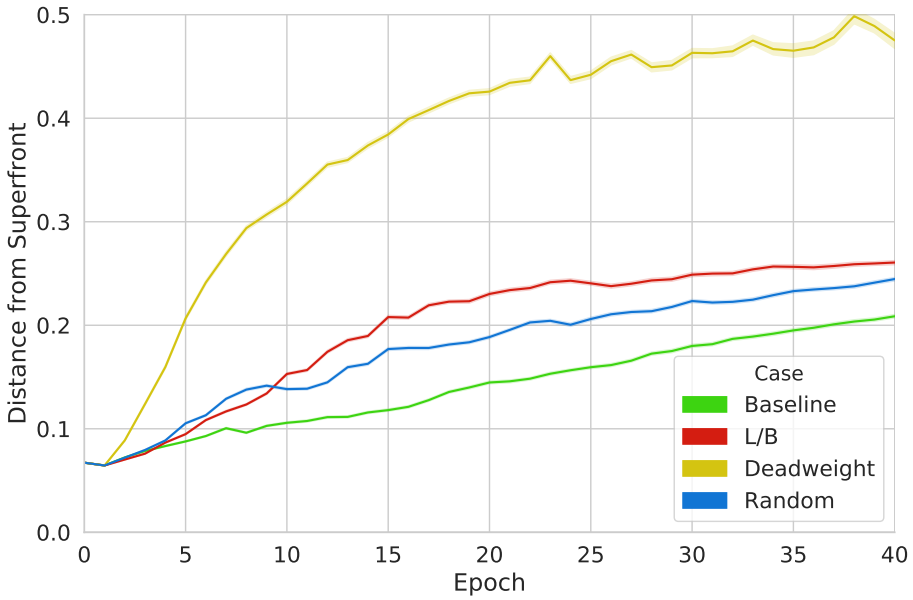


Figure 5.8: Distance from superfront of the feasible-set, $D_F(F(t))$.

getting increasingly farther from the superfront. Since DEADWEIGHT D_T is closer than BASELINE, but DEADWEIGHT D_F is much farther than any case, this confirms than DEADWEIGHT is concentrated in an area and not well-distributed across the superfront.

5.5 Generational Distance Results

To assess convergence and compare differences across cases, the analysis presented in Section 4.4 was conducted. In this analysis, the current Pareto-set $P(t)$ is evaluated relative to the previous Pareto-set $P(t-1)$ using the Generational Distance metric $GD(P(t-1), P(t))$. These results are shown in Figure 5.9 and demonstrate that all of the processes converge. Given that the previous analyses have shown large differences in each case's quality, this demonstrates that the phrase "optimal, converged solutions" does not necessarily imply any assessment of the solutions' quality. For the reference cases, BASELINE converges slower than RANDOM, which makes sense given that BASELINE is improving on solutions much more than RANDOM. L/B

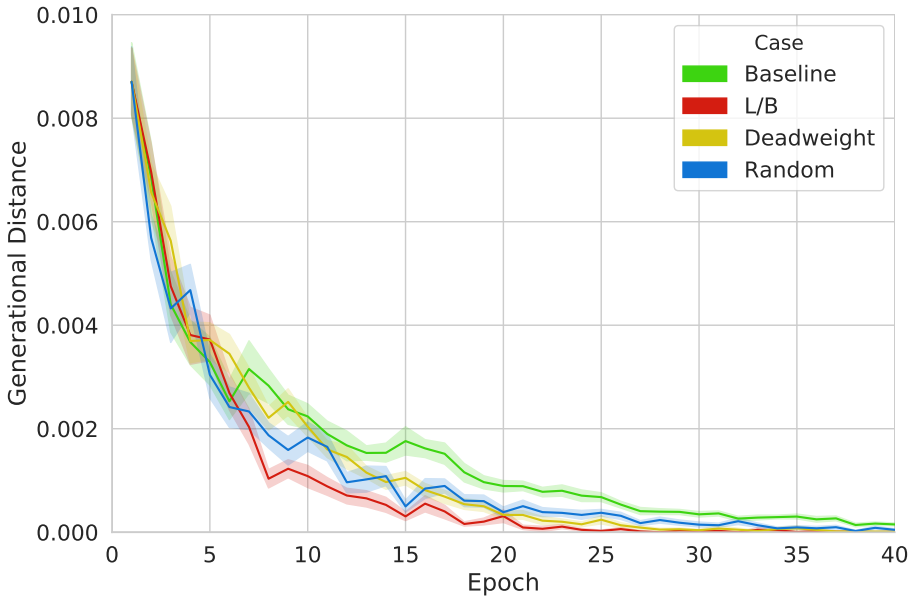
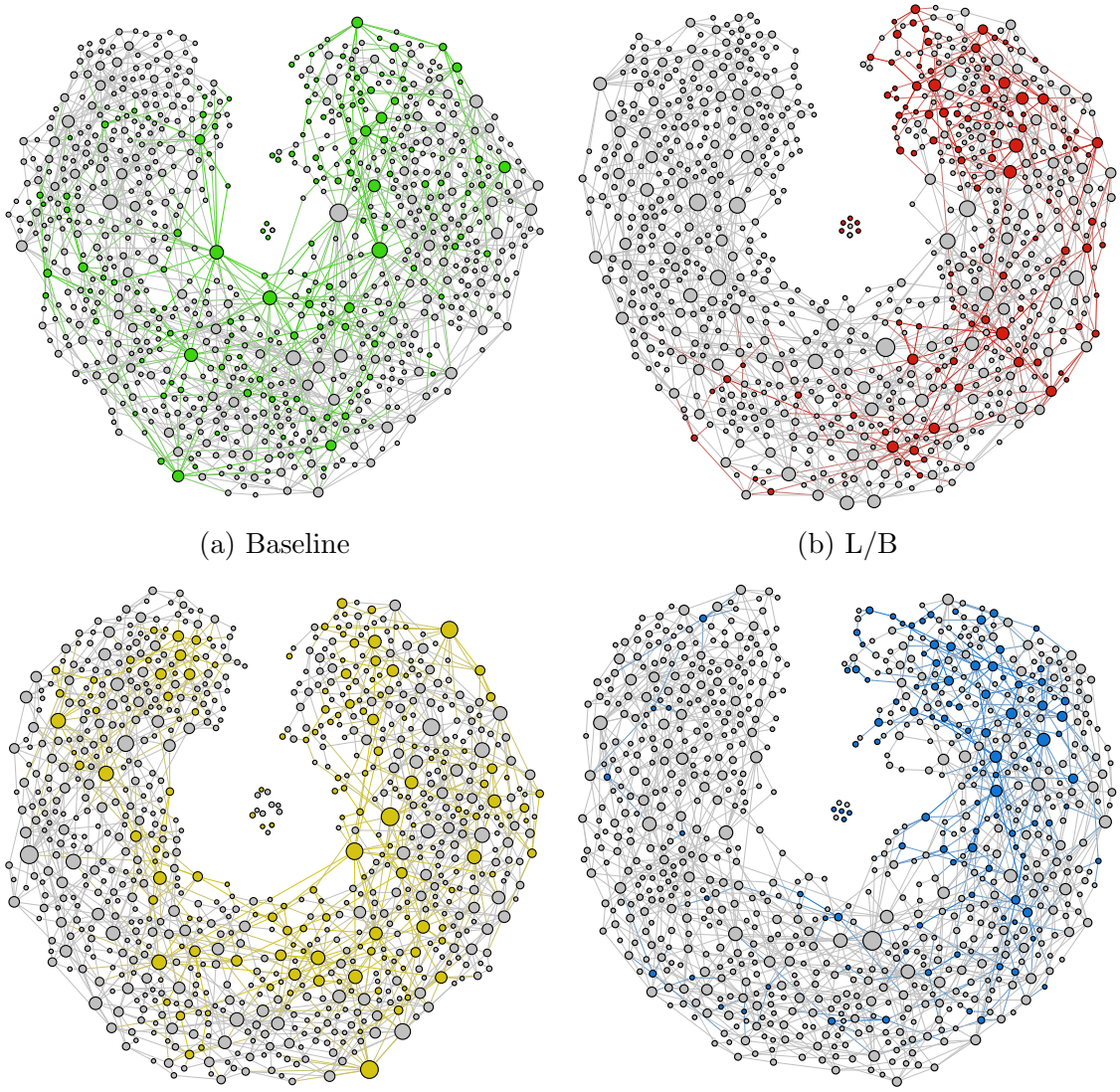


Figure 5.9: Generational Distance as a convergence metric, $GD(P(t-1), P(t))$.

converges faster than RANDOM; its lack of solution improvement results in quicker convergence. Initially, DEADWEIGHT is slightly slower than RANDOM, but then DEADWEIGHT converges before RANDOM. This also agree with the amount of solution improvement that DEADWEIGHT experiences over time.

5.6 Out-degree Results

Next the out-degree analysis is completed, as presented in Section 4.5. The PCNs are depicted in Figure 5.10, where a node's size illustrates its relative out-degree, and colored nodes are contained in $P_c(E)$. In BASELINE, optimal nodes are generated continually throughout the process. Additionally, optimal nodes have more influence than dominated nodes, in general. There are some large nodes that are not optimal; these solutions may be close to optimal, but that isn't assessed in this analysis. In RANDOM, the majority of optimal nodes are found early in the process. Since RANDOM is an unguided search with little solution improvement, the easiest optimal solutions to find are found early. Although optimal nodes are found later, they



(a) Baseline (b) L/B

(c) Deadweight (d) Random

Figure 5.10: Aggregate PCNs for each case. A node's size illustrates its relative out-degree, and colored nodes are contained in $P_c(E)$.

have low out-degree values; they are not perceived as being better by the solution generation process.

In L/B, most optimal solutions are found early as well, similarly to RANDOM; the second half of the search finds very few optimal solutions. Additionally, the most influential nodes (highest out-degrees) are not optimal nodes, indicating that optimal nodes are not as influential in the process. In DEADWEIGHT, optimal solutions

are found throughout, similar to BASELINE Additionally, optimal nodes have higher out-degree, indicating that optimal nodes are more influential in the process.

To consider all trials, logistic regression is used to model the probability that a node resides in $P_c(E)$ given its out-degree. This answers the question, "Do optimal nodes influence the process more?" These results are shown in Figure 5.11. For a high quality results, optimal nodes should have higher out-degree scores; the probability should increase with out-degree. This is seen in BASELINE, where the most influential nodes have an 80% probability of being optimal; out-degree is a good indicator of a node's optimality. RANDOM is less than BASELINE, as expected; the most influential nodes only have an 35% probability of being optimal. However, RANDOM also has a positive trend; this makes sense given that many optimal nodes are found in the initial population, which are more likely to have higher out-degree.

L/B is much less than BASELINE and similar to RANDOM; the most influential nodes only have an 30% probability of being optimal. In L/B, out-degree is a poor indicator of optimal nodes; L/B has poor generative quality within the model, thereby

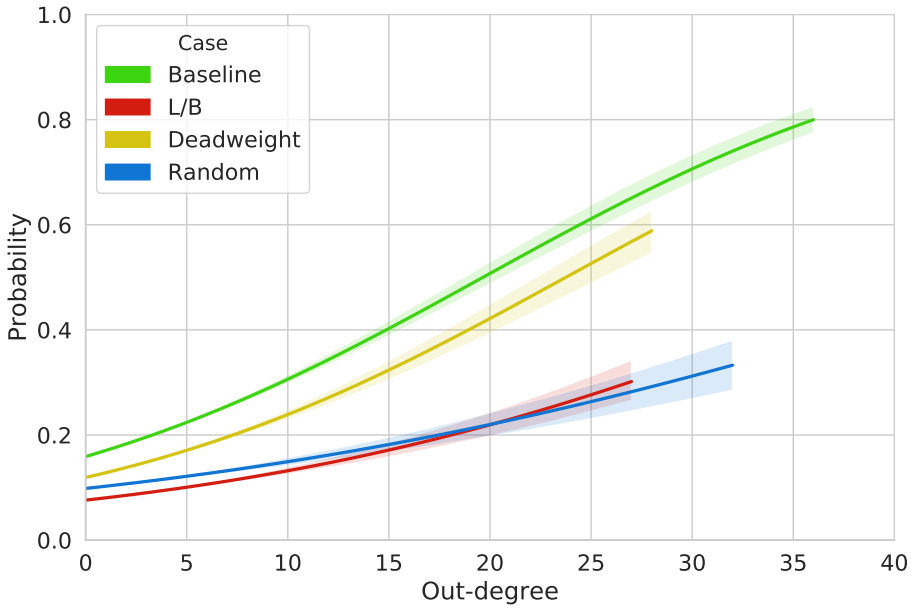
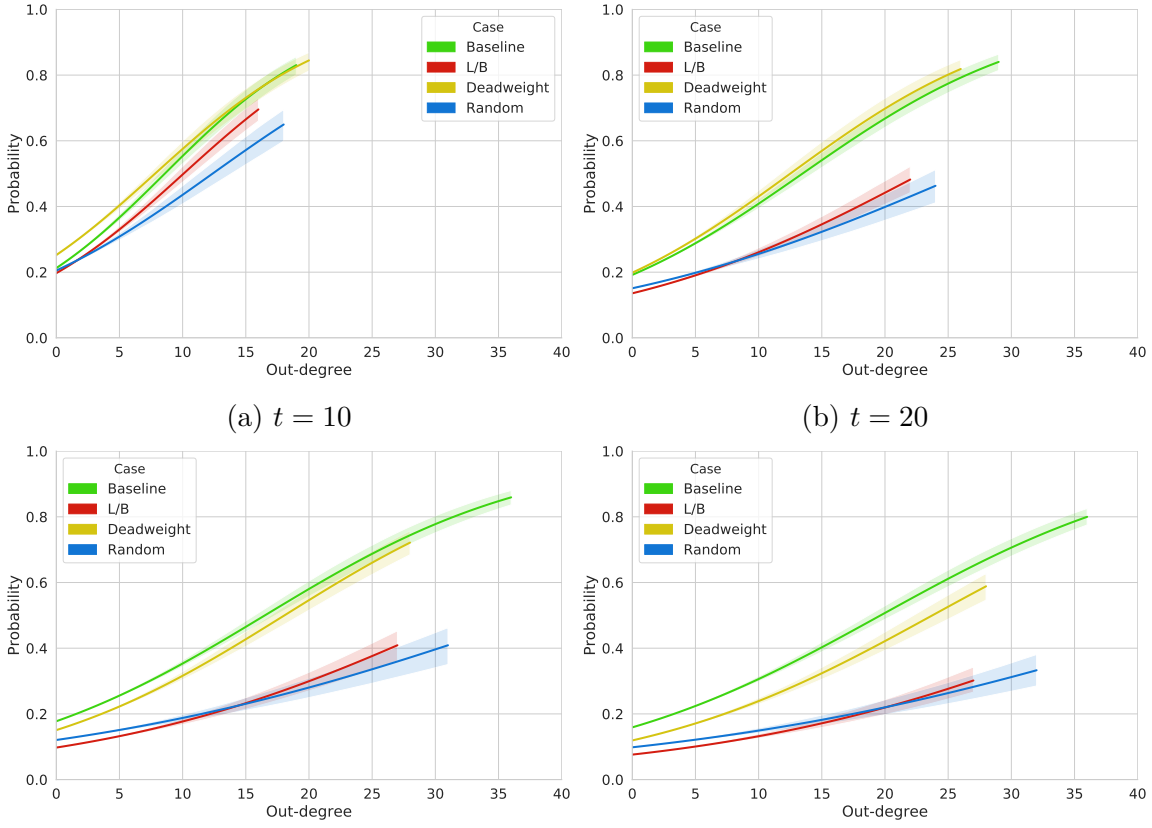


Figure 5.11: Probability of a node residing in $P_c(E)$ given its out-degree.

the model is not biased toward L/B. DEADWEIGHT is greater than RANDOM and less than BASELINE (although similar); the most influential nodes have a 60% probability of being optimal, demonstrating that optimal nodes are more influential in the process.

Figure 5.11 considers the aggregate PCNs at the end of the process ($t = 40$). The same process is be conducted at different time steps to understand the progression and to look for leading indicators of quality. Figure 5.12 shows these results at epochs 10, 20, 30, and 40. As t increases, BASELINE remains high and RANDOM decreases. In BASELINE, the influence of optimal nodes continues to be heavily utilized; whereas in RANDOM, this is not the case. As t increases, L/B behaves similarly to RANDOM: L/B is slightly higher than RANDOM at $t = 10$ and then progressively decreases



(a) $t = 10$ (b) $t = 20$

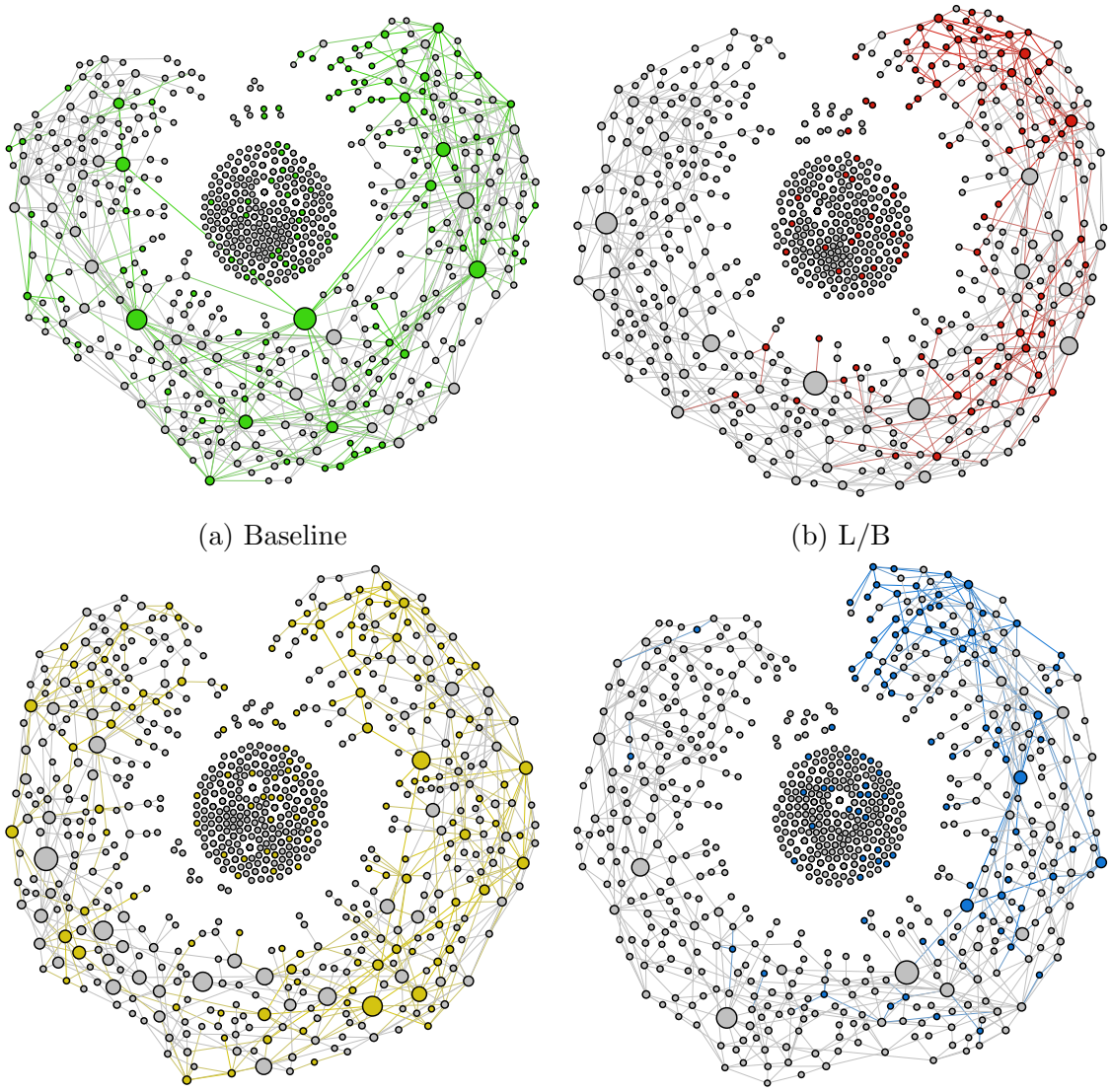
(c) $t = 30$ (d) $t = 40$

Figure 5.12: Probability of a node residing in $P_c(E)$ given its out-degree. Evaluating out-degree as a leading indicator of bias.

to be slightly lower than RANDOM at $t = 40$. Optimal nodes do not represent influential solutions within the process, indicating the model is not biased toward L/B. DEADWEIGHT is very similar to (if not slightly higher than) BASELINE at $t = 10$ and $t = 20$; the most influential nodes have a 90% probability of being optimal. DEADWEIGHT decreases to less than BASELINE at $t = 30$ (70%) and $t = 40$ (60%). However given the other analyses, DEADWEIGHT is not producing many feasible or optimal solutions at the end, and DEADWEIGHT converges toward the constraint boundary quickest.

5.7 Betweenness Centrality Results

Next the betweenness analysis is considered, as presented in Section 4.6. The resulting PPNs are depicted in Figure 5.13, where a node's size illustrates its relative betweenness, and colored nodes are contained in $P_c(E)$. In BASELINE, optimal nodes have highest betweenness: optimal nodes are most central in the network. Since the PPN represents the utilization of information within the solution generation process, this means that, for BASELINE, the information contained in optimal nodes is very important. In RANDOM, optimal nodes have low betweenness, especially later in the process (early nodes are predisposed to being optimal); the information contained in optimal nodes is not perceived as being better by the solution generation process. In L/B, optimal nodes have low betweenness, similarly to RANDOM; the most central nodes are not optimal nodes. DEADWEIGHT is between BASELINE and RANDOM; in the first half of the search (from 1:00 to 6:00), the most central nodes are optimal. However in the second half (from 6:00 to 11:00), the most central nodes are not optimal, although there are optimal nodes in the second half that have decent betweenness values. While RANDOM and L/B have only a few highly central nodes in the second half, DEADWEIGHT has many influential nodes in the second half.



(a) Baseline (b) L/B

(c) Deadweight (d) Random

Figure 5.13: Aggregate PPNs for each case. A node's size illustrates its relative betweenness, and colored nodes are contained in $P_c(E)$.

To consider all trials, logistic regression is used to model the probability that a node resides in $P_c(E)$ given its betweenness. This answers the question, "Are optimal nodes central to the utilization of information in the process?" These results are shown in Figure 5.14. In BASELINE, the most central nodes have more than a 95% probability of being optimal; betweenness is a good indicator of a node's optimality. RANDOM is much less than BASELINE but still has a positive trend; the most

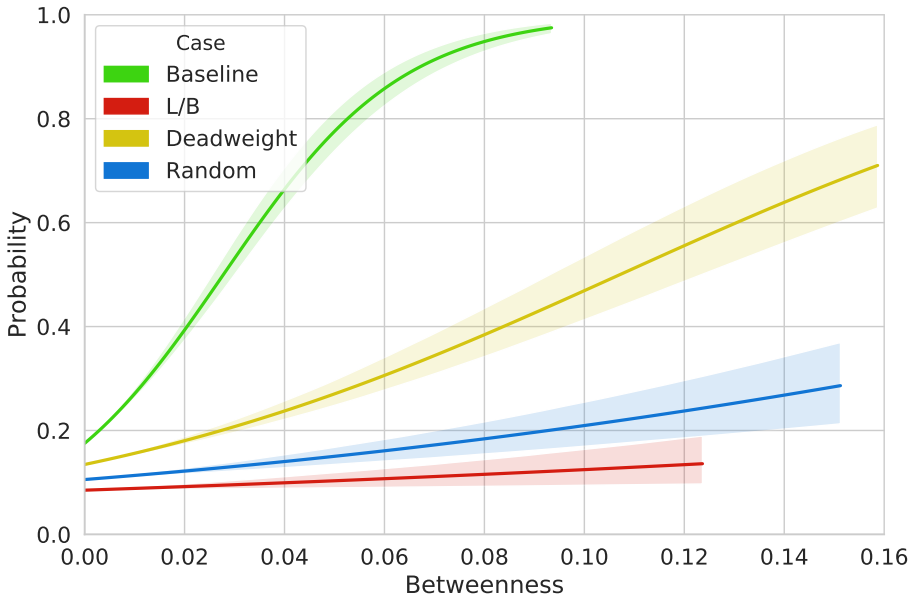


Figure 5.14: Probability of a node residing in $P_c(E)$ given its betweenness.

central nodes only have a 30% probability of being optimal. L/B is much less than BASELINE and even less than RANDOM; the most central nodes only have a 15% probability of being optimal; In L/B, betweenness is not a good indicator of optimal nodes, and the information contained in optimal nodes is not well-utilized by the process. DEADWEIGHT is greater than RANDOM but still less than BASELINE; the most central nodes have a 70% probability of being optimal. Optimal nodes are central, and the information contained in them is utilized by the process.

As discussed in Section 4.6, this analysis is primarily concerned with evaluating solutions that actually contribute to the process. Figure 5.15 shows the probability results when only considering nodes in the largest component of the PPN. This evaluates how central a node is within the community of utilized information. BASELINE has a strong positive trend with betweenness; the most central nodes have an 90% probability of being optimal. RANDOM is expected to be relatively flat, as it is; the most central nodes have a 15% probability of being optimal. L/B is less than RANDOM; the most central nodes have less than a 5% probability of being optimal. Furthermore, L/B has a negative trend with betweenness; the probability of a node

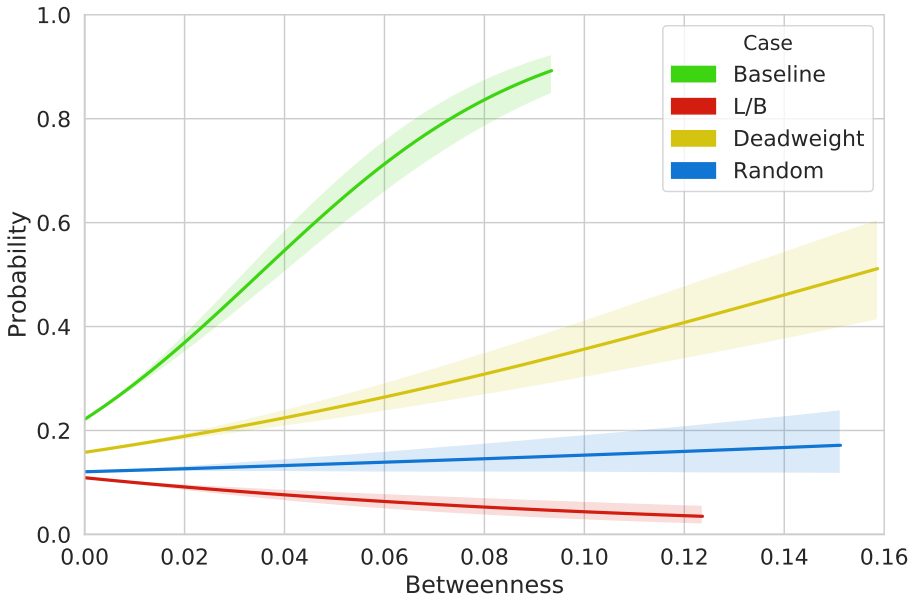


Figure 5.15: Probability of a node residing in $P_c(E)$ given its betweenness. Considering only nodes in the largest component of the PPN.

being optimal decreases as its information becomes more utilized; the model is definitely not biased toward L/B. DEADWEIGHT is greater than RANDOM but less than BASELINE; the most central nodes have a 50% probability of being optimal. DEADWEIGHT has the most uncertainty; as can be seen in the PPN visualization, DEADWEIGHT has more varied values of betweenness for both its optimal and dominated solutions.

5.8 Effects of Increasing Tournament Size

As discussed in Section 3.3.4, the tournament size can be increased to increase the amount of bias imposed by the ranking function. This is demonstrated using a cardinality analysis of the noncumulative feasible-set, $|F(t)|$. To conduct this analysis, additional HANs were generated for each of the cases using tournament sizes of 3 and 5; the other hyperparameters of the GA were unchanged. These results are shown in Figure 5.16. As k increases, BASELINE decreases less initially, but does not increase as much in the end. As expected, RANDOM does not change with k. As k increases,

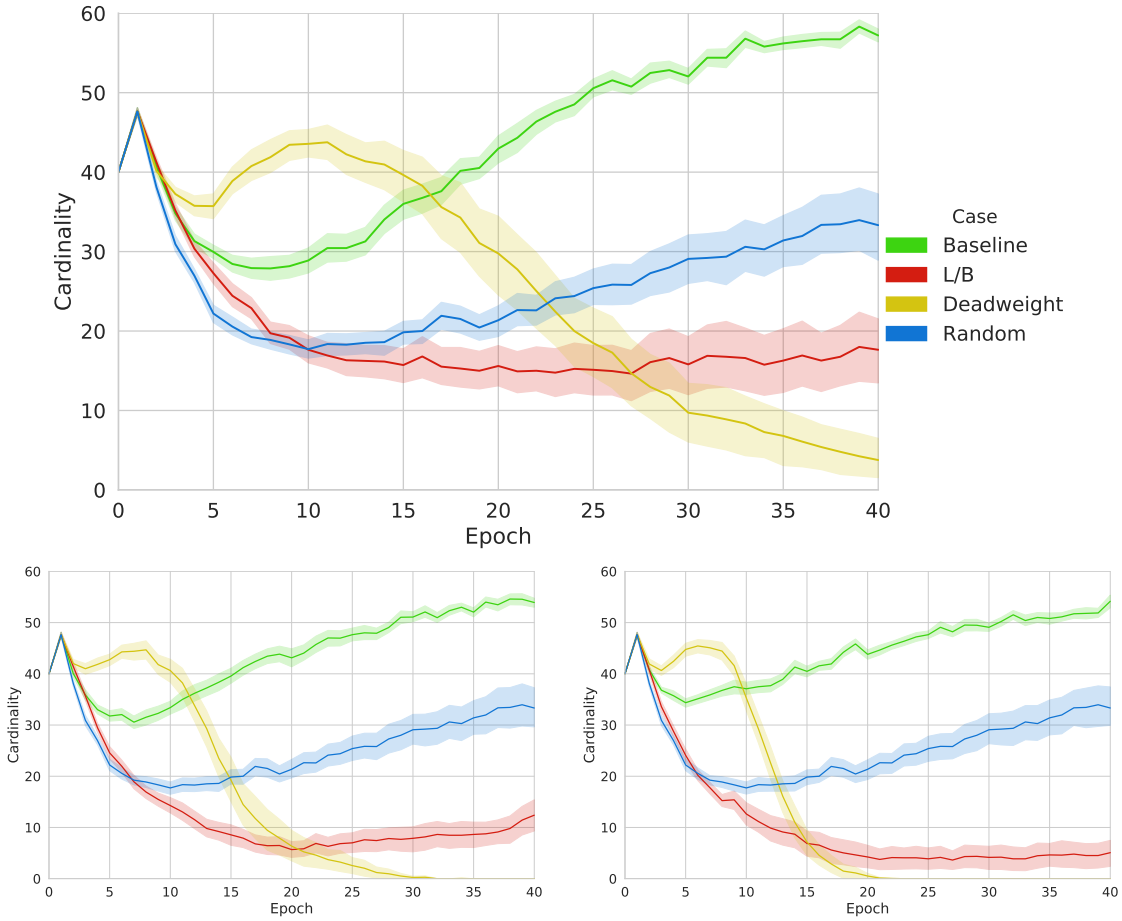


Figure 5.16: Cardinality of the noncumulative feasible-set, $|F(t)|$. Comparison of tournament sizes $k = 2$ (*top*), $k = 3$ (*bottom left*), and $k = 5$ (*bottom right*).

L/B decreases; fewer feasible solutions exist in the population each epoch; relative to RANDOM, L/B gets worse. As k increases, DEADWEIGHT decreases less initially, increases, and then decreases more rapidly; in fact DEADWEIGHT has no feasible solutions at $t = 30$ when $k = 3$ and at $t = 20$ when $k = 5$.

5.9 Conclusions

Identifying the underlying influences within design tools is crucial to ensure that design outcomes are not being predicated by the tools employed. Since the synthesis model was designed to produce parameters for bulk carriers, these results are understandable and expected. The bias toward deadweight and away from L/B align with

institutional knowledge regarding these vessels. As such, the methods and analysis created in this thesis provide designers with a sense of validation for the tool in its intended application. The analysis also indicates that the tool should not be utilized universally since it contains inherent influences from traditional bulk carriers; this includes simply trying to create ships with a maximum deadweight. While L/B was never a biasing factor (it always had lower quality) and although the model is biased toward deadweight, the results demonstrated the deadweight case does not capture the full intent of the model. This case study shows that the Sen bulker model has other design drivers beyond the simplistic view that the best bulk carrier is the one with the maximum capacity. The model also includes aspects of cost, and thus it is balancing both optimal cargo capacity and cost of that decision.

CHAPTER VI

Conclusions

6.1 Contributions

This thesis has been focused on the development of methods to answer the fundamental question, "How can an analysis tool be evaluated to determine if it is presenting biased information?" Toward this goal, the following are unique contributions of this research:

- Characterized a novel perspective of quality. Within the context of early stage ship design, the classical definition of quality associated with a design tool is insufficient. An extended definition of tool quality has been developed as part of this thesis. The contributions are:
  - Recognition that an honest assessment of quality requires new novel multiple contextual views.
  - Defined and developed two unique views of quality, *solution-centric* and *generative*.
  - Identified the factors necessary to evaluate quality and how evaluations of quality can be used to understand bias.
- Development of a novel framework to enable the evaluation of a design tool's biases.

  - Creation of a novel GA procedure that provides a dynamic environment of the solution generation process.
  - Modified the underlining process of a GA to enable the intentional imposition of bias.
  - Utilized this GA to establish a comparative context by defining reference and biasing cases.
  - Created two Hereditary-Amelioration Networks that are derived from the dynamics within the GA, captured the implied-causality behind solution evolution, and represented the utilization of information throughout the GA's optimization procedure.

- Developed novel solution-centric and generative analysis techniques that utilized the newly developed framework to evaluate a model's inherent biases.
  - Developed novel analysis metrics, extended existing metrics, and applied some existing metrics in a novel way to evaluate quality and understand bias.
  - These metrics include:
    * Cardinality of the Pareto-Set and Feasible-Set
    * Coverage
    * Superfront Relative Distance of the Pareto-set and feasible-set
    * Generational Distance
    * Out-degree
    * Betweenness Centrality

- Completed a representative case study that demonstrates the effectiveness of the developed framework and analysis methods.

6.2 Future Topics of Interest

As with any new and novel research activities there are many possible future opportunities for expansion. The focus of this thesis was the creation of the framework needed to develop the data needed for bias investigation as well as a set of analysis techniques that are sufficient to draw conclusion concerning bias of an early stage design tool. With the completion of this thesis there are several future topics concerning the framework developed in Chapter III and Chapter IV that are worth investigation. These include but are not limited to:

- Implementing the mutation operator within the current GA, which can prevent convergence in local minima by creating new individuals that are not solely derivatives of their parents. However to accomplish this, the HANs should be adapted to accurately represent these individuals.

- Developing strategies to define biasing cases that do not require the manual specification of the designer.

- Expanding the current GA to consider different selection and crossover operators. For example, implementing the selection operator as NSGA-II to impose diversity. Since tournament selection is modified to impose bias, a different method of imposing bias on the solution generation process would be required.

- Developing additional HANs to represent additional or different aspects of the solution generation process. For example, a multilayer network that represents (1) individuals on one layer and (2) inputs and objective values on other layers. Edges from the input layer to node layer would represent which inputs belong to which nodes; objective values would be similarly represented by edges from the node layer to the objective layer. Using this structure, analysis methods can be developed to investigate the dynamics of, influences within, and relationships

between the input-space and objective-space of the model.

- Investigation of HANs development in different optimization procedures. For early stage design purposes, an appropriate next step would be developing HANs for use in multidisciplinary optimizers.

- Investigating application of the framework to models without specified objectives.

- In the solution-centric analyses that evaluated distance, solution sets were evaluated discretely; that is, each solution was considered as a discrete point, and distance metrics were calculated using discrete points. Since solutions may be clustered in a certain area, distance evaluations using a discrete representation do not accurately reflect the design space characterized by the given solution set. Methods for determining continuous representations of discrete sets should be investigated to more accurately reflect the quality of solutions. For example, discrete solutions could be represented by their interpolated surface; then, a quality metric could be defined by surface area or by the integrated distance between two surfaces. This has particular significance to the superfront.

- Developing metrics that are applicable to objective-spaces with more than three dimensions. Research and practice have shown that Euclidean distance is a less than desirable measure of distance when operating in more than three dimensions.

- Investigating out-degree and betweenness as indicator variables for continuous evaluation of quality. In this work, the network metrics were assessed as indicators for a discrete result: optimal or not. This binary segregation does not reflect how close solutions are to optimal (similarly to the difference between coverage and superfront-relative-distance). Therefore, one example is using out-degree

as an indicator for distance to superfront.

- Developing conditional network metrics for the PCN to evaluate the influence that a node has on its descendant and the opportunity a node based on its ancestors.

- Developing analyses investigating influential communities within the PPN based on the network's structural aspects of network. In contrast, out-degree and betweenness are node-based evaluations and do not capture the structure of the network.

- In general, the development of solution-centric and generative metrics to evaluate the different aspects of quality.

- Evaluating results across different hyperparameters for the GA. In Chapter V, solutions were generated by implementing the GA using the same hyperparameters. This would be done (1) across a sweep of hyperparameters, (2) using optimal hyperparameters of the baseline case, and (3) the optimal hyperparameters of each case. By comparing across cases, analyses would further investigate the biases of the model by evaluating the differences of the resulting solutions and the utilization of information within their processes.

- Applying the framework to larger, more complex models.

APPENDIX

APPENDIX A

Bulk Carrier Synthesis Model

This appendix provides an overview of the bulk carrier synthesis model developed by *Sen and Yang* (1998). Additional information may also be found in *Yang et al.* (1990) and *Yang and Sen* (1996).

Inputs and Intermediate Functions

The model defines six inputs: length (L), beam (B), draft (T), depth (D), speed (V), and block coefficient (C_B). These inputs can be expressed by the input vector $\boldsymbol{x} = [L, B, T, D, V, C_B]$. Using these inputs, the model defines a host of intermediate functions:

$$\begin{aligned} \text{annual cost} &= \text{capital charges} + \text{running cost} \\ &\quad + \text{voyage cost} + \text{RTPA} \end{aligned} \tag{A.1}$$

$$\text{capital charges} = 0.2 \times \text{ship cost} \tag{A.2}$$

$$\begin{aligned} \text{ship cost} &= 1.3 \times (\text{steel mass})^{0.85} \\ &\quad + 3500 \times \text{outfit mass} + 2400 \times P^{0.8} \end{aligned} \tag{A.3}$$

$$\text{steel mass} = 0.034 \times L^{1.7} \times B^{0.7} \times D^{0.4} \times C_B^{0.5} \tag{A.4}$$

$$\text{outfit mass} = L^{0.8} \times B^{0.6} \times D^{0.3} \times C_B^{0.1} \tag{A.5}$$

$$\text{machinery mass} = 0.17 \times P^{0.9} \tag{A.6}$$

$$\text{P} = \Delta^{2/3} \times V^3 \times \frac{1}{b(c_B) \times \frac{V}{(g \times L)^{0.5}} + a\,(C_B)} \tag{A.7}$$

$$\Delta = 1.025 \times L \times B \times T \times C_B \tag{A.8}$$

$$\text{running cost} = 40000 \times DW^{0.3} \tag{A.9}$$

$$DW = \Delta - \text{light ship mass} \tag{A.10}$$

$$\text{voyage cost} = \text{fuel cost} + \text{port cost} \tag{A.11}$$

$$\text{fuel cost} = 1.05 \times \text{daily consumption} \times \text{sea days} \times \text{fuel price} \tag{A.12}$$

$$\text{daily consumption} = P \times 0.19 \times 0.024 + 0.2 \tag{A.13}$$

$$\text{sea days} = \frac{\text{round trip miles}}{24 \times V} \tag{A.14}$$

$$\text{round trip miles} = 5000\,(\text{nautical miles}) \tag{A.15}$$

$$\text{fuel price} = 100\,(\text{pounds/ton}) \tag{A.16}$$

$$\text{port cost} = 6.3 \times DW^{0.8} \tag{A.17}$$

$$\text{RTPA} = \frac{350}{\text{sea days} + \text{port days}} \tag{A.18}$$

$$\text{port days} = 2 \times \left(\frac{\text{cargo deadweight}}{\text{cargo handling rate}} + 0.5 \right) \tag{A.19}$$

$$\text{cargo deadweight} = DW - \text{fuel carried} - \text{crew, stores, and water} \tag{A.20}$$

$$\text{fuel carried} = \text{daily consumption} \times (\text{sea days} + 5) \tag{A.21}$$

$$\text{crew, stores, and water} = 2.0 \times DW^{0.5} \tag{A.22}$$

$$\text{cargo handling rate} = 8000\,(\text{tons/day}) \tag{A.23}$$

where $RTPA$ is round trips per annum, DW is deadweight, and g is the gravitational constant ($g = 9.8065\ m/s^2$). The functions $a(C_B)$ and $b(C_B)$ are regression equations based on Froude Number and a coefficient referred to as the *Admiralty Coefficient*, detailed in the original paper.

Objectives

The model defines three objectives:

$$\Omega_1 = \min(\text{transportation cost}) \tag{A.24}$$

$$\Omega_2 = \min(\text{light ship mass}) \tag{A.25}$$

$$\Omega_3 = \max(\text{annual cargo}) \tag{A.26}$$

which can be expressed by the objective vector $\mathbf{\Omega} = [\Omega_1, \Omega_2, \Omega_3]$. The functions that comprise the individual objectives are defined in terms of the model's intermediate functions as:

$$\text{transportation cost} = \frac{\text{annual cost}}{\text{annual cargo}} \tag{A.27}$$

$$\text{light ship mass} = \text{steel mass} + \text{outfit mass} + \text{machinery mass} \tag{A.28}$$

$$\text{annual cargo} = \text{cargo deadweight} \times \text{RTPA} \tag{A.29}$$

Constraints

The model defines dimensional and displacement constraints:

$$L/B \geq 6 \tag{A.30}$$

$$L/D \leq 15 \tag{A.31}$$

$$L/T \leq 19 \tag{A.32}$$

$$T \leq 0.45 \times DW^{0.31} \tag{A.33}$$

$$T \leq 0.7 \times D + 0.7 \tag{A.34}$$

$$DW \geq 3000 \tag{A.35}$$

$$DW \leq 500000 \tag{A.36}$$

powering constraints:

$$C_B \geq 0.63 \tag{A.37}$$

$$C_B \leq 0.75 \tag{A.38}$$

$$V \geq 14 \tag{A.39}$$

$$V \leq 18 \tag{A.40}$$

$$\frac{V}{(g \times L)^{0.5}} \leq 0.32 \tag{A.41}$$

and a stability constraint:

$$GM \geq 0.07 \times B \tag{A.42}$$

where

$$GM = KB + BM - KG \tag{A.43}$$

$$KB = 0.53 \times T \tag{A.44}$$

$$BM = \frac{(0.085 \times C_B - 0.002) \times B^2}{T \times C_B} \tag{A.45}$$

$$KG = 1.0 + 0.52 \times D \tag{A.46}$$

BIBLIOGRAPHY

BIBLIOGRAPHY

Andrews, D. J. (2012), Art and science in the design of physically large, *Proceedings of the Royal Society, A*, 891–912, doi:10.1098/rspa.2011.0590.

Audet, C., J. Bigeon, D. Cartier, and S. Le (2018), Performance indicators in multiobjective optimization, *European Journal of Operational Research*, pp. 1–39.

Chalfant, J. (2015), Early-Stage Design for Electric Ship, *Proceedings of the IEEE*, *103*(12), 2252–2266, doi:10.1109/JPROC.2015.2459672.

Coello Coello, C. A., G. B. Lamont, and D. a. V. Veldhuizen (2007), *Evolutionary Algorithms for Solving Multi-Objective Problems*, 800 pp., Springer, doi:10.1007/978-0-387-36797-2.

Collette, Y., and P. Siarry (2011), *Optimisation multiobjectif: Algorithmes*, Editions Eyrolles.

Dijkstra, E. W. (1959), A note on two problems in connexion with graphs, *Numerische Mathematik*, *1*(1), 269–271, doi:10.1007/BF01386390.

Dosi, G. (1982), Technological paradigms and technological trajectories. A suggested interpretation of the determinants and directions of technical change, *Research Policy*, *11*(3), 147–162, doi:10.1016/0048-7333(82)90016-6.

Gen, M., R. Cheng, and L. Lin (2008), *Network Models and Optimization*, 1–41 pp., Springer.

Gillespie, J. W. (2012), A Network Science Approach to Understanding and Generating Ship Arrangements in Early-Stage Design, Ph.D. thesis, University of Michigan.

Goos, G., et al. (2007), Evolutionary Multi-Criterion Optimization, in *Evolutionary Multi-Criterion Optimization*, Matsushima, Japan.

Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems: An introductory analysis with appication to biology, control, and artificial intelligence*, University of Michigan Press.

Holland, J. H. (1992), Genetic Algorithms, *Scientific American*, *267*(1), 66–72, doi:10.1038/scientificamerican0792-66.

Kassel, B., S. Cooper, and A. Mackenna (2010), Rebuilding the NAVSEA Early Stage Ship Design Environment, in *American Society of Naval Engineers Conference*.

Leveson, N. (2004), A new accident model for engineering safer systems, *Safety Science*, *42*(4), 237–270, doi:10.1016/S0925-7535(03)00047-X.

Li, M., S. Yang, S. Member, and X. Liu (2014), Diversity Comparison of Pareto Front Approximations in Many-Objective Optimization, *IEEE Transactions on Cybernetics*, *44*(12), 2568–2584.

Li, M., S. Yang, and X. Li (2015), A Performance Comparison Indicator for Pareto Front Approximations in Many-Objective Optimization, in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 703–710, doi: 10.1145/2739480.2754687.

Mackenna, A. (2011), Rapid Ship Design Environment, in *NDIA Conference on Physics-Based Modeling for US Defense*.

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

Naval Sea Systems Command (2012), Ship Design Manager (SDM) and Systems Integration Manager (SIM) Manual, *Tech. Rep. February*, Naval Sea Systems Command.

Newman, M. E. J. (2003), The structure and function of complex networks, *SIAM Review*, *45*(2), 167–256, doi:10.1137/S003614450342480.

Newman, M. E. J. (2010), *Networks: An Introduction*, Oxford University Press, Oxford.

Page, S. E. (2006), Path dependence, *Quarterly Journal of Political Science*, *1*, 87–115, doi:10.4324/9780203882313.

Papalambros, P. Y., and D. J. Wilde (2000), *Principles of Optimal Design: Modeling and Computation*, second ed., Cambridge University Press, New York.

Parker, M. C. (2014), A Contextual Multipartite Network Approach to Comprehending the Structure of Naval Design, Ph.D. thesis, University of Michigan.

Sargent, R. G. (2011), Verification and Validation of Simulation Models, *Proceedings of the 2011 Winter Simulation Conference*, pp. 183–198.

Savoie, T. B., and D. D. Frey (2012), Detecting mistakes in engineering models: the effects of experimental design, *Research in Engineering Design*, *23*, 155–175, doi:10.1007/s00163-011-0120-y.

Sen, P., and J.-B. Yang (1998), Multiple Objective Decision Making, in *Multiple Criteria Decision Support in Engineering Design*, chap. 4.4.1, pp. 150–157, Springer.

Shields, C. P. F. (2017), Investigating Emergent Design Failures Using a Knowledge-Action-Decision Framework, Ph.D. thesis, University of Michigan.

Van Veldhuizen, D. A. (1999), Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, Ph.D. thesis, Air Force Institute of Technology.

Wang, S., S. Ali, T. Yue, Y. Li, and M. Liaaen (2016), A Practical Guide to Select Quality Indicators for Assessing Pareto-Based Search Algorithms in Search- Based Software Engineering, in *IEEE International Conference on Software Engineering*, pp. 631–642.

Wu, J., and S. Azarm (2001), Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set, *Journal of Mechanical Design*, *123*(1), 18, doi: 10.1115/1.1329875.

Yang, J.-B., and P. Sen (1996), Interactive trade-off analysis and preference modeling for preliminary multiobjective ship design, *Systems Analysis, Modelling and Simulation*, *26*, 25–55.

Yang, J.-B., C. Chen, and Z. J. Zhang (1990), The Interactive Step Trade-Off Method (ISTM) for Multiobjective Optimization, *IEEE Transactions on Systems, Man and Cybernetics*, *20*(3), 688–695, doi:10.1109/21.57283.

Zitzler, E. (1999), Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Ph.D. thesis, Swiss Federal Institute of Technology Zurich, doi:citeulike-article-id:4597043.

Zitzler, E., and L. Thiele (1998), Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study, *Proceedings of the International Conference on Parallel Problem Solving from Nature*, *1*(September), 292–304, doi: 10.1007/BFb0056872.

Zitzler, E., K. Deb, and L. Thiele (2000), Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, *Evolutionary Computation*, *8*(2), 173–195.

Zitzler, E., L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca (2003), Performance assessment of multiobjective optimizers: An analysis and review, *IEEE Transactions on Evolutionary Computation*, *7*(2), 117–132, doi: 10.1109/TEVC.2003.810758.