GENERATIVE BAYESIAN NETWORKS FOR CONCEPTUAL AIRCRAFT DESIGN

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
AERONAUTICAL AND ASTRONAUTICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Emilio M. Botero
December 2019

This dissertation is online at: http://purl.stanford.edu/kh799xv6529

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Juan Alonso, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mykel Kochenderfer

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ilan Kroo

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Conceptual aircraft design, with its basis in engineering design, is an open-ended problem in which human designers synthesize solutions to meet a set of requirements. This open-ended problem inherently necessitates an iteration process combining creation and testing. Designers approach these problems with their intuition, experience, and knowledge of physics to achieve this iteration process. However, future aircraft design requirements break away from traditional norms, leaving the seasoned human designer with little intuition.

This work devises an architecture by taking cues from how human designers lead design processes. This architecture also becomes an iterative design loop with creation and testing. A probabilistic model, the generative Bayesian Network, suggests candidate designs for the architecture. The generative Bayesian Network decomposes the problem into physical components and finds the interactions between them, as a systems design. The generative Bayesian Network learns examples from a combination of historical data and generations analyzed with SUAVE, an aircraft conceptual design tool. SUAVE is especially suited for this process since it possesses the ability to analyze and optimize numerous types of aircraft. This architecture enables a human designer to consider concepts and configurations they might have otherwise not considered.

Four test cases examine the effectiveness of this architecture that combines SUAVE with a generative Bayesian Network. First, a simple beam design illustrates how multiple valid solutions may be found. Next, are two cases, that design with differing levels of complexity to compare against an existing medium-range airliner. Finally, an ultra-long-range aircraft is designed with mission requirements unprecedented today. The architecture returns feasible design concepts prepared for more in-depth design studies and development.

Acknowledgments

One cannot create a Ph.D. thesis in a bubble. Countless folks and organizations contributed to my success, and this very work. I want to thank everyone that has led me to and through Stanford to complete this dissertation. Although I cannot name everyone, I'd like to point out a few.

I've been extremely fortunate to be funded through a series of fellowships throughout my time at Stanford. The first being through Stanford Engineering, which funded my Master's degree. The next fellowship was the Department of Defense's National Defense Science & Engineering Graduate Fellowship (NDSEG). Finally, there was Diversifying Academia Recruiting Excellence (DARE) from Stanford VPGE. DARE has been excellent, and as they say so much more than just a funding source.

My committee is filled with superstars. My oral committee chair, Michael Saunders, created SNOPT, which I used countless times over the past 6 years. Bob Ormiston, who served as an oral committee member, has a wealth of knowledge in VTOL design and is always fun to fly RC planes with. It's been a pleasure working with Michael Kochenderfer, the POMDP master, in both a NASA Learn project and, more recently, with this thesis work. Professor Ilan Kroo is the legend of aircraft design and aerodynamics. Finally, my advisor Juan Alonso, possibly the most well-spoken individual in the world, who has provided me with fantastic research freedom.

Alonso's Aerospace Design Lab has felt like home to me since my first year as a Master's student. We've been way more than just an office: we grew plants, watched (and played) soccer together, and drank so much cafecito hc. I'm indebted to Trent Lukaczyk for guiding me into the lab, mentoring me, and then handing the reigns of SUAVE over to me. The entire SUAVE team has been awesome; we've put together a great code with a small but dedicated team. I want to thank Matthew for his hard work with the VLM. I'd also like to thank Wally Maier and Jessie Lauzon in the ADL for editing this thesis. Someone, please, water my avocado tree.

I've been lucky to have many great friends at Stanford and beyond! Being an Off Campus Housing Community Associate was a blast, and I met many wonderful people. I've had the pleasure

of snowboarding with many great folks through the Stanford Ski Team, the Auburn Ski Club, as well as in competition against the fastest riders in the world. Women in Aero/Astro (WIAA) has been a group that I worked with extensively. I met many amazing people through WIAA, and I've seen first-hand the way the department and the university have changed for the better. We joke that every cohort of Aero/Astro students is totally different. My year was legendary, and I've been great friends with all of them. I want to thank Aditya, Ben, Sumant, Adrien, Flora, Monica, and many many more. I've also had such a great group of friends from Embry-Riddle Aeronautical University that I still keep in touch with every day. My ERAU friends were the ones who convinced me to study engineering and then helped me through this Ph.D. with their moral support.

Finally, I must thank my whole family, whether they be from Maine or Colombia. My parents, who nurtured both my love of snowboarding and airplanes. Despite all of us being so totally different, I have wonderful siblings. My grandmother and Ron have always been there for me. My grandfather in Colombia started the family's airplane obsession. Finally, I have to thank, of course, my soft-coated wheaten terrier!

Preface

A Stanford Aero/Astro Ph.D. is a long meandering process of exploration. After passing qualifying exams, we're given little and sometimes conflicting advice on how to proceed forward. This environment is one where we learn by experience. In the end, we are expected to produce a seemingly cohesive narrative which we dub a thesis.

The purpose of a thesis is to demonstrate to the world new knowledge and summarize our work as a Ph.D. candidate. This small contribution encompasses efforts over the past few years. Yet, a short presentation and a final draft do not and cannot represent these efforts. These Ph.D. years included many research topics outside of the topics introduced here. For example, there is an entire NASA project that is irrelevant to this work, almost a forgotten year. In truth, this work only sums up about two years of my efforts. Despite that, this thesis built on the groundwork started many years before in SUAVE. For that, I am thankful.

The real intention of this thesis is to start something new. Indeed, this is not the first work on generative design, Bayesian Networks, conceptual aircraft design, or the like. However, I do believe that the future of engineering design will utilize these ideas. The hope is that this thesis helps one understand further and can build off this to design the future. That necessitates that this thesis is more than just words on paper to fill space. It must fully elucidate complex ideas to the reader without becoming tangential. If your thoughts after reading through are like this interaction from The Simpsons in which Homer Simpson became a newspaper food critic, then I have failed:

Editor: "This is a joke, right? I mean this is the stupidest thing I've ever read!"

Homer: "What's wrong with it?"

Editor: "... You make numerous threatening references to the UN and at the end you repeat the words 'Screw Flanders' over and over again."

Homer: "Oh, it's so hard to get to 500 words."

Contents

Abstract iv

Acknowledgments v

Preface vii

1 Introduction 1

1.1 The Engineering Design Process . . . 3

1.1.1 Design Thinking . . . 4

1.1.2 Design Phases . . . 5

1.2 The Aircraft Design Process . . . 6

1.2.1 Conceptual Aircraft Design . . . 7

1.3 Generative Aircraft Design . . . 9

1.4 Contributions . . . 12

1.5 Dissertation Outline . . . 13

2 Generative Bayesian Network 14

2.1 Introduction . . . 14

2.2 Bayesian Networks . . . 15

2.2.1 Hybrid Bayesian Networks . . . 16

2.3 Conditional Probability Distributions . . . 17

2.3.1 Discrete Nodes . . . 17

2.3.2 Continuous Nodes . . . 18

2.4 Component-Based Networks . . . 20

2.4.1 Concurrent Systems and Subsystems . . . 23

2.5 Learning . . . 24

2.5.1 CPD Learning . . . 24
2.5.1.1 Discrete . . . 24
2.5.1.2 Continuous . . . 26
2.5.2 Structure Learning . . . 27
2.5.2.1 Score . . . 28
2.5.2.2 Search . . . 29
2.6 Design Sampling . . . 32
2.6.1 Nodal Sampling . . . 32
2.6.2 Forward Sampling . . . 33
2.6.3 Amplification . . . 34
2.7 Conclusions . . . 35

3 Architecture **36**
3.1 Introduction . . . 36
3.2 Component-Based System Design . . . 37
3.3 Sizing, Analysis, and Pruning . . . 38
3.3.1 Sizing . . . 38
3.3.2 Analysis . . . 39
3.3.3 Pruning . . . 40
3.4 Design Space Exploration . . . 40
3.4.1 Sampling Exploration . . . 40
3.4.2 Amplification Exploration . . . 41
3.5 Overall Architecture . . . 41
3.5.1 Convergence . . . 46
3.6 MAP and MPE . . . 47
3.7 Conclusion . . . 49

4 SUAVE **50**
4.1 Introduction . . . 50
4.2 Architecture . . . 51
4.2.1 Mission Structure . . . 52
4.2.2 Energy Networks . . . 53
4.2.3 Optimization . . . 54
4.3 Capabilities . . . 55

4.3.1 Aerodynamics . . . 55
4.3.1.1 Vortex Lattice Method . . . 56
4.3.2 Weight Estimation . . . 58
4.3.2.1 Main Wing Weight Estimation . . . 58
4.3.3 Center of Gravity Estimation . . . 61
4.3.4 Stability . . . 62
4.4 The Future of SUAVE . . . 63
4.5 Conclusions . . . 63

5 Test Cases **65**
5.1 Introduction . . . 65
5.2 Beam Design . . . 66
5.2.1 Introduction . . . 66
5.2.2 Beam Problem . . . 66
5.2.3 Architecture . . . 69
5.2.3.1 Seed Designs . . . 70
5.2.4 Results . . . 71
5.2.4.1 Convergence . . . 71
5.2.4.2 Learned Network . . . 75
5.2.4.3 Sampled Results . . . 76
5.2.5 Conclusion . . . 77
5.3 Medium Range Airliner . . . 77
5.3.1 Introduction . . . 77
5.3.2 Design Requirements . . . 78
5.3.3 Seed Designs . . . 79
5.3.4 Analysis . . . 80
5.3.5 Setup . . . 82
5.3.6 Results . . . 84
5.3.6.1 Convergence . . . 84
5.3.6.2 Final Designs . . . 84
5.4 Wing Planform Design . . . 87
5.4.1 Introduction . . . 87
5.4.2 Setup . . . 87

5.4.3 Results . . . 90
5.4.3.1 Convergence . . . 90
5.4.3.2 Final Design . . . 90
5.5 Kangaroo Airliner . . . 93
5.5.1 Introduction . . . 93
5.5.2 Design Requirements . . . 94
5.5.3 Results . . . 94
5.5.3.1 Convergence . . . 94
5.5.3.2 Final Designs . . . 95
5.5.4 Further Studies . . . 97
5.5.4.1 Additional Configurations . . . 98
5.5.4.2 Static Stability . . . 100
5.6 Conclusion . . . 100

6 Conclusions and Future Work **103**
6.1 Conclusions . . . 103
6.2 Future work . . . 104

A Medium Range Airliner Convergence **106**
A.1 Input-Output Comparison . . . 106
A.2 Learned Graph Structure . . . 107

B Wing Planform Design Convergence **109**
B.1 MPE Samples . . . 109
B.2 Input-Output Comparison . . . 112
B.3 Learned Graph Structure . . . 117

C Kangaroo Airliner Convergence **118**
C.1 MPE Samples . . . 118
C.2 Input-Output Comparison . . . 122
C.3 Learned Graph Structure . . . 127

Bibliography **128**

List of Tables

3.1 CPD: Y given X 48
3.2 CPD: Z given Y 48
3.3 CPD: Z given Y, X=0 48

5.1 Beam Material Properties 69
5.2 Beam Architecture Setup 69
5.3 Seed Beams 70
5.4 Seed Beam Results 71
5.5 Beam Results 77
5.6 Medium-Range Airliner Requirements 78
5.7 Medium-Range Airliner Architecture Setup 83
5.8 Discrete Options 83
5.9 Medium-Range Airliner Specifications 86
5.10 Discrete Options for Wing Segments 89
5.11 Segmented Wing Medium-Range Airliner Requirements 89
5.12 Segmented Wing Medium-Range Airliner Architecture Setup 89
5.13 Segmented Wing Medium-Range Airliner Iterations 90
5.14 Segmented Wing Medium-Range Airliner Results 90
5.15 Segmented Wing Medium-Range Airliner Final Specifications 92
5.16 Kangaroo Airliner Requirements 94
5.17 Kangaroo Airliner Iterations 95
5.18 Kangaroo Airliner Specifications 95

List of Figures

1.1 V/STOL Wheel of Misfortune 2
1.2 Conceptual Design Sketch by Kelly Johnson in 1958 8
1.3 A Probabilistic Model for Component-Based Shape Synthesis 9
1.4 Case-Based Reasoning 10
1.5 Design Flow using Knowledge-Based Engineering 11

2.1 Example Bayesian Network 15
2.2 Dining Room Table 17
2.3 Discrete Node 18
2.4 Continuous Node 19
2.5 Component Node 20
2.6 Simply Connected Network 22
2.7 Learned Network 22
2.8 System of Systems 23
2.9 Edge Operations 30

3.1 Sampling Exploration 42
3.2 Amplification Exploration 43
3.3 Notional Design Flow 44
3.4 CPD 47

4.1 Example Mission Profile 52
4.2 Solar Network 54
4.3 Blended Wing Body Pressure Coefficient 56
4.4 B737 Class Aircraft with Control Points from Native Vortex Lattice Method 57

5.1 Illustration of a Simply Supported Beam 67
5.2 Illustration of a Cantilevered Beam 67
5.3 Initial Beams 72
5.4 Beam Case: Iteration 1 72
5.5 Beam Case: Iteration 2 72
5.6 Beam Case: Iteration 3 72
5.7 Beam Case: Iteration 4 72
5.8 Beam Case: Iteration 5 72
5.9 Beam Case: Iteration 6 73
5.10 Beam Case: Iteration 7 73
5.11 Beam Case: Iteration 8 73
5.12 Beam Case: Iteration 9 73
5.13 Beam Case: Iteration 10 73
5.14 Beam Case: Iteration 11 73
5.15 Beam Case: Iteration 12 74
5.16 Beam Case: Iteration 13 74
5.17 Beam Case: Iteration 14 74
5.18 Beam Case: Iteration 15 74
5.19 Initial Beam Network 75
5.20 Learned Beam Network 76
5.21 Bombardier/Airbus A220 79
5.22 All Seed Vehicles with Trapezoidal Wings 80
5.27 Medium-Range Airliner 85
5.28 All Seed Vehicles with Segmented Wings 88
5.33 Medium-Range Airliner with Segmented Wings 91
5.38 Kangaroo Airliner MPE Sample 96
5.43 Kangaroo Airliner Amplification Alternative 97
5.48 Alternative Kangaroo Configurations 99
5.49 Longitudinally Unstable Aircraft Iteration 100
5.50 Next Iteration with four horizontal tails 101

A.1 Iteration 1: Final Iteration 107
A.2 Medium Range Airliner Learned Network 108

B.3 Iteration 2 . . . 110
B.6 Iteration 3 . . . 110
B.9 Iteration 4 . . . 111
B.12 Iteration 5 . . . 111
B.13 Iteration 1 . . . 112
B.14 Iteration 2 . . . 113
B.15 Iteration 3 . . . 114
B.16 Iteration 4 . . . 115
B.17 Iteration 5: Final Iteration . . . 116
B.18 Medium-Range Airliner with Segments Learned Network . . . 117

C.3 Iteration 0 . . . 119
C.6 Iteration 1 . . . 119
C.9 Iteration 2 . . . 120
C.12 Iteration 3 . . . 120
C.15 Iteration 4 . . . 121
C.18 Iteration 5: Final Iteration . . . 121
C.19 Iteration 1 . . . 122
C.20 Iteration 2 . . . 123
C.21 Iteration 3 . . . 124
C.22 Iteration 4 . . . 125
C.23 Iteration 5: Final Iteration . . . 126
C.24 Kangaroo Airliner Learned Network . . . 127

Chapter 1

Introduction

If it looks right, it'll fly right.

Old Aviation Saying

Spurred by new technologies and business models, new aircraft design concepts are evolving at a pace unseen in decades. This evolution in design is particularly evident for those attempting to solve urban air mobility challenges [1–3]. The number of aircraft design concepts for urban air mobility aircraft was little more than single digits up until 2017. At the time of this writing, eVTOL News has tracked the number of concepts to over 200 different designs [4]. When browsing the designs proposed, one may find a vast array of configurations.

Each design has a flavor, or perhaps gimmick, which the designer believes provides an advantage. While themes exist, under further examination they are nothing more than electrified versions of past designs. The original Wheel of V/STOL Misfortune, shown in Figure 1.1, categorizes V/STOL aircraft by configuration [5].

A designer may choose to reinvent the proverbial wheel or accept it. If designers accept it, as is proposed in this work, they can learn from the past and build for the future. Accepting the wheel is not to say that every concept has been devised. On the contrary, these design concepts guide us towards things that are likely to work and for which purposes. The author would argue that there are configurations yet to be discovered.

Any concept or configuration will have a specific set of trade-offs. This fact is especially apparent in the urban air mobility space. However, when the performance of different classifications of configurations are compared, they tend to cluster [4]. The pure multi-rotor based configurations are agile and predominately used for short-range missions, while as mission

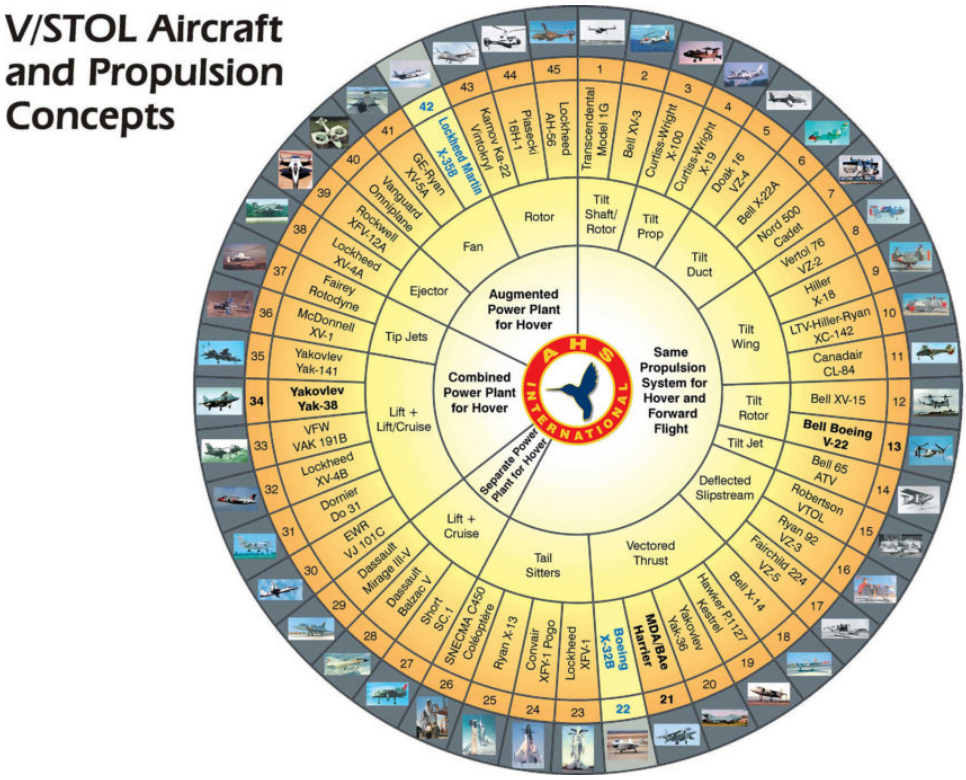


Figure 1.1: V/STOL Wheel of Misfortune [5]

range requirements become longer more wing-based configurations are used. There is not a hard dividing line between concepts here, however. That is where there is an opportunity: how does one produce a configuration and create a conceptual design?

This thesis proposes a framework to expand the configurations to be considered in the design process. If methodologies can provide an array of choices to the designer to select from, they may be supplied with ideas and concepts they may not have considered otherwise.

Historical aircraft and trends tend to aircraft design significantly. Many aircraft design books and guides are built off of correlative data from predecessor vehicles and experiments [6–10]. This thesis proposes a new way to use this data to make predictions. However, historical data is not the sole source.

Physics-based analysis is injected to fill the gaps in our knowledge. Not everything has been tried in real life, or there may not be data available; it's essential to provide input to fill the void. If the design process is only based on historic data, the human designer may be limited to designs like

those already in existence.

Even with data and physics, aircraft design is inherently an iterative process. By automating the search, this loop is closed faster. This automated search reduces the time to complete the design and allows for the recycling of previously generated results. Every data point is valuable, and when iterating through the design process, the work generated is saved. The time spent generating results where it was not as intended is not wasted time. While true for a human designer who naturally is always learning, this is even more relevant for a computer. These data points may be preserved for the same design process or even a future design case.

The results produced at the end of this framework are physically-feasible designs. They meet the customer requirements. However, it may not be optimal by any metric. The designer can then progress the design through optimization and compare the trade-offs between design objectives. Human designers can then iterate with the customer and suggest some candidate designs with trade-offs. Providing a menu of options may uncover the solution to the customer's real underlying problem.

1.1 The Engineering Design Process

Regardless of the end goal, engineering design processes, thinking, and approaches predominantly follow the same trends. Engineering design can be complicated, but at a high-level design follows a procedure composed of simple phases. These phases, outlined in Section 1.1.2, are grounded in some principles. These principles hold whether the product to design is an airplane, a toothbrush, a table, or even software.

The designer must navigate the "design space." The design space is a mental construct: It encompasses all potential solutions to the design problem [11]. For any given set of requirements, there could be zero viable solutions or potentially an infinite amount. The goal of the designer is to find one or more of these solutions and subsequently select one to construct.

Engineering design is a particularly difficult task because it is open-ended and ill-structured [11]. There exist multiple solutions to the same problem in the design space. Take an example: For eating utensils, you may use a fork, a spoon, chopsticks, or a knife and many variations within. There exists no simple mathematical formula to follow to get to the best solutions. Furthermore, what is the "best" solution? Is it the optimal solution? But what is considered optimal? Lowest cost? Lowest weight? Strongest? Safest? Each one of these objective functions has a different set of trade-offs. There is no right answer.

However, there are many wrong answers. These can be obvious, designs that do not meet needs. For our previous utensil example, using a butter knife on a steak just is not going to cut it. Over-designed answers are wrong answers, too [11]. In our steak example, we could guillotine it, but that would be overkill and hence over-designed. Once the requirements are met, the marginal benefit of excess performance decreases. This concept is known as saturation. Ultimately, a customer may not be willing to accept an over-performing design if it costs significantly more.

With engineering design, the goal is to search for viable solutions to meet a need. With the open-ended nature of these problems, the onus is on the designer to choose the "right" answer. The processes that guide the designer are based on the ideas of 'design thinking.'

1.1.1 Design Thinking

With the unbounded nature of design, it may seem to the untrained eye as a freeform and almost random process. Although creativity is a necessity, it is, in fact, a rigorous but rewarding process. At the root of design is a way of methodically approaching a design problem, *design thinking*. Design thinking "integrates human, business, and technological factors in problem forming, -solving, and -design." [12]. Design thinking is widely applicable to solving engineering problems and even outside engineering [12, 13].

Design thinking requires two separate sets of diverging and converging processes. The first set of processes is in observing and synthesizing, to create empathy with the users. The second set of processes is in creating a solution. The diverging states expand the solution space: a search for possibilities. The converging parts are a down-selection where some solutions are excluded as unviable. It is critical throughout the processes, however, that the designer not self-censor, that is, hold back.

Self-censoring reduces designer creativity as they would ignore a potential solution because of perceived drawbacks. Many of the diverging processes help in disinhibiting the designer's creativity. These diverging processes include synectics, analogy, fantasy, empathy, and inversion [11, 14, 15]. These diverging processes are generally viewed under the guise of brainstorming.

Essential to design processes is iteration [11, 12, 14, 15]. Iteration is all-encompassing and manifests itself within the brainstorming processes, within the design phases, between the design phases, within the designer's life experiences, and more. Iteration never ends. As an example, in studies, those who have performed the egg-drop exercise once before are far more successful the second time having learned lessons. If given multiple attempts and chances to iterate, the participants perform better [12]. Therefore, the designer is always learning and improving.

Design thinking encompasses several stages within an iteration, beginning with empathy, define, ideate, prototype, test, and assess [16]. These periods exist at both the highest levels and the smallest levels. They permeate the processes, whether realized or not by the designer. These live within the phases of design.

1.1.2 Design Phases

There is a typical set of engineering design phases. Several different sources have different naming conventions [11, 12, 14, 15], but they follow similar flows. Design starts with specifications, moves to conceptual design, preliminary design, and finally, detail design.

The first phase, specifications, involves translating customer needs, or tasks to be performed [15], into a set of requirements. This translation is known as a "problem definition" and encompasses objectives, constraints, and performance targets. These metrics place a value on particular design features to ensure compliance with the needs [11]. These can be numerical in nature or more abstract, such as aesthetics [11]. To create these specifications, it may require iteration with the client, i.e., the market, to elaborate and clarify [11, 15].

The next phase is conceptual design. Conceptual design takes the requirements identified in the prior phase and creates a design scheme with a set of analyzed specifications. This synthesis stage identifies potential problems, searches for solution principles, creates concept variants, and then compares them [15].

Preliminary design refines the concept created in conceptual design [11, 15]. The design layouts are created in more detail [15]. Furthermore, the design may be optimized to meet a particular objective. As the designer examines the concept further, they scrutinize the analysis results. Fortunately, at this stage, the designers are still able to make rather substantial changes.

Detail design is the final main phase of design. This phase ends with the definitive layout [15] as manufacturing must commence after this stage. Drawings, schematics, and the like necessary for construction are the primary concern. At this stage, engineering communication, the real deliverable of the engineering team, is essential [11].

However, this is not the only way to view the design phases. Although similar on many levels, an additional phase may be added. This phase is concept design [15]. Concept design is slightly different from conceptual design (it can also be viewed as a sub-phase). In concept design, the designer thinks abstractly about the physical principles that embody the design. The goal is to generate a proof of concept: evidence that the product obeys a physical law. At this phase, materials may be compared, and different geometry types are explored. Although these topics are generally explored

in conceptual design, there is a deeper fundamental abstract level. An example is comparing light generation from fire to electricity. This example is irrespective of the notion of a candle or a light bulb, just identifying the underlying physics. Physical principles are the key to concept design.

The efforts of this thesis are devoted to aircraft design but are generalizable to any engineering design. However, the current state of aircraft design has its nuances rooted in historical norms.

1.2 The Aircraft Design Process

In the early infancy of aviation, there was little designer intuition to build upon. When Otto Lilienthal created his first gliders, he used birds as a source of inspiration [17]. Combined with iteration (he produced over 18 different models), he eventually found designs that would perform as he expected. Although his life was tragically cut short after an accident, later pioneers of aviation built on his success. Octave Chanute learned from Lilienthal's successes, making ever-better gliders having benefited from the lessons Lilienthal found the hard way. Chanute then continued iterating. With further progress in designs, especially in the area of control, the Wright Brothers had solid understandings of aerodynamics and design by the time they attempted powered flight [17]. Current designs no longer solely draw inspiration from nature, but from the minds of physics and experimentation: iteration in progress.

Aircraft design is typically divided into three main phases [6, 8, 18, 19]. These are much the same as the general engineering design phases, as shown in the prior section: conceptual, preliminary, and detail design. There are other phases, such as market research and flight testing, but they do not require the same type of design processes. The traditional stages generally have formal reviews at the end before continuing into the next phase. At each stage, the aircraft is further derisked, meaning more of the specifics of the design have been completed and analyzed, providing confidence to the designer. This process is generally iterative, as increasingly detailed analysis models reveal drawbacks in prior attempts. Changes are continually being made throughout the process, and it is not uncommon to return to earlier stages of design when something does not perform as predicted. Throughout the design process, the costs, both computationally and monetarily, increases dramatically. Conceptual design can be completed by a small team with limited resources rather quickly. But detail design requires significant resources. At the academic level for students, there are several books on aircraft design that typically treat it as a cookbook-type process [6–8, 10, 19].

Conceptual design is the phase when significant choices are being made. From the overall requirements, the designer makes big-picture choices such as what type of powerplant

will be used, how many wings, the tail configuration, the landing gear configuration, and where the passengers are seated. Further, the vehicle is sized. At first, sizing means to select a wing area and thrust requirement but then goes into dimensioned layouts. Some sources perform sizing before the layout [7], while others decide the layout before sizing [6]. The key here is that a set of requirements drive the configuration making process. Market demands and regulations drive the requirements themselves. In conceptual design, it has been the designer's task to take these requirements and transform them into sketches.

After conceptual design, the vehicle undergoes further refinement. At this stage, many parameters are still fluid; the designer may choose to perform optimization to find the best possible shape. One of the end goals of preliminary design is an outer mold line, where the external geometry is fixed. Throughout preliminary design, additional analyses at higher levels of fidelity are introduced to supplement and revise the work done in conceptual design.

The final phase, detail design, is where the design is completed such that drawing and schematics can be sent to manufacturing to build an aircraft. There should be the utmost confidence in the engineer's mind how the vehicle will perform in flight.

1.2.1 Conceptual Aircraft Design

The work presented here pertains more directly to conceptual design; therefore, this section expands more into the conceptual design phase. However, the methodologies presented in this thesis can be used throughout the design process as they extend to all of design.

As mentioned prior, from a set of requirements, designers create sketches. These sketches are created through intuition and creativity in an attempt to satisfy the requirements. Figure 1.2 shows an example of an aircraft design sketch from 1958 [20]. Here, the legendary Kelly Johnson has sketched a supersonic airplane that has various performance parameters and sizing associated with it. His experience designing aircraft, combined with his knowledge and intuition of physics, lead to this sketch. It is this human touch that guides conceptual design to take specifications to concepts.

In the process of creating these sketches, there are many iterations as designers consider their options. The designer has two main ways of iterating; they may either batch out concepts to make comparisons or iterate directly on a particular concept. Both are necessary at various times.

Driving this iteration is analysis. After a design iteration is sketched, a human designer calculates how it performs aerodynamically and gains an idea of whether this concept may be feasible. There is also an inherent human aspect, as no analysis can be encompassing. An analysis, for aviation at least, can never be perfectly accurate and take into account every possible factor. However,

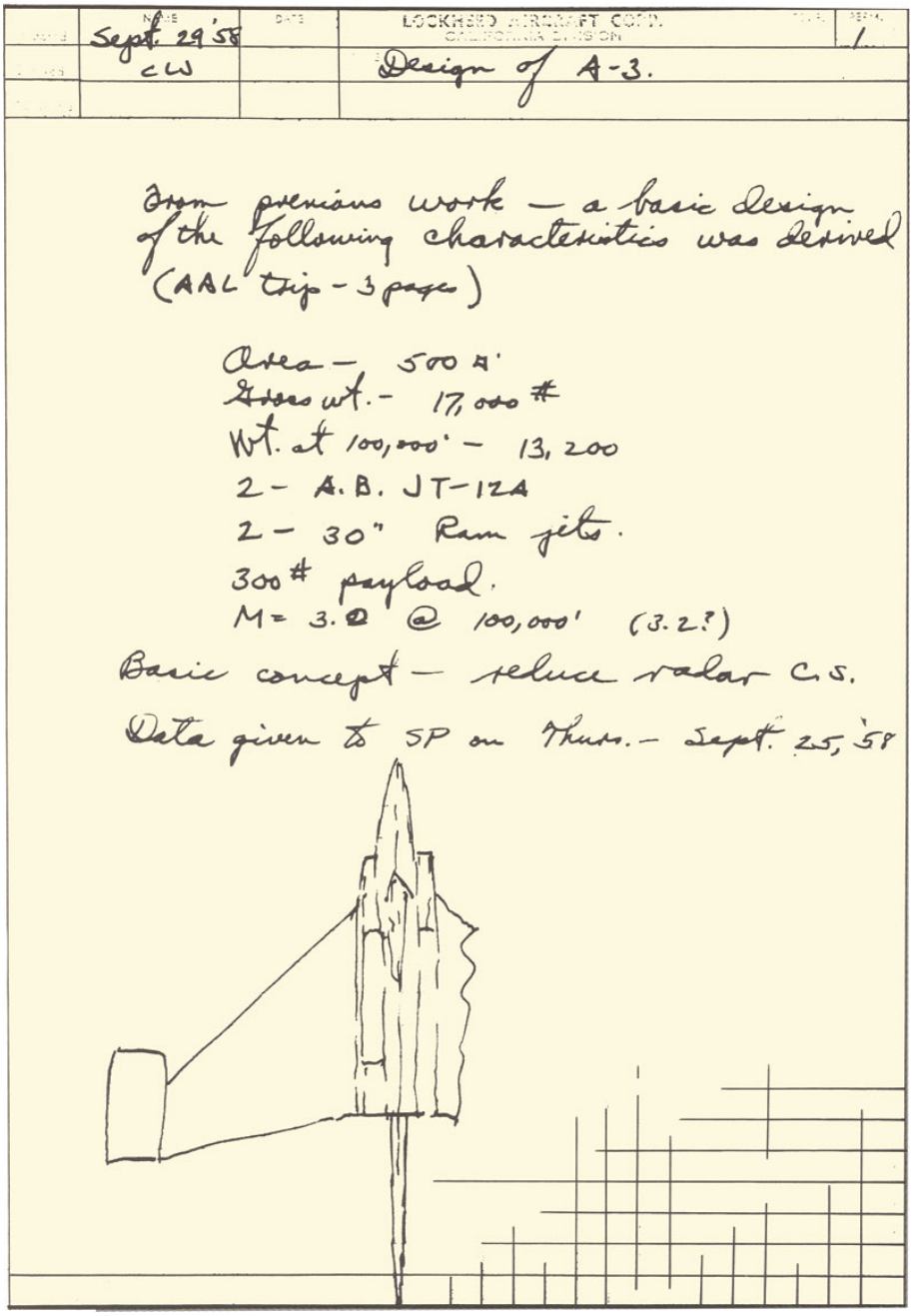


Figure 1.2: Conceptual Design Sketch by Kelly Johnson in 1958 [20]

the human eye is still incredibly useful in judging the practicality of a design. For example, certain aspects such as what makes a safe design are harder to encode in the analysis. It just may not be

entirely possible to let a computer design from start to finish... at least not yet.

1.3 Generative Aircraft Design

The new aircraft designs outlined at the outset of this chapter leverage new technologies that do not have extensive flight history. For designers, they cannot just draw on the past. The designers may not have full intuition for how to design appropriately given this new technology. What is envisioned here in this work is to change the aircraft conceptual design workflow fundamentally. This work aims to more quickly iterate, find better solutions, and do this with less labor. One day the exact algorithms used for this may not be the ones used in this work, but the overall ideas may be.

This work was motivated by the questions those who were designing eVTOL configurations were asking combined with algorithms. The algorithms specifically being Probabilistic Graphical Models [21]. The work by Kalogerakis et al. established a way to segment designs into components and used learned connections between them to create new ones [22]. These methods were demonstrated on construction vehicles, tables, chairs, and airplanes, as in Figure 1.3. Further extensions of these methods showed their efficacy for generating Super Mario Brothers levels [23]. In Figure 1.3, the green airplanes represent the initial database of designs while the blue aircraft are synthesized using a Bayesian Network. A more in-depth examination of this work is shown in Chapter 2.

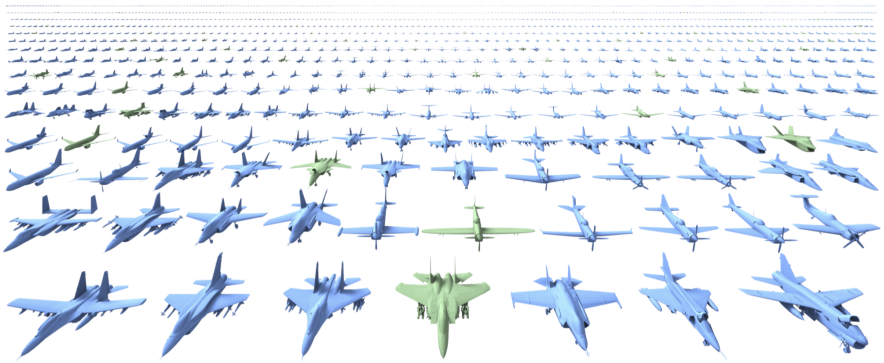


Figure 1.3: A Probabilistic Model for Component-Based Shape Synthesis [22]

The work proposed here is not the first to attempt to design aircraft configurations from the aerospace perspective automatically. There have been two original works that attempt to do similar

things but use fundamentally different approaches. The first is by Date Rentema, who uses Case-Based Reasoning (CBR) to suggest candidate designs [24]. The second is by Gianfranco La Rocca in his dissertation that uses Knowledge-Based Engineering (KBE) [25].

The work by Rentema on Case-Based Reasoning stores a set of aircraft designs with particular specifications. When called upon for new requirements, the CBR selects the closest example in the set of designs. Then CBR finds the vector distance from the desired results to the closest example. It then projects that vector into the solutions space to modify the base example. In essence, it is finding a similar vehicle and stretching it to meet the requirements. Figure 1.4 illustrates the CBR process graphically.

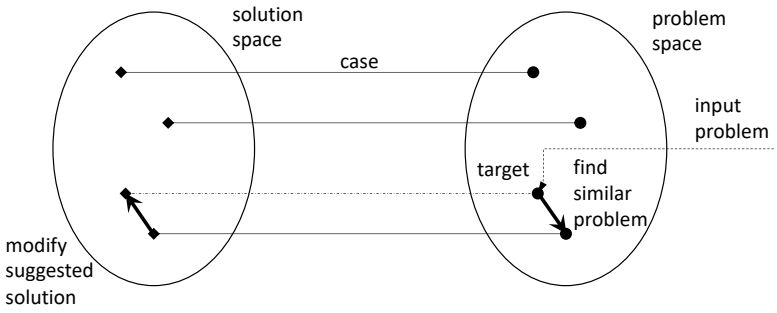


Figure 1.4: Case-Based Reasoning

La Rocca took a different approach to configuration design. KBE utilizes a set of programmatic rules that are embedded in a codebase [26–29]. KBE is useful because it means the results can be grounded in some physical rules. However, the rules include not only physics but also assumptions by the programmer, hindering the creativity of the results. The methods knowledge base does not directly use an aircraft database. Instead, the aircraft database is used to validate the knowledge base. The overall design flow in which La Rocca used is shown in Figure 1.5. This flow, shown in Figure 1.5, is useful in that it creates an iterative loop and set of analyses to ensure that the design solution meets the requirements based on analysis.

Another work worthy of mention is that of Peter Gage's thesis [30]. Gage used a variable complexity genetic algorithm (VCGA) [31]. This VCGA allows the genetic algorithm to expand its scope, basically extending the encoding length. One example Gage began with was a simple wing and then allowed it to change the number of segments. The VCGA found a better solution across test cases than the gradient-based optimizer. This work illustrated how difficult it is to find a global optimum in aircraft design using gradient-based optimization. Although there was a discussion of

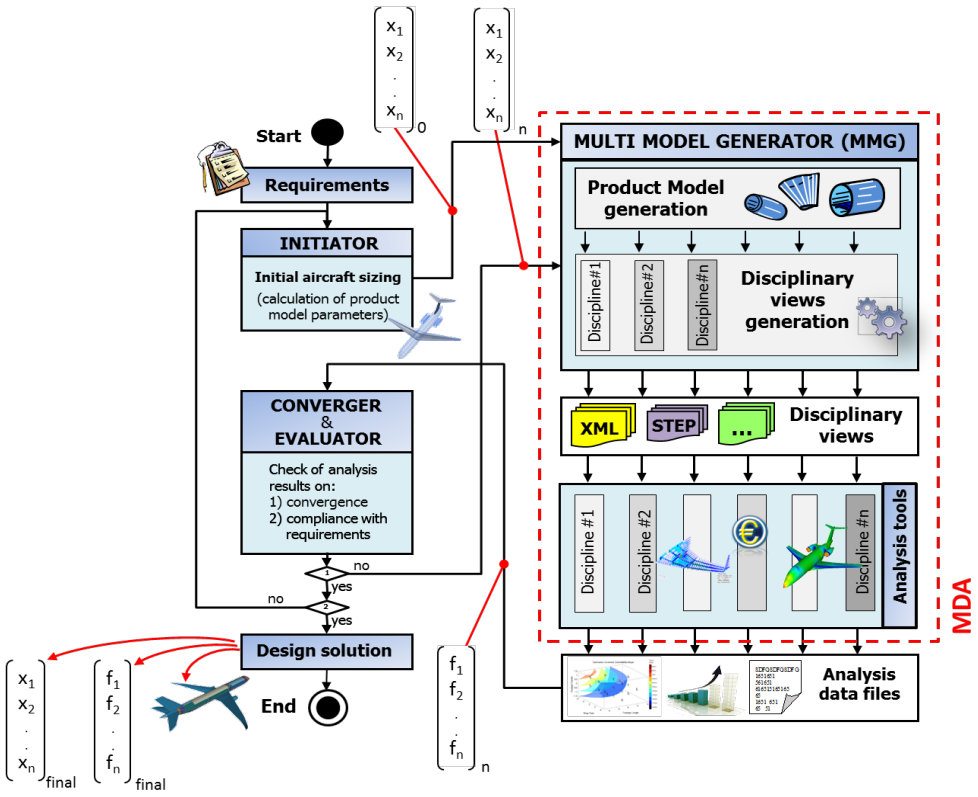


Figure 1.5: Design Flow using Knowledge-Based Engineering (Reproduced with Permission) [29]

the applicability to a full aircraft design, no real efforts were made to demonstrate the usage.

1.4 Contributions

There are three main contributions in this thesis. These are enumerated below and further expanded upon:

- Developed numerous modules integral to SUAVE for aircraft design that extends its ability to analyze, optimize, and now design;
- Created methodologies based in Bayesian Networks for the generation and design of engineering systems by incorporating physics-based analyses with historic seed data;
- Demonstrated the use of these methodologies to show applicability towards aircraft conceptual design with multiple use cases.

The first contribution, concerning SUAVE, is far-reaching in the codebase. Many years of development efforts went into making SUAVE, an aircraft design environment [32]. The author has thousands of commits to the SUAVE repositories to grow it to where it is today for the world to use. The ability to analyze such a wide range of vehicles in one single analysis tool is unprecedented. This comes from the flexibility of the code to be adapted and built upon and, most importantly, the drive towards physics-based analyses while removing assumptions when possible. Some of the unique specific contributions to SUAVE are a flexible *Data* class, weights models, post-stall aerodynamics models, battery models, UAV specific models, propeller models, the mission architecture, and importantly the optimization frameworks. The list of new features and methods continues on and on.

The second contribution is the methods developed for generative Bayesian Networks. The generative Bayesian Network itself is component-based and specifically implemented for design. This network is a hybrid network, which decomposes components into discrete decisions and continuous decisions. The network is also able to learn from limited data by using surrogates inside the models to make predictions. This network is then wrapped into a design architecture that combines SUAVE's physics-based analysis to close the design loop and design feasible systems.

Combining the generative Bayesian Network methodologies with SUAVE created something powerful. To demonstrate this power, a series of use-cases that are each more difficult than the last is presented. These use-cases present designs applicable to current real-world mission requirements to design conceptual level aircraft. These designs demonstrate practical, realizable aircraft. The results illustrate the ability to both synthesize configurations a designer may not have conceived of while doing so very quickly.

1.5 Dissertation Outline

Chapter 2 discusses our method of generating designs, the Generative Bayesian Network. With the theory developed for the Generative Bayesian Network, a design loop is created in Chapter 3. However, without robust analyses, the designs generated are of no use to compare; therefore, Chapter 4 provides background information to SUAVE, an open-source aircraft design environment. Then several test cases are shown in Chapter 5, including a model problem and full aircraft design examples. Finally, there are conclusions and thoughts for the future in Chapter 6.

Chapter 2

Generative Bayesian Networks

When something exceeds your ability to understand how it works, it sort of becomes magical.

Sir Jony Ive

2.1 Introduction

This chapter provides the background and derivation of a generative Bayesian Network. This work is not intended to provide a complete explanation into the field of Probabilistic Graphical Models, or even Bayesian Networks [21]. Bayesian Networks are explained with enough information to provide the reader with a complete understanding of the work contained in this thesis.

Bayesian Networks were selected for a variety of reasons related to design. First is the intuition that can be gathered and encoded in a Bayesian Network. Next is the relationships that can be established in a Bayesian Network through edges. In systems design, often a decision for one system or component must be made before the next [33], requiring these edges. The third is the simplicity and computational efficiency relative to Markov Random Fields, which generally require some type of convergence of the network due to the cyclic nature. This combination of intuition, directed nature, and simplicity allows us to experiment and use Bayesian Networks in the early stages of engineering conceptual design with rapid iteration.

Bayesian Networks are used to simulate the kinds of decisions that take place in engineering systems designs. One insight from design is that engineering systems are fundamentally composed of components whose interactions are designed to meet a higher-level set of requirements. These interactions are encoded between components in the Bayesian Network as edges that provide information to downstream components. The downstream components use this information to select how

many to have and then design them. This is accomplished by learning from examples of successful engineering designs to predict future design configurations.

Bayesian Networks, as described here, have the advantage of being able to make accurate predictions from limited data. This is due to several reasons. First, human designer input in nodal selection limits the domain to a reasonable size. Second, with Bayesian Networks, the designer may tailor the network to encode priors into the distribution to augment the limited data with designer intuition. Finally, the network knows how well it fits the distribution based on the limited dat,a as is explained in structure learning in Section 2.5.2. Engineering designs may have very few existing data points to learn from, yet there is enough information for a seasoned expert to provide intuitive input from example. This work aims to provide this same kind of learned intuition into a rigorous statistics-based process.

Within this chapter, Bayesian Networks are examined in Section 2.2. The Conditional Probability Distributions associated with Hybrid Networks are described in Section 2.3. Next, one of the crucial contributions of this work is explained, the Component-Based Network in Section 2.4. Then, moving to how the parameters of these networks are learned in Section 2.5. There is a discussion on generating configurations via sampling and amplification in Section 2.6 before concluding in Section 2.7.

2.2 Bayesian Networks

In classical studies of Probabilistic Graphical Model, and thus, Bayesian Networks, multinomial or binomial data types are typical introduced first because they can be represented by simple probability distributions. Bayesian networks arise when there are multiple layers of conditional probability distributions. Conditional probability distributions are fully described in Section 2.3. Each conditional probability distribution can be viewed as a process where the ancestor node has some observed, or presumed, state variable that affects the current state variable. Let us examine a basic example; take Figure 2.1.

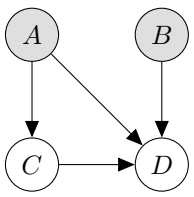


Figure 2.1: Example Bayesian Network

In this example there are two observed states variables, A and B. From A, a conditional probability distribution is used to estimate the probabilities of a given state variable for C. Given the knowledge from A, B, and C one can then evaluate the conditional probability distributions for D. The edges indicate the direction of information travel. For an extension to a real-world example, imagine if A and B are performance characteristics of a vehicle, say speed and range, while C and D are components, such as wing design and fuselage design. Knowing the speed and range of the vehicle, one can make predictions on the wing or fuselage. At a high-level, the Bayesian Networks discussed here use these concepts to synthesize.

2.2.1 Hybrid Bayesian Networks

To fully capture the probabilities associated with engineering systems design, the methodologies must handle complex data types. A hybrid network is used to accommodate these complex data types. A hybrid network is composed of mixed discrete and continuous decision processes [21, 34]. In real-world engineering design problems, human designers are often confronted with a variety of these types of decisions.

Introduced here is a basic type of engineered design as an example. This design is a table; which serves as a simple canonical test case in other works [22]. The author's table is seen in Figure 2.2. A table is composed of two main component categories: tops and legs. Discrete decisions are made as to the number of legs and type of tabletop. Continuous decisions are made for dimensions of the table, spanning in scale from a side table to a dinner table and beyond. This is an excellent example of a hybrid network; containing both discrete and continuous parameters in the same space.

A hybrid network necessitates that both discrete and continuous inputs are appropriately handled. In some cases, discrete inputs must create continuous outputs, vice-versa, and any combination thereof. Furthermore, discretization of a continuous space can become quickly intractable by creating large data requirements and the loss of information associated with discretization [21]. Therefore, in this work, continuous outputs are addressed with a modified Conditionally Linear Gaussian (CLG), as well as discrete decisions via the use of logistic regression with a Softmax output.

Figure 2.2: Dining Room Table

2.3 Conditional Probability Distributions

2.3.1 Discrete Nodes

Discrete decisions must handle cases with both discrete and continuous parents. Logistic regression is used to handle discrete decisions. Many of the methods are found in [21, 22]. The values of the discrete and continuous parents are combined into a single vector, $\boldsymbol{x}$. The kernel of the logistic regression function uses a linear fit as in

$$\ell_j(\boldsymbol{x}) = \beta_{j0} + \boldsymbol{\beta_j}^T \boldsymbol{x}. \tag{2.1}$$

The result of the kernel is fed into the Softmax output function in

$$p(Y = j|\boldsymbol{x}) = \sigma = \frac{\exp \ell_j(x_1, x_2, ..., x_n)}{\sum_{k=0}^{J} exp(\ell_k(x_1, x_2, ..., x_n))}. \tag{2.2}$$

The Softmax function has $N-1$, or J, possible discrete options. In this case, logistic regression is used to represent a multinomial from 0 to J. For example, which style tabletop could be a discrete decision, 0 could represent a rounded style while 1 could be a rectangle. The output of the function represents the probability of the j-th feature of the Softmax. In the table example, the probability of each tabletop style. The mechanics of the operations overall can be seen in Figure 2.3. Further information regarding the learning of the parameters and how logistic regression is used is elaborated upon in Section 2.5.1.1.

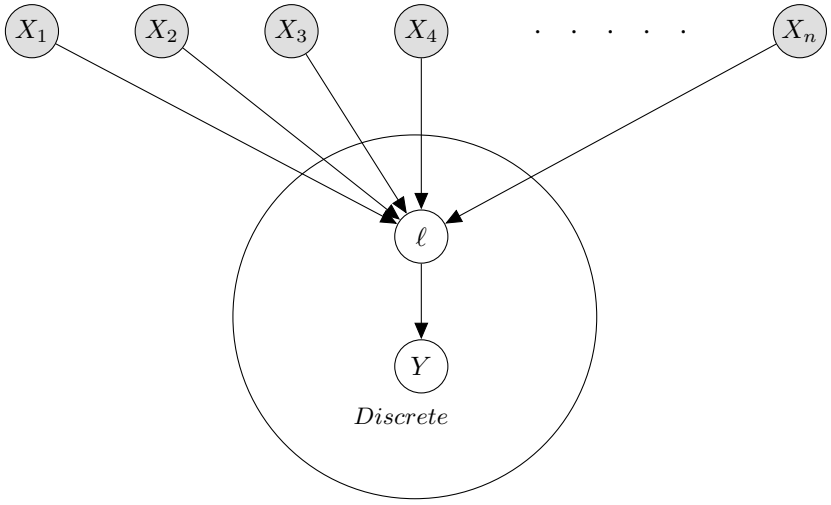


Figure 2.3: Discrete Node

2.3.2 Continuous Nodes

Continuous subnodes provide continuous decisions after making discrete choices. An example of a continuous decision could be the width of a dining table. Prior work uses Conditionally Linear Gaussians (CLG) for hybrid networks [21, 22, 35]. The formula for a simple CLG with a single binomial discrete parent

$$p(Y|\boldsymbol{C},\boldsymbol{D}) = \begin{cases} \mathcal{N}\big(\beta_0 + \boldsymbol{\beta_1}^T\boldsymbol{C}, \boldsymbol{\sigma_2}\big) & if D = 0 \\ \mathcal{N}\big(\beta_3 + \boldsymbol{\beta_4}^T\boldsymbol{C}, \boldsymbol{\sigma_5}\big) & if D = 1 \end{cases}. \tag{2.3}$$

CLGs are unlikely to be tractable in this context because of the extensive possibilities in engineering design; some combinations of discrete parameters are not in the observed design instances when data is limited. Therefore, it may not be possible to learn all weights for every discrete combinatorial

parameters.

This lack of data necessitates modification to the CLG. One of the first solutions tried was to treat the discrete inputs as if they were continuous in nature. This approach is intuitive as many discrete choices in engineering are multinomial choices, such as the number of rotors in a multirotor UAS. For this case, a simple Linear Gaussian Model is used,

$$p(Y|\boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{\beta} + \boldsymbol{B}^T\boldsymbol{x}, \boldsymbol{\Sigma}\big). \tag{2.4}$$

In the Linear Gaussian Model, $\boldsymbol{x}$ contains discrete and continuous parameters. This approach is similar to what is done with the logistic regression model in Section 2.3.1. Although the results are adequate, better approximations can be made.

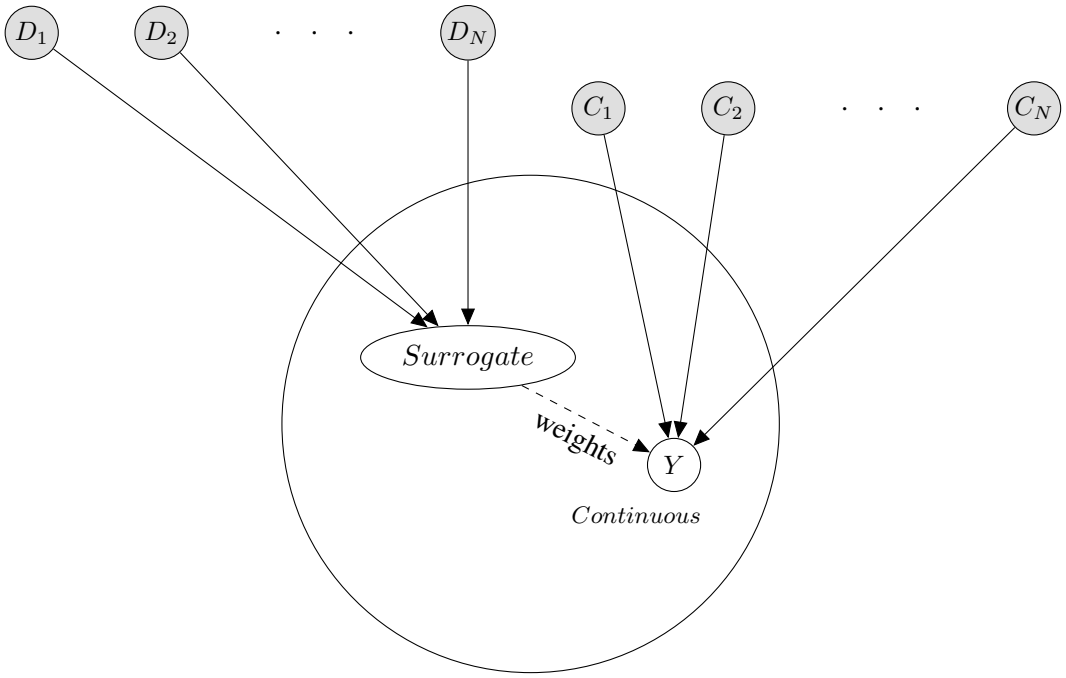


Figure 2.4: Continuous Node

Return to the original CLG, Equation 2.4, and observe that there are a set of weights, $\boldsymbol{B}$, as well as a covariance, $\boldsymbol{\Sigma}$, for each discrete combination. Next, the discrete inputs D are mapped through a surrogate model to the weights, $\boldsymbol{B}$, and the covariance, $\boldsymbol{\Sigma}$. Subsequently, a single Linear Gaussian with weights fit from a surrogate is produced,

$$p(Y|\boldsymbol{C}, \boldsymbol{D}) = \mathcal{N}\big(\boldsymbol{\beta_D} + \boldsymbol{B_D^T}\boldsymbol{C}, \boldsymbol{\Sigma}\big), \tag{2.5}$$

that only uses the continuous inputs inside the Gaussian kernel. Of particular note is that this methodology works with both univariate Gaussian distributions, as well as multivariate Gaussian distributions. The schematics of how the continuous element of a node work is in Figure 2.4.

Many options exist for a surrogate model, and in this case, a Gaussian Process Regression (GPR) is used to fit the weights vector and the covariance. A linear regression fit to estimate weights of weights was also attempted for comparison purposes. However, linear regression is a poor substitute as it is not guaranteed to be exact at known observed data points.

2.4 Component-Based Networks

One of the fundamental characteristics of engineering design is the observation that a design is composed of a system of components or decisions. To design a component itself requires decision making. The components interact with one another, forming a product intended to fulfill specific design requirements. This observation allows us to intuitively model an engineering system by grouping the design decisions, both continuous and discrete, of an entire design into components categories. For example, components for our table are the legs and tabletops. From the component categories, nodes are created. Each component class is represented with a node.

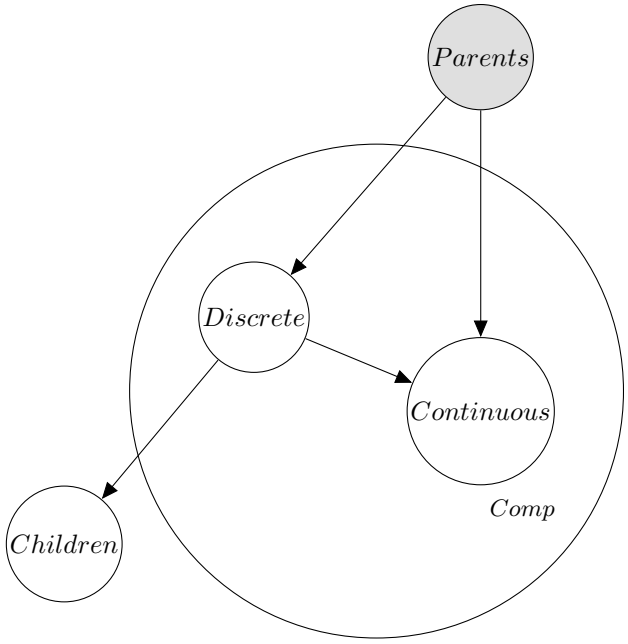


Figure 2.5: Component Node

A similar component breakdown was created in [22]. Another source provided a probability a design met requirements given component decisions [36]. However, the derivations here deviate considerably to allow for greater generalization and more logical flow. This work by Kalogerakis et al. relies on learned latent design categories made of discrete decisions [22]. When using latent design categories, the designs are grouped by topologies (all four-legged dining tables are separated from three-legged coffee tables). Although demonstrated with the use of aircraft, as well as other types of designs, the method by Kalogerakis et al. would not enable a broader range of topologies not observed prior. The goal of this thesis is to create new things. Furthermore, this restricts learning. For example, one may not want to segregate military aircraft from civilian aircraft. The underlying hypothesis is that learned information from military aircraft can help civilian designs and vice-versa. For this reason, the latent style is ignored, and the discrete interactions directly incorporated instead. This means that the component-based architectures must be general in this work.

A component node is described in Figure 2.5. This component node has observed parents, containing potentially both discrete and continuous data. Inside the component node is a discrete decision, modeled as in Section 2.3.2. The parents, as well as the discrete decision, feed into the continuous decision. This continuous decision is modeled as in Section 2.3.2. Notice that the discrete decisions are the only outputs of the component node. The reason for this is to ensure consistency in data sizes, between children component types, i.e., a helicopter does not have a wing aspect ratio, a continuous parameter, to transfer to children components.

Next, a simply connected network is assembled. The simply connected network is shown in Figure 2.6. This network works with the premise that each of the component nodes only parent are the design requirements. These requirements, chosen by the network architect, or human designer, drive the rest of the analysis. Usual choices for continuous requirements for aerospace designs could be things such as design cruise speed or range, for example. However, this does not limit us to only continuous requirements. One could add in discrete parameters as well. An example of a discrete parameter could be classifications of vehicle types, and in this way one could also group vehicles by observed category types if desired (a difference here is that if one were to group vehicles in this way, some data would be shared compared to the work by Kalogerakis which segregates classifications of data). One of the central aspects of the requirements is that it permeates all levels of the network; i.e., the requirements drive the network.

Once a simply connected network is created, then a more complex network may be learned. The learned network, as in Figure 2.7, has learned edges that connect various components. This learned

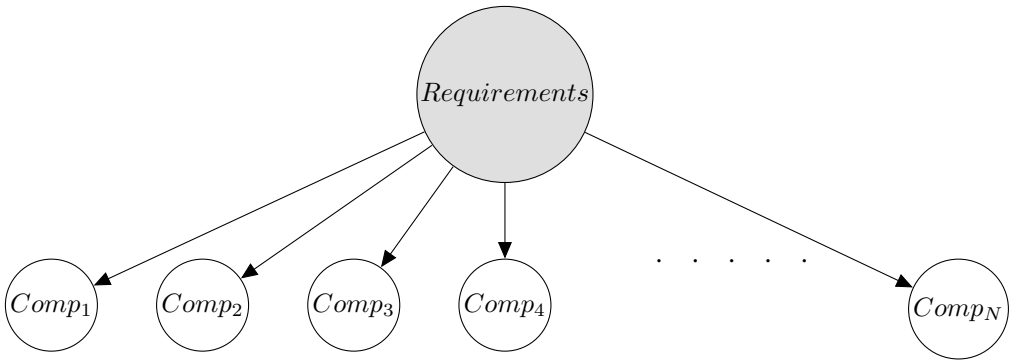


Figure 2.6: Simply Connected Network

network is created in a process described in Section 2.5.2. The edges allow us to statistically model the interactions between component designs. The edges for an aircraft could be, for example, from the wing to the horizontal tail. Most aircraft designers would agree that there is an interaction that the main wing affects the horizontal tail. The learned network must still result in a Bayesian Network: no loops may be induced. Furthermore, the fact that the requirements drive the network never changes. Even if the statistical link is weak, the requirements edge may not be removed to disconnect components. Consequently, unconnected component nodes are not allowed.

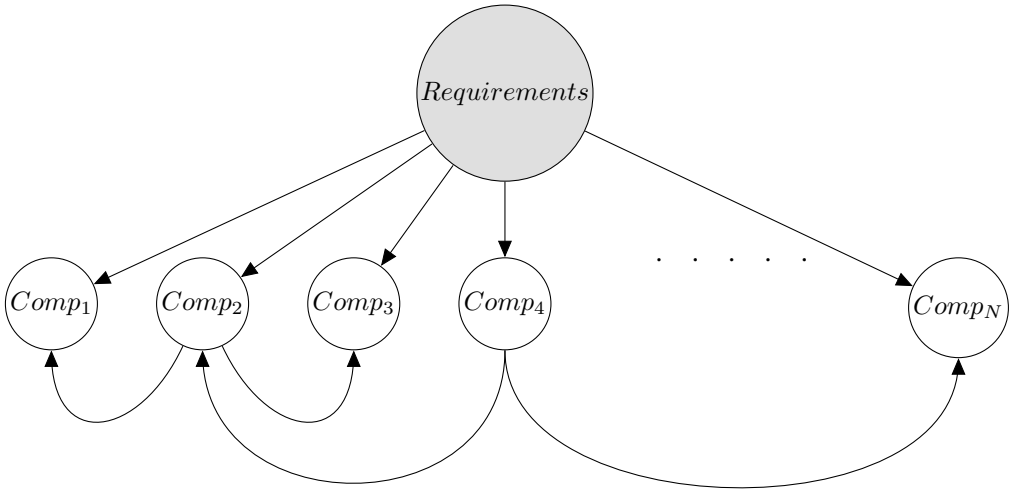


Figure 2.7: Learned Network

2.4.1 Concurrent Systems and Subsystems

Using the ideas for component-based networks, one can extend this to systems that are not just component-based, but the components themselves are composed of subcomponents. This regressive feature is the concept of a system of systems approach. In Chapter 5, subcomponents are used to allow for more complex wing shapes, as the wing becomes more than just a simple trapezoid.

This requires only a slight change to the network, with a similar line of thinking where edges from requirements to components were forced. Subcomponents are forced to be connected to their parent component. This parent edge is not allowed to be modified. Otherwise, the subsystem components behave as if they are full components in the network and can influence other systems and subsystems if further edges are learned.

One of the primary advantages of casting a component as a subcomponent is the ability to pass continuous decisions from the parent component into the subcomponent. This allows the network to make nuanced decisions based on the parent component.

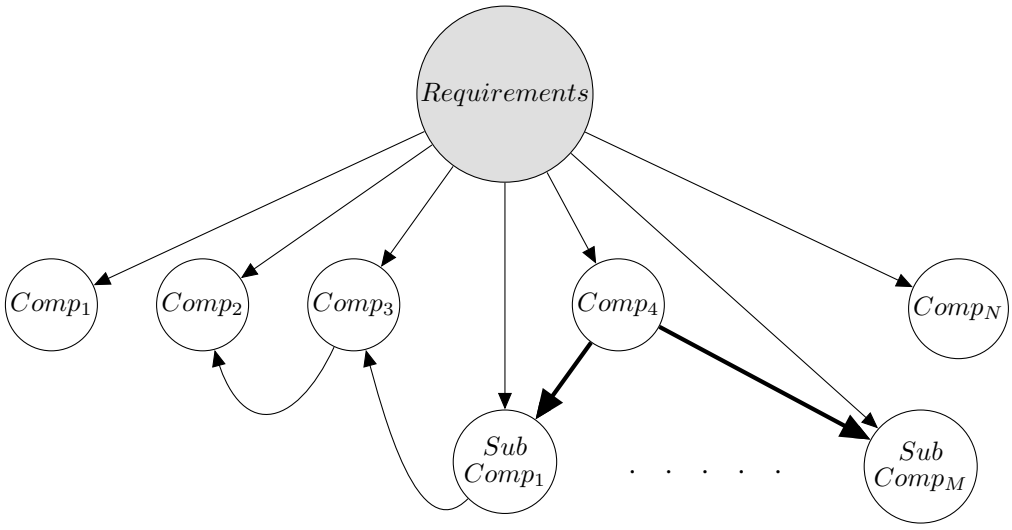


Figure 2.8: System of Systems

The ideas of system of systems are shown in Figure 2.8. Here, there are $Subcomp_1$ through $Subcomp_M$, specifically being child components of Component 4. Hence, they are a subsystem of Component 4.

2.5 Learning

With a simply connected network of human designer selected components complete, the learning may commence. Learning is done with several steps. The most basic is learning the Conditional Probability Distributions for each subnode, discrete and continuous, to encode the distributions. Next is how the edge structure of the network is learned. Learning a structure requires a score to compare different edge configurations as well as an algorithm to find the best structure to maximize this score. Learning requires a database of existing designs the human designer supplies. This database must segment into compatible components to be used.

2.5.1 CPD Learning

The Conditional Probability Distributions (CPD) described in Section 2.3 require that the weights are trained for both the continuous and discrete cases. Maximum Likelihood Estimation (MLE) is used by observing the seed data [21]. Training the CPD's is done by collecting all instances of observed design configurations and extracting the Parent data {C, D} as well as the nodal observation, (Y). An extension of the MLE equation, as in

$$J = \frac{1}{M} \sum_{m}^{M} x[m] \tag{2.6}$$

is used for both discrete and continuous subnode CPDs is used [21].

2.5.1.1 Discrete

For discrete nodes the Continuous Parents, C, are combined with the Discrete Parents, D, into a single vector, X. Once the observed data, X, as well as the observed nodal data, Y, are collected then the weights, $\boldsymbol{\beta}$, are fit using a gradient-based optimization process.

The Y observations are collected and represented by a set of "one-hot" style vectors. Each observed configuration has a "1" placed at the decision in the vector that corresponds with the design decision and 0 elsewhere, hence the name "one-hot." For example, a table with four legs would have a 1 placed at the fifth index in the array, assuming you can have legless chairs as an option as well. This "one-hot" vector can be visualized as follows

$$\begin{aligned} \text{Table Legs: } & [0, 1, 2, 3, 4, 5] \\ \text{One-Hot: } & [0, 0, 0, 0, 1, 0] \end{aligned} \tag{2.7}$$

.

In the process of researching software packages, including TensorFlow [37] and scikit-learn [38], for logistic regression calculation, it was found that these available codes were relatively slow given the number of times CPDs must be recalculated. As a consequence, a new codebase was created, and the following is the derivation of these methods.

To optimize the weights, they must first be initialized. Weights are uniformly distributed between -0.5 and 0.5; thus, they have a mean of zero. By placing the mean to zero, there is no initial bias. Furthermore, one cannot initialize all values to zero as it would have a symmetry property that is harder to break during optimization.

Then an optimization problem is formulated. For the accompanying code of this thesis, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is used for optimization, but other optimizers would work and have been used successfully. If BFGS fails, the code then attempts other optimizers with a new set of initial conditions until the optimization process converges. A loss function is selected to force the predicted probabilities, σ, to align with the MLE probabilities. The loss function chosen is the Mean Squared Error,

$$J = \frac{1}{M}\sum^{M}\big(Y-\sigma\big)^2. \tag{2.8}$$

Analytical gradients are calculated to get an exact and expedient calculation of the gradients for optimization. The partial derivatives of the loss function are taken with respect to the weights, $\boldsymbol{\beta}$. Then the differentials are broken down in as in

$$\frac{\partial J}{\partial \boldsymbol{\beta}} = \frac{\partial J}{\partial \sigma}\frac{\partial \sigma}{\partial \ell}\frac{\partial \ell}{\partial \boldsymbol{\beta}} \tag{2.9}$$

$$\frac{\partial J}{\partial \boldsymbol{\beta_0}} = \frac{\partial J}{\partial \sigma}\frac{\partial \sigma}{\partial \ell}\frac{\partial \ell}{\partial \boldsymbol{\beta_0}}. \tag{2.10}$$

One may easily differentiate these equations resulting in

$$\frac{\partial J}{\partial \sigma} = Y - \sigma \tag{2.11}$$

$$\frac{\partial \sigma}{\partial \ell} = \sigma(1 - \sigma) \tag{2.12}$$

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \frac{x}{M} \tag{2.13}$$

$$\frac{\partial \ell}{\partial \boldsymbol{\beta_0}} = \frac{1}{M}. \tag{2.14}$$

This formulation can also be modified slightly to, in a sense, get a prior distribution. For example, if the designer wishes to place some uncertainty on observed data that is derived from analysis rather than test data, the designer would want to "flatten" if it were the distribution to place more weight towards certain discrete configurations. This leverages the "one-hot" style to make it a "some-hot" vector. With a maximum value set by the human designer, say 95%, then 5% of the weights may be uniformly distributed to other combinations as well. The concept of "some-hot" is illustrated,

$$\begin{aligned} \text{Table Legs: } &[\quad 0, \quad 1, \quad 2, \quad 3, \quad 4, \quad 5] \\ \text{Some-Hot: } &[0.01, 0.01, 0.01, 0.01, 0.95, 0.01] \end{aligned}. \tag{2.15}$$

The intuition behind this comes from the observation that in engineering design, multiple designs can meet the same higher-level requirements. Under limited data circumstances, it can become necessary to acknowledge that an observed design is not the only possible solution. This process, in essence, creates a prior that uniformly weights other options as well to be non-zero and aid in later stages of design exploration.

2.5.1.2 Continuous

Continuous subnodes learning is presented here. Learning the parameters for continuous subnodes also uses the MLE strategies introduced in Equation 2.6. However, one must consider each observed discrete combination of continuous values separately to allow us to learn the surrogate. First, how each observed discrete combination is trained is discussed.

First, the weights, $\boldsymbol{B}$ and $\boldsymbol{\beta}$, are estimated. Estimation is more natural than the discrete case because of the linearity.

$$\hat{Y} = \boldsymbol{\beta} + \boldsymbol{B}^T \boldsymbol{C} \tag{2.16}$$

shows the predicted mean, given some continuous parent values. Least squares estimates are used for the weight terms due to the linearity,

$$\boldsymbol{B} = \arg\min_{B} \sum^{M} (Y_m - \hat{Y}_m)^2. \tag{2.17}$$

With the values of the weights determined, the covariance now requires attention. Because the continuous parent variables are the requirements of the design, it is assumed that these are perfectly observed. Thus, there is no uncertainty in the parents. Because of this, our uncertainties arise solely from inadequacies in modeling: they are epistemic in nature. The following equation,

$$\hat{e}_m = Y_m - \hat{Y}_m, \tag{2.18}$$

is used to calculate the error.

With this assumption, the covariance needed for the Gaussian can be calculated,

$$\Sigma = \frac{\hat{e}_m^T \hat{e}_m}{M}. \tag{2.19}$$

With each combination of observed parent discrete variables, a set of weights has been learned. Now, these weights can be fit into a surrogate process using Gaussian Process Regression (GPR). Many different surrogate types may be useful here, including but not limited to, Support Vector Machines regression (SVM) and K-Nearest Neighbor regression (KNN). For this work, scikit-learn's GPR functions using a Radial-Basis Function (RBF) kernel multiplied by a constant plus a constant are used. This is a standard kernel used for GPR.

2.5.2 Structure Learning

Structure refers to the edges and their directionality in a Bayesian Network. Deciding these edges is known as learning the structure. Once a structure is selected, the CPDs must also be trained as described above in Section 2.5.1. Learning the structure of a Bayesian Network requires two things, a scoring metric and a search algorithm. Both of these requirements are covered here. Different options exist for scoring a structure, but they are all derived from the log-likelihood of the structure fitting our data set. For searching, many alternatives exist, but one suitable for this work is presented.

2.5.2.1 Score

Given two candidate network structures, one must use a criterion to select the preferred structure. Introduced here is the idea of scoring a network using this criterion. Many different scoring metrics and techniques exist, but all are based on how well the learned network fits the data. These are divided into two different types, Bayesian Scores [35], and information-theoretic scores. However, it has been shown in general that information-theoretic scores perform better [39]. Log-likelihood is introduced as it is the foundation of information-theoretic scores to begin.

To calculate the log-likelihood of the data set,

$$\hat{ll} = \log P(\mathcal{D}|\theta, G) = \sum_i \sum_n \log p(x_{n,i}|x_{n,pa_x}, \theta_i), \tag{2.20}$$

is used. To perform this computation requires progression through the network summing the log probabilities of each node given an observation of an instance of the data. A perfect fit would make a log-likelihood of zero. Therefore, the log-likelihood is necessarily negative.

Ultimately, however, solely using log-likelihood as a score has a significant drawback. Even if the connection between nodes is weak at best, it is invariably better to add an edge. This relates to Section 2.3. The data of the components may be independent, yet noise in the data may mislead us to add a spurious edge to improve the value. The best scoring network under a log-likelihood score is fully connected [21]. A fully connected network, however, is undesirable as it is computationally wasteful to have to calculate the relationship between potentially ineffective edges. A fully connected network adds time with limited benefits and potential issues with distributions. A fully connected network may also overfit the data, reducing the generality. Three candidate solutions are presented as a solution.

The first candidate solution to adding spurious edges is the Bayesian Information Criterion (BIC),

$$BIC = ln(M)k - 2(\hat{ll}). \tag{2.21}$$

The BIC was intended as a way to weight the number of parameters and the size of the data set with the fitness. The number of instances in the training data is M, while the number of parameters in a network is k. Typically, k also includes the number of nodes. However, the designer selects the nodes, and they are consistent between graph structures. So, k does not include the nodal count. The BIC score should be minimized, as is standard with optimization [40].

The Akaike Information Criterion (AIC),

$$AIC = 2k - 2(\hat{ll}), \tag{2.22}$$

is considered next. The AIC was explored as another option because it performs well with small data sets. The AIC is only concerned with the number of edges explicitly and not the size, M, of the data [41].

Finally, the Corrected AIC (AICc) [41] is examined. The AICc is formulated for tiny datasets; adding an extra correction factor to the AIC,

$$AICc = 2k - 2(\hat{ll}) + \frac{2k^2 + 2k}{M - k + 1}. \tag{2.23}$$

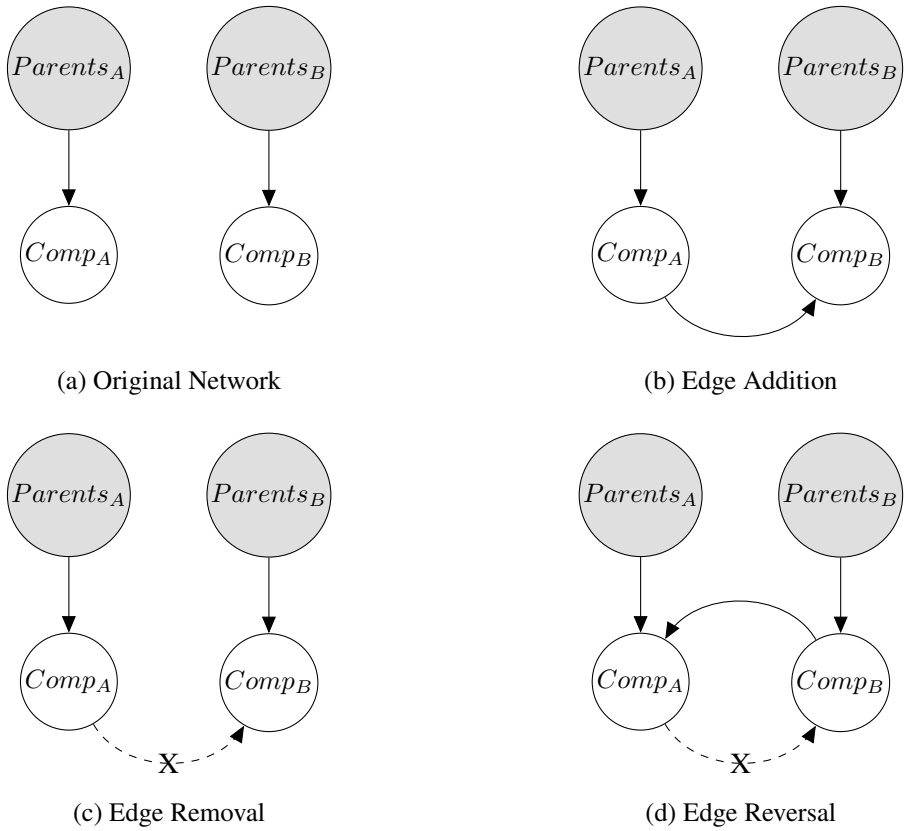
This correction factor tends to zero as the data set becomes large. As the number of examples in the data set, M, increase the third term diminishes. As with all of these scores, the AICc should be minimized, similar to posing an optimization problem. The AICc strikes a balance between the number of data points with the need for edges to represent the underlying distribution of the data. The AICc is used for all of the test cases in Chapter 5.

One can then look at the change in the scores between structures to solve this optimization problem. Once a baseline score is established, simple permutations of the graph structure can be compared to verify the improvement of the score. These simple changes of one edge at a time are inexpensive to compare without analyzing the entirety of the network.

2.5.2.2 Search

With a quantifiable way of comparing graph structures, how graph structures are generated is investigated. Initialization begins with the simply connected network, as shown before in Figure 2.6. The simply connected network is seen as the minimum for this type of application as it is assumed in this problem that components in system design are not independent. Accordingly, unconnected nodes are not allowed; there must always be some connection to the root Requirements of the engineering system to any node.

Next, the options for connecting nodes to learn the statistical interactions between components are reviewed. There exist three options to change a graph: Edge Addition, Edge Removal, or Edge Reversal. Of course, an edge that does not exist cannot be removed or reversed. An arbitrary example of this can be seen in Figure 2.9. Although intuitive, it is necessary to be clear what



(a) Original Network

(b) Edge Addition

(c) Edge Removal

(d) Edge Reversal

Figure 2.9: Edge Operations

options may be performed. When examined node by node, complex networks are made up of only operations of addition, removal, or reversal.

Algorithmically a choice is made of whether to perform one of the three options on a set of nodes to improve the scoring metric. Many algorithms exist for searching for the optimal network. However, let us step back and make a critical observation about structure learning, which limits our approach to the simplest of algorithms. Koller and Friedman [21] observed:

> "... when doing density estimation from limited data, it is often better to prefer sparser structure. The surprising fact is that this observation applies not only to networks that include spurious edges relative to G*, but also to edges in G*. That is, we can sometimes learn a better model in terms of generalization by learning a structure, with fewer edges, even if this structure is incapable of representing the true underlying distribution."

With this observation in mind, the limited data available in engineering systems designs increases the likelihood that it is a small change from our initial Simply Connected Network in Figure 2.6.

With the intuition that a basic network is likely to be the best, one can surmise that an optimal network is only a limited number of operations from the Simply Connected Network. Therefore, a local search is performed. A local search is not only likely to find the optimal network or nearly optimal network, but it has the advantage of being computationally tractable given the fact an exhaustive search is super-exponential in computational complexity [21, 35]. Many different search algorithms exist [42, 43]. Implemented here is a modified first-ascent hill climbing method with some additional randomization, shown in Algorithm 1.

Algorithm 1 Structure Search

function STRUCTURE_SEARCH(G_0, iterations)
$G \leftarrow G_0$
$G' \leftarrow G_0$
  for i in $iterations$ **do**
    draw 2 nodes in G'_i: A and B
    Randomly: Addition, Removal, or Reversal in A and B
    if G'_i is $Cyclic$ **then**
      $G'_i \leftarrow G'_{i-1}$
      $Continue$
    end if
    if $score(G'_i) \leq score(G)$ **then**
      $G \leftarrow G'_i$
    end if
  end for
  return G
end function

There exist some drawbacks of Algorithm 1. First, there is no guarantee of finding a global optimum. Second, this method is prone to plateauing. Once a local optima is found, it will keep returning to the architecture the same optima until another better scoring structure is found. Finally, the structure may not even be modified, therefore progress immediately plateaus at the Simply Connected Network. However, because the algorithm continues to keep looking, it is possible to become "unstuck" given enough iterations. One positive aspect is that for each round of iterations, a known number of steps are run. In the following chapter, Chapter 3, mechanisms are provided between major iterations: restarts to address issues at a higher level.

2.6 Design Sampling

With a fully learned structure of CPDs from data, the network can probabilistically generate engineering systems. At our disposal are two methods: forward sampling and amplification. These are used for differing reasons. Forward sampling is used to stochastically generate a single design from the learned network. Amplification is used to generate a host of designs given a threshold. Amplification is necessary for engineering systems designs as multiple feasible designs may meet the product requirements: if you want a table, you could have a dinner table, a coffee table, an end table, or a bar table. A designer must be able to explore the most likely configurations.

The component nodes must be able to be sampled to enable both forward sampling and amplification, including both the discrete and continuous subnodes. First, the constituent parts of both sampling and amplification are explored.

2.6.1 Nodal Sampling

When beginning either amplification or sampling, there must be a system of navigating the topology of the learned structure: a topological sort. The topological sort provides us with an nodal ordering that is used to generate designs. A simple topological sort method is presented in Algorithm 2. Once there is a set of ordered nodes from the topological sort, there is a way to progress through the graph structure to sample or amplify. The topological sort is included in later algorithms for sampling and amplification of graph structures.

Algorithm 2 Topological Sort

```
function TOPOLOGICAL_SORT(graph)
    nodes ← {Requirements, Comp_1, Comp_2, ...Comp_N}
    ordered_nodes ← {Requirements}
    while node in nodes do
        if parents of component are in ordered_nodes then
            ordered_nodes ← node
        else
            nodes ← node
        end if
    end while
    return ordered_nodes
end function
```

- Discrete Subnode Sampling

  Sampling a discrete subnode of a component node is done using Equation 2.2. A discrete subnode sampling provides a vector of probabilities for each of the *N* discrete options, which sum to one. The result from drawing a sample uniformly from 0 to 1 is allocated towards one of the *n* options. The result provides a discrete numerical decision. *n* empty component types are instantiated and added to our engineering system. If this discrete node is categorical, i.e., stylistic such as a tabletop shape, then *n* represents the component category.

- Discrete Subnode Amplification

  Amplification of configurations starts with Equation 2.2. From the probabilities vector returned, select the *n* components with a probability over a designer-selected threshold. In cases where multiple options are viable, these are generated as well, hence amplifying specific options.

- Continuous Subnode Sampling

  For sampling, the learned Gaussian is utilized, and a random variate is selected from the distribution. After sampling, the Probability Density Function (PDF) value is retained for later comparison.

- Continuous Subnode Most Probable Explanation

  With continuous subnodes, there an infinite number of possibilities for each component parameter. When amplifying the database, the most probable explanation (MPE) may be needed. This is simple, as this is just the result of Equation 2.16 bypassing the Multivariate Gaussian. The MPE limits the possibilities to allow for a reasonable number of designs. Further, using the MPE focuses on the most likely feasible sizing for each configuration.

2.6.2 Forward Sampling

With the nodal procedures established, algorithmically, how designs are sampled is examined. Forward sampling belongs to a class of approximate inference methods with a long history [21, 44, 45]. Algorithm 3 shows the forward sampling algorithm. Although the data pertaining to existing engineering systems designs may not be exact to what a human designer wishes to sample, they can still use the learned graph to meet a new set of requirements. The sampling algorithm is fed the requirements the human designer wishes to meet.

Algorithm 3 Forward Sample

```
function FORWARD_SAMPLE(graph, Requirements)
    instantiate Sample
    ordered_nodes ← Topological_Sort
    for component in ordered_nodes do
        Get observed parent features
        sample discrete subnode
        instantiate components as comp from discrete subnode sample
        sample continuous subnode into comp
        Sample ← comp
    end for
    return Sample
end function
```

One particular note, sampling provides only one complete instantiated design at a time. Furthermore, subsequent samples are likely not going to be the same or similar given identical input specifications. So, a single sample may not be indicative of the performance of the learned network's capability of synthesizing new designs. This is ever more apparent when the sample is drawn with a set of system requirements radically different from the observed data. Regardless of whether the specifications of the requirements are observed prior or not, one must sample multiple times to capture the distribution truly..

2.6.3 Amplification

The reason for amplification is to produce a database of candidate feasible design solutions. A threshold is set for each component to adhere to. Using a threshold is different from setting a likelihood for a configuration; instead, this progresses component by component. Because of the topological sort, amplification is done at every component, therefore if a low threshold is provided, an exponentially large number of configurations may be created. The idea of amplification is how the designer "discovers" new configurations. This is a fundamental observation in the work of Kalogerakis et al. [22].

Algorithm 4 shows how the database of samples is created. It is up to the designer to select the appropriate threshold. It is possible that if the threshold is overly large, an empty system may be created. If the threshold is too small, far too many would be synthesized.

Algorithm 4 Amplification

```
function AMPLIFICATION_SAMPLE(graph, empty_config, threshold, Requirements)
    ordered_nodes ← Topological_Sort
    configs ← {empty_config}
    for component in ordered_nodes do
        Get observed parent features
        enumerate discrete subnode into comp given threshold
        while config in configs do
            for enumerations do
                instantiate component as comp
                sample continuous subnode into comp
                config ← comp
            end for
        end while
    end for
    return configs
end function
```

2.7 Conclusions

This chapter covered the basics of Bayesian Networks for use in this work. The central concept explored was the component-based network that requires a hybrid approach that can handle both continuous and discrete variables. Following that, this chapter explored how to learn the structures of this Bayesian Network. Sampling and amplification were covered in this chapter. In the following chapter, explore how these techniques are used to design new configurations and how to include analysis in the design loop.

Chapter 3

Generative Design Architecture

3.1 Introduction

The Generative Bayesian Network (GBN) created in Chapter 2, while a useful tool, may not produce realizable vehicles on the first pass without further learning. Instead, methodologies to incorporate the Generative Bayesian Network into the design loop must be devised. Aerospace design is a loop; iteration and refinement are key even when the process is entirely human-led. Here, this mentality continues, and many of the same processes a human designer would do are automated, saving the human designer time and resources.

Returning to the idea of a design loop, what happens when a generated design fails to meet the requirements? Does this mean that the network architecture is worthless? No, like a human designer, the methodologies also learn from their failures. It is critical to embrace failures as a learning mechanism.

The initial training of the network begins with the Seed Design Database. These seed designs are historical aircraft (but could be any design the human designer wishes to pursue) that have a long lineage of successful use and well-known specifications. However, when designing a vehicle whose requirements are radically different from the observed specifications of the Seed Design Database, one cannot expect the network to at first generate perfect designs that satisfy the requirements. Think of how the first helicopter designers felt when the only flying machines were airplanes. New designs must be cultivated to learn from and further explore the design space.

This process of cultivating examples can be viewed in multiple lights. As covered in Chapter 2, there are two primary techniques for generating designs: amplification and sampling. Both of

these techniques are utilized to aid in relearning. The end goal is to train a network that accurately generates designs that meet the design requirements through a convergent design loop.

This chapter is divided into several sections to explore further how one can best leverage a Generative Bayesian Network in the design of systems using analysis. The following section, Component-Based Systems Design, explains more about how components factor into generation. Section 3.3 explains how analysis can be fed back into the design flow. Design Space Exploration, Section 3.4, covers the different mechanisms and flows of exploration. Section 3.5 explains how everything connects at a high level to create a convergent process. Then a discussion of some of the nuances between a maximum a posteriori and a most probable explanation sample in Section 3.6 is included. Finally, the conclusions summarize this chapter in Section 3.7.

3.2 Component-Based System Design

The first significant assumption for working with the generative Bayesian Network is that the design process is causal. This means that one decision about one component directly effects another. While the directed and acyclic nature precludes a direct cyclic interaction between the component nodes, it does not mean that synergistic design effects cannot be included in the design. As an example, consider an aircraft with a wing and propellers. The acyclic nature means that a wing may be designed and sized before the propellers. Thus, the wing is not sized to take advantage of distributed propulsion effects that later on placed propellers may provide. This would be the case after one design iteration. However, there is a fundamental interaction between the two components. After analysis, the designs which have the wing and propeller interacting perform differently. From further iterative learning, the network can take advantage of distributed propulsion by assigning higher probabilities of those designs if it finds it to be advantageous.

The next primary assumption is in how a human designer discretizes components. This component-based nodal setup was inspired by the work of Kalogerakis et al. [22]. How the components are divided, and what parameters the network learns, however, is up to the designer. At this current point of time, the methodologies cannot, even if one wanted to, replace a human designer. Component selection is a process of intuition and depends on the granularity the designer seeks to pursue as well as the analyses available.

By dividing the domain into distinct components, effects can be represented and captured without enumerating every discrete design combination. Later examples will illustrate the combinatorics of the design possibilities. However, if at all possible, it is essential to minimize the number

of combinations generated for learning. For example, in an aircraft configuration, one can observe a monoplane and a biplane, and perhaps a single-engine design and a twin-engine design. This represents four possible designs: single-engine monoplane, twin-engine monoplane, single-engine biplane, and twin-engine biplane. The key here is that the number of engines, while perhaps statistically linked to the wing selection does not need to be. The edge and the interaction are learned and can be from incomplete data. It is not necessary to observe all four combinations of vehicles to learn. For example, perhaps a single-engine monoplane and twin-engine biplane are in the data set, yielding enough information to make later predictions. When cultivating these new designs, the goal is to fill in these holes in our knowledge without every combination.

Learning these interactions is crucial, and this extends beyond the methodologies explained in Chapter 2. Much of the architecture explained in this chapter is about obtaining a structure that models these interactions via edges. However, as necessary is curating a database to learn from. This is implicit; enough designs are generated via sampling and amplification. Not only is it essential to generate useful data points but the right amount. Ideally, not too many as to overwhelm the AICc score from Section 2.5.2.1 and add spurious edges or bias our continuous regressions. Combined, one could end up with a self-fulfilling prophecy: little exploration occurs, and the ideal solutions may not be found as the same insufficient concepts continue to be regurgitated. The goal of this architecture is to create a graph that makes accurate designs for the given requirements.

3.3 Sizing, Analysis, and Pruning

Once a design is generated, the first thing any designer wants to do, of course, is to see how it performs. Does this design meet the requirements that it was supposed to meet? To get to that point, the system must conduct sizing, perform a set of analyses, and parse the results.

3.3.1 Sizing

As mentioned prior, vehicles that are generated are initially incomplete. When creating the network, the designer chooses which parameters are generated. The remaining parameters are calculated. The designer has many options in choosing which parameters to use when configuring the network. Although, in practice, the code that was created in this work queries the analysis software to determine which component classes are available, there are still options left to the designer. First is the bounds of discrete options. For example, if the designer explicitly wanted to design an airplane (as opposed to a helicopter), it must have at least one wing and a maximum number of wings

allowed (would a designer be interested in 100 wings?). The designer then selects which specific continuous parameters are generated. It was found that selecting as many non-dimensional parameters as possible allows designs to scale appropriately. For example, by placing component origins non-dimensionally in an aircraft, if one were to re-scale the fuselage for a higher passenger count, the tail feathers will move aft rather than staying fixed in space. Many parameters in aerospace are already non-dimensional, such as wing aspect ratio or turbofan bypass ratio, so these are natural choices.

Now, assuming the generated aircraft has been instantiated with many non-dimensional parameters, bounds must be enforced. The designer must know from experience which are physical limitations, such as negative aspect ratio (an undefined quantity), and which bound the design space to create reasonable designs. This is done by sequentially checking the results before continuing to traditional aerospace sizing. If a generated quantity is outside of a bound, then it is brought back to the limit.

After bounding the generated design, it is then taken through a sizing operation. The sizing operation is left to the designer to craft. An example of a sizing operation that must be performed is calculating the wingspan from the aspect ratio and wing area. Sizing also includes more detailed computations, including computing component weights and, thus, aircraft weight statements. Sizing continues until all values necessary for a complete analysis are calculated.

3.3.2 Analysis

The analysis commences after sizing the generated vehicle. This section is not intended to explain a specific analysis rather some general guidelines. The analysis used for many of the test cases, SUAVE, is covered in detail in Chapter 4.

The set of analyses performed on a design must be consistent. Consistency is meant in two ways. First, every design that can be generated should be captured with one set of analyses. Second, the outputs of the analyses should deterministically produce results. For example, although possible, one could use Computational Fluid Dynamics (CFD) for aerodynamic analysis in some cases and handbook correlations for others. These analyses, however, are not consistent as one cannot expect the results to match between these varying levels of fidelity without adjusting for uncertainties.

3.3.3 Pruning

Despite our efforts to keep "bad samples" retaining samples which analysis could not converge is not helpful. For a variety of reasons, the results may be un-physical. The network must learn from physical results. Learning from un-physical results could lead to continued generation of un-physical results. Moreover, some of these pruned samples have cases of infinities or division by zero. These must be removed, hence *pruning* the results.

3.4 Design Space Exploration

Exploring the vast expanses of the design space begins with concepts from Chapter 2. This includes sampling and amplification exploration. With that knowledge, the architecture can be examined in more detail. Recall from Section 1.1.1 that design has divergent and convergent processes, amplification and sampling are how the algorithms "brainstorm" new designs.

3.4.1 Sampling Exploration

Section 2.6.2 covered how an individual sample is generated. However, the goal of exploration is not to generate a single design. Instead, the goal is to establish a database of designs that can be used to further learn from. Utilizing the ideas of Sizing, Analyzing, and Pruning from Section 3.3, a streamlined block is created that is built upon in the overall architecture to be shown later.

Having diversity is one critical issue in generating a database of designs. If training with limited data, biases may exist, which over-predict the probabilities of particular configurations. Repeated forward sampling of the same overall set of requirements is likely to create many similar, if not, identical samples. Learning from too similar samples creates a self-fulfilling prophecy. Thus, it is best not to continue sampling the exact requirements the designer wishes to meet. Instead, sampling is done over a range.

The requirements samples are generated in log-space. This allows the algorithm to capture significant differences in design spaces. In the case of aerospace, if sampling is conducted in a linear space to try to cover vehicles ranging in speeds from multirotors to supersonic transports, the samples generated would be biased towards higher speeds designs relative to the multirotor. Similar trends are seen in other types of designs, such as in the Beams shown in Section 5.2. An order of magnitude difference, O_{max}, which is explained further in Section 3.5.1, is used. From this

magnitude difference, an upper bound is set, S^+, and lower bound, S^-. These are used in,

$$S^+ = \log_{10}(Specs \times O_m) \tag{3.1}$$

$$S^- = \log_{10}(\frac{Specs}{O_m}), \tag{3.2}$$

to select the log space bounds. Next, applying a uniform random number generator, $\mathcal{U}$, one can generate the requirements they wish to sample on

$$S = 10^{((S^+ - S^-)\mathcal{U}(0,1) + S^-)}. \tag{3.3}$$

The entire flow of a Sampling Block is shown in Figure 3.1. Beginning with a design database, the block performs a search to relearn the GBN. The relearning may be toggled off if the human designer had recently called for a structure search. Regardless, the GBN is repeatedly sampled using the requirements generated by Equation 3.3.

The next parts may be performed in parallel: sizing, analyzing, and pruning. Exploiting the parallel processes here can significantly speed up the computations. Any reasonably accurate analysis in aerospace will be far more costly than any operation for the GBN. Minimizing the wall time of the analysis portion is essential. Finally, the new samples are added to the Design Database.

3.4.2 Amplification Exploration

Coexisting with sampling is amplification. The Amplification Block is shown in Figure 3.2. The Amplification Block is very similar to the sampling block with a few subtle differences. First, amplification works for only one set of input requirements to create many output designs. Therefore, sampling is done over a range with a loop to create further designs. All that is necessary is to set the relearning parameters and the requirements the designer wishes to enumerate upon. Then the designer sets whether this a Most Probable Explanation (MPE) sample or a random sample on the continuous parameters. The Amplification Block will be used in the overall architecture shown in Section 3.5.

3.5 Overall Architecture

The overall architecture used in this work can be viewed in several different analogous lights. One way which is the one that is often mentioned throughout this thesis, as a human designer

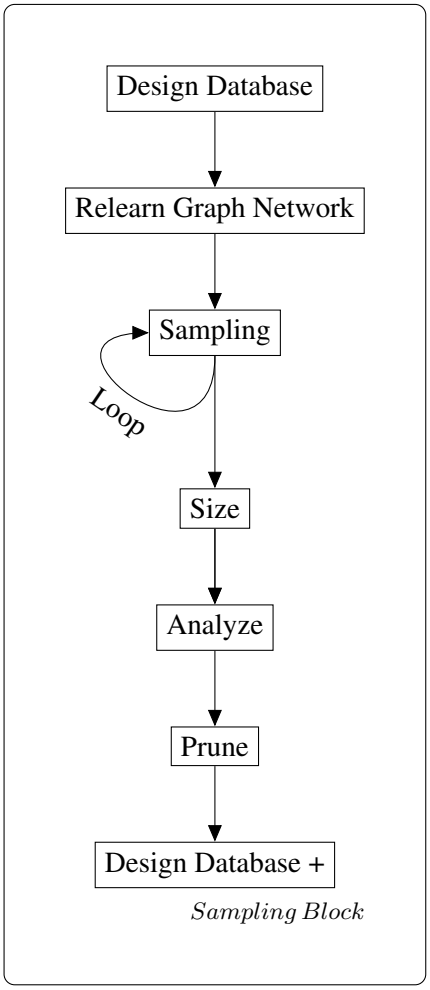


Figure 3.1: Sampling Exploration

iterating designs by learning from their successes and failures. Another way of viewing this is as a stochastic optimization process; the design space is considered a distribution and iteratively fit. This section touches a bit on both.

Stochastic optimization methods, such as cross-entropy, and population-based optimization methods, such as a genetic algorithm, have similar parallels to the overall architecture described in this work [46]. One of the most similar algorithms, the Bayesian Optimization Algorithm (BOA) is a type of Estimation of Distribution Algorithm (EDA) [47]. In the BOA, a Bayesian Network is used instead of a Genetic Algorithm to create a population of strings rather than using typical random mutations. Otherwise, it behaves much the same as a Genetic Algorithm.

However, there are a few crucial differences that result from the goals of this work. First, there

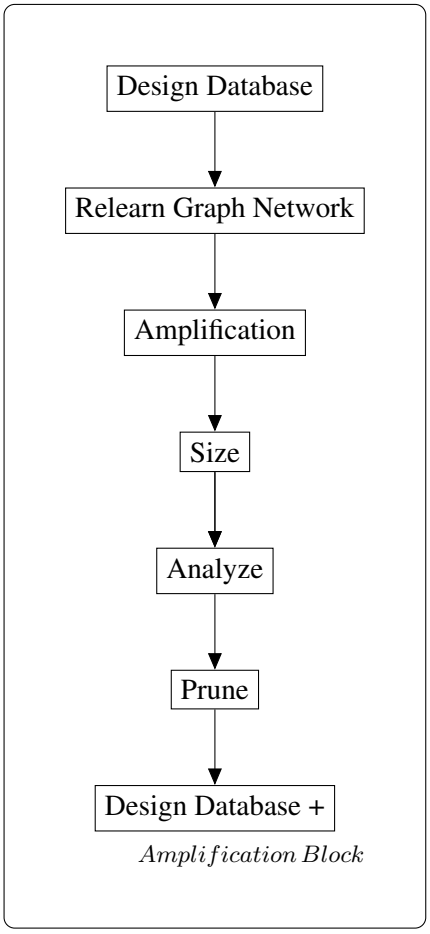


Figure 3.2: Amplification Exploration

is no specific intention to create an "optimal" design. The designs generated must be feasible, i.e., they must be close to the designer's desired requirements, and they must be physically realizable. In many stochastic optimization or population-based optimization methods, there are techniques to select the best samples from the prior iteration using an objective function. Instead, all pruned samples are kept as valuable insights can be gleaned to fill our knowledge gaps in our network. Second, there is no explicit enforcement of constraints (although constraints could be met through pruning). Third, the architecture does not continue for a fixed number of iterations. Instead, it continues for convergence to a set of solutions that will be explained in Section 3.5.1.

Now that the building blocks have been laid, let us examine what this process looks like to a human designer. Figure 3.3 shows a top-down look at how this is used for design. Beginning from the Seed Design Database, an initial simply connected network is created. Then the amplification

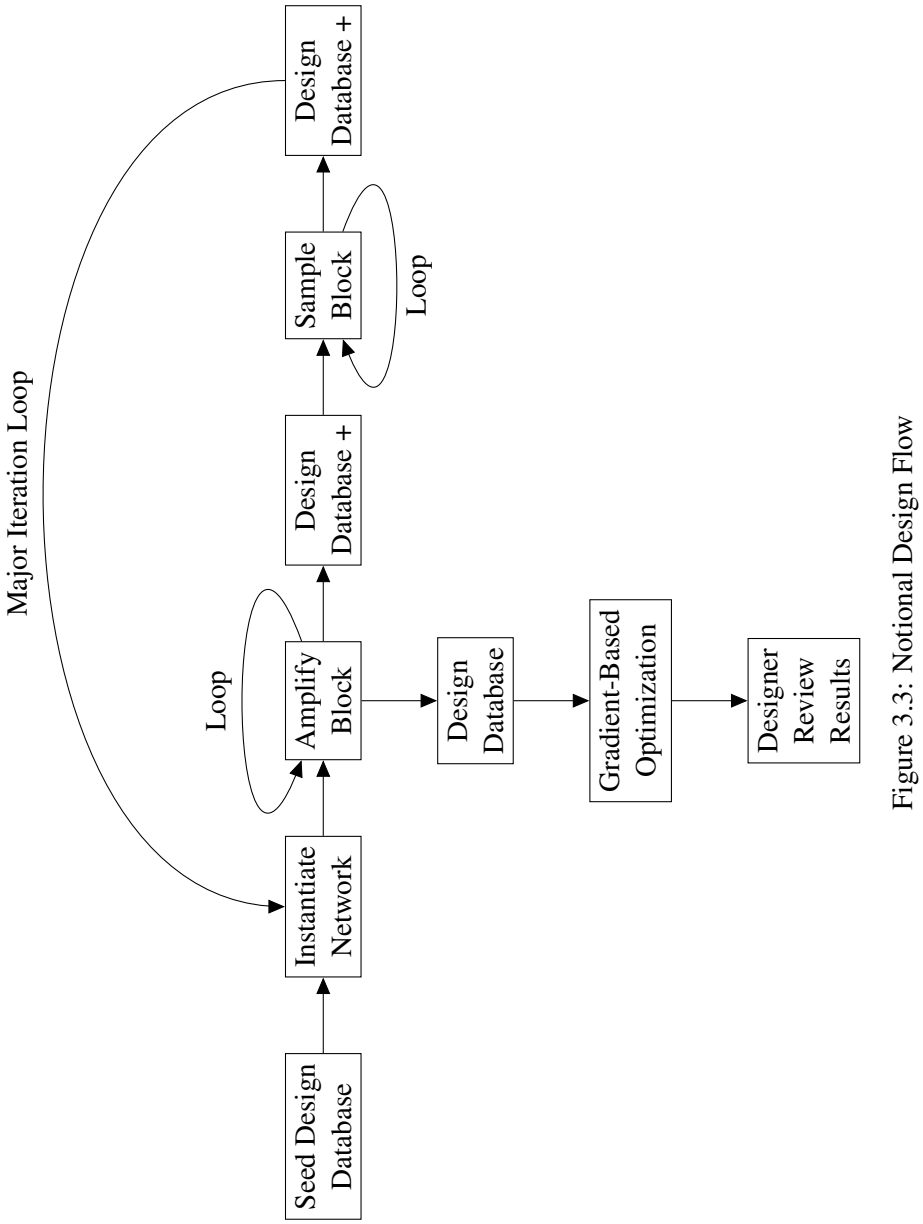


Figure 3.3: Notional Design Flow

block is utilized, and it may be interrogated to create MPE samples followed by Random samples. With further samples in our new Design Database, the Sample Block can be repeatedly used. For example, the first use of the Sample block can skip the graph structure searches, while the second instance may have further graph structure searches.

At the end of the loop, there is potentially an extensive Design Database. However, that new database of samples is discarded after the next graph generation. The next iteration commences with only the Seed Design Database. The reasons for this stem from the Bias-Variance trade-off [48]. Although the GBN will continue to create designs that meet the criteria, they may start creating undesirable designs. Some bias towards historical seed designs, rather than generated designs, is essential. If the pool of designs continued to grow, the GBN would be overwhelmingly filled with generated designs and overwhelm the data of the seed designs. This, of course, comes at the cost of variance; thus, the GBN cannot be expected to capture the breadth of the design space fully. Indeed, the way the GBN architecture searches is evolutionary, rather than revolutionary.

From the human designers' perspective, having some biases in the Seed Design Database is comforting. These seed designs have years of analysis, flight test, and/or operational use. The confidence of knowing the designs that are generated are based on things that work is appealing. Perhaps one can generalize the analogy further, imagine if one were to learn solely from statically stable aircraft designs. The hope is that the vehicles generated would also be statically stable. The GBN does not realize it is doing this, but implicitly the geometric layout is stable, and that will be mimicked.

Returning to Figure 3.3, a lower branch is seen below the Amplification Block. This lower branch is used after convergence to the final graph of the GBN. This lower branch begins with an Amplification Block. Here the final sets of designs are created. The Design Database that is enumerated from the Amplification does not contain the Seed Design Database; rather, these are new designs. Next, these samples are optimized via a gradient-based optimization to minimize an objective function with a variety of constraints that the human designer selects. Furthermore, if there is some difference between the desired specifications and the results, these may be rectified quickly with optimization. After that is complete, then the designer has a set of optimal design configurations to review. Notionally, the human designer would down-select from this optimal set by choosing qualitatively superior designs. These qualities could include safety, aesthetics, familiarity, or more.

With the down-selected optimal designs, the human designer would then perform further analysis and design. Further design and analysis reduce uncertainties about the performance. At some

point, the designer may find that the performance is not up to the desired specifications under further scrutiny. Not all is lost, as has been mentioned throughout this work, there is no "bad sample." This fully-fleshed out design is placed back into the Seed Design Database and used to help us as the design process is restarted. This process may continue until the designer finds *THE design* they genuinely want.

3.5.1 Convergence

There are two options to loop the overall architecture from Figure 3.3: a fixed number of Major Iterations and a convergence process. The fixed number of iterations is an obvious case where the designer selects a number from experience. Consider the convergence case where the Major Iterations loop while a tolerance set by the designer is not satisfied. There are two ways a tolerance can be set. Both examine the maximum error between the MPE sample and the desired specifications. As discussed earlier, further design and optimization are necessarily performed after. Some deviation from our desired requirements is accepted, knowing that feasible designs were sought. One must balance exploration with exploitation. Optimal designs with the exact desired performance will appear after optimization.

The first way of accepting the network as "converged" or trained is to set some designer approved bounds. Because the designer will be tweaking the design later, they know they can change the vehicle some amount. For example, they may be willing to accept a vehicle that is within 100 nautical mile range when they are seeking a vehicle that has a 3000 nautical mile range. To an experienced designer, this is an easy change. Nevertheless, they may not be as willing to accept an unstable design, so the bounds would be tighter in that case. The acceptance criterion is

$$Converged = \begin{cases} \text{true}, & \text{if } |MPE - Reqs| \leq bounds \\ \text{false}, & \text{otherwise.} \end{cases} \tag{3.4}$$

The other tolerance option is also set by comparing an MPE sample at the end of a Major Iteration with the requirements set by the designer. The error, calculated by

$$E = \left| \frac{Reqs - MPE}{Reqs} \right|, \tag{3.5}$$

continues or ends the while loop. However, more useful is the Order of Magnitude Error, O_E,

$$O_E = \frac{Reqs + E}{Reqs}. \tag{3.6}$$

From the Order of Magnitude Error, the sampling range for the next Major Iterations Sampling Block may be obtained. If O_E is large, it indicates the network is not well fit to synthesize vehicles of the requirements provided. Therefore, a wider range of samples is necessary to hone towards feasible solutions. A coefficient, K, is chosen, which, if less than 1, helps shrinks the search space even further. Larger than 1, it can be thought of as damping the convergence. This error used regardless of the convergence criteria,

$$O_m = KO_E. \tag{3.7}$$

Initial iterations tend to have a vast search space. However, in practice, the search space progressively shrinks, and amplification expands the number of feasible configurations very quickly. Therefore, more designs with similar feasible specifications, yet different configurations, are enumerated. Recall from Section 1.1.1, that design thinking requires diverging and converging processes. This architecture functions in this way to diverge, or search widely, and then converge, or shrink the search space. It is to be expected that if the process is not converging that it would expand the search until feasible designs are found.

3.6 MAP and MPE

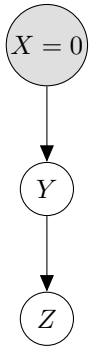


Figure 3.4: CPD Bayesian Net

An experienced reader may notice one particular issue in the sampling methodology; sampling generates and uses the most probable explanation (MPE). Using MPE is an embedded assumption

of the work. As a quick review, at each node, the most likely scenario is selected. However, if one considers the results at the vehicle level, this may not yield the vehicle in which is most likely!

Take this quick example: assume there is a simple Bayesian Network with a basic chain shape, X, Y, Z as shown in Figure 3.4. X is observed to be 0. Let us also create two tables of binomial outcomes for CPD's for Y and Z. If MPE sampling is used, the first choice would be to select Y to be 0 given the CPD in Table 3.1. Then, at the next step for Z, Z would be set to 0 given the CPD in Table 3.2. That is how the algorithm works. However, that is not the most likely outcome!

Table 3.1: CPD: Y given X

	$P(Y\|X=0)$
Y=0	0.51
Y=1	0.49

Table 3.2: CPD: Z given Y

	$P(Z\|Y=0)$	$P(Z\|Y=1)$
Z = 0	0.51	0.01
Z = 1	0.49	0.99

Because of the simplicity of this problem and the chain format, one can very easily compute all outcomes in Table 3.3. Notice that the probability of our MPE is 0.2601, which while high, is not the maximum. The maximum occurs at $Z = 1$. Now the reader is left wondering what happened. This example has developed an edge case that illustrates the concept of maximum a posteriori (MAP) [21]. Only after careful examination, this result is evident.

Table 3.3: CPD: Z given Y, X=0

	$P(Z\|X=0,Y=0)$	$P(Z\|X=0,Y=1)$
Z = 0	0.2601	0.0049
Z = 1	0.2499	0.4851

The maximum a posteriori (MAP) is the value that best fits the data. However, a MAP is computationally more difficult to solve. In the interest of time, the architecture was chosen to converge on MPE rather than MAP. MPE is a straightforward algorithm. Even in our simple case, it did find the second most likely result. For many cases, the MAP is the same as the MPE. Using MPE was an assumption made when crafting this architecture.

Now the reader may be questioning the validity of the methods given that it does not necessarily return the highest likelihood result. However, the methods described here account for this difference. By using amplification steps, at least one feasible solution will be uncovered, and therefore, the MAP should be as well. Furthermore, it would be an oversight as a designer to only examine one output. Instead, the designer must consider multiple configurations as it is hard to be sure which one a priori is the best. Remember, it requires a bizarre set of values to create a case such as the one illustrated above, just further reminder not to be fixated on a single configuration.

3.7 Conclusion

This chapter covers how a generative Bayesian Network is used in a design problem at an abstract level. Some musings were included here of how a human designer should approach component-based systems design. Starting with the ideas of iteration, a series of blocks to design were built. These blocks include an amplification block and a sampling block to extend a design database.

The amplification and sampling blocks are composed of sub-blocks. These sub-blocks include the accompanying generative Bayesian Network process, as well as a Sizing block, an Analysis block, Prune block, and a Relearning block.

Careful sampling methodologies are taken to generate designs of the most value for learning. By automatically re-scaling the sampling bounds, further hones our design space to convergence. The architecture can run major iterations for a fixed number of iterations or use a Most Probable Explanation sample as a stopping criterion for iterations. Finally, a potential pitfall was exposed when exclusively using MPE results. Instead, designers should always use amplification results to compare discrete configuration differences. The next chapter discusses one of the sub-blocks, the Analyze block using a software package named SUAVE.

Chapter 4

SUAVE

An Aerospace Design Environment for the Future

4.1 Introduction

SUAVE is an aircraft conceptual design tool written in Python. Every major analysis discipline of aircraft design is represented in SUAVE's codebase. SUAVE was explicitly created because analysis gaps were found in existing tools [32]. Existing tools were unable to analyze unconventional vehicle designs credibly. With future concepts incorporating new topologies and technologies, it became necessary to start anew, and SUAVE was born.

Owing to the successful release of SU2, an open-source CFD code [49], SUAVE followed suit in releasing the code to the open-source community. The primary motivator for open source code is openness and transparency in research. SUAVE uses the LGPL-2.1 license. This permissive license is the same as SU2 [50]. This specific open-source license allows academia, industry, and government entities to easily collaborate without issues related to intellectual property.

Open source is a benefit to the aerospace research community for several reasons. First is that the code has more eyes on it to check for errors, the additional scrutiny also allows the code to evolve. Second, new research can quickly build on top of prior research. Every evolutionary step does not require "reinventing of the wheel," new research can work on top of existing code. Finally, it broadly accelerates research by easily sharing ideas online.

SUAVE was created not only for the future of aircraft design but to use modern coding practices. It was made as a modular library that removed as many assumptions as possible. In this way, it is expandable and customizable. The code's flexibility allowed later versions to build it into a true

multifidelity code [51], to enable optimization [52, 53], and multifidelity optimization [54]. At this time, the code expanded from its initial use as a code for transport class aircraft into drones and more [55, 56].

What makes SUAVE so uniquely suited to use with Generative Models is Method-Attribute Orthogonality [32]. Method-Attribute Orthogonality enables the aircraft data to exist separately from the analyses. This orthogonality permits the aircraft to be decomposed into constituent components, that may or may not be present in any instantiation of an aircraft, and any variety of analyses can be used to evaluate the instantiated aircraft. These component instantiations themselves are class-based; thus, the generative model only needs to have access to the component classes to query.

This chapter describes how SUAVE functions and some of the specifics related to using it with Generative Bayesian Networks for the results presented in Chapter 5. First, the Architecture is broken down in Section 4.2, then an overview of methods is presented in Section 4.3. Subsequently, there is a discussion of the future of SUAVE in Section 4.4. Finally, there are conclusions in Section 4.5.

4.2 Architecture

SUAVE is more than just methods that calculate aircraft performance; it was drafted with a core set of utilities to drive processes and handle data. The *Data* class allows the designer to access many critical aspects of the vehicle to gain valuable insight into performance while also reducing overhead and spaghetti code and, most importantly, allowing for human readability.

Essential to SUAVE are the analyses. At a basic level in SUAVE, the analyses themselves are nothing more than a sequential list of methods to be executed. In practice, the flow is as follows; a vehicle is instantiated with component subclasses containing all dimensional data. Then, the vehicle can be split into varying vehicle configurations, for example, with flaps extended, a family of variants such as the 737-800 and 737-900, or as extensive as a family of quadcopters to airplanes [55]. With configurations completed, analyses can be specified for each. The analyses encompass any discipline that is necessary for aviation/aerospace and generally at different levels of fidelity. For example, an aerodynamic analysis could utilize a vortex lattice method or SU2. With analyses specified, either point analyses can be performed, or full missions can be flown. The following sections will cover how a mission is converged, elaborate on energy networks, and explain how vehicle optimization is performed, respectively.

4.2.1 Mission Structure

With a vehicle and a full set of analyses, SUAVE can then "*fly*" the aircraft through a mission. A mission is composed of various fixed segments. An example of a mission profile for a Boeing 737 class aircraft can be seen in Figure 4.1. Each segment represents a portion of the flight in which some change in the aircraft is occurring. These segments are divided into categories such as climb, cruise, descent, and more. These segments are further discretized into control points with flight conditions resolved at each point.

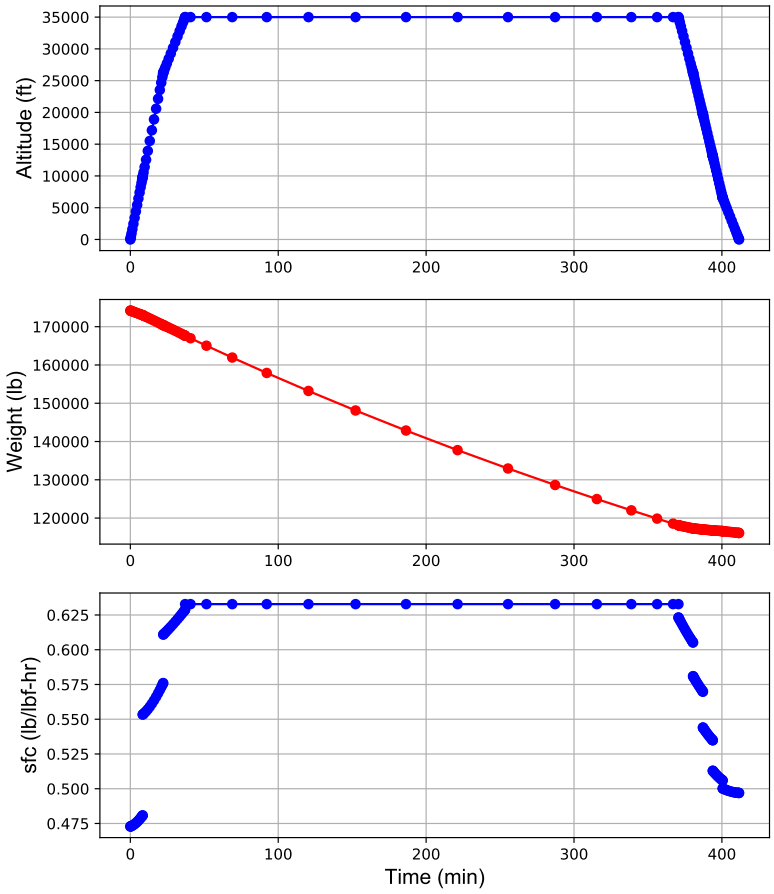


Figure 4.1: Example Mission Profile

The segments are generally solved sequentially, although the capability exists to solve them together in one system. The numerical solutions to the segments are accomplished via pseudospectral methods using Chebyshev polynomials [32]. This methodology allows for convenient and efficient differentiation and integration operators. For example, using these operators, from velocity, one can obtain both acceleration and distance.

For any given segment, there exist known conditions, as well as unknown variables. These unknowns are generally control variables that describe the trajectory of the vehicle. After propagating the unknowns through the analyses, a set of residuals is returned to a root-finding algorithm. SUAVE utilizes MINPACK's "hybrid" algorithm [32] by default. The structure is flexible enough such that it can be used to find optimal trajectories if need be by just changing to an optimizer.

The mission architecture can also regress into itself for segments of segments. For the cases shown in Chapter 5, the vehicle's range must be calculated, given a landing weight. In this case, the mission is solved as a segment itself. This "mission" is a variable distance cruise segment. A new variable, distance, and residual, mass, is used and converged as a larger system of equations encompassing the entirety of the mission.

4.2.2 Energy Networks

SUAVE can analyze a vast array of propulsion systems, including turbofans, afterburning and non-afterburning turbojets, electric ducted fans, internal combustion propeller-based powerplants, battery-powered electric propeller, rockets, solar power systems, and more. The plethora of propulsion system options has been enabled in SUAVE by the Energy Networks. An example from a Solar UAV Network is shown in Figure 4.2 [32].

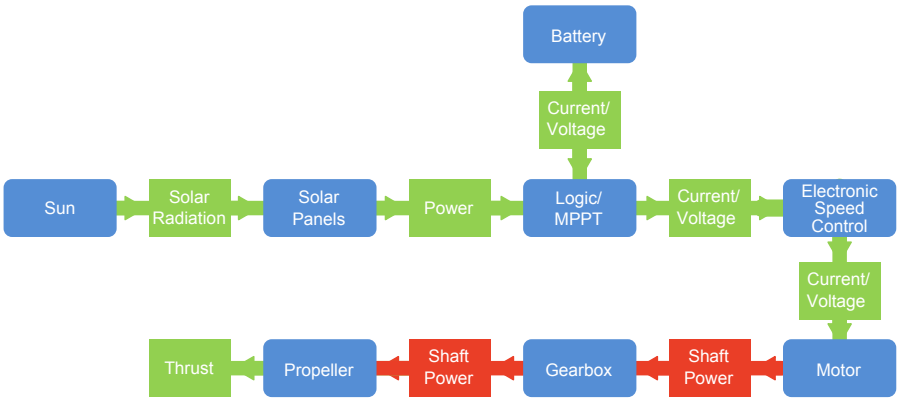


Figure 4.2: Solar Network

Energy Networks differ from traditional propulsion analyses by the code architecture. A network is broken down into physical component classes, and process classes, which, when combined, describe the energy flow of the system. For example, a battery class transfers energy to the speed controller class and then on to a motor class. By breaking down components into a class structure, new networks can be created by linking together components. Additionally, networks can be made into networks of networks. The systems can be made arbitrarily complex, if necessary.

The interface to the mission is simple. When presented with input throttle settings and flight conditions from the mission architecture, the network must return a thrust vector, a mass rate, and the applied power to the vehicle. This interface is common for all networks. Each component class performs the necessary calculations in succession by the links set up in the network. For the work conducted in Chapter 5, exclusively uses turbofan and turbojet energy networks.

4.2.3 Optimization

To perform optimization, SUAVE transforms into a wrapper to be driven [52]. SUAVE then operates as a black-box function; the outputs from the optimizer become inputs to SUAVE. An objective function value and constraint values are then passed to the optimizer. In keeping with the ethos of SUAVE, any parameter in SUAVE is accessible to an optimizer. All of these capabilities are made possible through the *Nexus*.

The *Nexus* inherits from the *Data* class with executable methods. The *Nexus* contains the vehicle configurations, the analyses, the mission segments, the results, and the setup of the optimization problem. When called, the *Nexus* operates on the data to perform analysis. Configuring the *Nexus*

requires a series of files that require setup before use. The first is the *Optimize* file; this is the optimization problem and encompasses the design variables, objectives, and constraints. The next is the *Vehicle* script, which contains the geometries to be optimized. The *Analyses* file spells out which of the configurations uses which analyses and for which reason. The *Mission* file sets each of the flight profiles to simulate. Finally, the *Procedure* file is a set of user-defined functions that are executed that are unique to the optimization problem to be solved. Often, users have specific sizing or requirements that are performed at each iteration.

Since the *Nexus* abstracts SUAVE away to a black-box, any optimization package may be used. This includes both gradient-based and gradient-free global optimizers. For gradients, the *Nexus* can provide finite-difference gradients or outside packages that can take parallelized finite-difference gradients. Many different packages can interface with SUAVE, including basic SciPy packages as well as pyOpt, pyOptSparse, and IPOPT [50].

4.3 Capabilities

This section describes some of the analyses used for generative design in Chapter 5. Some of these analyses were created specifically for generative design to ensure that they would work with arbitrary configurations. Generated components may be sized and placed almost without limits. Thus, the analyses should account for these configurations.

4.3.1 Aerodynamics

SUAVE, being a multifidelity design tool, has many different aerodynamic analyses available [32, 54]. These include the default Fidelity Zero aerodynamic analyses, OpenVSP analyses, SU2 CFD, Athena Vortex Lattice (AVL), lifting line models, and an AERODAS model for small drones [55]. As many of these models are inviscid, the viscous drag components are calculated via a drag build -up. An example of a SU2 analysis can be seen in Figure 4.3.

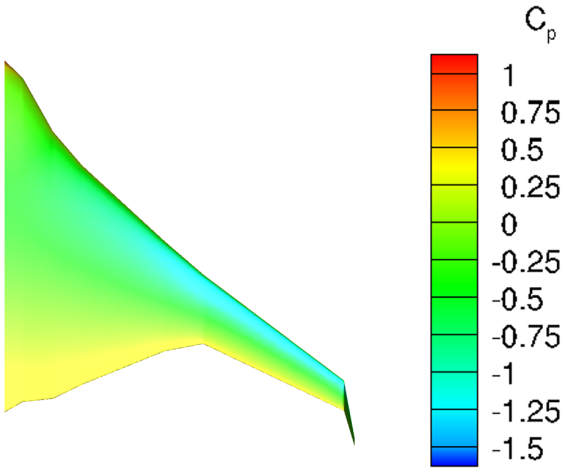


Figure 4.3: Blended Wing Body Pressure Coefficient

The viscous drag build-ups are based on the notes by Kroo [18]. One of the primary sources of drag is the parasitic drag caused by skin friction. Skin friction drag is built up from skin friction coefficients by the components Reynolds numbers. Form factors by Shevell correct for shape [9]. The skin friction coefficient can be modeled as laminar, turbulent, or transition at a fixed length scale. Further information on drag calculations can be found in [32, 50].

For the complex new geometries that a generative model can create, a general and accurate aerodynamic model must be used. The Fidelity Zero model uses a Weissinger Vortex Lattice Model. This model, while an advancement from classic lifting line models by allowing for swept wings, does not consider full configurations together. Therefore, the aerodynamic interactions between wings of all types are not considered. For quick analyses of simple configurations, this may be acceptable. However, the cases in Chapter 5 necessitate a quick but comprehensive evaluation.

Other analyses were considered for these test cases. Using Euler CFD analysis from SU2 is far too slow. AVL also has two main issues. The first issue was convergence time and stability; for some arbitrary cases and angles of attack, AVL failed to converge. The second was code speed, as it is necessary to write input files, use system calls, and read in files. This overhead added considerable evaluation time. For the cases in Chapter 5, uses a new Vortex Lattice Model (VLM).

4.3.1.1 Vortex Lattice Method

With the need for a fast, yet reasonable level of fidelity, a new native SUAVE vortex lattice method (VLM) was created. By starting from scratch, the code could be as flexible as SUAVE has

promised. One of the significant shortcomings of other analyses, especially analyses external to SUAVE such as AVL, is the inability to handle propeller-wing interactions. Given that the purpose of this work is to one day enable new vehicles such as those for UAM, this is a necessary feature. The purpose of this section is not to detail the development of the VLM method, but rather introduce the reader to the types of analyses used in Chapter 5.

The underlying methodologies have been developed elsewhere for many years [57, 58]. However, there are a few reasons to start from scratch. These reasons include code speed, code stability, analysis integration, and the ability to handle supersonic flight.

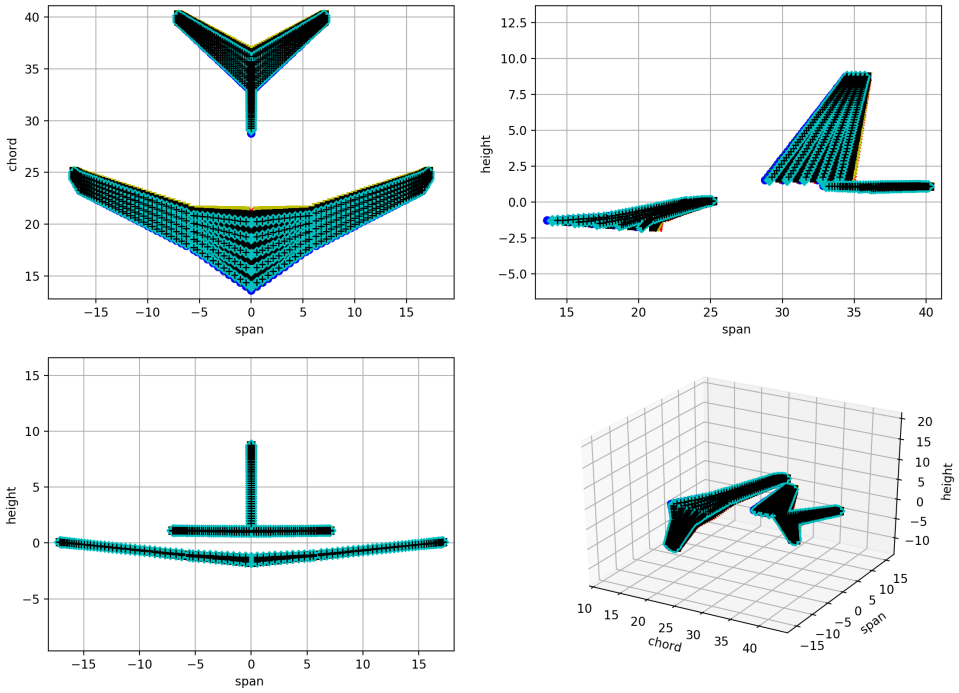


Figure 4.4: B737 Class Aircraft with Control Points from Native Vortex Lattice Method

With one code, both subsonic and supersonic regimes may be analyzed. The grid is stretched to handle subsonic compressibility effects. For supersonic flight, a cone of influence is used to reduce the vortex interactions [59].

There are several code speed enhancements. When used with a mission, surrogate models for lift

and drag coefficients are constructed. Because lift from a VLM method is linear, it builds a simple linear model. Similarly, for drag, a quadratic function is fit. These surrogates reduce function calls. These are not applicable when using a propeller-based propulsion system with wing-propeller interactions; in this case, the mission must directly solve the VLM integrated with the blade element momentum results from the propeller. The code is vectorized; this means that the VLM only needs to be solved once to resolve all angles of attack and Mach numbers necessary to build the surrogate models.

4.3.2 Weight Estimation

As a further expansion of SUAVE methodologies, the vehicle class can now regress through its own data structure to query components for weights. Hence, it can determine the total mass of the vehicle. For weight estimation for vehicle components, many of the functional methods already in SUAVE for use in transport category aircraft are utilized [32]. These methodologies for component weight estimation originated in the work by Kroo [18]. However, the methodologies lacked credible weight estimates for complex wing designs.

4.3.2.1 Main Wing Weight Estimation

Starting from the same fundamental equations that Kroo began with, an analytical expression for wing weight was derived. First, beginning with the bending moment of the wing box. Equation 4.1 shows how the bending moment relates to the thickness of the section, t, the stress, σ, and the loaded area, A,

$$M_b = 2\sigma \frac{A}{2}\frac{t}{2} = \sigma A \frac{t}{2}. \tag{4.1}$$

Rewriting Equation 4.1 in terms of an allowable stress yields Equation 4.2. Equivalently, this can be written in terms of area, A,

$$\sigma_{allow} = \frac{2M_b(y)}{tA} \tag{4.2}$$

$$A = \frac{2M_b(y)}{t\sigma_{allow}}. \tag{4.3}$$

Material density and area are integrated over the span to calculate the weight of the bending material of the wing box. Assuming constant material properties, the material density, ρ, is removed

from the integral. Next, using the relation for area arrives at an expression for the weight in terms of bending moment. For the remainder of these derivations, σ, is the allowable stress,

$$W_b = 2\int_0^{b/2} \rho A dy = \frac{4\rho}{\sigma}\int_0^{b/2} \frac{M_b(y)}{(t/c)c} dy. \tag{4.4}$$

However, the bending moment itself is a function of the spanwise lift loading. At any spanwise station, the lift outboard causes a bending. Therefore, the bending moment is an integral over the lift distribution, l,

$$M_b(y) = \int_y^{b/2} (y) dy. \tag{4.5}$$

To model the lift distribution, one could either take this directly from an aerodynamic solution or assume a functional shape, as shown in Equation 4.6. In this case, an elliptical lift loading is assumed. Although an elliptical wing minimizes induced drag, the spanwise loading is weighted further towards the tips compared to a linear loading. Therefore, this is a conservative weight estimate assuming the induced drag is optimistic,

$$(y) = L_0\sqrt{1 - \frac{2y}{b^2}^2}. \tag{4.6}$$

Normalizing the y ordinate over the span results in η. Then, combining Equations 4.5 and 4.6 results in

$$W_b = \frac{2\rho b^2 L_0}{\sigma}\int_0^1 \int_\eta^1 \frac{\sqrt{1-\eta^2}}{c(\eta)(t/c(\eta))} d\eta d\eta. \tag{4.7}$$

This double integral in Equation 4.7 is excellent for wing shapes, which can be parameterized over the entire span. However, in SUAVE, complex wing shapes are defined using segmented wings. The discrete nature of these necessitates that the integral is only valid between sections. It is assumed that between segment breaks, the functional shape is linear; if the user wants smoothly varying wing segments, additional segments should be added. The weight becomes a sum of double integrals,

$$W_b = \frac{2\rho b^2 L_0}{\sigma}\sum_{n=1}^{m}\int_{\eta_{n-1}}^{\eta_n}\int_\eta^1 \frac{\sqrt{1-\eta^2}}{c(\eta)(t/c(\eta))} d\eta d\eta. \tag{4.8}$$

The prior equations expressed the thickness of the segment in terms of wing chord and thickness to chord ratio. The thickness is only necessary to solve for bending, not the wing chord directly.

Assuming a linear shape, one may arrive at an expression for thickness as Equation 4.9. The η_{n-1} term indicates the value at the last segment station location. By grouping terms, the expression is simplified, and called A, B, and C,

$$t(\eta) = c(\eta_{n-1})(t/c(\eta_{n-1})) - \frac{c(\eta_{n-1})(t/c(\eta_{n-1})) - c(\eta_n)(t/c(\eta_n))}{\eta_n - \eta_{n-1}}(\eta - \eta_{n-1}) = A - B(\eta - C). \tag{4.9}$$

Combining Equations 4.8 and 4.9 produces,

$$W_b = \frac{2\rho b^2 L_0}{\sigma} \sum_{n=1}^{m} \int_{\eta_{n-1}}^{\eta_n} \int_{\eta}^{1} \frac{\sqrt{1-\eta^2}}{A - B(\eta - C)} d\eta d\eta, \tag{4.10}$$

which is a more compact representation. Now solving the double integral of Equation 4.10 results in,

$$\begin{aligned}
W_b = \frac{2\rho b^2 L_0}{\sigma} \sum_{n=1}^{m} \Big(& (\frac{1}{4B^3}(2A^2\pi\eta + 4ABC\pi\eta + B^2(-1+2C^2)\pi\eta - \\
& 2(2A^2 + 4ABC + B^2(-1+2C^2))\sqrt{1-\eta^2} + \frac{2}{3}B\sqrt{1-\eta^2}(3A\eta + B(-1+3C\eta+\eta^2)) + \\
& 2B(A+BC)\text{ArcSin}[\eta] - 2(2A^2 + 4ABC + B^2(-1+2C^2))\eta\text{ArcSin}[\eta] - \\
& \frac{4(A+BC)^2\sqrt{-A^2-2ABC-B^2(-1+C^2)}\text{Log}[A+BC-B\eta]}{B} - \\
& \frac{4(A^3+3A^2BC+B^3C(-1+C^2)+AB^2(-1+3C^2))\eta\text{Log}[A+BC-B\eta]}{\sqrt{-A^2-2ABC-B^2(-1+C^2)}} + \\
& \frac{8(A+BC)^2\sqrt{-A^2-2ABC-B^2(-1+C^2)}\text{Log}[-A-BC+B\eta]}{B} + \\
& (4(A^3+3A^2BC+B^3C(-1+C^2)+AB^2(-1+3C^2))\eta\text{Log}[-B+A\eta+BC\eta - \\
& \sqrt{-A^2+B^2-2ABC-B^2C^2}\sqrt{1-\eta^2}])/(\sqrt{-A^2-2ABC-B^2(-1+C^2)}) - \\
& \frac{1}{B}4(A+BC)^2\sqrt{-A^2-2ABC-B^2(-1+C^2)}\text{Log}[-B+A\eta+BC\eta + \\
& \sqrt{-A^2+B^2-2ABC-B^2C^2}\sqrt{1-\eta^2}] + \\
& \frac{1}{B}4(A+BC)\sqrt{-(A^2+2ABC+B^2(-1+C^2))^2}\text{Log}[A^2\eta + 2ABC\eta + B^2(-1+C^2)\eta + \\
& \sqrt{-(A^2+2ABC+B^2(-1+C^2))^2}\sqrt{1-\eta^2}])\Big)\Big|_{\eta_{n-1}}^{\eta_n}.
\end{aligned} \tag{4.11}$$

Equation 4.11 holds for segments that linearly vary. However, this breaks down in the case a segment is constant in thickness as B becomes infinite. Returning to Equation 4.8, thickness term is removed from the integral. Especially relevant here is that both derivations can exist together, such that a wing can have both tapered and straight segments as only the integral needs to be substituted. The result for straight segments is much simpler,

$$W_b = \frac{2\rho b^2 L_0}{\sigma} \sum_{n=1}^{m} \frac{1}{c(\eta_n)(t/c(\eta_n))} \int_{\eta_{n-1}}^{\eta_n} \int_{\eta}^{1} \sqrt{1-\eta^2} d\eta d\eta \tag{4.12}$$

$$W_b = \frac{2\rho b^2 L_0}{\sigma} \sum_{n=1}^{m} \Big(\frac{1}{3} (\frac{1}{8} (-\eta_{n-1}(5-2\eta_{n-1}^2)\sqrt{1-\eta_{n-1}^2} - 3\text{ArcSin}[\eta_{n-1}]) + \frac{1}{8}(\eta_n(5-2\eta_n^2)\sqrt{1-\eta_n^2} + 3\text{ArcSin}[\eta_n])) \Big). \tag{4.13}$$

Returning to L_0, which was not fully defined earlier. An ultimate lift on the main wing, as assumed by Kroo in [18] is used,

$$L = n_{ult}\sqrt{\text{ZFW TOW}} = 2L_0 \int_0^1 \sqrt{1-\eta^2} d\eta. \tag{4.14}$$

If multiple wings are present, the ratio of the wing to the total main wing area is used. Integrating over the lift distribution and solving results in

$$L_0 = \frac{2n_{ult}\sqrt{\text{ZFW TOW}}}{\pi}. \tag{4.15}$$

These equations account for the primary bending structure but do not account for the secondary structure weights. The secondary structure includes things like skins, ribs, and more. A correction factor is necessary. Using the same correlations established in [18], one can simply correct for secondary structure based on the wing reference area. The correlations for simple wings derived in work by Kroo cannot be directly compared as the wing lift distribution used is not elliptical. Therefore, only the linear terms can be added directly for secondary structural weight.

4.3.3 Center of Gravity Estimation

The methods established by Kroo [18] are used to calculate the centers of gravity for the aircraft,. However, one innovation in SUAVE for this work is the automated center of gravity estimation. As

part of a sizing procedure, each vehicle component can have CG assigned relative to its origin. Now every physical component in the vehicle can be queried to compute the total vehicle moment, much like the weight estimation methods. Now the vehicle level CG can be computed.

4.3.4 Stability

SUAVE contains an array of stability analyses: from handbook methods to an interface with AVL [51] to CFD analysis. However, for the work in this thesis, simple methods were utilized. These are simple as just calculating static margin.

The aircraft's neutral point can be calculated using the lift curve slopes from the new vortex lattice method described in Section 4.3.1. The work conducted in this thesis, which pertains to transport class aircraft in cruise, focuses on the moments due to lift. Furthermore, as this is conceptual level design, it assumed that the airfoils are symmetric, and for small angles of attack, the moment is negligible. One can show the force due to lift for each surface is

$$F_{wing} = 0.5\rho V^2 S_{wing} C_{L\alpha_{wing}} \alpha. \tag{4.16}$$

Moments are taken to be at the origin of the aircraft. Then it is assumed the forces act at the aerodynamic center of each wing. The distance from the origin to the wing aerodynamic center is the length l. The sum of the moments for all wings about the origin is taken. The distance can easily be calculated as in

$$NP = \frac{\sum lF}{\sum F} = \frac{\sum l S_{wing} C_{L\alpha wing}}{\sum S_{wing} C_{L\alpha wing}}. \tag{4.17}$$

Several terms drop out as they are common to the summation, including the angle of attack and the dynamic pressure terms. Dropping these terms assumes that the dynamic pressure loss on the tail feathers caused by the wing is negligible. Next, the center of gravity is calculated, as discussed in Section 4.3.3. The distance between the neutral point, NP, and the center of gravity, CG, normalized by the main wing mean aerodynamic chord is the static margin. These methods for calculating neutral point disregard many viscous effects, thus they can only be used for initial conceptual design.

4.4 The Future of SUAVE

SUAVE has tremendous power already. However, many areas could be improved. Some of the shortcomings are in the architecture, while others are in future analyses to expand the toolbox.

One area of future research to upgrade the architecture is in enabling gradients throughout the code. Many of the high fidelity analyses require external interfaces that may not be simply-differentiated analytically. For current optimization cases, the optimizers must finite-difference through the *Nexus* to provide gradients. In the future, to tackle more complex design cases that may require optimization with many design variables, it will be crucial to have gradients without finite-differencing SUAVE. It is suggested to use automatic differentiation to extract gradient information. Automatic differentiation is essentially chain rule, which breaks down the code into simple to differentiate chunks. Concurrently, there is the possibility to enable parallelization and compilation of these efforts. The end goal is speed without sacrificing capabilities and code readability.

One of the new analyses suggested for SUAVE is a Free Wake model. This model is used for mid-fidelity rotorcraft analysis. This would enhance SUAVEs abilities to tackle many of the UAM type aircraft that are being suggested. Although the current analyses handle wing/propeller interactions, an accurate simulation of rotors at a higher level of accuracy would lend credibility to pure rotorcraft simulations.

The interfaces with SU2 could be improved to smooth the integration. Currently, the SU2 interface relies on writing files, using system calls, and reading results files. Instead, it should be done with pure Python code. Furthermore, the hope is one day to have automatically generated meshes for RANS analyses. Having automated RANS capabalities would enable true high-fidelity simulations at the conceptual level.

Finally, with many of these unconventional designs, simple weight correlations fail. For primary weight estimation, SUAVE has build-up methods. However, it lacks a high-fidelity simulation for structural analysis, design, and primary weight estimation. Therefore, a finite element based structural model would be a highly useful addition to the code base.

4.5 Conclusions

This section encompasses the basics of SUAVE and how it is used to generate results in this work. SUAVE has grown tremendously since its inception in 2013. A chapter in a thesis could never fully encompass the work therein. Furthermore, SUAVE is ever improving with a team of

devoted developers. Anything written here will only hold valid at the time of publication. However, there are overarching themes of SUAVE that hold.

The first is the architecture of SUAVE being flexible. By adhering to Method-Attribute Orthogonality, different analyses can be quickly substituted without changing the vehicle definition. Missions in SUAVE are segment-based and solved using collocation methods. Energy networks are a way to link physical components to model the flow of energy and power to allow for the simulation of complex propulsion systems. The *Nexus* allows SUAVE to be driven as a "black-box" function by external optimizers.

SUAVE can be viewed as a library of analysis. The wide range of general analyses allows SUAVE to handle a broad array of aircraft types. Several new analyses were created to complete this work. These include a new native vortex lattice method and a main wing weight estimation method for segmented wings.

Although currently, SUAVE possesses a variety of capabilities, future vehicles will require new methods for credibly analysis and design. Built on modern code and ever-evolving SUAVE is well poised for the future of aerospace.

Chapter 5

Test Cases

Aeronautics was neither an industry nor a science. It was a miracle.

Igor Sikorsky

5.1 Introduction

This chapter builds upon the theory and methodology presented in the previous four chapters to show the effectiveness of the methods. Four test cases are shown. Each successive case becomes more complicated, starting from a simple initial test case. The four design cases investigated include a structural beam design and three aircraft design problems. The goal is to explore how the algorithms behave and illustrate that they create credible designs.

There is one distinct difference between the architecture presented in Chapter 3, and the ones presented here in this chapter. At no point are any of the designs passed through an optimizer. The results are in the raw form from the algorithms to provide the reader with a better idea of the power of these methods to see how well they perform without post-processing.

Further optimization would yield final realizable results. As is discussed in Chapter 3, optimization is an essential part of the design process using these methodologies. The generative Bayesian Network can create feasible results but does not ensure optimality. An optimal configuration would not only meet specifications exactly, but they would represent truly the best a configuration could perform given an objective.

5.2 Beam Design

5.2.1 Introduction

The methodologies explained in Chapters 2 and 3 are general to any system that can be decomposed into components. In the long term, this work may extend to design things far beyond aircraft and into systems humanity has yet to envision. To begin, let us examine something that most aerospace engineering students first begin analyzing: a beam.

A beam is arguably one of the most straightforward systems. Analyzing a given beam design is quite straight-forward, and any solid mechanics textbook has rigorous solutions. At first glance, it seems like it is only one piece, but in fact, it can be viewed as a sequence of decisions that can be cast as components. The components of the beam are broken into the boundary conditions, the materials, and the dimensions. From those different decisions, one can arrive at widely different levels of performance.

While the beam problem may be viewed as simplistic, it is a proving ground for far more complex systems. With respect to the behavior of a beam, the analysis will yield results that change linearly with modifications in width, but non-linear with respect to other dimensions. Width and height alterations to the beam geometry non-linearly change the results due to the boundary conditions and moment of inertia. Furthermore, for any given set of requirements, more than one solution will satisfy. The power of the methodologies shown in this thesis is that they can uncover multiple valid feasible solutions within a design space. The beam problem should thoroughly expose that capability.

5.2.2 Beam Problem

This beam problem requires a specific load and displacement at the material yield strength. The load specified is the total integrated force of a distributed load. This beam problem will have multiple feasible solutions. A further simplification, as a low-fidelity analysis is sought, is that the beam is only subjected to Euler-Bernoulli bending: shear deformation is ignored, and it is only valid under small changes in slope. Using Euler-Bernoulli bending is generally a reasonable assumption for slender beams. The use cases for this are somewhat nebulous, as most beams will be used at far less than the yield strength. What matters is that the network predicts designs based on learning from consistent analysis, and the analysis itself, as well as the solution results, are not trivial to predict, i.e., linear.

There are many possible boundary conditions for beams. Two of the most common, and also

familiar to most engineers are selected: a Cantilevered boundary condition and a Simply-Supported boundary condition. The Simply-Supported beam, as seen in Figure 5.1, is pin supported at one end and roller supported at the other. This setup results in the maximum deflection at the midpoint.

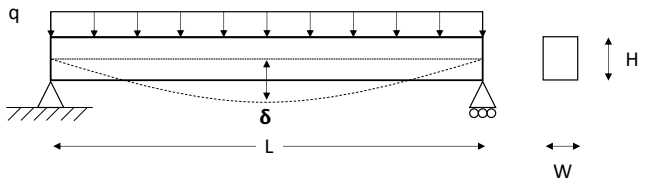


Figure 5.1: Illustration of a Simply Supported Beam

The Cantilever beam, as seen in Figure 5.2, is fixed at one end and free at the other, resulting in maximum deflection at the tip. Given an identical set of dimensions, material, and load, the Cantilever beam will deflect more than the Simply-Supported beam.

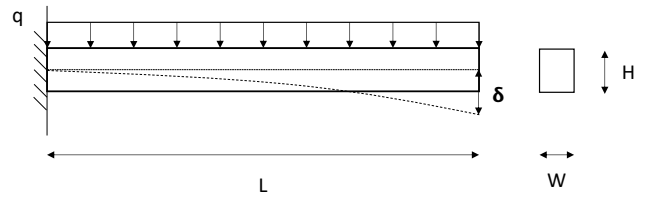


Figure 5.2: Illustration of a Cantilevered Beam

A simple analytical analysis is used for both sets of boundary conditions. The moment for a beam is computed from stress,

$$\begin{aligned} \sigma &= \frac{My}{I} \\ M &= \frac{I\sigma}{y}. \end{aligned} \tag{5.1}$$

For simplicity, a rectangular cross-section of height, H, and width, w, was selected. The moment of inertia for a rectangular cross-section is

$$I = \frac{wH^3}{12}. \tag{5.2}$$

For both beams, the distributed load, q, is defined to be the maximum load f_{max} divided by the length, L. The maximum bending load for a simply supported beam occurs at the midpoint, while the cantilever beam's maximum bending load occurs at the attachment. The moments for both beams can be found to be in terms of their dimensions and the distributed load,

$$\begin{aligned} M_{SS} &= \frac{q\sigma L^2}{8} \\ M_C &= \frac{q\sigma L^2}{2} \end{aligned}. \tag{5.3}$$

The integrated maximum load, f_{max}, can be solved for in terms of the yield stress and dimensions. The maximum load for both boundary conditions,

$$\begin{aligned} f_{max_SS} &= \frac{8\sigma I}{yL} \\ f_{max_C} &= \frac{2\sigma I}{yL} \end{aligned}. \tag{5.4}$$

The deflections at this load condition are solved in terms of the distributed load, q:

$$\begin{aligned} \delta_{max_{SS}} &= \frac{5qL^4}{384EI} \\ \delta_{max_C} &= \frac{qL^4}{8EI} \end{aligned}. \tag{5.5}$$

These derivations yield the two performance criteria: displacement at yield, δ, and the maximum integrated load, f_{max}. Design requires material properties, including modulus of elasticity, E, and yield stress, σ. Two materials commonly used in aerospace are selected: 7075-T6 aluminum and 4130 steel alloys. The material properties for both of these are shown in Table 5.1.

The yield strengths of both materials are rather close. However, the modulus of elasticity values vary significantly. This means that one material cannot just substitute for the other and still satisfy

Table 5.1: Beam Material Properties

Material	Yield Stress (MPa)	Modulus of Elasticity (GPa)	Source
Aluminum	503	71.7	[ASM Data Sheet] [60]
Steel	435	205	[ASM Data Sheet] [61]

the deflection criterion.

5.2.3 Architecture

Using the architecture presented in Chapter 3, one can build up the parameters for each major iteration. Some of the more basic configurations are used here to prove the concepts established in Chapters 2 and 3. Again, the intention is to utilize the relatively inexpensive functional cost and limited design space of a beam relative to an aircraft conceptual design to verify the functionality of the codebase. A table summarizing the configurations of the architecture are outlined in Table 5.2.

Table 5.2: Beam Architecture

Major Iterations: 15			
Amplification Block 1			
Tolerance	**MPE**	**Iterations**	**Searches**
0.00	TRUE	1	0
Amplification Block 2			
Tolerance	**MPE**	**Iterations**	**Searches**
0.00	FALSE	10	0
Sampling Block 1			
Iterations	**Samples Per Iteration**	**Order of Magnitude Range**	**Searches**
5	5	0.4	0
Sampling Block 2			
Iterations	**Samples Per Iteration**	**Order of Magnitude Range**	**Searches**
5	5	0.4	10

Rather than iterating over an unknown number of major iterations until convergence, a fixed number is used. Fifteen major iterations are chosen. The reason for a high number of major iterations is that this case seeks to "over converge," which is to go beyond convergence. In this case, more than just the MPE sample taken at the end of iteration should be fully converged. For complex systems, there may be zero to infinite possible configurations; it is not realistic given the architecture to ensure a nearly infinite number of converged solutions. For this beam case, there is a priori

knowledge that there are four discrete cases. A large number of major iterations are chosen to ensure the Bayesian network has an opportunity to learn all of these thoroughly. To further simplify, a static order of magnitude sampling range is chosen.

For the architecture, four design blocks are used. The first two utilize amplification, while the second two are sampling blocks. The first amplification block is configured with a tolerance of zero and using MPE continuous sampling that will generate each of the four possible beam configurations. The second amplification block is similar to the first but provides a random sample on the continuous parameters. This is done to produce diversity by ensuring at least two samples of each beam configuration are generated at each major iteration.

The two sampling blocks serve two slightly different functions. The first block will add additional samples to our design database. The second block generates designs while simultaneously refitting and searching the graph structure. Both will generate up to 25 samples. The second block will perform ten consecutive structure searches at each iteration; therefore, it will modify 50 edges for each major iteration to seek a better graph representation. By further sampling beyond the MPE, one can search wider and better fit the real underlying distributions.

5.2.3.1 Seed Designs

Specifications must be established, and seed designs added to the Bayesian Network to begin the iteration process. The specifications are 5 cm of deflection at the yield strength of the material while supporting 5 million Newtons of force. These values did not represent any realistic or useful scenario and were chosen for roundness.

Now two different designs, as can be seen in Table 5.3, are used as the Seed Design Database. The results of their analysis can be seen in Table 5.4. The results are the order of magnitude correct but vary widely from the intended goals.

Table 5.3: Seed Beams

Beam	Material	Boundary Condition	Width (m)	Height (m)	Length (m)
B1	Steel	Cantilever	0.5	0.6	3.5
B2	Aluminum	Simply Supported	0.1	0.7	6.0

Table 5.4: Seed Beam Results

Beam	Deflection (m)	Deflection ($\Delta\%$)	Yield Load (N)	Yield Load ($\Delta\%$)
Design Goal	5.0000E-02	N/A	5.000E+06	N/A
B1	2.1662E-02	56.68	7.4571E+06	49.14
B2	7.5164E-02	50.33	5.4771E+06	9.54

5.2.4 Results

Since the purpose of the beam problem is not to design beams that will be constructed for use but rather demonstrate the methods of developed in preceding chapters, the results will be examined with a focus primarily on the process rather than products. First, the raw convergence of the network for each major iteration will be plotted. Next, the final network structure will be examined. Finally, the raw results will be compared.

5.2.4.1 Convergence

Recall for this problem, a fixed number of major iterations, 15, were selected to ensure the problem is converged for more than just the single MPE sample taken. Because of the limited number of design parameters for this beam problem, all of the parameters may be tractably plotted together. At each iteration, six plots are made. These include three limit load plots and three deflection plots. For the limit load and deflection, the target is plotted as a red dashed line with the sampling range in grey dashed bars. Proper sampling should fall between the two grey bars, as this is the sampling range. Since there are four discrete choices, two boundary conditions with two materials, each can be colored uniquely. Seen is a simply supported steel beam in red, a simply supported aluminum beam in yellow, a cantilever steel beam in black, and finally, a cantilever aluminum beam in green.

As one examines each iteration, there are a few things to be mindful of. First, after several iterations, there should be no bias to any specific discrete configuration; with only two seed beams, those samples should not weight strongly. Second, the sampling should fall closer within the grey bars of the sampling range as the network is better trained. Third, after the final iteration, all configurations should be nearly equally likely. Finally, because the solution to this problem is undoubtedly non-unique, there should be a wide swath of viable (or nearly viable) designs.

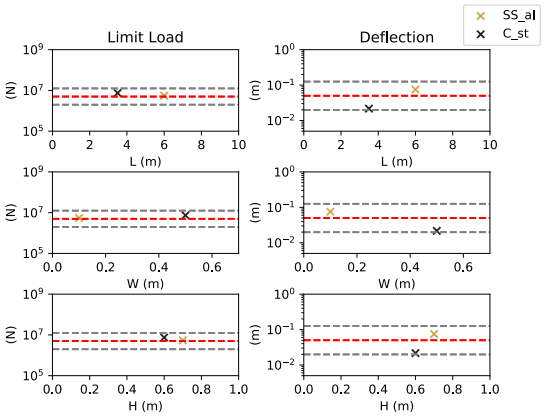


Figure 5.3: Initial Beams

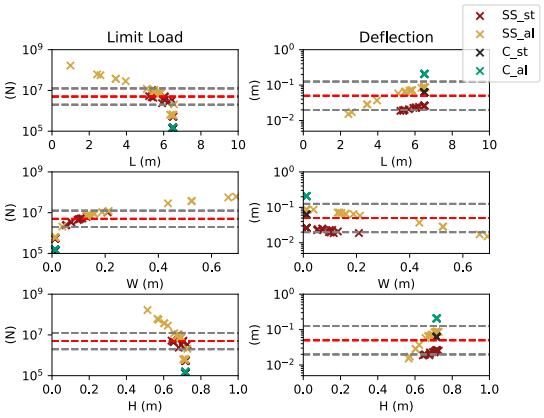


Figure 5.4: Beam Case: Iteration 1

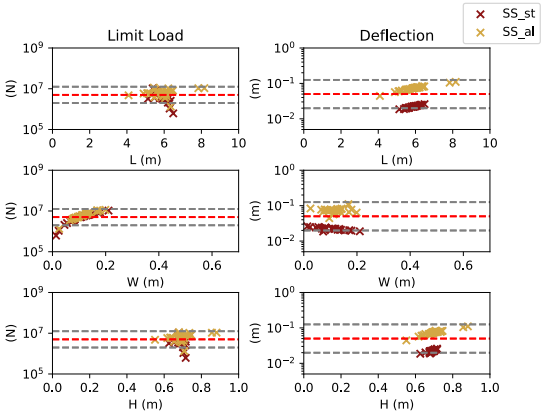


Figure 5.5: Beam Case: Iteration 2

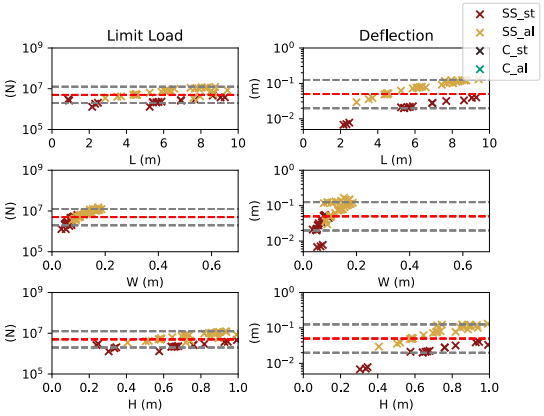


Figure 5.6: Beam Case: Iteration 3

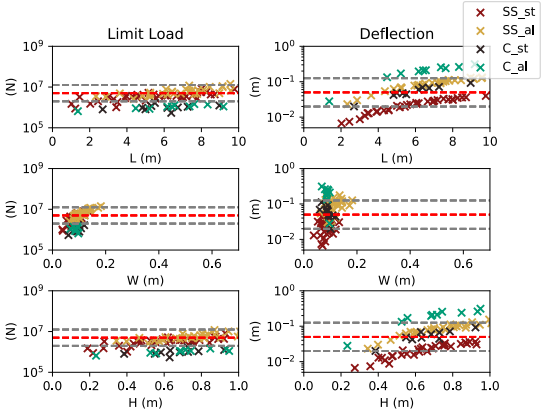


Figure 5.7: Beam Case: Iteration 4

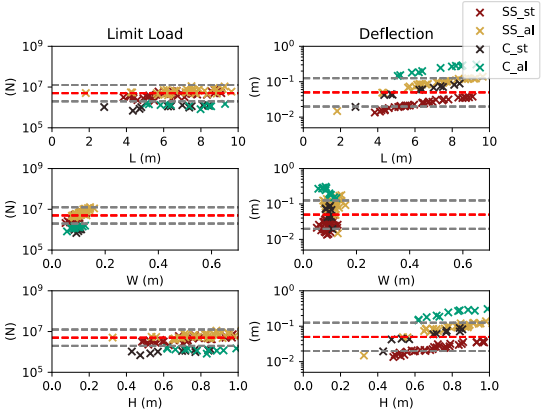


Figure 5.8: Beam Case: Iteration 5

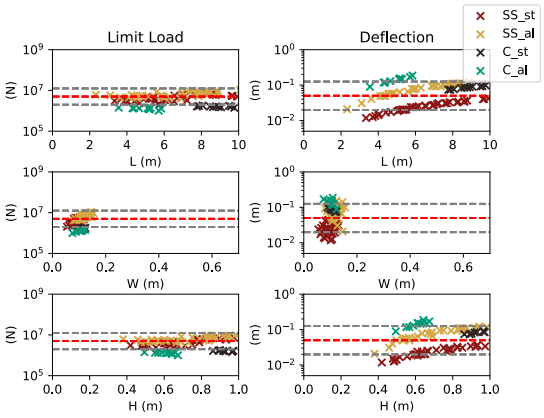


Figure 5.9: Beam Case: Iteration 6

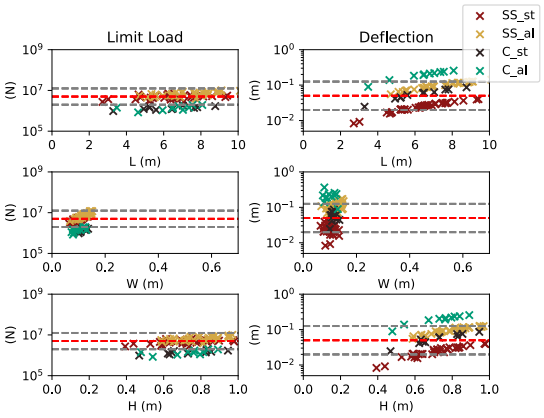


Figure 5.10: Beam Case: Iteration 7

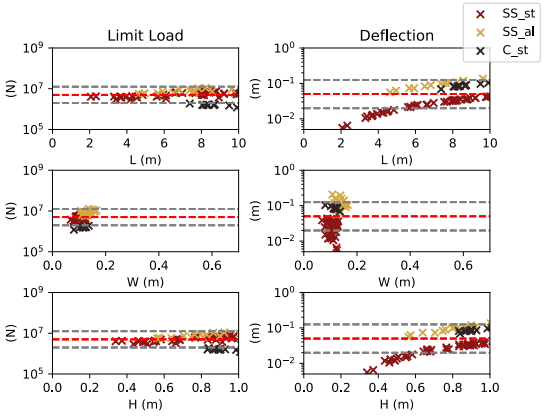


Figure 5.11: Beam Case: Iteration 8

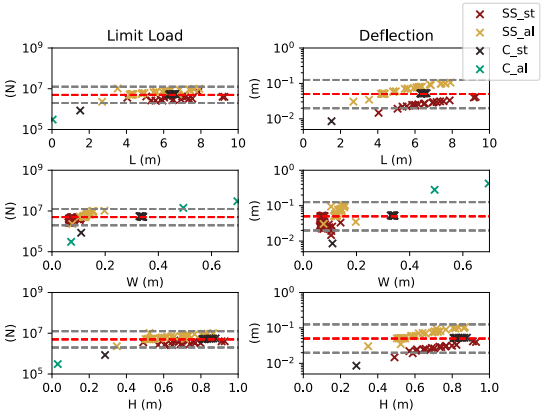


Figure 5.12: Beam Case: Iteration 9

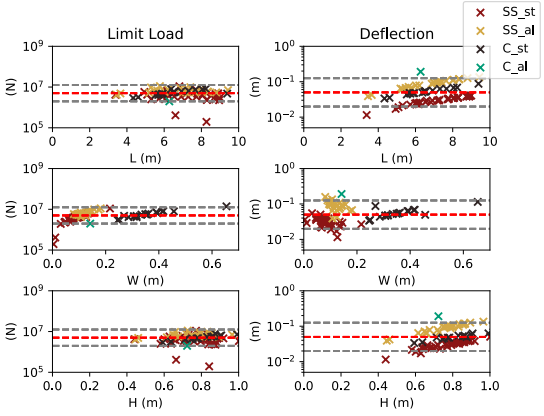


Figure 5.13: Beam Case: Iteration 10

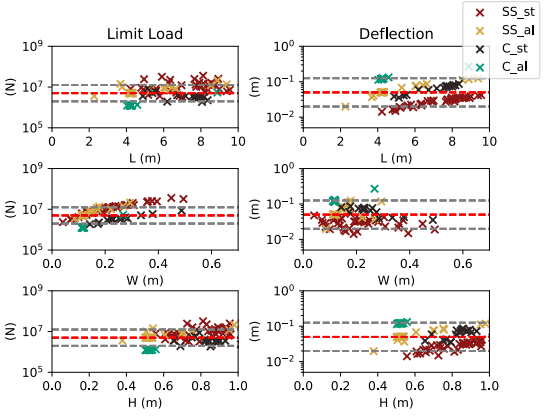


Figure 5.14: Beam Case: Iteration 11

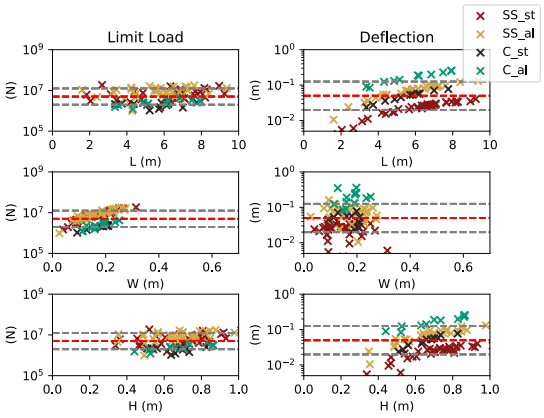


Figure 5.15: Beam Case: Iteration 12

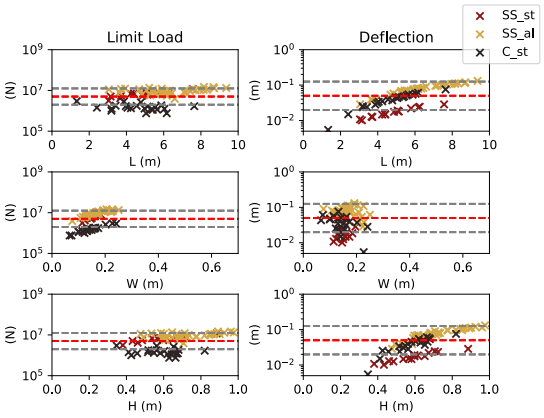


Figure 5.16: Beam Case: Iteration 13

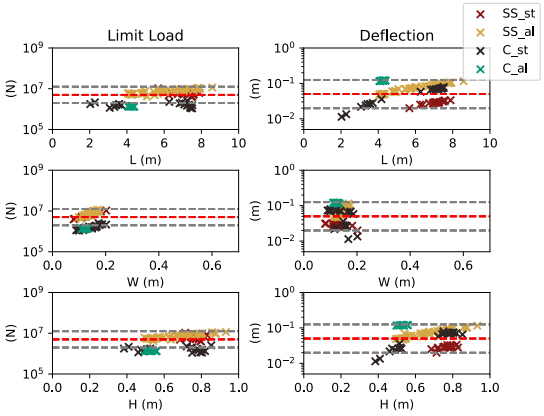


Figure 5.17: Beam Case: Iteration 14

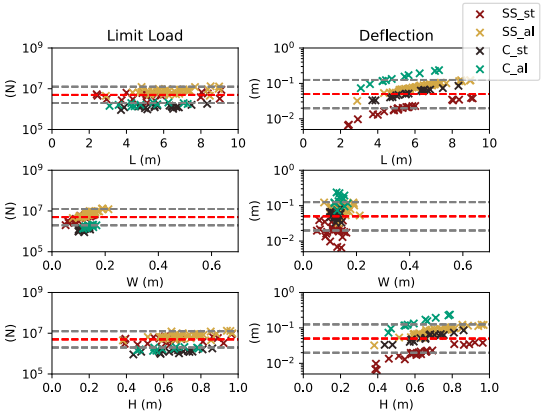


Figure 5.18: Beam Case: Iteration 15

Several things may be observed after a review of the iterations. First, is that not all configurations appear at every iteration. Because the tolerance in the architecture is 0, this indicates that non-realizable beam dimensions are being sampled. Second, in some iterations, one particular configuration will dominate the sampling process, verifying our hypothesis that only sampling without amplifying will skew the results and learning. Finally, and perhaps the most significant takeaway is the network does not "get stuck," or fixate, on any particular configuration. At the end of the iteration process, all configurations are sampled thoroughly and represented. This comes with a caveat; it does show that the convergence process can, in a sense, diverge. That is to say; the next major iteration can do worse than a prior iteration. This is seen in the 13^{th} major iteration, where a configuration is no longer represented. However, by the 15^{th} major iteration, this has recovered.

5.2.4.2 Learned Network

Recall when configuring the architecture, the second sampling blocks had integrated graph structure searches. The algorithm for structure searches runs for a fixed number of samples. With ten searches on each sampling iteration, five sampling iterations, and 15 major iterations require a total of 750 total structural search samples through the iteration process. A structure that would make sense to a designer is sought.

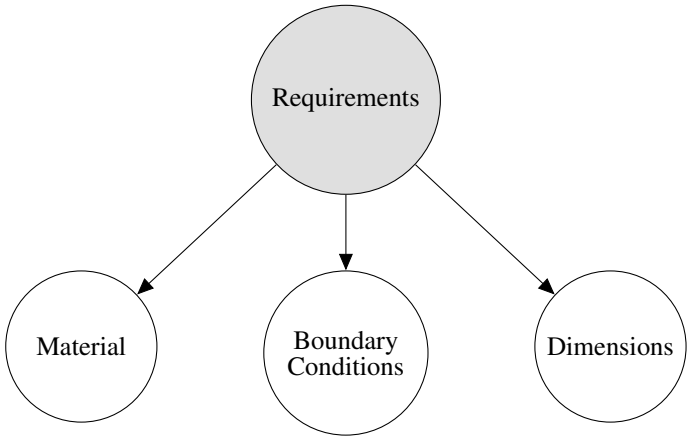


Figure 5.19: Initial Beam Network

First, examine the initial structure in Figure 5.19. The requirements feed three decisions: material, boundary conditions, and dimensions. The final network structure is seen in Figure 5.20. This structure results in a sequence of design decisions flowing from material, to boundary conditions, then from material and boundary conditions to dimensions.

The graph structure can be seen to be fully connected. This structure is similar to another type of generative Bayesian Network, known as an autoregressive Bayesian Network [62]. This name comes from time-series models where one state can help predict the next but is useful in other contexts too. Although mathematically, one could create an alternate equivalent equation that does the same as this network, by casting this problem as a generative Bayesian Network, one can represent this distribution more compactly. Representing the variables as nodes, scales linearly in complexity with variables, rather than exponentially with variables [62]. Exponential scaling in variables for a simple beam problem is tractable but becomes intractable for intracate designs such as an airplane. The fully connected equivalent equation for an aircraft would be extremely complicated.

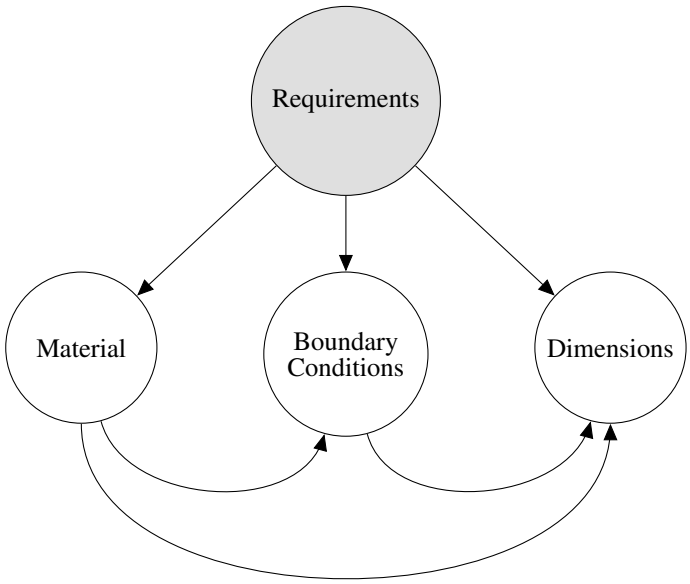


Figure 5.20: Learned Beam Network

5.2.4.3 Sampled Results

Finally, let us examine the MPE samples after the 15^{th} major iteration. This is done in two ways. First, the standard MPE sample is created to find the most likely single sample. Next, amplification is used with a tolerance of 0.0, which will create all discrete combinations. Then all four possible configurations with MPE continuous features are compared. The results are tabulated in Table 5.5. The single MPE sample, B, a steel simply supported beam, is boxed out. This design deviates in deflection by only 2.4% and in yield load by 13.4%. However, when examining the results in-depth, one can see that D, an aluminum simply supported beam, performs closer to the design goals than the MPE sample.

It seems counter-intuitive that sample D performs closer to the design goal than B. Let us dig into why this may be expected. First, this problem formulation does not lend particularly well to any specific configuration. Second, the large number of amplification data points taken means that the system should not be biased towards any configuration. For this specific case, a fully converged network would prescribe a 50% probability for both the material selection and boundary condition. Any single MPE sample taken could only be selected from numerical noise. Furthermore, a most probable explanation sample is not the maximum a posteriori, there is no expectation that this sample is, in fact, the sample that fits best (at least according to the network)! For further discussion

Table 5.5: Beam Results

Characteristic	A	B	C	D
Material	Steel	Steel	Aluminum	Aluminum
Boundary Condition	Cantilevered	Simply Supported	Cantilevered	Simply Supported
Length (m)	6.8030	11.4610	3.1166	4.2090
Width (m)	0.3717	0.0665	0.3898	0.1235
Height (m)	0.9537	1.1341	0.5298	0.5176
Deflection (m)	5.1487E-02	5.1201E-02	6.4306E-02	5.0025E-02
Deflection ($\Delta\%$)	2.97	2.40	28.61	0.05
Yield Load (N)	7.2063E+06	4.3285E+06	5.8874E+06	5.2732E+06
Yield Load ($\Delta\%$)	44.12	13.43	17.75	5.46

on MPE versus MAP and how they differ see Section 3.6.

5.2.5 Conclusion

In this section, a model problem using a simple beam design to test our generative Bayesian Network architecture was made. This beam design required a specific deflection and distributed load under the yield strength of the beam. The beam design was chosen for its simplicity, as well as inherent non-linearity. After 15 major iterations, convergence was examined. The final beam designs as generated from amplification were reasonably close to the design requirements. This serves an excellent example of how the MPE sample may not be the "best" sample. The value of the MPE is shown to be close enough, given the simplicity in calculating. This model problem has shown how the architecture used with the generative Bayesian Networks can be used to design.

5.3 Medium Range Airliner

5.3.1 Introduction

Realistic aircraft design problems can now be tackled having exhibited the applicability of generative Bayesian Networks on a simple beam. The first test case seeks to design something in which there is a known solution. The intention is to have this act as a simple cross-validation study. What is needed is a traditional design with a known existing solution for comparison. Of course, one would do a trivial test to ensure that an MPE sample would reproduce a given training data point. A problem is posed with ordinary mission requirements, yet different from the training data, to test

if the system generates a design in which a human designer would expect. The first example is a medium-range airliner based on the Airbus A220-100.

The Airbus A220-100 began life as the Bombardier CSeries before being purchased by Airbus [63]. The Airbus A220-100 is a modern design of a traditional configuration. It serves as an excellent test case here. The specifications are readily available, and the dimensions and other high-level parameters are known.

Although this serves as a more realistic test case, no optimization is performed on the vehicle. This case is not what would be done in practice for real aircraft design. Not optimizing serves several purposes. It is not known which, if any, objective function the designers of the A220 used, which would drastically alter the outcome. Having a raw aircraft before optimization provides us with a clear understanding of the performance of the network. Finally, this simplifies the problem and reduces computational time.

5.3.2 Design Requirements

The A220-100 is a medium-range airliner. It is larger than a typical regional jet, yet smaller than common narrow-body airliners such as the Airbus A320. The range of the A220 is also longer than typical regional jets. This aircraft is well suited to long, thin routes. A three-view drawing of the original CSeries before re-branding can be seen in Figure 5.21 [64].

Table 5.6: Medium-Range Airliner Requirements

	Airbus A220-100	Convergence Range
Range (nm)	2760	$\pm$ 300
Passenger Count (1 Class)	120	$\pm$ 8
Cruise Speed ($KTAS$)	447	$\pm$ 0
Static Margin (%)	52.0	$\pm$ 7
ToC Throttle (%)	90.0	$\pm$ 10
ToC Coefficient of Lift	0.4	$\pm$ 0.1

The design case references the data provided in the Airport Planning Publication (APP) [65]. Since a typical airliner can be loaded in a variety of ways depending on the mission range, the design point on the provided payload-range diagram is used. For this, the aircraft is loaded to the maximum takeoff weight, MTOW, with 120 passengers for a still air cruise range of 2750 nautical miles. The requirements for the A220 are summarized in Table 5.6. ToC denotes Top of Climb or the beginning of the cruise segment of the flight.

Figure 5.21: Bombardier/Airbus A220 [64]

Not all of these values that are provided for the A220 are publicly available. Therefore, for static margin, throttle, and coefficient of lift, a combination of expert judgment and similar vehicle results are used. This makes the concept of a convergence range much more valuable, as there are uncertainties in the design requirements; there is no pressing need to force exact design results.

5.3.3 Seed Designs

Recall from Chapter 3, the need to begin with a Seed Design Database. For this design case, a specially curated database of seed designs is used. This database includes aircraft models that have undergone extensive verification and validation exercises previously [32, 51, 52]. The vehicles can be seen in Figure 5.22. The Airbus A220-100 falls in size between the Embraer E-190AR (third from the left) and the Boeing 737-800 (far right). However, there is added a Boeing 777-200LR model that is also a conventional tube and wing aircraft. An Aérospatiale/BAC Concorde and the Boeing BWB-450 were included in the set of seed designs to make things more interesting. These two unconventional vehicle designs are used to test the algorithms. One would presume a well-trained network would produce a design more similar to the Embraer E-190AR and the Boeing 737-800; it would be concerning if the network architecture suggests a design like the Concord.

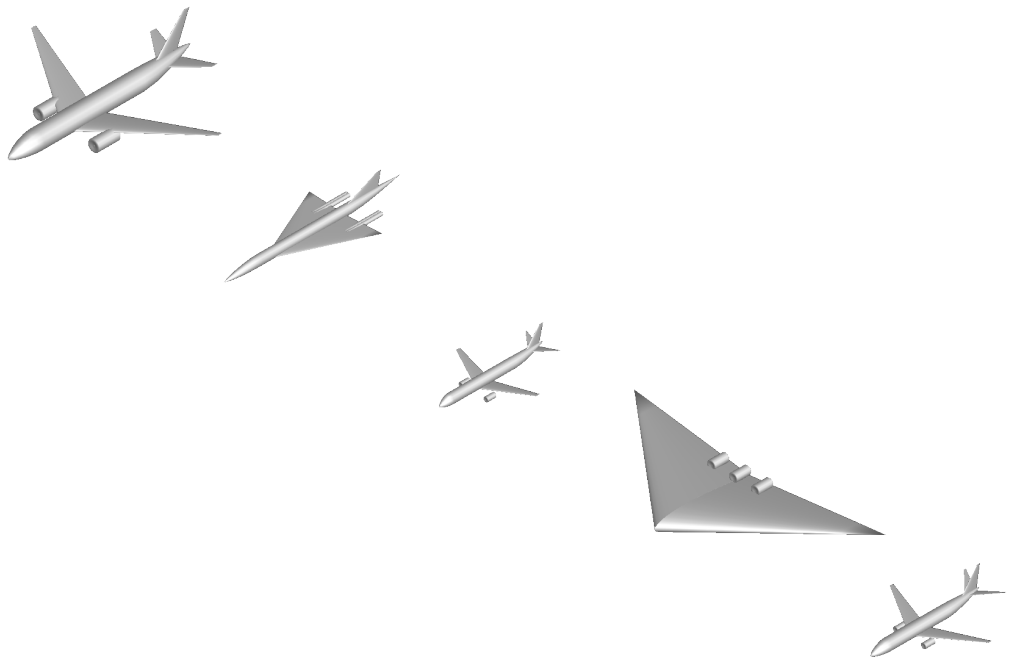


Figure 5.22: All Seed Vehicles with Trapezoidal Wings

5.3.4 Analysis

This test case will utilize SUAVE as the analysis. The analysis is set up in such a way that it is computationally quick yet comprehensive enough for conceptual design. There is always a trade-off between the accuracy of a result and time for computation. Indeed, further parameters could have been used for the specifications. After experimenting, it was found that these inputs yield adequate

results for conceptual design. Further refinement of the produced designs could easily be performed using an optimizer.

The nature of the sampling blocks and the amplification blocks lend well to parallelization. The entire SUAVE analysis is batched using Python's multiprocessing pool functions to parallelize. All of the generated designs are combined into a list for analysis. The result is another list of aircraft post-sizing with their analyzed results.

Each aircraft is first sized. A set of routines are used to ensure values are between preset ranges set by the designer to be physical; for example, any wing aspect ratio must be greater than 0. Next, geometric relations are used to finalize the dimensions of the vehicle. Many of these relations are simple and are necessary to complete the data structure for analysis. For example, given a specified quarter chord sweep, one will calculate the leading edge sweep for analysis.

Several assumptions are made based on regulatory requirements and technology levels. The first being that the aircraft complies with FAA Part 25 regulations as this sets the design loads for the structures. Next, for engine technologies, the same temperature and pressure ratio values that had been prescribed for a CFM-56 in prior publications [32] are used. Holding these values fixed, the engine is treated as a "rubber engine," meaning that the engine is re-scaled as each component is sized for a design sea-level static thrust and bypass ratio. This fixes a consistent level of technology between generated designs. Similarly, using the same concepts for afterburning turbojets based on data from the Concorde. When placing the engines, symmetry is imposed. If a network attempts to design an airplane with both turbofans and turbojets, both engine sets will be throttled the same, and the thrust will be treated as additive.

For analyses, a unified set of analyses that may work for an arbitrary vehicle type is necessary. An aerodynamics model is utilized, which includes the vortex lattice method outlined in Section 4.3.1.1. A unified set of equations will automatically switch for supersonic and subsonic flight, such that the same analyses can be used for all transport class vehicles. The weights methods are based on the derivations elaborated on in Section 4.3.2. A full set of stability methods is not used, only the static margin calculation from Section 4.3.4.

For the mission, takeoff and landing segment analyses are not used. While SUAVE is capable of analyzing a vehicle through the entirety of a mission profile, this simplification is made to save time. Takeoff and landing requirements could be met after setting a constraint to a post-processing optimizer. Instead, the aircraft is analyzed, assuming to cruise at 35,000 feet. Because takeoff and landing are not analyzed here, the lift coefficient in cruise helps to appropriate size the main wing. The lift coefficient is on the lower side, and higher lift coefficients could be used. In Section 5.3.2,

one may notice that the cruise speed is peculiar in that the convergence range is 0. This is due to the fact the cruise speed is fed forward to the mission directly. Design cruise speed is useful as metric in specifications as it corresponds with many design choices such as wing sweep. Furthermore, one could start running optimal segments, which would decide the cruise speed. However, optimal cruise segments are not done for simplicity. An additional 45-minute reserve segment is added to the mission at the cruise speed to comply with regulatory requirements. As mention in Section 4.2.1, the landing weight is used for the mission to calculate the range. The reserve segment does not receive range credit. Although this mission setup neglects the climb and descent, for many vehicles of longer-range, these segments account for only a small percentage of the total flight length and fuel burn. This is not a limitation of SUAVE or any of the methods shown within, merely a simplification.

Once the mission results are acquired, the results are pruned. If any value is infinite or Not a Number (NaN) in the vehicle, it is immediately discarded. It is possible to create engines from combinations of bypass ratio and sea-level static thrust, which are nonphysical, yielding NaN values. All vehicles which successfully converge the mission and do not have any suspect values contained within the data structure are added to the vehicle list, which becomes a database.

5.3.5 Setup

The architecture and hyperparameters used to generate results are provided here. In comparison to the earlier beam problem, it is not necessary to generate every possible configuration. The purpose of this exercise is to ensure the methods in this work return results that are to be expected, similar to the real Airbus A220-100. Therefore, for amplification steps, the tolerance value is increased from 0 to 0.0005. This tolerance is still a rather low value, enabling some design space exploration.

The architecture here is also different from the beam problem in one critical way; no longer is the architecture iterating for a fixed number of major iterations. Instead, the network utilizes the concepts in Section 3.5.1 to allow the network to converge. In contrast to the beam problem, this formulation, again, does not ensure that every possible feasible configuration is uncovered. However, for this case, as a test, it is beneficial. For an industry level design, it would be prudent to decrease amplification tolerances values and perhaps run for additional iterations post-convergence to widen the results. The configuration of the architecture is shown in Table 5.7. The vehicle database is purged after each major iteration and re-initialized at the next step with the seed vehicles. This purging is similar to the beam problem formulation.

Another critical difference between the prior beam problem and the setup shown in Table 5.7 is

Table 5.7: Medium-Range Architecture Setup

Amplification Block 1			
Tolerance	**MPE**	**Iterations**	**Searches**
0.005	TRUE	1	0
Amplification Block 2			
Tolerance	**MPE**	**Iterations**	**Searches**
0.005	FALSE	10	0
Sampling Block 1			
Iterations	**Samples Per Iteration**	**K Coefficient**	**Searches**
15	5	1.0	0
Sampling Block 2			
Iterations	**Samples Per Iteration**	**K Coefficient**	**Searches**
15	5	1.0	10

the use of the K coefficient. The K coefficient is set to 1, meaning that the range of the sampling search space is double the error in the prior MPE sample. The intention is to quickly converge to a solution rather than search the broader design space.

For the Bayesian Network itself, bounds must be set for the discrete options. There are more components in SUAVE, and, in fact, in the Bayesian Network, the primary components are listed in Table 5.8. It is crucial to consider the fact that a design may have both turbofans and turbojets. Furthermore, it may seem peculiar that the options are 0 or 1, how would one design a twin-engine vehicle? SUAVE handles multiple identical propulsors as a floating-point number as an architectural choice. The number of propulsors of identical propulsors is set in the Bayesian Network by a continuous parameter. There must be at least one Main Wing. This is a design choice, as jet-packs are not being considered here. With this setup, there are on the order of 10^7 discrete combinations. This is far more than the four discussed for the beam problem.

Table 5.8: Discrete Options

Component	Discrete Options
Turbofan	0 - 1
Turbojet	0 - 1
Horizontal Tail	0 - 6
Vertical Tail	0 - 3
Fuselage	0 - 5
Main Wing	1 - 4

5.3.6 Results

5.3.6.1 Convergence

In total, only one major iteration was required for convergence. However, the wall time for the entire test case was only 4 minutes on a 15-inch 2018 MacBook Pro with a 2.9 GHz 6-Core Intel Core i9 processor using 12 threads. A total of 160 aircraft evaluations were required. The Input versus Output comparison plots and final GBN structure are shown in Appendix A. The meanings of these plots are further explained in the Appendix.

5.3.6.2 Final Designs

After the convergence process, the final amplification process only created one viable configuration. Other configurations were created, but are unfeasible. This is due to the nature of the problem, as well as the settings used. Given that there are similar enough seed examples, and the K Coefficient was set to be low, little design configuration exploration occurred. Instead, the iteration focused on sizing to exact requirements. Unsurprisingly, the resulting configuration was a conventional tube and wing configuration with low pod-mounted wing engines. 3-View and left isometric images of the resulting vehicle can be seen in Figure 5.27.

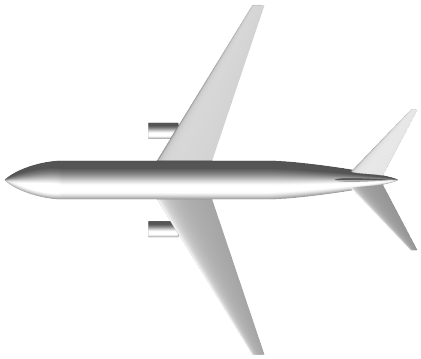


Figure 5.23: Top View

Figure 5.24: Left View

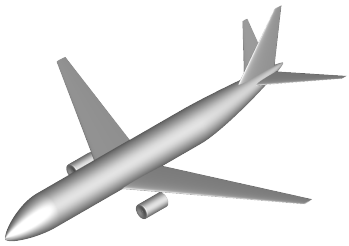


Figure 5.25: Left Isometric View

Figure 5.26: Front View

Figure 5.27: Medium-Range Airliner

The fact that the resulting design in Figure 5.27 is so unremarkable is remarkable in itself. Given the broad boundaries, the search space is quite extensive. Using SUAVE and data, this resulted in an airliner that most folks would not have known were not designed by a human. Every aspect

of the design is generated automatically, including the CAD, to create the images. This result, in a sense, not only verifies the architecture and GBN, it further shows the capabilities of SUAVE. Further comparison of the results compared to the real Airbus A220-100 can be seen in Table 5.9.

Table 5.9: Medium-Range Airliner Specifications

	Airbus A220-100 [65]	Generated Medium-Range Airliner
Range (nm)	2760	2523
Passenger Count (1 Class)	120	121
Static Margin (%)	N/A	50.0
ToC Throttle (%)	N/A	0.86
ToC Coefficient of Lift	N/A	0.398
MTOW (kg)	63,049	55,249
Fuselage Length (m)	34.9	36.47
Fuselage Width (m)	3.5	3.61
Wing Area (m^2)	112.3	110.3
Wingspan (m)	35.1	32.4
Wing Aspect Ratio	10.97	9.5
Total Sea-Level Static Thrust (kN)	207.2	150.0

The comparisons in Table 5.9 show items which are design parameters from the network explicitly, such as the aspect ratio, and calculated parameters, such as passenger count. The largest differences between the generated aircraft and the A220-100 lie in the wingspan, maximum takeoff weight (MTOW), and sea-level static thrust. Unlike a typical cross-validation study, it is not practical to examine an error criterion directly. This is for several reasons. One study cannot prove, without a doubt, the efficacy of the methods directly. An error needs a comparison to be of use; is this an acceptable error? Furthermore, there are inherent modeling limitations that prevent one from ever truly comparing.

One of the limitations is the fact that the Airbus A220-100 has a sister vehicle in the A220-300. The A220-100 and A220-300 have significant part commonality, including the same wing design. Since one wing design must support a larger aircraft design, in the A220-300, the smaller A220-100 has a slightly oversized wing. The real A220 is a compromise, and that would not be reflected in our design setup. The remainder of the dimensions is quite similar. With simple optimization or simple sizing, the range requirements could be met with additional fuel.

One should not focus on the A220-100 being *the* solution to these design requirements. Many

other considerations are taken when designing a vehicle beyond those used in this problem. Further considerations include manufacturability, costs, safety, aesthetics, and more. If one had a group of human designers, each would have their own opinions and ideal design. As such, there is no reason to believe one could, or even want to reproduce a vehicle precisely like the A220-100. There is nothing to indicate that this airplane represents the pinnacle of human design capabilities. This test case only serves as a check to ensure the final results create reasonable results in a reasonable amount of time. It exposes how one can use these methods to design a realizable aircraft.

5.4 Wing Planform Design

5.4.1 Introduction

After viewing the results of the last A220-100 design, there may be some wondering about the resulting design. It looks at first glance to be close, yet something is missing (besides the engine pylons). The wing design looks unrealistic. Real transport class aircraft have a much more complex wing shapes than a simple trapezoidal wing; most have a Yehudi, winglet, and other breaks in the wing design. Now, the same A220-100 design is examined but with adding the ability to attach up to 10 segments on each wing. In this case, the same seed aircraft are used, but they now have segments, as seen in Figure 5.28.

Adding wing segments seems like a small addition. Nevertheless, adding wing segments requires systems of systems. This test case illustrates for the first time how systems of systems can be designed. This case also shows how the Bayesian Network scales to more complex problems that yield further refinement in results. The wing segments can interact with other systems themselves to form ever deeper relationships.

5.4.2 Setup

Much of the setup and motivation from Section 5.3 holds the same for this analysis. However, a few changes are highlighted in this section. Each main wing now has the option between 1 to 10 segments, as shown in Figure 5.10. This simple change has made our problem significantly more complex, with the order of magnitude of discrete configurations now at 10^{14}. Unlike in the beam problem earlier, it is now impractical to observe one of each possible configuration through brute force.

The most substantial change made, seen in Table 5.11, is that the coefficient of lift requirement

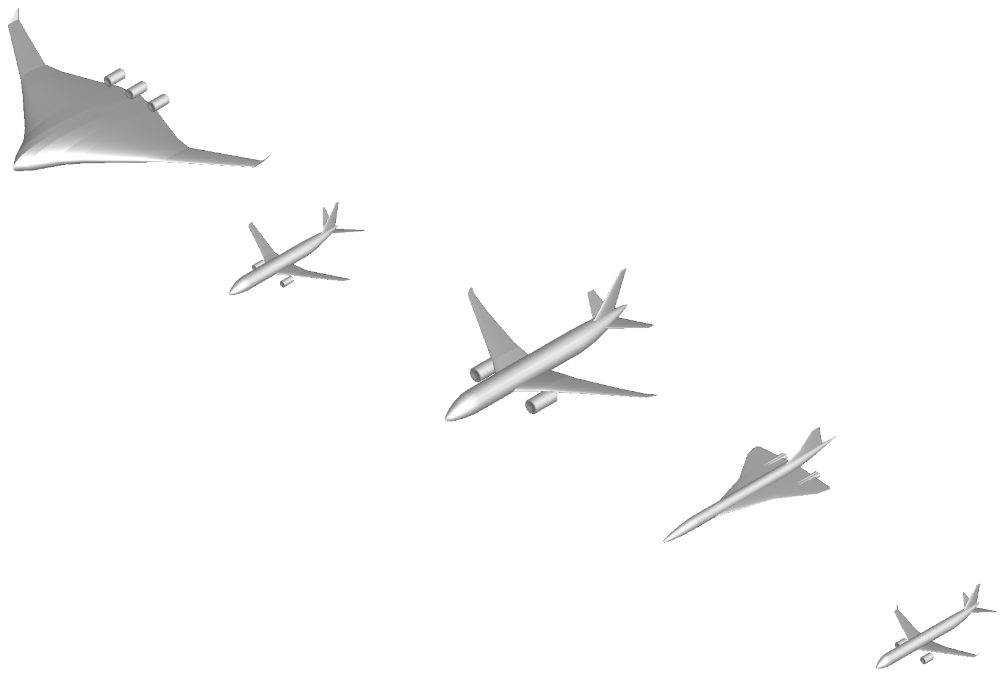


Figure 5.28: All Seed Vehicles with Segmented Wings

has been removed. Instead, for this case, aircraft with extremely large coefficients of lift at the beginning of cruise are discarded. For the convergence ranges, they are set to be 15% of the desired value. The architecture stays much the same as before from Section 5.3.

Table 5.10: Discrete Options for Wing Segments

Component	Discrete Options
Turbofan	0 - 1
Turbojet	0 - 1
Horizontal Tail	0 - 6
Vertical Tail	0 - 3
Fuselage	0 - 5
Main Wing	1 - 4
Segments	1 - 10

Table 5.11: Segmented Wing Medium-Range Airliner Requirements

	Airbus A220-100	Convergence Range
Range (nm)	2760	$\pm$ 415
Passenger Count (1 Class)	120	$\pm$ 18
Cruise Speed ($KTAS$)	447	$\pm$ 0
Static Margin (%)	58.8	$\pm$ 9
ToC Throttle (%)	90.0	$\pm$ 13.5

Table 5.12: Medium-Range Architecture

Amplification Block 1			
Tolerance	**MPE**	**Iterations**	**Searches**
0.005	TRUE	1	0
Amplification Block 2			
Tolerance	**MPE**	**Iterations**	**Searches**
0.005	FALSE	10	0
Sampling Block 1			
Iterations	**Samples Per Iteration**	**K Coefficient**	**Searches**
15	5	1.0	0
Sampling Block 2			
Iterations	**Samples Per Iteration**	**K Coefficient**	**Searches**
15	5	1.0	10

5.4.3 Results

5.4.3.1 Convergence

This case converged in five iterations using 80 minutes of wall time on the same laptop as the prior case. This case converged in far more than the one iteration and four minutes. This is despite having similar parameters as the prior simple wing case. The extra complexity of the wing design requires further iteration than the prior case.

A table of iterations and generated samples are shown in Table 5.13. Appendix B has further information, including each MPE sample taken at the end of each iteration. The MPE samples from each iteration show the configuration changes throughout the process. This includes adding a wing segment through the process.

Table 5.13: Segmented Wing Medium-Range Airliner Iterations

Iteration	1	2	3	4	5	Total
Total Generated Samples	10	15	21	45	130	221

5.4.3.2 Final Design

After convergence, only one viable design was created via amplification. The final vehicle is shown in Figure 5.33. The resulting configuration is as one would expect and at first, appears very similar to the real Airbus A220-100. A comparison of the generated results and the requirements are tabulated in Table 5.14. This vehicle has the same number of wing segments as the E-190AR and B737-800. However, the resulting shape of the wingtip is different. Therefore, the configuration is fundamentally the same, but the dimensions are different. However, as mentioned in Section 5.4.3.1, through the iteration process, the number of segments do changes. The GBN is attempting various configurations and finding the ones that work the best.

Table 5.14: Segmented Wing Medium-Range Airliner Results

	Range (NM)	Passengers	Cruise Speed (KTAS)	Static Margin	Throttle
Requirements	2760	120	470	58%	90%
Generated	2688	132	470	62%	79%

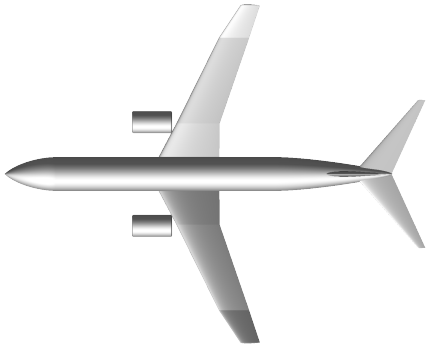


Figure 5.29: Top View

Figure 5.30: Left View

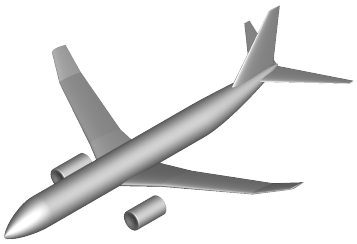


Figure 5.31: Left Isometric View

Figure 5.32: Front View

Figure 5.33: Medium-Range Airliner with Segmented Wings

The dimensions of the real Airbus A220-100 are compared against the two test cases in Table 5.15. One thing to keep in mind is that the convergence bounds between the two test cases, the segmented wing airliner fuselage has differing dimensions due to changes in the fineness ratio,

Table 5.15: Segmented Wing Medium-Range Airliner Final Specifications

	Airbus A220-100	Generated Medium-Range Airliner	Generated Medium-Range Airliner with Segments
Range (nm)	2760	2523	2688
Passenger Count (1 Class)	120	121	132
ToC Throttle (%)	N/A	0.86	0.78
MTOW (kg)	63,049	55,249	65,233
Fuselage Length (m)	34.9	36.5	36.6
Fuselage Width (m)	3.5	3.6	3.2
Wing Area (m^2)	112.3	110.3	122.3
Wingspan (m)	35.1	32.4	31.8
Wing Aspect Ratio	10.97	9.5	8.27
Total Sea Level Static Thrust (kN)	207.2	150.0	310.0

accounting for an additional 11 passengers. The engine thrust for the segmented wing airliner is significantly higher as should be expected since the top of climb throttle is 79%. However, the remainder of the results is in line with the prior case and the real Airbus A220-100.

5.5 Kangaroo Airliner

5.5.1 Introduction

One of the most storied airline routes of all time is known as the Kangaroo Route. Traveling over 9,700 nautical miles from London, England to Sydney, Australia, this route has been an unattainable nonstop voyage aboard an airliner throughout history [66]. Early attempts of serving this route via air required multiple stops and, even today, it still requires at least one stop. Only two flights on an airliner have completed the voyage. The first to complete the flight was a 747 without passengers or cargo, specially formulated fuel, and it did not even taxi under its own power [67]. More recently, a 787-9 completed the flight with a small contingent of journalists under strong tailwinds to gather data [68]. Completing this flight nonstop under realistic conditions would connect the world in a way that generations of passengers have only dreamed of, saving both airlines and passengers time and money. The economics for Qantas is undeniable, as just this single route accounts for 13% of their capacity [66].

As of this writing, Qantas is seeking proposals from major manufacturers to design and build an aircraft to serve the Kangaroo Route [69]. Boeing is planning on offering a variant of the new 777-X, while Airbus is offering an A350 variant. Neither airframer wants to make an aircraft from the ground up, and would rather modify an existing one. The resistance to design a new airframe leaves Airbus short on range capabilities to serve London to Sydney [70]. Furthermore, uncertainties in weather may limit the aircraft's usage due to flight cancellations, not an ideal scenario. There is demand, yet none of the manufacturers are willing to create a new airliner that will completely satisfy this design need. Manufacturers are worried that designing an aircraft for one airline is a risky proposition. Hence, the proposals fail to meet Qantas's requirements exactly.

Here, a clean sheet aircraft able to meet or exceed Qantas's needs is designed. Having multiple customers is always better than just one for financial stability. This airplane goes beyond... more range. This mission requires an aircraft with a range of 11,000 nautical miles. This range is half of the circumference of the Earth. There should be no city pair in which cannot be served. This range opens up routes beyond just the Kangaroo to city pairs airlines never dreamed of connecting, further spurring demand and increasing reliability in case of inclement weather or geopolitical diversions.

5.5.2 Design Requirements

This ultra-long-range aircraft seeks to fulfill the requirements of Qantas with excess range. Although, as discussed in Section 1.1, this is saturation for Qantas's range, it does expand the market possibilities. Starting from a clean sheet allows the market to exactly specify the design requirements. The requirements are shown in Table 5.16.

Table 5.16: Kangaroo Airliner Requirements

	Requirements	Convergence Bounds
Range (nm)	11000	$\pm$ 700
Passenger Count (1 Class)	300	$\pm$ 15
Cruise Speed ($KTAS$)	500	$\pm$ 0
Static Margin (%)	40	$\pm$ 10
ToC Throttle (%)	90.0	$\pm$ 10
ToC Coefficient of Lift	0.4	$\pm$ 0.1

The range is 11,000 nautical miles with a convergence bound of 700 nautical miles. The additional 45-minute reserve is also added. The cruise speed is set to 500 knots (Mach 0.87); this provides exactly 22 hours in flight. From the author's perspective, any more time on-board would be torture. The passenger count is set to be 300 passengers as Qantas required. The remainder of the parameters is set similar to other cases and for roundness. The setup is fundamentally the same as the two prior test cases, including having segmented wings and the same seed aircraft.

5.5.3 Results

5.5.3.1 Convergence

The Kangaroo airliner converged to a feasible solution in five major iterations. The number of converged samples at each iteration are shown in Table 5.17. In total, 698 total vehicle evaluations were necessary for convergence. Convergence required 34 minutes of wall time on the same laptop as the prior cases. For a vehicle with less applicable designs than the A220-100, this is considered a success in having converged so quickly.

Table 5.17: Kangaroo Airliner Iterations

Iteration	1	2	3	4	5	Total
Aircraft	91	147	144	159	157	698

5.5.3.2 Final Designs

From the post-convergence amplification block, two vehicles were created. These vehicles can be seen in Figures 5.38 and 5.43. The first sample, the MPE sample, is a conventional tube and wing aircraft. Significantly, it has long raked wingtips. Raked wingtips like these are a feature not seen in the design database. Although the study of raked wingtips extends as far back as 1921 [71], these are becoming prevalent on longer-range aircraft such as the Boeing 787 [72]. Aircraft designs with a variety of wingtip geometries have been studied for years [73, 74].

The second viable amplification vehicle has sharp wingtips. When comparing the dimensions and performance of the two vehicles in Table 5.18, the two are quite similar. Despite differing mainly in the wingtip geometry, the second vehicle has a range that is almost 2000 nautical miles less. Some of this can be attributed to the reduced maximum takeoff weight reducing the possible fuel load. Likely the majority of the reduction in the range can be contributed to the wingtip. The throttle setting is at 144% (more power than is available) for an aircraft with the same installed sea level static thrust, indicating that more thrust is required to fly this aircraft, resulting from higher drag.

Table 5.18: Kangaroo Airliner Specifications

	Most Probable Explanation	Amplification Alternative
Range (nm)	10560	8692
Passenger Count (1 Class)	305	305
Static Margin (%)	35.0	33.2
ToC Throttle (%)	91.9	144.0
ToC Coefficient of Lift	0.40	0.38
MTOW (kg)	307,990	304,364
Fuselage Length (m)	56.83	56.83
Fuselage Width (m)	5.49	5.49
Wing Area (m^2)	284.1 0	262.4
Wingspan (m)	57.43	57.43
Wing Aspect Ratio	11.61	12.57
Total Sea Level Static Thrust (kN)	1018.2	1018.2

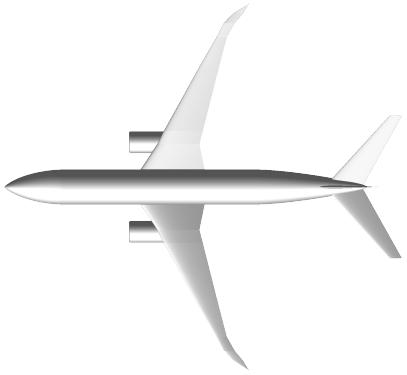


Figure 5.34: Top View

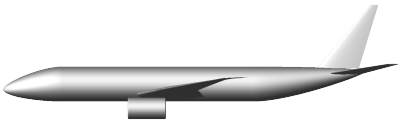


Figure 5.35: Left View

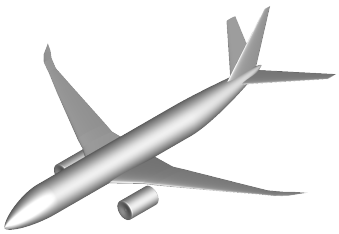


Figure 5.36: Left Isometric View

Figure 5.37: Front View

Figure 5.38: Kangaroo Airliner MPE Sample

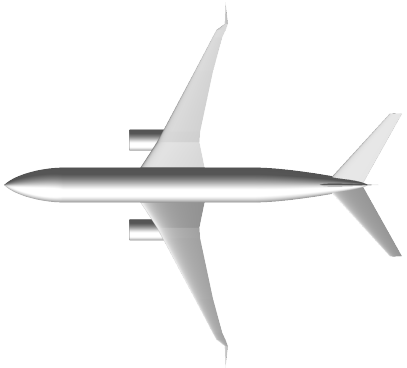


Figure 5.39: Top View

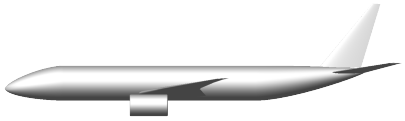


Figure 5.40: Left View

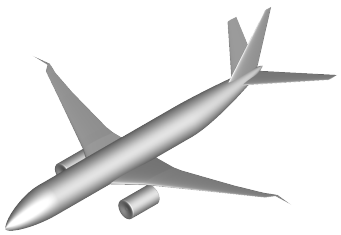


Figure 5.41: Left Isometric View

Figure 5.42: Front View

Figure 5.43: Kangaroo Airliner Amplification Alternative

5.5.4 Further Studies

This case was repeated multiple times to understand the behavior of this system further. Moreover, owing to the ability to converge a solution quickly, it is in the human designer's best interest

to try variations for further consideration. Of particular note is that although many of the alternative configurations seem to be similar at first glance, in fact, they represent discrete alternatives. They are traditional in the aircraft designs with tube and wing configurations but have differences that are worth examining.

Built into the code was the ability to seed the pseudorandom number generators. This work relies heavily on pseudorandom number generators. Recall, from Section 2.5.1.1, that discrete CPD weights are initialized randomly before optimization, from Section 2.5.2.2, that the structure search will randomly select both edges and operations. Finally, both sampling and amplification rely heavily on random sampling. Changing these random seeds may have a drastic effect on convergence.

This Kangaroo case made it far more apparent the effects of seeding. The seed aircraft are not directly applicable as in the Medium Range Aircraft Case. Therefore, there is far more variability as the feedback process from analysis driven by analysis. The convergence history, as well as the generated aircraft, vary. However, this allows us to find some impressive results.

5.5.4.1 Additional Configurations

An array of some of the different possible configurations is presented in Figure 5.48. The first configuration has elegantly sweeping wings using the maximum number of segments to obtain the continuously sweeping leading edge. Of these additional designs presented, this is the most practical.

The next design has four engines and an enormous horizontal tail. This configuration, most likely amplified for its large wing to support enough weight to fuel four engines, is impractical based on the highly swept horizontal tail. Further analysis and design would be required to determine critical design points such as stall characteristics, low-speed controllability, and aerodynamic flutter.

The aircraft in Figure 5.46 has three engines, with one under-slung in the middle. This trijet design also incorporates forward-swept wingtips. Concerningly, these wingtips may be susceptible to flutter.

The final design has double vertical tails with aft-mounted engines. This exotic configuration is similar to the MIT D8 with some key differences [75]. The engines of this configuration are placed outboard of the vertical tails instead of in between. The D8 is designed with boundary layer ingestion. However, the D8 places the engines in very close proximity, which is a potential issue in the case of an engine failure. This new design has the engines farther apart and separated by a vertical tail. Further analysis and design would be required, but this configuration may have the ability to combine the efficiencies of boundary layer ingestion with a higher level of safety.

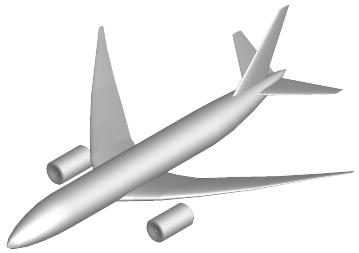


Figure 5.44: Alternative Configuration 1

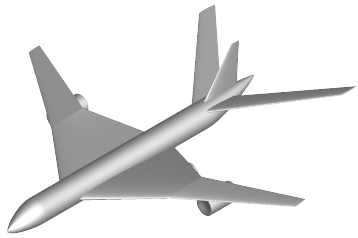


Figure 5.45: Alternative Configuration 2

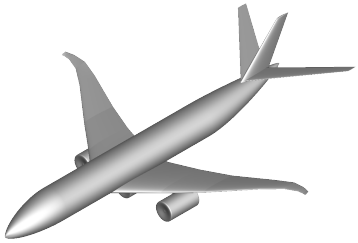


Figure 5.46: Alternative Configuration 3

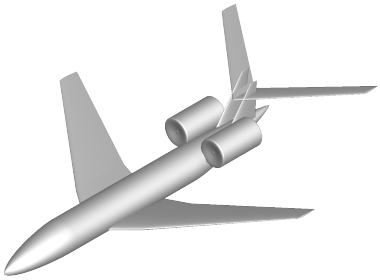


Figure 5.47: Alternative Configuration 4

Figure 5.48: Alternative Kangaroo Configurations

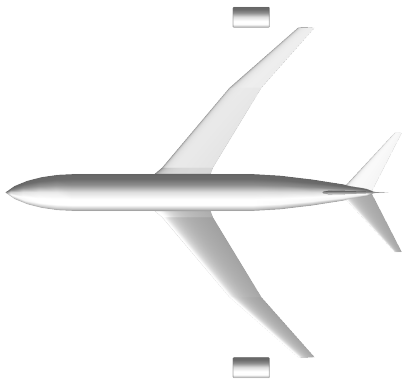


Figure 5.49: Longitudinally Unstable Aircraft Iteration

5.5.4.2 Static Stability

Through the course of iterating, one common problem which is easy to visualize is a negative static margin. A seasoned aircraft designer can look at a design and determine a design is unstable. In this case, the aircraft shown in Figure 5.49 is longitudinally unstable. However, at the next iteration, shown in Figure 5.50, illustrates an attempt at a stable aircraft.

While this vehicle has now satisfied the stability requirements, it has also done something unrealistically impractical. Nevertheless, it did solve the problem creatively. This example serves us in two ways: It illustrates to the designer a weakness in their analysis and provides further understanding. Rather than moving the engines further forward, or shifting the main wing, the system just added more horizontal stabilizers. Four horizontal stabilizers in this fashion are impractical. Nevertheless, there may be a design in which multiple horizontal tails are practical (perhaps not with four). Furthermore, the engine placement is questionable. In this case, the analysis did not capture engine-out performance; thus, the spanwise location of the engine has no feedback mechanism.

5.6 Conclusion

This chapter carefully built up a series of test cases to illustrate the use of a generative Bayesian Network for design cases. Four test cases were explored, each building upon the last for increasingly

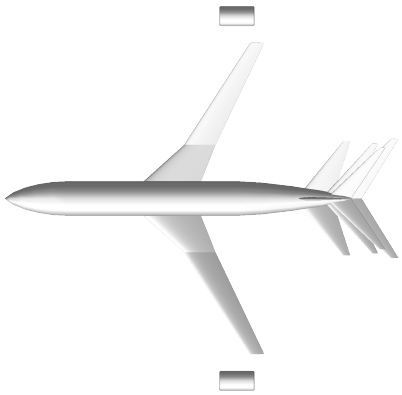


Figure 5.50: Next Iteration with four horizontal tails

more complex designs. The first design case was the Beam test case. Next, was a Medium Range Airliner in the A220-100 with simple trapezoidal wings. Subsequently, that case was re-examined with segmented wing designs. Finally, an ultra-long-range airliner for the Kangaroo capped off the results to produce a result never before seen.

The beam test case illustrated one of the most straightforward systems that could be designed. This case, while not useful in its setup, has all of the vital hallmarks of elaborate engineering systems designs. The analysis, while analytical and straightforward, is highly non-linear. A simple architecture was used, yet the results were excellent, given little data. This case demonstrated how limited data could be augmented through sampling and analysis. The configuration options yield wildly different results providing an array of options to the designer.

The next case, the medium-range airliner, was an initial verification of the aircraft design capabilities. By choosing an aircraft with known requirements, this case tests if the architecture and the analysis can reproduce designs similar to those in existence. As this is conceptual design, one cannot expect results to match exactly. However, the results yielded an aircraft similar in configuration and size to a real Airbus A220-100.

Further making the medium range airliner more realistic, segmented wing design was added to the setup. This seemingly small change resulted in creating a vast discrete configuration space to search. This case highlights the ability of the architecture to design systems of systems. The

resulting aircraft is very similar to an Airbus A220-100, more so than the simple wing design case.

The final test case captured an aircraft that does not exist today. This ultra-long-range airliner is unlike anything in our design database. This test case highlights the ability of the methods to design a vehicle that is challenging to a human designer. Recall the motivation of this work is to enable designers to devise things never before seen. The results are astonishing and beautiful in itself.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

At the heart of this thesis is conceptual aircraft design. At the beginning of this work, Chapter 1, described and motivated the meaning of design. Design is an iterative process where one creates to fulfill a need. Design is notably open-ended, resulting in potentially numerous feasible solutions. A new tool for human designers is established to address the future of aircraft design, where new technologies and mission requirements drastically alter the design landscape. This new tool is to provide the human designer with alternative feasible aircraft solutions they may not have considered themselves.

Bayesian Networks possess many inherent qualities that lend to design. Building off of fundamental theory in Bayesian Networks, a hybrid Bayesian Network is created, which is tailored to component design. This component network is driven by high-level design requirements and handles subcomponents. This Bayesian Network learns from existing design examples and provides mechanisms to "fill-in" gaps where there are no suitable examples. With a fully trained network, one can synthesize designs using two different methods: amplification and sampling.

Design, however, is an iterative process requiring creation and testing until one converges on a final solution. Therefore, much as a human designer would, the generative Bayesian Network is combined with analysis to "test" how the synthesized designs would perform. This architectiure creates a feedback loop that helps the generative Bayesian Network converge on a design or a set of designs in which can meet the requirements.

For the airplane examples here, SUAVE, a conceptual aircraft design analysis and optimization tool is used. From the overarching code structure, SUAVE is uniquely suited to handle new

technologies and incorporate multiple information sources. SUAVE is component-based, thus integrating well with the physical component structure of the generative Bayesian Network. SUAVE makes as few assumptions as possible by relying on physics rather than pure correlations, lending credibility to non-traditional aircraft design analysis.

Four different test cases are examined with various levels of complexity. Each case builds upon the last to add further realism. The first case, a beam that must support a specified load and deflection at the material yield stress, created four unique beams that closely meet the criteria. Next, a medium-range airliner with simple wing shapes is designed to compare against a real-world airplane in the Airbus A220-100. The generated aircraft is similarve to the existing Airbus A220-100. A more sophisticated version with segmented wing shapes is then completed. In a minimal amount of iterations, the architecture arrives at an aircraft, which is remarkably similar to the A220. Finally, demonstrating the capabilities of the architecture to create new concept design alternatives, the system was used to design an aircraft that does not exist but has a demonstrated industry need. Several ultra-long-range airplane designs, for the Kangaroo route, are created and can cover an unprecedented distance. However, these ultra-long-range aircraft are just the beginning; the future of this technology will continue.

6.2 Future work

This thesis intends to be useful in laying the groundwork for future research in engineering design. The methods and algorithms may change, yet the goals and ideas continue. With that in mind, there are ways to improve this work, establishing a new research direction.

The fundamentals of Bayesian Networks and Deep Learning are inextricably linked. One may have noticed that the discrete node behavior is nothing more than logistic regression. Moving beyond shallow networks, when data permits, will allow for richer representations of the underlying distribution. Deeper networks are particularly obvious for the conditionally linear Gaussian. A linear function is not an adequate characterization of the entirety of the design space, which limited the effectiveness of this generative Bayesian Network.

If the conditionally linear Gaussian was replaced and further improvements made in generality, one could use this for more than one set of design requirements. Currently, the architecture only trains for a single set of requirements. However, if the network could accurately represent a broader range of designs, one could use this effectively for multiple designs. This is useful because it would not require significant relearning when the requirements are changed.

Saving time when learning is an area that requires further research. SUAVE cannot currently handle gradients without finite differencing. If gradients could be passed from the analysis functions back through the Bayesian Network, one could more effectively train a deeper network. Further research is required into how best to link the two. However, automatic differentiation could be used effectively with SUAVE.

The Bayesian Network also extensively relies on Most Probable Explanation (MPE) samples. As explained in Section 3.6, this may not, in all circumstances, represent the best design. It would be worth time examining the Maximum a Posteriori (MAP) to consider the trade-offs in time and effectiveness. Related to this trade-off is the fact the convergence is based on a single MPE sample. The goal of this work is to elucidate new design options for the designer. Convergence should be based on having multiple feasible designs from an amplification step that can be compared. The medium-range airliner lends well to a single viable configuration, yet future designs have much more variability in design options.

This work should be expanded to new test cases. Although presented as conceptual design methodologies here, everything generalizes to preliminary and detail design. One day these algorithms could be used to refine designs further. For example, with an outer mold line established, the designer turns their attention to structural layouts in preliminary design, and yet again uses a GBN to draft the layout.

Finally, as was motivated in the introduction, these methods were made to design for a new frontier. Further use with actual engineering designers would provide enormous feedback. This feedback would lead the way in how to integrate these methods into the design cycle in practice best. Using this work to design eVTOL applications could enlighten the solution to our urban mobility problems.

Appendix A

Medium Range Airliner Convergence

This Appendix contains many of the plots and illustrations that show the progression of the designs not shown in Section 5.3 for concision. This appendix expands upon that to show how the vehicle converged through major iterations. The set of plots in Section A.1 compares the inputs versus the outputs. Section A.2 shows the resulting graph structure for the converged vehicle.

A.1 Input-Output Comparison

These input versus output plots help the designer understand the performance of the algorithm. Initially, the GBN is not well trained, a set of input requirements to the GBN does not produce results that satisfy those requirements. As the network converges, the inputs begin to match the outputs. A completely trained network should be a line with a slope of 1 and an intercept of 0 ($X = Y$). However, it is natural for scatter to occur, given that amplification and sampling are intended to induce learning. Amplification steps, in particular, are for exploration. Furthermore, the designs are only sampled within a range of Xs. A line that extends to the origin is never to be expected. Since, in this case, only one major iteration was necessary for convergence, a sloping trend of samples does not materialize.

The area of interest for these plots lies in a narrow region. The plots below contain an inset area that remains consistent for comparison. However, the main axes are allowed to scale to show all data points. One thing of note is that all data points are valuable: for example, a design that has four of the five criteria met is still valuable and perhaps requires a few tweaks to meet the requirements. In other words, when examining the plots, one cannot see the correlation between one requirement to another.

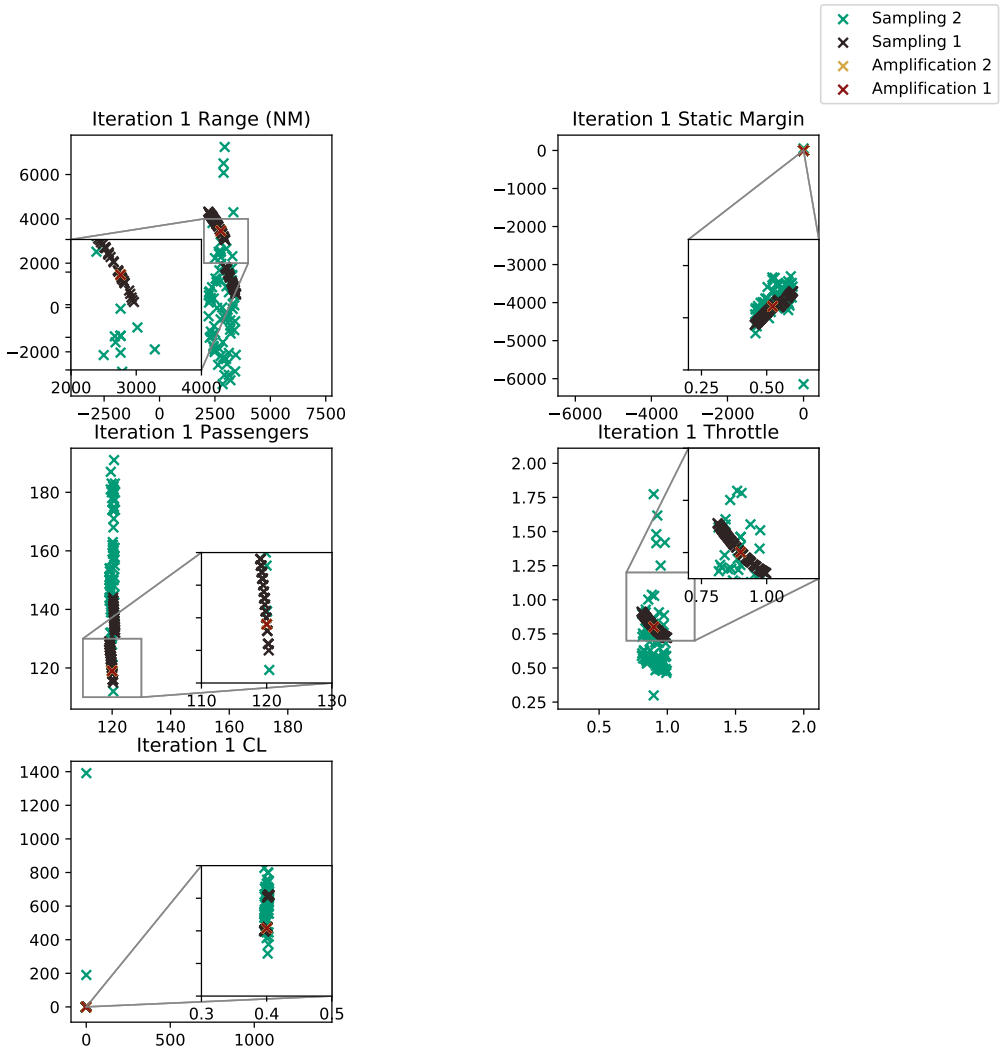


Figure A.1: Iteration 1: Final Iteration

A.2 Learned Graph Structure

The graph structure shown in Figure A.2 is raw, as it is generated from NetworkX. One interesting note is that every component is labeled as "class" at the beginning. These labels are an artifact of clever programming practices. Rather than name each node with a string, each node name is a link to an un-instantiated class. When querying a node, all information can be derived directly from

the name itself and each node exist independently of SUAVE. This ensures the architecture and the generative Bayesian Network are portable from SUAVE. They are not interwoven.

Of note is that the resulting network, while dense, is not fully connected; not every possible edge is added. Therefore, structure learning is behaving as intended and adding edges only as needed.

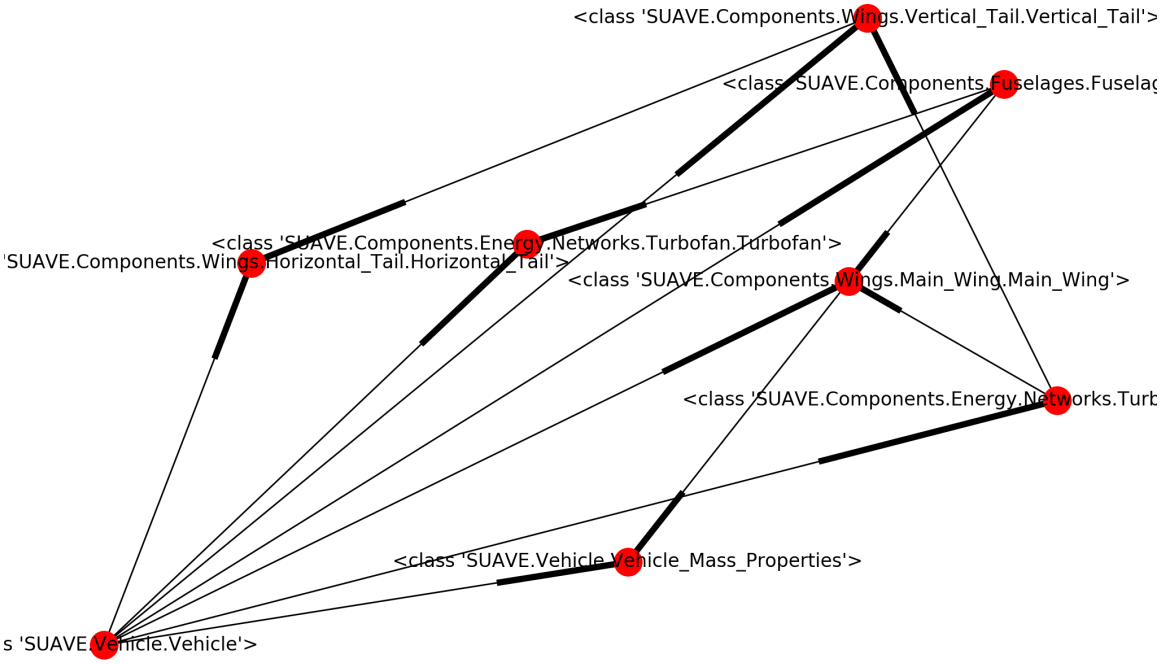


Figure A.2: Medium Range Airliner Learned Network

Appendix B

Wing Planform Design Convergence

Much like the prior appendix section, this section contains further data regarding the convergence of the test case in Section 5.4. First, in Appendix B.1, images of each MPE sample taken at the end of each iteration are shown. This gives a visualization of how the algorithms converge. Iteration 1 failed to produce a viable MPE sample; as such, no output is available. However, iterations two through five have vehicle illustrations.

Appendix B.2 contains all of the iteration plots for the inputs and outputs. For further information on how to interpret these plots, see Appendix A.1 for guidance. Finally, Appendix B.3 has the final graph structure after convergence.

B.1 MPE Samples

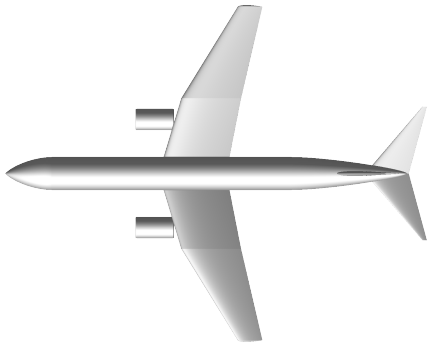


Figure B.1: Segmented Wing Medium-Range Airliner Top View

Figure B.2: Segmented Wing Medium-Range Airliner Front View

Figure B.3: Iteration 2

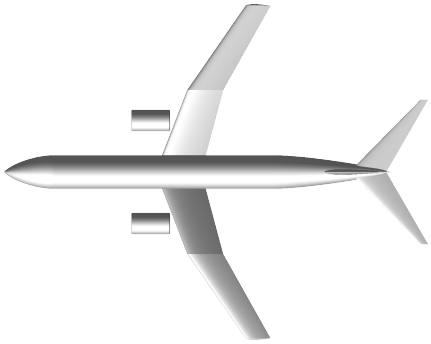


Figure B.4: Segmented Wing Medium-Range Airliner Top View

Figure B.5: Segmented Wing Medium-Range Airliner Left View

Figure B.6: Iteration 3

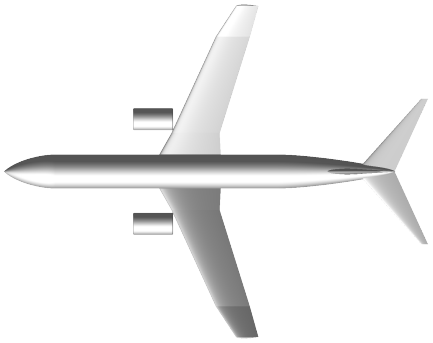


Figure B.7: Segmented Wing Medium-Range Airliner Top View

Figure B.8: Segmented Wing Medium-Range Airliner Left View

Figure B.9: Iteration 4

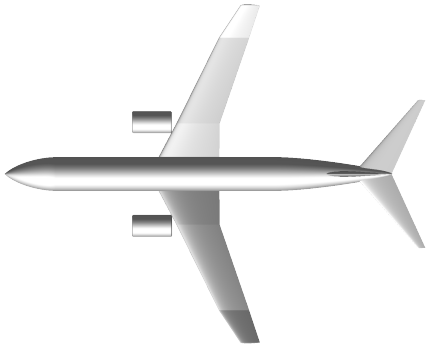


Figure B.10: Segmented Wing Medium-Range Airliner Top View

Figure B.11: Segmented Wing Medium-Range Airliner Left View

Figure B.12: Iteration 5

B.2 Input-Output Comparison

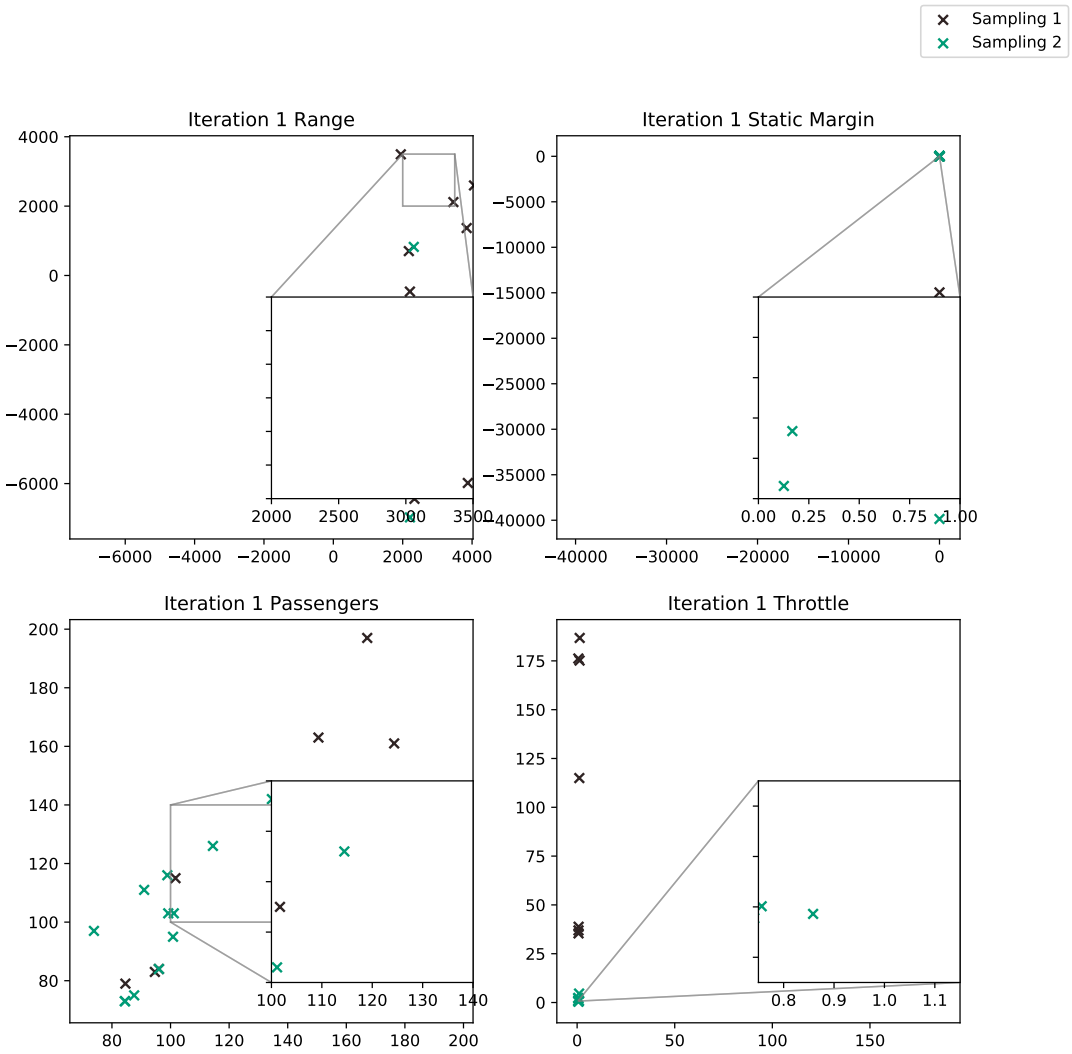


Figure B.13: Iteration 1

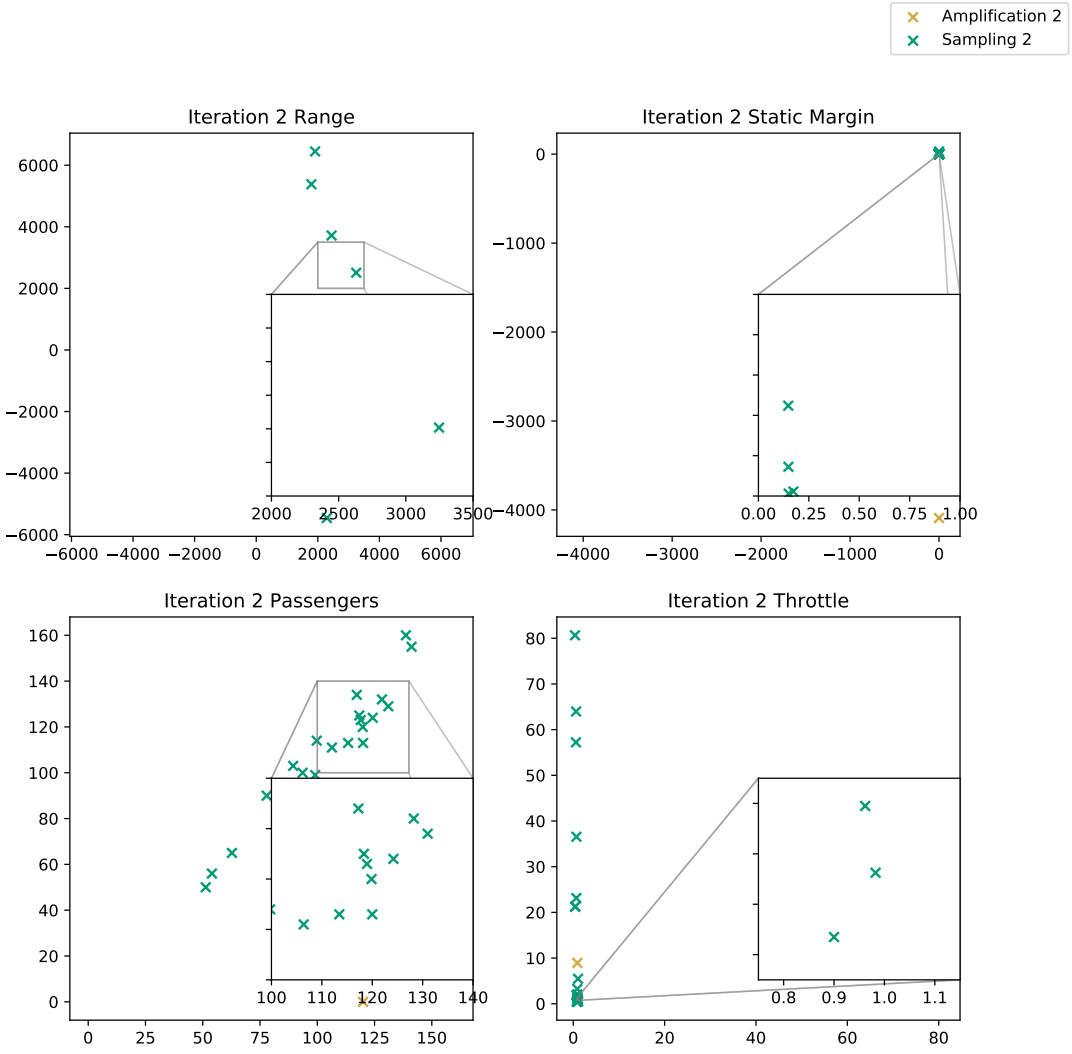


Figure B.14: Iteration 2

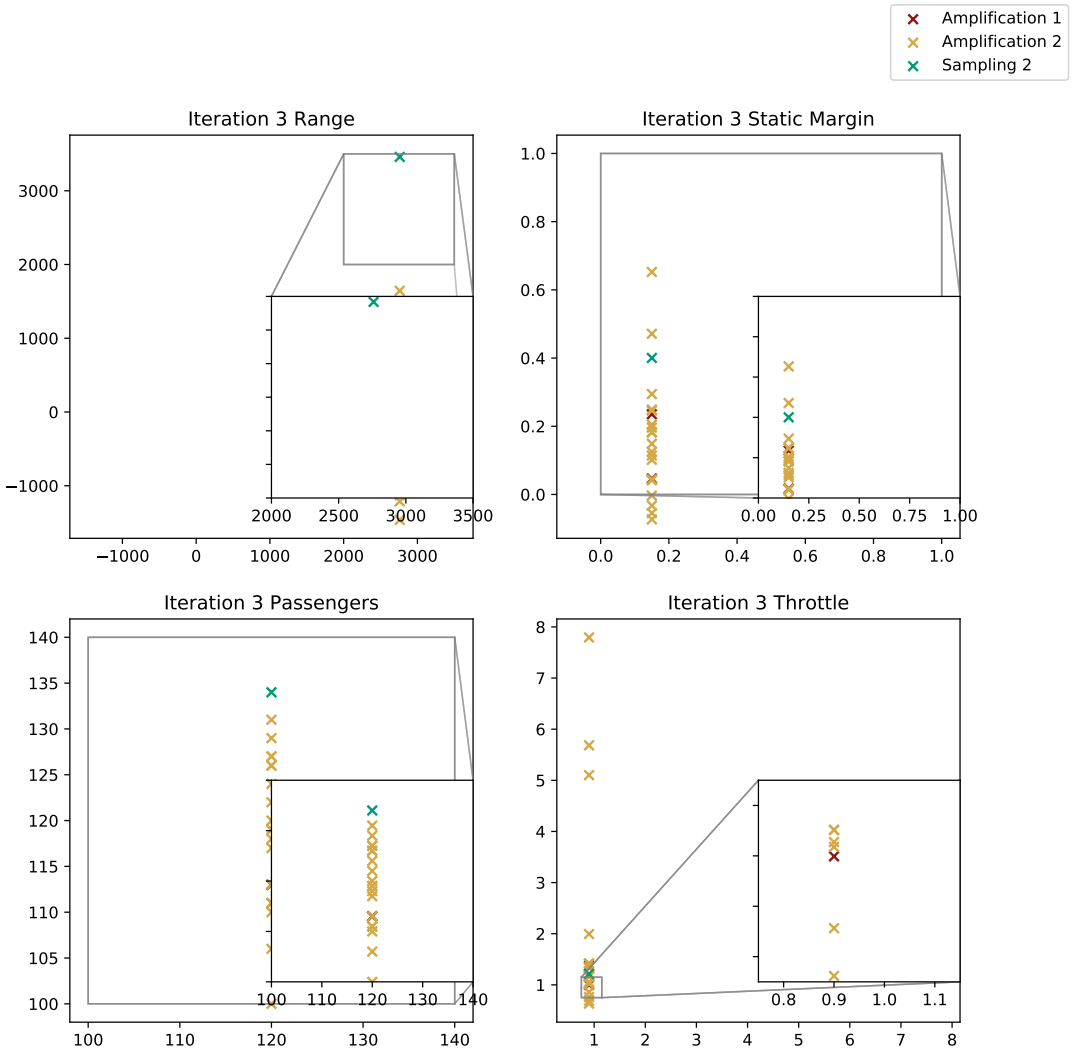


Figure B.15: Iteration 3

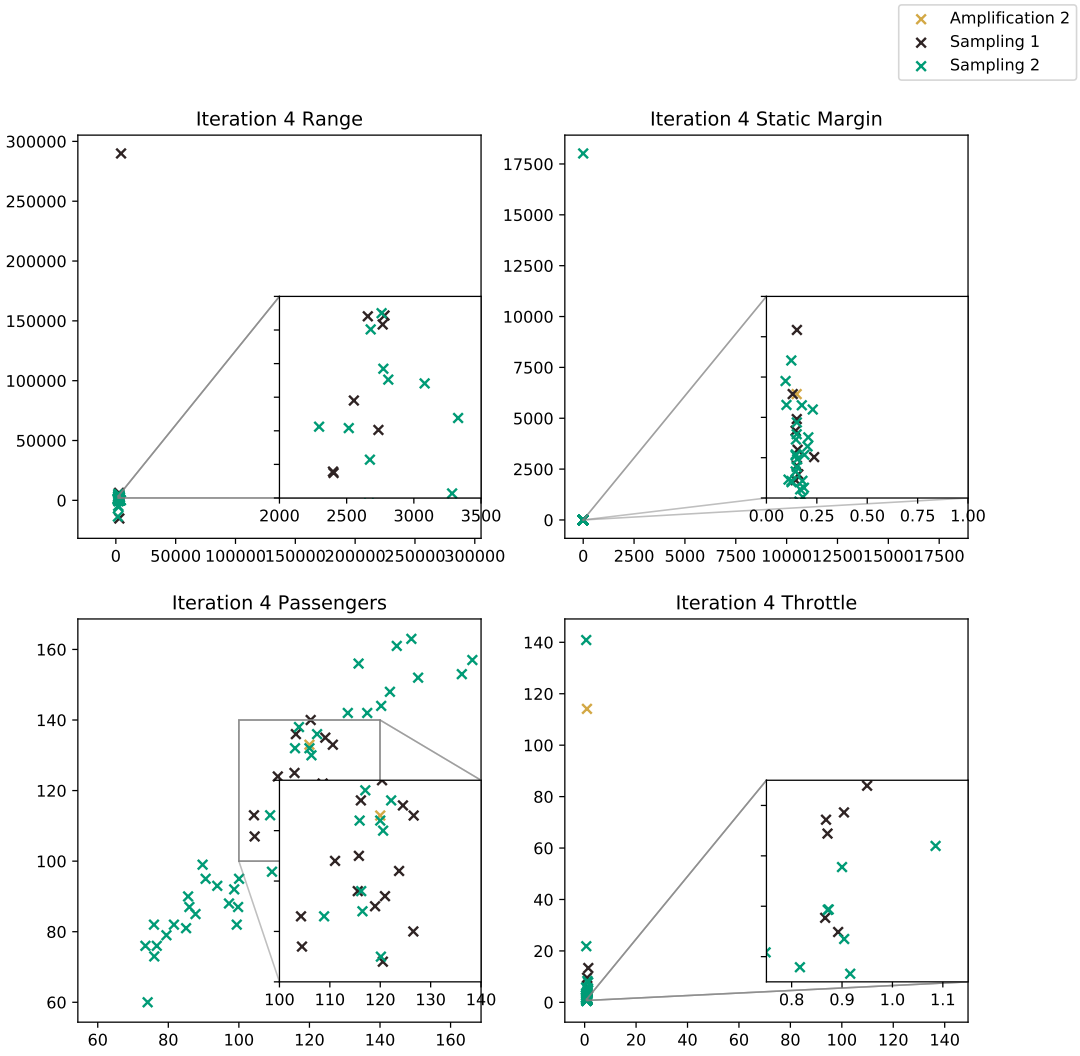


Figure B.16: Iteration 4

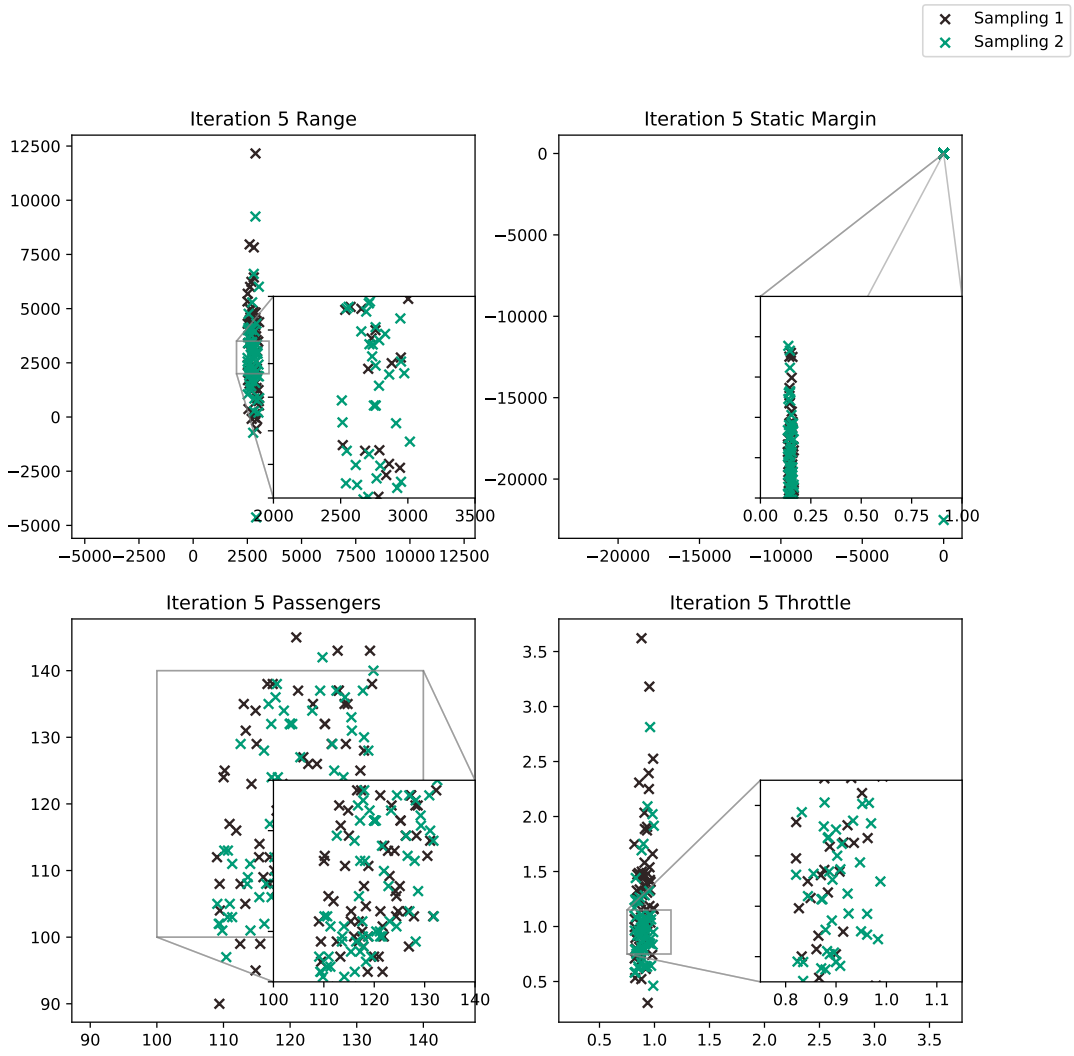


Figure B.17: Iteration 5: Final Iteration

B.3 Learned Graph Structure

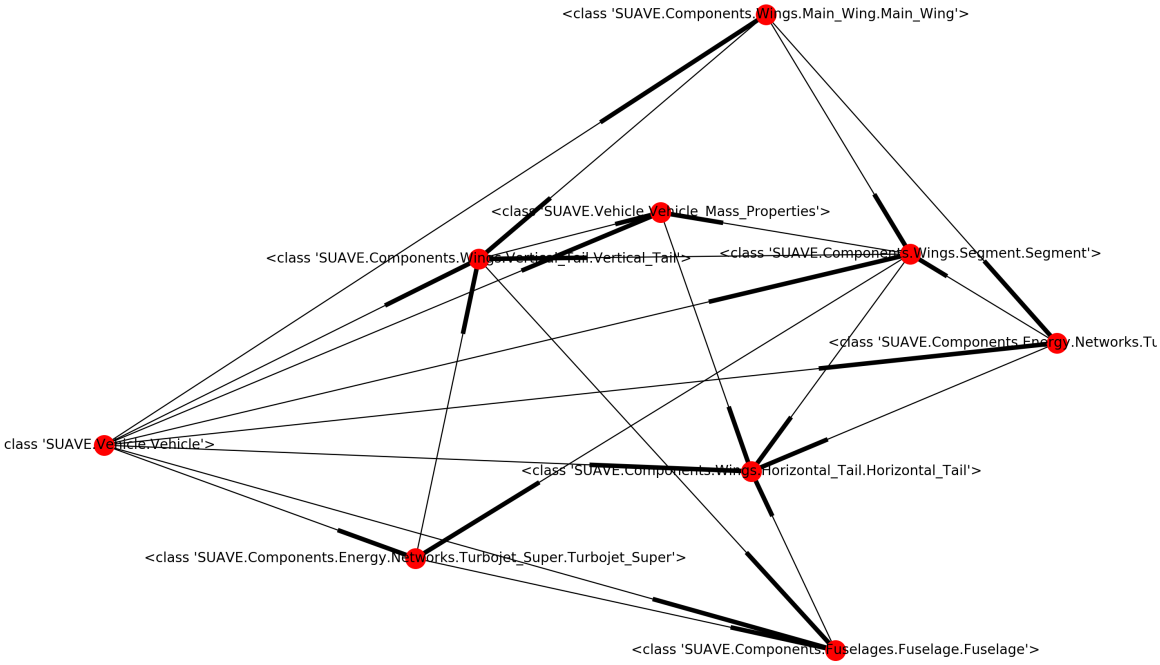


Figure B.18: Medium-Range Airliner with Segments Learned Network

Appendix C

Kangaroo Airliner Convergence

Similar to the past two appendices, this appendix contains additional figures to accompany Section 5.5. This appendix is divided into three sections. The first contains images of the MPE samples taken at each major iteration. This set of MPE samples includes a zeroth iteration MPE sample; this sample is drawn after only training on the seed vehicles. The next section has input versus output comparison plots. For further information regarding how to interpret these plots, see Appendix A.1. Finally, the resulting graph structure is shown in Section C.3.

C.1 MPE Samples

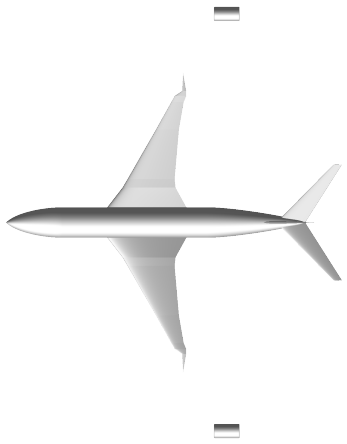


Figure C.1: Kangaroo Airliner Top View

Figure C.2: Kangaroo Airliner Front View

Figure C.3: Iteration 0

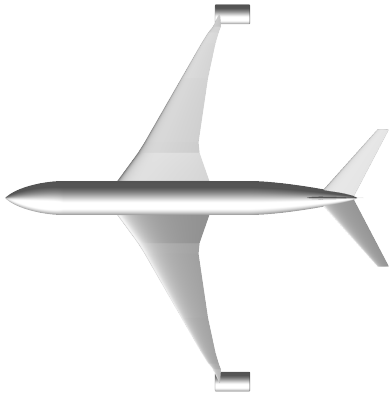


Figure C.4: Kangaroo Airliner Top View

Figure C.5: Kangaroo Airliner Front View

Figure C.6: Iteration 1

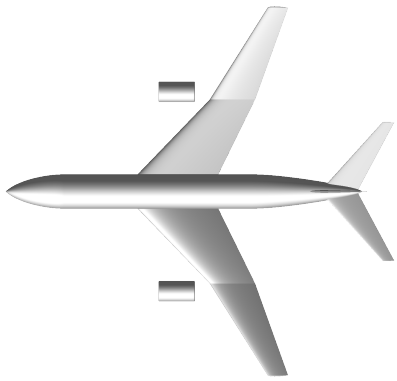


Figure C.7: Kangaroo Airliner Top View

Figure C.8: Kangaroo Airliner Front View

Figure C.9: Iteration 2

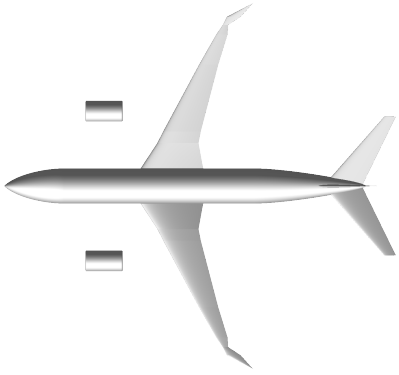


Figure C.10: Kangaroo Airliner Top View

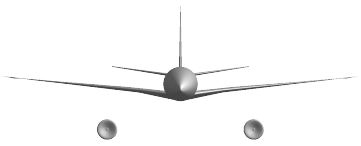


Figure C.11: Kangaroo Airliner Front View

Figure C.12: Iteration 3

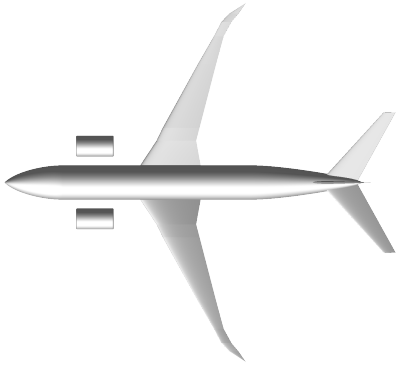


Figure C.13: Kangaroo Airliner Top View

Figure C.14: Kangaroo Airliner Front View

Figure C.15: Iteration 4

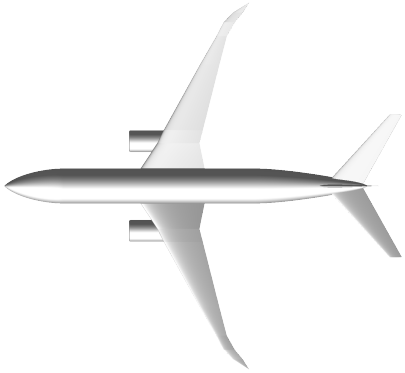


Figure C.16: Kangaroo Airliner Top View

Figure C.17: Kangaroo Airliner Front View

Figure C.18: Iteration 5: Final Iteration

C.2 Input-Output Comparison

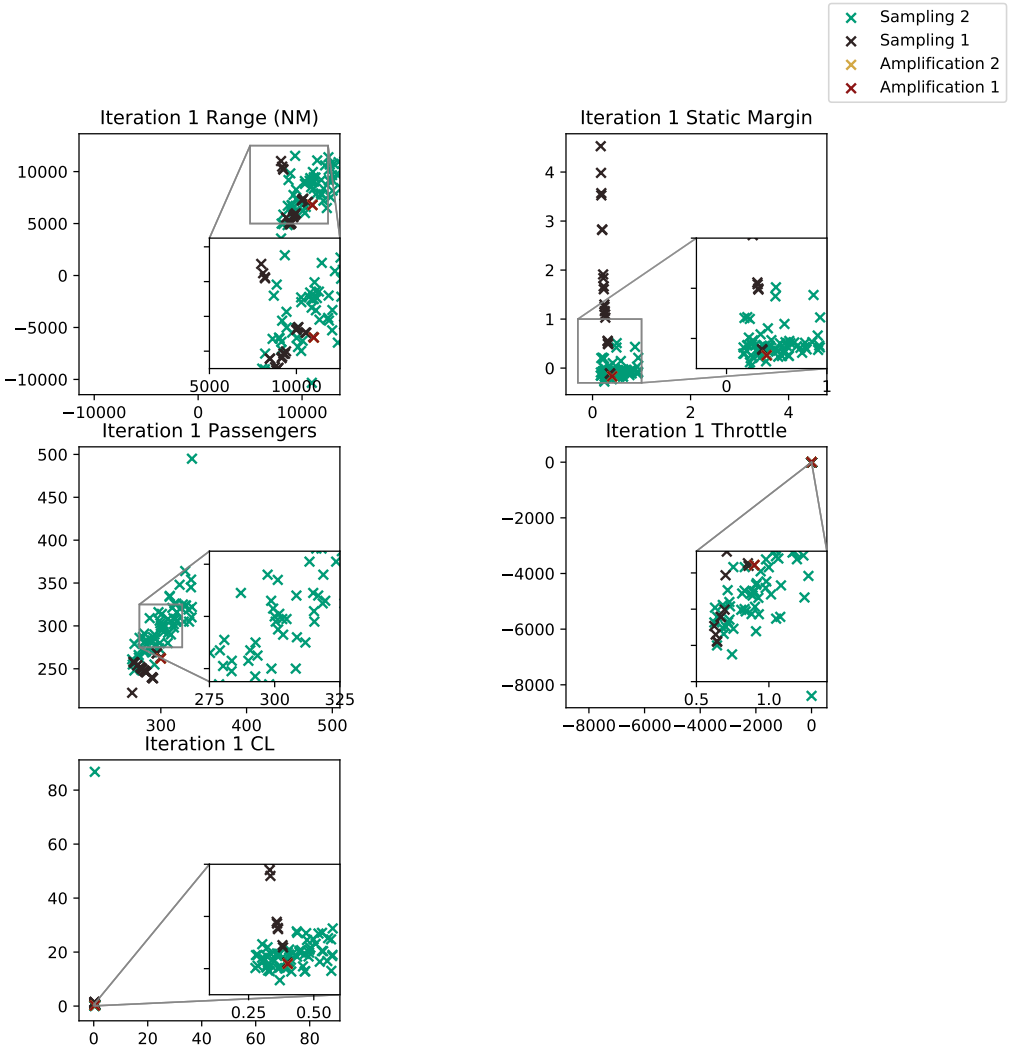


Figure C.19: Iteration 1

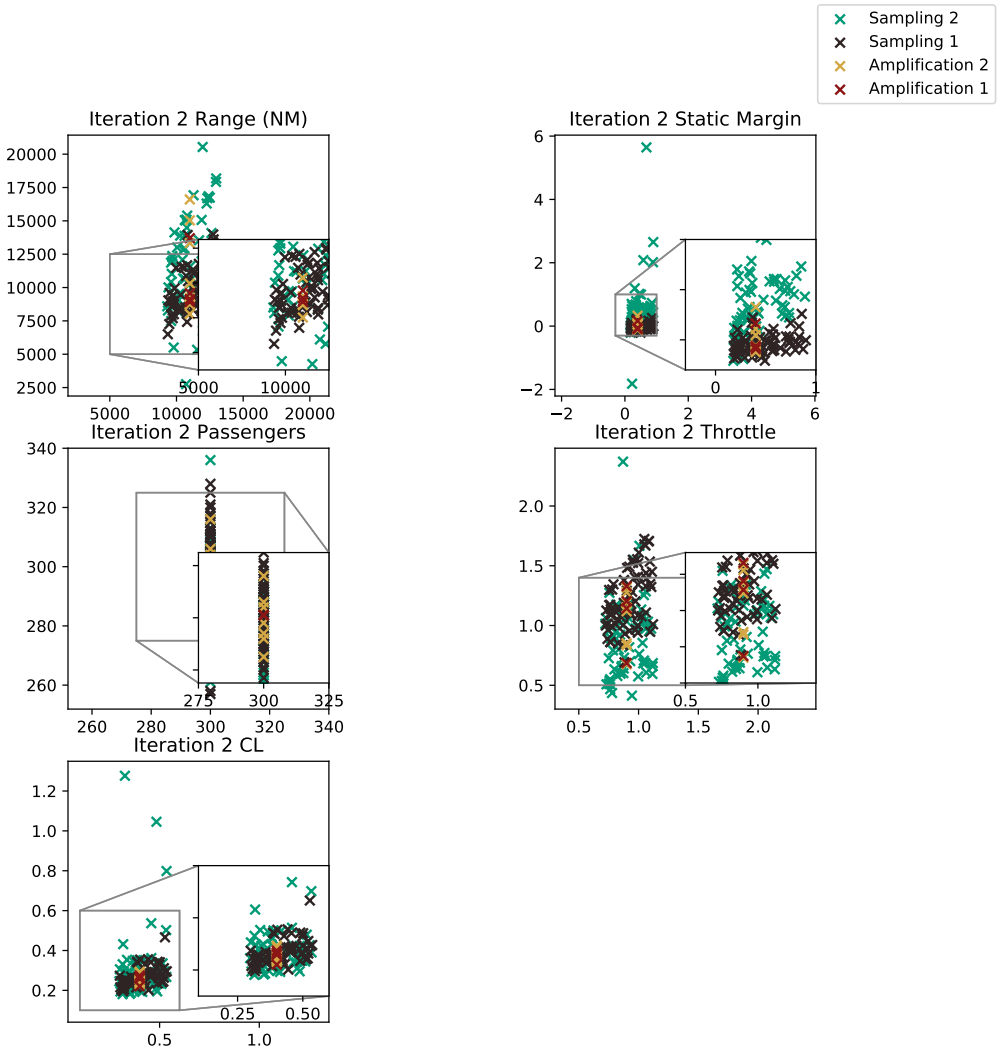


Figure C.20: Iteration 2

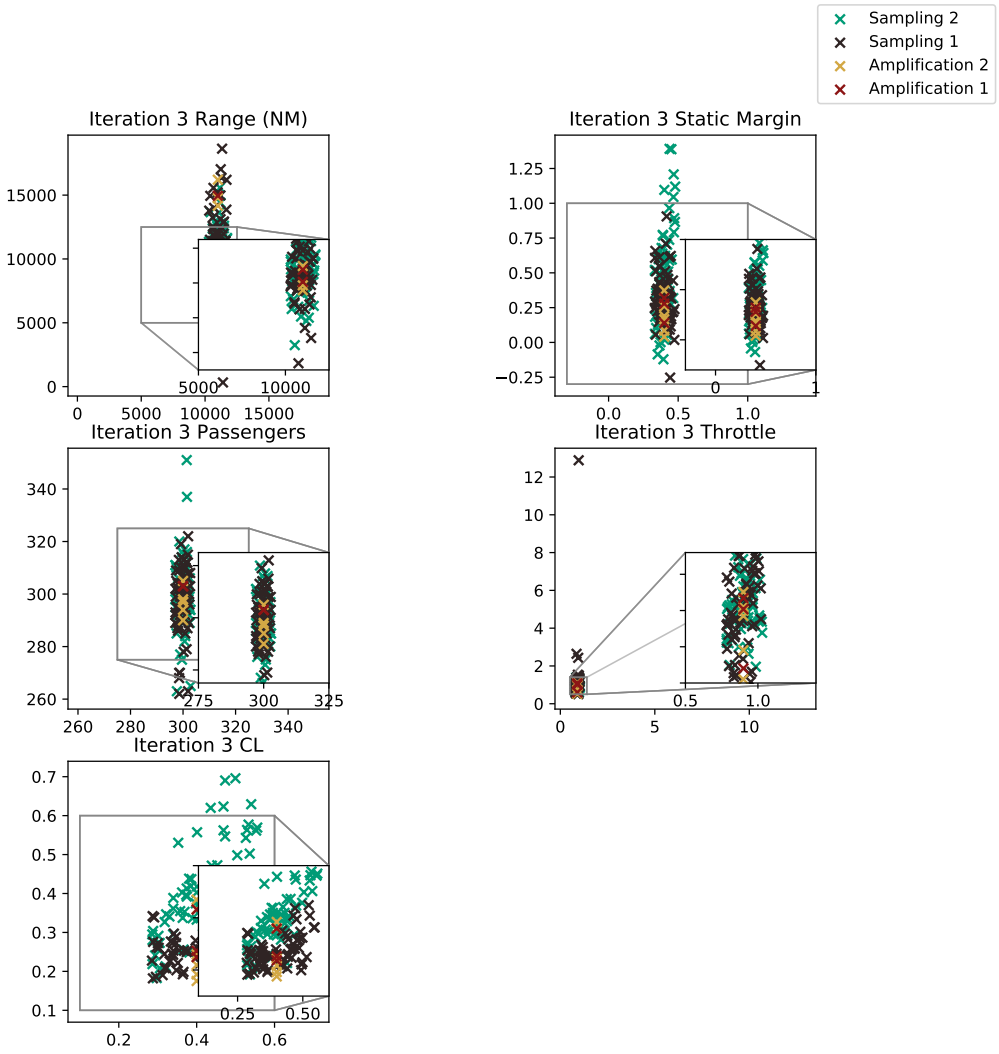


Figure C.21: Iteration 3

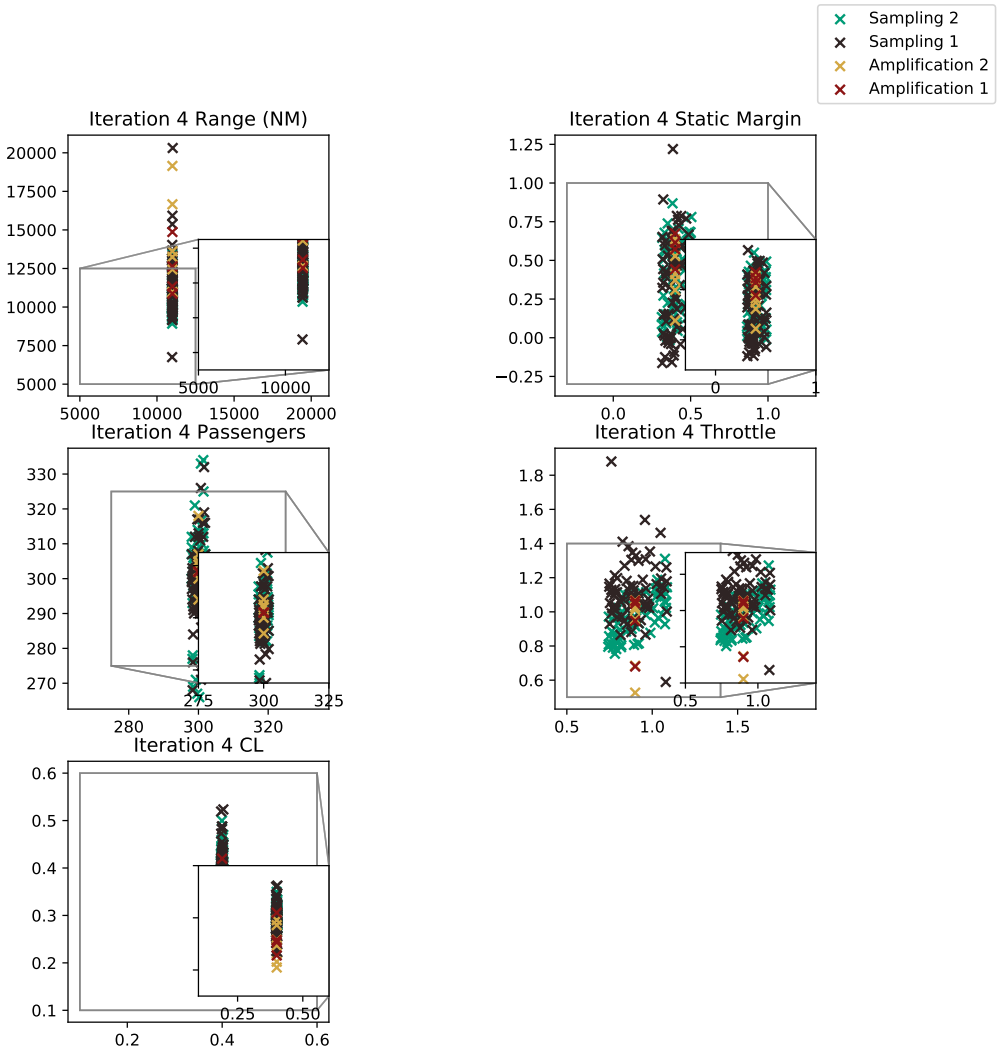


Figure C.22: Iteration 4

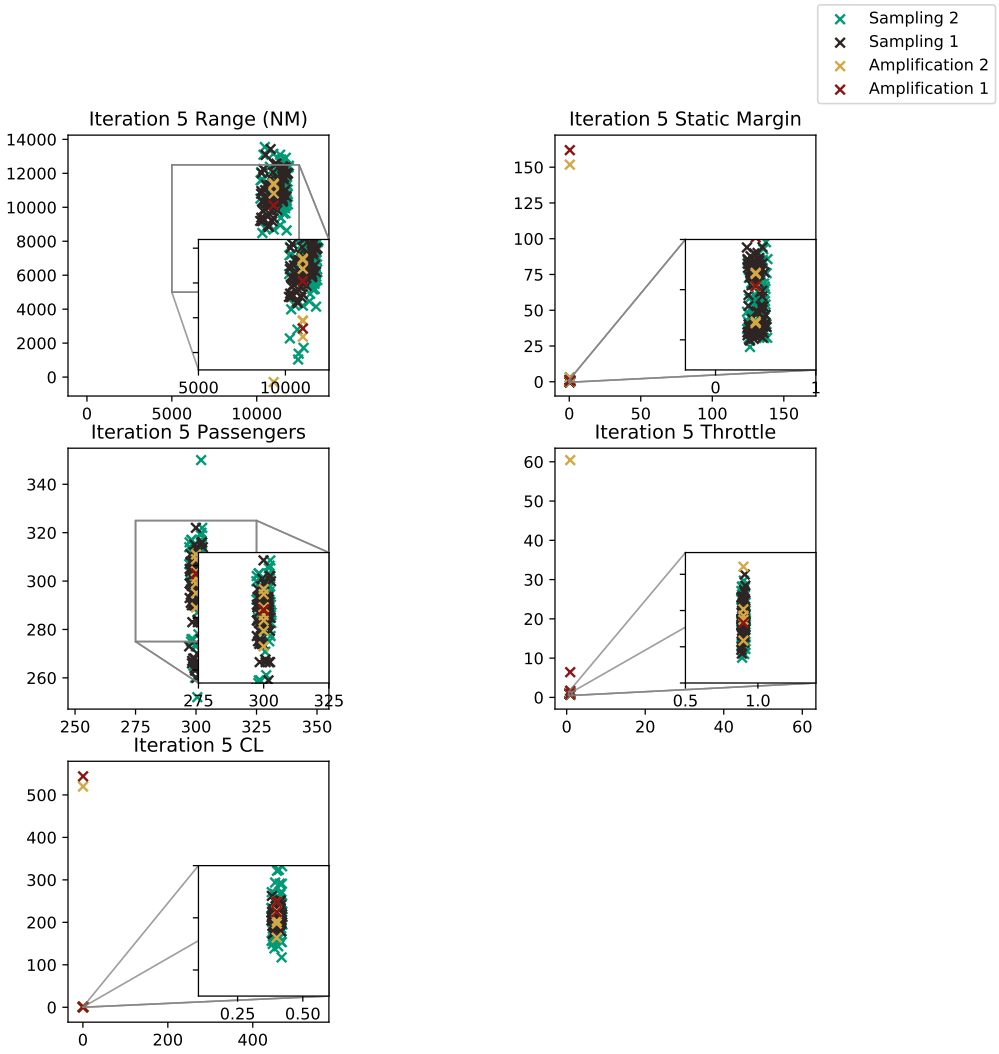


Figure C.23: Iteration 5: Final Iteration

C.3 Learned Graph Structure

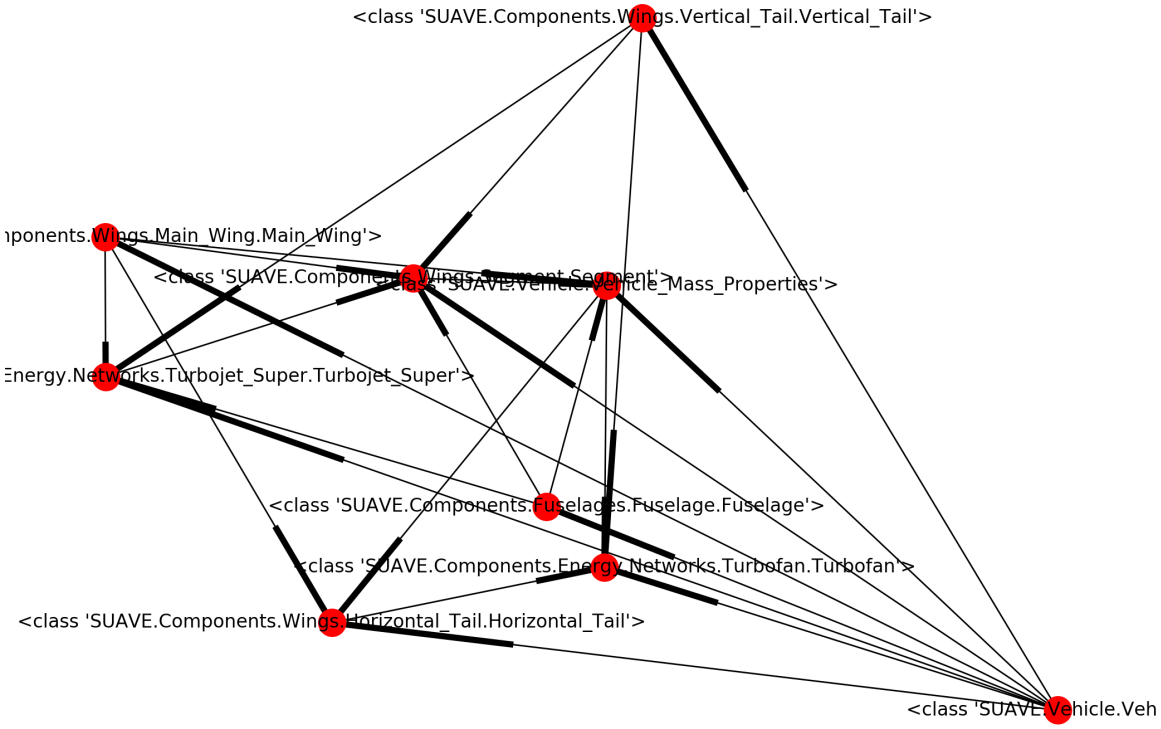


Figure C.24: Kangaroo Airliner Learned Network

Screw Flanders. Screw Flanders. Screw Flanders.

Bibliography

[1] Jeff Holden and Nikhil Goel. Uber elevate: Fast-forwarding to a future of on-demand urban air transportation. Technical report, Uber, October 2016.

[2] Shahab Hasan. Urban air mobility (uam) market study, 2018.

[3] A Datta, S Elbers, S Wakayama, J Alonso, E Botero, C Carter, and F Martins. Commercial intra-city on-demand electric-vtol status of technology. *AHS/NARI Transformative Vertical Flight Working Group*, 2, 2018.

[4] evtol aircraft directory. `http://evtol.news/aircraft/`. Accessed: 2019-10-20.

[5] Michael J Hirschberg. V/stol: the first half-century. *Vertiflite*, 43(2):34–54, 1997.

[6] Daniel Raymer. *Aircraft design: a conceptual approach*. American Institute of Aeronautics and Astronautics, Inc., 2018.

[7] Jan Roskam. Preliminary sizing of airplanes. *Airplane Design*, pages 5–196, 1988.

[8] Egbert Torenbeek. *Advanced aircraft design: conceptual design, analysis and optimization of subsonic civil airplanes*. John Wiley & Sons, 2013.

[9] Richard Shepherd Shevell. *Fundamentals of flight*. Prentice Hall, 1989.

[10] Leland M Nicolai and Grant E Carichner. *Fundamentals of aircraft and airship design, volume 1–aircraft design*. American Institute of Aeronautics and Astronautics, 2010.

[11] Clive L Dym and Patrick Little. *Engineering design: A project-based introduction*. John Wiley and sons, 1999.

[12] Hasso Plattner, Christoph Meinel, and Ulrich Weinberg. *Design-thinking*. Springer, 2009.

[13] Rim Razzouk and Valerie Shute. What is design thinking and why is it important? *Review of educational research*, 82(3):330–348, 2012.

[14] Rudy Eggert. *Engineering design*. Pearson/Prentice Hall, 2005.

[15] Ken Hurst. *Engineering design principles*. Butterworth-Heinemann, 1999.

[16] Michael G Luchs, Scott Swan, and Abbie Griffin. *Design thinking: New product development essentials from the PDMA*. John Wiley & Sons, 2015.

[17] National Air and Space Museum. *Otto Lilienthal and Octave Chanute: Pioneers of Gliding*. Smithsonian Institution, National Air and Space Museum, 1980.

[18] I M. Kroo. Aircraft design: Synthesis and analysis, 01 2006.

[19] Thomas C Corke. *Design of aircraft*. Prentice Hall Englewood Cliffs, NJ, 2003.

[20] Kelly Johnson. Sketch Johnson Lockheed a-3. "https://commons.wikimedia.org/wiki/File:SketchJohnsonLockheedA-3.jpg",note = Accessed: 2019-7-17, 1958.

[21] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[22] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):55, 2012.

[23] Matthew James Guzdial. *Combinational Machine Learning Creativity*. PhD thesis, Georgia Institute of Technology, 2019.

[24] Date Willem Egbert Rentema. *AIDA. Artificial Intelligence supported conceptual Design of Aircraft*. PhD thesis, Delft University of Technology, 2004.

[25] Gianfranco La Rocca. *Knowledge based engineering techniques to support aircraft design and optimization*. PhD thesis, Delft University of Technology, 2011.

[26] Gianfranco La Rocca. Knowledge based engineering: Between ai and cad. review of a language based technology to support engineering design. *Advanced engineering informatics*, 26(2):159–179, 2012.

[27] S TONG. Coupling artificial intelligence and numerical computation for engineering design. In *24th Aerospace Sciences Meeting*, page 242, 1986.

[28] ILAN KROO. A quasi-procedural, knowledge-based system for aircraft design. In *Aircraft Design, Systems and Operations Conference*, page 4428, 1994.

[29] Jaroslaw Sobieszczanski-Sobieski, Alan Morris, and Michel Van Tooren. *Multidisciplinary design optimization supported by knowledge based engineering*. John Wiley & Sons, 2015.

[30] Peter J Gage. *New approaches to optimization in aerospace conceptual design*. PhD thesis, Leland Stanford Junior University, 1995.

[31] Peter J Gage, IM Kroo, and IP Sobieski. Variable-complexity genetic algorithm for topological design. *AIAA journal*, 33(11):2212–2217, 1995.

[32] Trent W Lukaczyk, Andrew D Wendorff, Michael Colonno, Thomas D Economon, Juan J Alonso, Tarik H Orra, and Carlos Ilario. SUAVE: an open-source environment for multi-fidelity conceptual vehicle design. In *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3087, 2015.

[33] Alexander Kossiakoff, William N Sweet, Samuel J Seymour, and Steven M Biemer. *Systems engineering principles and practice*, volume 83. John Wiley & Sons, 2011.

[34] Antonio Salmerón, Rafael Rumí, Helge Langseth, Thomas D Nielsen, and Anders L Madsen. A review of inference algorithms for hybrid bayesian networks. *Journal of Artificial Intelligence Research*, 62:799–828, 2018.

[35] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[36] John Jones, Yang Xiang, and Stefan Joseph. Bayesian probabilistic reasoning in design. In *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, volume 2, pages 501–504. IEEE, 1993.

[37] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore,

Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[39] Alexandra M Carvalho. Scoring functions for learning bayesian networks. *Inesc-id Tec. Rep*, 12, 2009.

[40] Stefano Ermon. Structure learning for bayesian networks. `https://ermongroup.github.io/cs228-notes/learning/structure/`. Accessed: 2019-10-10.

[41] D Anderson and K Burnham. Model selection and multi-model inference. *Second. NY: Springer-Verlag*, 63, 2004.

[42] Gregory F Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

[43] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better bayesian network structures: Constraint-based, score-based or hybrid algorithms? *arXiv preprint arXiv:1805.11908*, 2018.

[44] M Henrion. propagating uncertainty by logic sampling in bayes. *networks,Technical Report, Department of Engineering and Public Policy, Carnegie-Mellon University, Pittsburgh, PA*, 1986.

[45] Homer L Chin and Gregory F Cooper. Stochastic simulation of bayesian belief networks. *arXiv preprint arXiv:1304.2722*, 2013.

[46] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[47] Martin Pelikan and David E Goldberg. Hierarchical bayesian optimization algorithm. In *Scalable optimization via probabilistic modeling*, pages 63–90. Springer, 2006.

[48] Dev Rajnarayan and David Wolpert. *Bias-Variance Trade-offs: Novel Applications*, pages 101–110. Springer US, Boston, MA, 2010.

[49] Francisco Palacios, Juan Alonso, Karthikeyan Duraisamy, Michael Colonno, Jason Hicken, Aniket Aranake, Alejandro Campos, Sean Copeland, Thomas Economon, Amrita Lonkar, et al. Stanford university unstructured (su 2): an open-source integrated computational environment for multi-physics simulation and design. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 287, 2013.

[50] Emilio Botero, Tim MacDonald, Jmvegh, Trent Lukaczyk, Carlos Roberto Ilário Da Silva, Tmomose, Mclarke2, Tarikorra, Awendorff, Jordan Smart, Wally Maier, Pedro Gonçalves, Tstfrancis, Anilvar, and Fcapristan. suavecode/suave: Suave 2.0.0. `https://zenodo.org/record/2564444`, 2019.

[51] Timothy MacDonald, Emilio Botero, Julius M Vegh, Anil Variyar, Juan J Alonso, Tarik H Orra, and Carlos R Ilario da Silva. SUAVE: An open-source environment enabling unconventional vehicle designs through higher fidelity. In *55th AIAA Aerospace sciences meeting*, page 0234, 2017.

[52] Emilio M Botero, Andrew Wendorff, Timothy MacDonald, Anil Variyar, Julius M Vegh, Trent W Lukaczyk, Juan J Alonso, Tarik H Orra, and Carlos Ilario da Silva. SUAVE: An open-source environment for conceptual vehicle design and optimization. In *54th AIAA Aerospace Sciences Meeting*, page 1275, 2016.

[53] Andrew Wendorff, Emilio Botero, and Juan J Alonso. Comparing different off-the-shelf optimizers' performance in conceptual aircraft design. In *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3362, 2016.

[54] Timothy MacDonald, Matthew Clarke, Emilio M Botero, Julius M Vegh, and Juan J Alonso. Suave: an open-source environment enabling multi-fidelity vehicle optimization. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4437, 2017.

[55] Emilio M Botero and Juan J Alonso. Conceptual design and optimization of small transitioning uavs using SUAVE. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4149, 2017.

[56] Matthew Clarke, Jordan Smart, Emilio M Botero, Walter Maier, and Juan J Alonso. Strategies for posing a well-defined problem for urban air mobility vehicles. In *AIAA Scitech 2019 Forum*, page 0818, 2019.

[57] John J Bertin and Michael L Smith. *Aerodynamics for engineers*, volume 5. Prentice Hall Upper Saddle River, NJ, 1998.

[58] Joseph Katz and Allen Plotkin. *Low-speed aerodynamics*, volume 13. Cambridge university press, 2001.

[59] Tomas Melin. A vortex lattice matlab implementation for linear aerodynamic wing applications. *Royal Institute of Technology, Sweden*, 2000.

[60] Matweb asm material data sheet 4130 steel. `http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=m4130r`. Accessed: 2019-10-10.

[61] Matweb asm material data sheet 7075 aluminum. `http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=ma7075t6`. Accessed: 2019-10-10.

[62] Stefano Ermon and Aditya Grover. Autoregressive models. `https://deepgenerativemodels.github.io/notes/autoregressive/`. Accessed: 2019-10-10.

[63] Benjamin D Katz. Airbus seeks 100-plus orders for Bombardier jet now called A220. `https://www.bloomberg.com/news/articles/2018-07-10/airbus-seeks-100-plus-orders-for-bombardier-jet-now-called-a220`. Accessed: 2019-10-14.

[64] Paolo Cerutti. CS100 CS300 3view. "`https://commons.wikimedia.org/wiki/File:CS100_CS300_3view.jpg`",note = Accessed: 2019-11-29, 2011.

[65] Airbus Canada. A220 airport planning publication app. Technical Report BD500-3AB48-22000-00 Issue No. 020, Airbus Canada, Airbus Canada Limited Partnership Customer Services 13100 Henri-Fabre Blvd., Mirabel, Quebec Canada J7N 3C6, June 2019.

[66] Reuters News Agency. London to Sydney non-stop flights 'real possibility' within five years, says Qantas. `https://www.telegraph.co.uk/news/2017/04/06/london-sydney-non-stop-flights-real-possibility-within-five/`, Apr 2017. Accessed: 2019-11-11.

[67] Alinn Winn. Qantas flies London-Sydney non-stop. *Flight International*, 1989(2606):88, Aug 1989.

[68] Richard Quest and Barry Neild. London to Sydney flight breaks world record, Nov 2019.

[69] Tamara Hardingham-Gill. Which plane will fly London-Sydney nonstop? `https://www.cnn.com/travel/article/qantas-london-sydney-airplane/index.html`, Jun 2019. Accessed: 2019-11-10.

[70] David Kaminski-Morrow. A350-1000 to be Airbus's lead candidate for project sunrise. `https://www.flightglobal.com/news/articles/a350-1000-to-be-airbuss-lead-candidate-for-project-461544/`, Oct 2019. Accessed: 2019-11-10.

[71] Frederick Harwood Norton. An investigation on the effect of raked wing tips. *National Advisory Committee for Aeronautics*, 1921.

[72] Boeing Commercial Airplanes. *787 airplane characteristics for airport planning*. Boeing Commercial Airplanes, 2018.

[73] Andrew Ning and Ilan Kroo. Multidisciplinary considerations in the design of wings and wing tip devices. *Journal of Aircraft*, 47(2):534–543, 2010.

[74] Andrew Ning and Ilan Kroo. Tip extensions, winglets, and c-wings: conceptual design and optimization. In *26th AIAA Applied Aerodynamics Conference*, page 7052, 2008.

[75] Mark Drela. Development of the D8 transport configuration. In *29th AIAA Applied Aerodynamics Conference*, page 3970, 2011.