AN ABSTRACT OF THE DISSERTATION OF

Ada-Rhodes Short for the degree of Doctor of Philosophy in Mechanical Engineering presented on June 13, 2018.

Title: Autonomous Decision Making Facing Uncertainty, Risk, and Complexity.

Abstract approved:

_____

Bryony L. DuPont

The creation of autonomous decision makers to handle potentially hazardous or logistically complicated tasks is desirable for the purpose of reducing human labor and exposure to risk, and for increasing the general performance of autonomous systems in a wide variety of environments. However, traditional autonomous decision makers perform poorly when faced with the uncertainty, risk, and complexity of these situations. In this dissertation, I explore these problems through three distinct case studies, and make significant improvements in the performance of autonomous decision makers through the application and synthesis of concepts from decision theory, risk assessment, and computational cognition. The first case study is a terrain navigation problem which shows how the availability of information and information fidelity affect performance, and develops and validates techniques to overcome these problems. Next, an automated design problem is used to explore and evaluate techniques for handling intractably large decision spaces. Finally, a mission planning and command simulation is performed to investigate how multiple time delineated decisions can be made efficiently. Through these three case studies, I show that autonomous decision-makers can be made to perform efficiently and effectively under uncertain, high-risk, and incredibly complex circumstances.

Autonomous Decision Making Facing Uncertainty, Risk, and Complexity

by
Ada-Rhodes Short

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 13 2018
Commencement June 2018

Doctor of Philosophy dissertation of Ada-Rhodes Short presented on June 13, 2018

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Ada-Rhodes Short, Author

ACKNOWLEDGEMENTS

My work is the result of an ever expanding line of friends, mentors, and teachers that helped me along the way despite many unexpected twists and turns along the way. I offer my sincerest thanks to my advisor Dr. Bryony DuPont who rescued my research and brought me to Oregon State University, to my interim advisor Dr. Alexandra Newman at the Colorado School of Mines, and to Dr. Carolyn Skurla my undergraduate advisor and mentor at Baylor University. I would also like to thank Dr. Irem Tumer, Dr. Rob Stone, Dr. Matt Campbell, and Dr. Jenny Hutchings for serving on my doctoral committee and offering their time, effort, and wisdom. Finally I would like to thank my partner in life and constant comrade Míkasì Goodwin for supporting me through the late nights, early mornings, and long days.

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

Autonomous decision making is the process of using computational techniques to gather information and then use that information to make informed decisions without human intervention. An autonomous decision maker is a system capable of autonomous decision making. A common class of autonomous decision makers is mobile robots [1], which often must make navigation decisions in their operating environments. In contrast, an autonomous decision-maker could also be a completely immobile system, such as an application that helps a user select outfits from their wardrobe [2].

Developing autonomous decision makers designed to handle potentially hazardous or logistically complicated tasks is desirable because of the potential to reduce human labor and exposure to risk, and increase the general performance of autonomous systems in a wide variety of environments. However, traditional autonomous decision makers perform poorly when faced with the uncertainty, risk, and complexity of these situations. In this dissertation, I explore these problems through three distinct case studies and make significant improvements in the performance of autonomous decision makers through the application and synthesis of concepts from decision theory, risk assessment, and computational cognition.

In the first case study, I simulated a terrain navigation problem which explores how information fidelity and the decision horizon affect decision maker performance and examines how variable risk attitudes [3] affect performance in hazardous environments. This case study will show how autonomous decision makers can adapt to uncertain environments and perform in higher information fidelity environments

(like reality) while utilizing lower information fidelity models. The second case study is an automated design problem that explores the behavior of decision-making techniques in intractably large and complex decision spaces. This case study will show which techniques and algorithm characteristics are most effective when dealing with large, open end, non-monotonic decision spaces. The third case study uses a mission planning and command simulation to investigate how to make multiple time-sensitive decisions efficiently. This case study will demonstrate how an automated decision maker can utilize concepts from computational cognition to handle large complex and uncertain decision spaces efficiently.

1.1 Motivation

Humans are not good at making decisions when faced with uncertainty (choosing an action based off of imperfect information, with unknown outcomes [4]) [5], risk [6], or immense complexity [7], Because of these potential shortcomings in human decision making, it is highly desirable to create autonomous decision makers that can handle these problems. However, the same factors that make these problems difficult for humans make them infeasible for modern autonomous decision makers without using extreme quantities of computational resources such as supercomputers. For example, AlphaGo was recently developed to play the ancient Chinese game of Go and was able to beat the best players in the world but to do this; it first had to play more games of Go than have ever been played by humans [8]. This strategy would not be very effective for most real-world problems because stopping and running a large number of simulations every time the decision maker encounters a new problem is very time consuming and computationally expensive. Additionally, simulations of most real-

world problems will never capture the full information fidelity of the problem. The work I present in this dissertation aims to address this problem by exploring how to make difficult decisions while using minimal computational resources through the empirically informed selection of existing algorithms, as well as the development of new techniques inspired by computational cognition. This research goal is encapsulated in the following thesis statement:

> Traditional autonomous decision-making techniques struggle to perform under conditions of uncertainty, risk, and extreme complexity; however, performance can be significantly improved through the application and synthesis of techniques from decision theory, risk analysis, game theory, and computational cognition.

In the context of this work, uncertainty describes the unknown in a problem space; it is all of the factors that are not known by the decision maker but affect performance. Risk is the probability of a negative event occurring multiplied by the consequence of that event, giving an expected average consequence. Conversely, reward is the probability of a positive even occurring multiplied by the benefit gained from that event. For this work, a system is considered to have complexity if the decision making space is large and multimodal making it difficult to handle through brute force search methods.

The findings of this work are broadly generalizable to many existing problems in decision making, and the methods and techniques developed throughout this work have mainly applications.

1.1.1 Generalizability

The work presented in this dissertation is broadly generalizable to autonomous decision-making problems and also improves understanding of human decision making. Several specific contributions to autonomous decision making in general include:

- Exploration of the consequences of information fidelity in decision-making models and techniques to overcome the information fidelity gap
- Understanding of how risk attitudes influence autonomous behavior
- Improved concepts for autonomous navigation
- Identification of characteristics and challenges of the design decision tree
- Introduction and explanation of Color Graph as a way to explore techniques that utilize design decision trees efficiently
- Investigation of the challenges of multi-objective mission structures and the introduction of techniques for rapid, low computational expense techniques for overcoming them

1.1.2 Specific Applications

Due to the broad generalizability of this work, it has applications in many domains including autonomous mobile robots, design and manufacturing, logistics operations, and space mission command and control. Autonomous mobile robots often have to make navigational and operational decisions facing uncertainty, risk, and complexity. The traditional approach for this problem tends to be applying methods that use increased computational power to improve performance such as many machine

learning methods. For example, this could involve taking a picture of the environment, using off-board computed machine vision methods to identify every object, constructing a detailed model of an environment, and then using a supercomputer to run a Monte Carlo Tree Search to find an acceptable route before sending it back to the robot.

The techniques explored in this work use methods inspired by computational cognition, decision theory, and game theory to improve the efficiency of the decision maker allowing an autonomous mobile robot to adapt to unknown objects and situations it encounters while using less computational resources.

These techniques could enable the development of more robust planetary exploration rovers that require less human input to operate and autonomous probes that could enable us to more effectively explore the cosmos and collect valuable scientific data without having to deal with large signal delays, or they could lead to the creation of home robots that are more capable of adapting to the chaos of human life [9].

This research could also enable new forms of automated design where a human dictates their wants and needs for a product. Not only could a new product be designed to fit their specifications, but the manufacturing process can be laid out from start to finish including everything from material sourcing and assigning tasks to craftspeople and machines, to finding the best shipping route, dramatically decreasing human labor.

Furthermore, by using some of the techniques for fault detection and response, a manufacturing line could be set up to detect problems as they occur and automatically find solutions without human intervention. These same techniques could also be used

to develop space probes that are capable of piloting themselves and making their own decisions.

1.2 Research Approach

The general approach of this work focuses on exploring challenging decision-making problems and identifying common characteristics that exist between them. Next methods drawing from decision theory, risk analysis, game theory, and computational cognition are explored and evaluated on their ability to address the problem. Finally, the insights gained through empirical study are used to develop new techniques that address the problem more effectively. Several of the biggest problems identified in this work include the information fidelity gap, design decision tree searches, and mission control and command decisions. All of these problems are present in the three case studies.

1.2.1 The Information Fidelity Gap

Models used by decision makers (both human and machine) are inherently low fidelity when compared to the information fidelity of reality. I call this problem the *information fidelity gap*, which means that even supposedly deterministic systems will not behave exactly as expected leading to potentially unexpected outcomes. Additionally, anything outside of the model is effectively a *black swan* (something that has not been seen before and therefore treated as impossible), meaning that there are gaps in the decision maker's ability to handle unexpected events. Traditional techniques for addressing the information fidelity gap involve increasing model fidelity and computational power necessary to utilize this model. However, model fidelity will

continue to struggle to reach the information fidelity of reality in uncontrolled environments [10], so instead, I explore ways to bridge the information fidelity gap through various techniques.

1.2.2 Design Decision Tree Searches

A design decision tree is an extensive form representation of the decision space for a design problem that I introduce in this dissertation. Design decision trees have four unique properties: multimodality, opaque intermediate states, unboundedness, and increasing internal locations.

The *multimodality* of the design tree makes finding the global optimum difficult as local minima may exist at different levels and different branches of the tree. For example, when designing a chair, a seat with a cushion on the floor is a simple but good design that is higher quality than the same seat with a single leg on the bottom of the seat. However, adding a stable number of legs to the bottom of the seat would be preferable to a seat on the ground.

The second property of *opaqueness* means that only completed designs can be evaluated and reliable information about the final design cannot be gleaned from intermediate states.

The design decision tree is also *unbounded*. This means that potentially infinite decisions can be made in the design space. Using the chair example again, the property of unboundedness means that a chair could be made with armrests, and the armrests could have cup holders, and there can be an unreasonably large number of cup holders potentially. The unboundedness leads into our final property *increasing internal locations*.

Increasing internal locations mean that as decisions are made in the design decision tree, it creates new locations from which additional subsequent decisions can be made, resulting in a greater number of potential choices as the candidate solutions increase in complexity. Again, using the chair example, every time an armrest is added the number of current design decisions is increased by the new potential of adding a cup holder. The Color Graph problem was developed to possess all four of the properties while being rapidly evaluable with minimal computational resources.

Further discussion of these properties can be found in Section 5.3.1.

1.2.3 Mission Control and Command Decisions

One desirable application of autonomous decision makers is the management of large multi-agent teams for the completion of multiple objectives through the performance of tasks. An example of this is mission control and mission command decisions, such as those made by humans on earth when managing a planetary exploration rover, or those made by a commander during a space mission when allocating tasks and giving orders. However, this approach is also generalizable to other multi-agent applications. The challenges faced are the same as those faced by a product manager bringing a product to market. This class of problem can be broken down into three sub-problems: 1) ordering objectives, 2) ordering tasks to complete objectives, 3) assigning agents and equipment to tasks. Each of those sub-problems, however, is computationally difficult and is further complicated by the fact that decisions often need to be made rapidly and consider unanticipated conditions. In this dissertation, I address these problems by exploring what strategies are most effective for each

problem under specific conditions and then develop a method to select the best strategy for the problem at hand.

2. LITERATURE REVIEW

The work presented in this dissertation builds on existing concepts from decision theory, computational cognition, functional modeling, risk analysis, game theory, tree search, and automation. This literature review describes relevant work in each of these fields and provides some detail on the concepts that informed the work. Figure 1 summarizes which fields contributed to the study of each of the three problems.

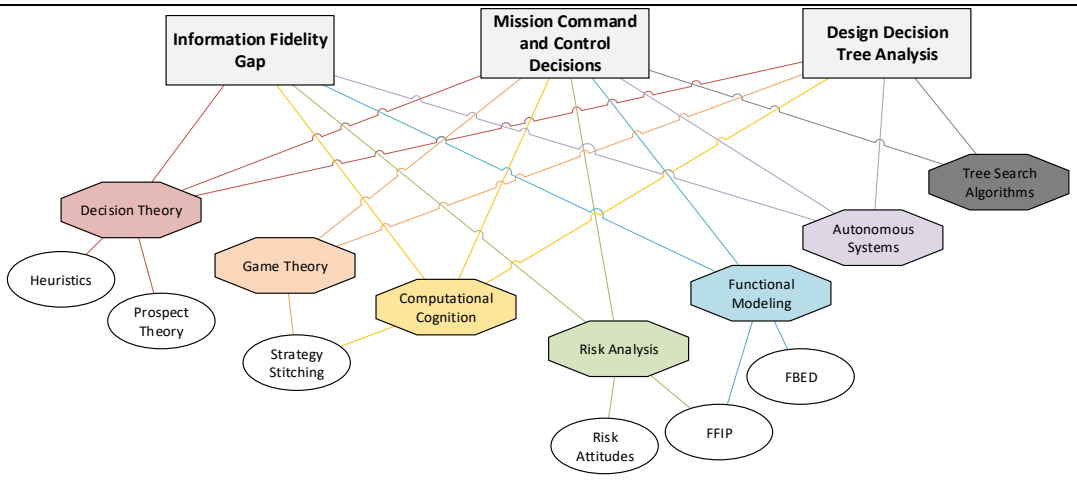


Figure 1: Contributions of Existing Research Areas to the Current Work

2.1 Decision Theory

The concepts of prospect theory [11,12] from psychology and expected utility theory [13,14] from economics formed the foundational basis that became decision theory in the mid-twentieth century, but since then the field has grown to encompass a wide variety of theories about how to make decisions. While human decision makers often do not use expected utility theory in many of their decisions and are likely to rely on heuristic biases instead [11], it is a rational way to make decisions when facing risk

and used it often throughout this work. Decision theory also includes how decisions are informed and what biases influence the decision making. A form of bias that I utilize extensively throughout this dissertation is a risk attitude, which is an individual's bias towards acceptable levels of risk for reward [3]. Heuristics are another type of decision-making bias that comes up throughout the work. In decision making, a heuristic is a rule of thumb used to more efficiently make a complex decision and decisions under uncertainty [5]. Heuristic methods for decision making are often employed in the form of algorithms that aim to make *good* decisions efficiently as opposed to *optimal* decisions inefficiently.

2.1 Computational Cognition

The study of computational cognition uses modern computational techniques from statistics, data analysis, machine learning, and artificial intelligence to improve humanity's understanding of cognitive science. However, this relationship is not unidirectional, and computational cognition also includes the using insights about cognition and cognitive science to inform computational methods. Currently, the study of inference [15–19] is one of the primary focuses of computational cognition research, but work is also being carried out on a variety of topics including reasoning [20], visual perception [21], reverse engineering of brains [16].

Generally, computational models of cognition exist on three distinct levels: hardware, algorithmic, and computational [22]. The hardware model level describes how processes are physically performed in a brain, algorithmic models focus on how processes are executed during decision making, and computational models represent cognition in functional mathematics terms. A common approach is to describe methods

for exploring these levels as either a top-down approach that beings computational analysis that informs the development of algorithms and places them in a larger cognitive structure, or as a bottom-up approach, which begins with neural mechanisms and works towards computational behavior as shown in Figure 1 [19]. In this dissertation, I take a top-down approach to studying and developing an autonomous decision maker. I first explore how simple computational approaches can be used to inform decision facing uncertainty, risk, and complexity. Next, I examine how algorithms utilizing these functions behave under test conditions. Finally, I construct a cognitive structure that properly applies algorithms to problems.

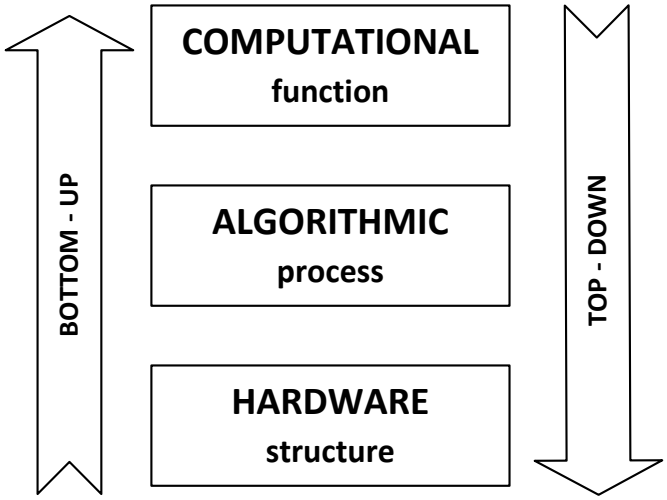


Figure 2: Bottom-Up and Top-Down strategies for studying cognition

2.2 Functional Modeling

Functional models represent not only what is physically present in a system, but describe what a system does and how it behaves. The Functional Basis for Engineering Design (FBED) is a commonly used approach for functional modeling [23]. FBED represents a system or system-of-systems using an established lexicon of terms that

represent flows (the materials, energy, and information acting on or being acted upon by the system) and functions that act on the flows [23–26]. This general approach is common amongst functional modeling methods and is employed throughout this research [26,27]. Other notable functional modeling techniques such as Flow Block Diagrams (also known as Functional Flow Diagrams (FFD)) [28,29] are difficult to apply to systems that are less linear, potentially resulting in tangled networks of functions and flows that are difficult or impractical to analyze, or must be simplified to the point where they provide an inaccurate representation of the system and its associated dynamics. However, FBED provides concise definitions of functions and flows that describe most engineered systems and allow for a greater degree of non-linearity.

The process of developing an FBED model is given as follows:

1) *Generate a Black Box model.* A black box model takes the highest-level-possible view of the system and only considers flows into and out of the overarching system model.

2) *Create function chains for each input flow and order them concerning time*. This step consists of following a flow from its entrance into the system, through all sub-systems that interact with the flow, and finally exiting the system. Then place all the systems that interact with the flow in chronological order from the perspective of the flow.

3) *Aggregate function chains into a functional model*. In the final step of FBED, combine the functional chains to determine the underlying functional structure of the

system. This paper uses FBED because of the advanced risk analysis methods that have been developed utilizing the FBED lexicon [30–34].

Functional models present a promising foundation for work in autonomous decision making because they represent the complex interconnected behavior in systems and enables broader automated problem solving [35–38]. Additionally, methods such as FBED enables various forms of risk analysis to be applied to a system, informing design and management decisions [39–42]. For example, the Active Mission Success Estimation (AMSE) method applied in parts of this paper utilizes a functional model of an entire mission framework and analyzes how decisions made in that framework affect the probability of mission success over time [43].

2.3 Risk Analysis

In both academia and industry, practitioners employ a variety of risk analysis methods to inform decision making, policy, and design, including:

- Failure Modes and Effects Analysis (FMEA) [44]
- Reliability Block Diagrams (RBD) [45]
- Probabilistic Risk Assessment (PRA) [26,46]
- Function Failure Design Method (FFDM) [34]
- Function Failure Identification and Propagation (FFIP) [39,47]
- Flow State Logic (FSL) [48]
- Failure Flow Decision Functions (FFDF) [36,37]

While methods like FMEA, RBD, and PRA can be used effectively to inform decisions facing risk, the quality of the failure model and practicality of implementation

limit their feasibility. However, in the past decade, several methods of risk analysis have been developed using functional modeling methods such as FBED [26]. The Function Failure Design Method (FFDM) [34] combines FMEA and FBED to provide decision makers with a method for choosing specific functions based on past component failure data. Additionally, methods like Function Failure Identification and Propagation (FFIP) [39,47] which models failure flows through FBED models and Flow State Logic (FSL) [48] which further refines FFIP and provides a complete representation of the analyzed system's failure state use functional representations to model system failure. These methods can be enhanced to enable autonomous navigational and mission control decisions to be made through the addition of Failure Flow Decision Functions (FFDF) [36,37] which are a tool that determines an optimal decision when faced with problems of system survivability. Within this dissertation risk analysis techniques were used by autonomous decision makers to maximize mission success while minimizing human work hours [38,46,49–52].

Risk analysis methods represent system failure through various failure distributions. A commonly used approach is the use of the hazard rate, λ, which describes the expected number of system failures over a specified time interval. A failure distribution is calculated from the hazard rate by using an appropriate distribution. For example, in Equation 1 an exponential distribution is used to calculate the probability of survival of a system or sub-system at a given time [53]:

$$S(t) = e^{-\lambda t} \quad [1]$$

The expected failure rate can be found by subtraction the expected survival rate from 1 as shown in [2.

$$F(t) = 1 - S(t) = 1 - e^{-\lambda t} \quad [2]$$

Throughout this dissertation hazard rates and failure rates are used as a metric for assessing autonomous decision-making performance when facing a risk of system failure.

2.3.1 Risk Attitudes

Risk attitude refers to an individual's or organization's preference towards acceptable levels of risk and reward [54,55]. Risk is a parameter that describes the probability of a negative event occurring multiplied by the severity of the event. Risk attitudes vary greatly between individual decision makers and are often influenced by experience, environment, culture, working conditions, and various other factors. Risk attitudes are often represented using a biased utility function [56] with a constant that pushes the function towards either risk or reward. Risk attitudes can be placed along a spectrum between risk aversion and risk tolerance. Risk aversion refers to the desire not to accept more risk in exchange for a reward. Using the example of a human crossing the road, a risk-averse individual will more likely go to the crosswalk and wait for the proper lights and signal to cross the street. Risk tolerance is the preference towards higher risk in exchange for a reward. A risk-tolerant individual is more likely to jaywalk to get to the other side sooner by exposing themselves to risk.

2.4 Game Theory

Game theory is the study of mathematical modeling and analysis of games. In this context, a game is any situation in which an agent makes a decision that influences the payoff of the situation. This definition is very broad and encompasses problems that would traditionally be thought of as a game, such as tic-tac-toe [57], as well as problems that would not be traditionally thought of as games such as war [58]. In general, a game can be classified by whether or not it is cooperative or non-cooperative, symmetric or asymmetric, zero-sum or non-zero-sum, sequential or simultaneous, has perfect or imperfect information, and how long it lasts [59]. Often games are represented in an extensive form by a game tree [60], consisting of branches representing choices made by the players of the game, and terminating in leaf nodes that contain payoffs for the players. While not always the case, the typical goal of each player is to maximize their payoff. The solution to many collaborative and competitive games is a Nash equilibrium, which describes the situation where neither player can improve their payoff by changing only their strategy [61–63]. In this dissertation, I draw on various techniques and methods from game theory including strategy stitching [64], techniques for modeling and analyzing infinitely long games, and tree search algorithms often used on game trees [65–67]. These techniques enable me to model and analyze abstract decision-making problems quantitatively.

2.5 Tree Search Algorithms

The proposed work employs various forms of graph search algorithms. Generally, graph search algorithms can be characterized as a Depth-First Search (DFS) [68],

Breadth-First Search (BFS) [69], or a best-first search [70]. Depth-first searches are methods that attempt to travel as far down a path as possible, before returning to the last node for which there are new paths still. A breadth-first search searches all of the nodes connected to a starting node, before moving to all of the nodes connected to the newly discovered nodes. Best-first search methods are much more diverse and generally employ a heuristic equation to determine the quality of the connected nodes before selecting the best one. Examples of best-first searches include the greedy search and *A** [71]. In my work, I apply tree search algorithms to decision trees that I developed. Searching the trees allows me to find good solutions to complex problems, and the faster an algorithm can find a solution the larger the decision-making problems can become.

2.6 Autonomous Systems

Autonomous systems are human-made systems capable of acting on their own without human intervention. Generally, autonomous systems can be described as mobile or immobile. A mobile autonomous system is an autonomous system capable of movement through space. Common examples are autonomous mobile robots [1], self-driving cars [72], or aerial drones [73]. Immobile autonomous systems are autonomous systems that do not move through space. Some common examples include home automation systems [74], digital assistants [75], or robotic systems that stay in one place, such as robotic arms on manufacturing lines. Autonomous decision makers are a type of autonomous system that can be either mobile or immobile but is defined by their ability to make decisions based on inputs.

2.6.1 Autonomous Mobile Robots

William Grey Walter's robotic tortoises, which navigated towards lights in the early 1950s and had control circuits containing vacuum tubes for decision making [76,77] were the earliest autonomous mobile robots, and over the past several decades, autonomous mobile robots have advanced significantly. Modern autonomous mobile robots utilize advanced controls to command their complex systems [1] and have taken a wide variety of novel forms [78–81]. Of particular interest in this work are the National Aeronautics and Space Administration's (NASA) two currently active rovers on the Martian surface: Opportunity and Curiosity [82,83], and of even more interest are the two non-operational Martian rovers, Spirit and Sojourner [84]. Spirit utilized the same design as Opportunity and arrived on Mars at approximately the same time. However, Spirit failed prematurely due to becoming trapped in a dust pit and losing mobility. This shows how the dangers of unknown terrain and the large signal delay before human mitigating action can be initiated are a significant threat in robotic planetary exploration. Additionally, Curiosity is facing a similar problem due to unexpected damage to its wheels from unexpected terrain conditions [85].

2.6.2 Path Planning

Route planning was first developed in the late 1960's [86] and has been used in a wide variety of fields, including transportation infrastructure, aerospace, automotive, and robotics [87–90]. In the field of robotics, route planning can be used in commercial applications in warehouse settings, security, tour guiding, or exploration. NASA researchers developed the OASIS autonomous science system [91] which provides a

method for planning rover scientific activities. OASIS allows for the management of long-term objectives with opportunistic scientific actions while generating a mission and route plan. Many route planning techniques involve the use of optimization techniques [92] to determine the best available path. These specific optimization objective functions [93] can vary, but generally are a non-linear constrained function evaluated the direct linear distance between two discrete points. PHM-enabled route planning has been under development for the past several years [94,95] and has shown to be effective, but has not taken into account varying risk attitudes in their decisions. In my work, I use path planning both in a traditional application so that agents can find the best path when making navigational decisions, but I also adapt techniques from path planning so that they can be used to find a "path" through a decision space where multiple choices need to be placed in order.

3. MATERIALS AND METHODS

Asking a question is an important first step in the scientific process, but a proper experimental design is critical to getting an answer to that question that is insightful, repeatable, and contributes to human progress. In this dissertation, I am not only interested in the design and evaluation of effective autonomous decision-making methods, but also in methods that are poorly suited to the problem. By thoroughly studying poor solutions we can glean important insights to improve the design of good solutions, and by recording and discussing that work, we allow others to learn from our mistakes as well. Some of the evaluated autonomous decision makers discussed in this section have clear flaws but understanding the design of why the decision maker and how the flaws develop is critical to moving the work forward.

3.1 General Approach

The general approach for each case study presented in this dissertation is the same and consists of 1) defining the problem, 2) modeling the problem space in MATLAB [96], 3) design the autonomous decision makers to be evaluated, 4) evaluate the decision makers statistically, and 5) synthesize a better decision making method when appropriate. However, despite this commonality, the three case studies presented are distinct with very little overlap. In the following chapters, we discuss each case study in enough detail to be repeatable by future researchers.

3.1.1 Mathematical Method Selection

For the work presented in this dissertation, I wanted to be as consistent as possible in my approach. To that end, MATLAB [96] was selected as the primary computational

tool used to represent the problem, development of the decision makers, and evaluation of results. MATLAB was chosen over other potential options (such as Python [97], Java [98], or C# [99]) because it is capable of everything from environmental simulation and functional modeling to statistical evaluation with minimal effort compared to other solutions. Additionally, while other tools may have been better suited for specific tasks, structures within them can be overly specialized and make porting to other languages difficult, limiting the ease of application for the work. MATLAB, on the other hand, is relatively simple and ubiquitous and should allow for researchers to easily recreate the scripts and functions used in each case study.

4. RISK-INFORMED AUTONOMOUS ROVERS

Terrain navigation is a common but difficult problem in autonomous decision making with no clear globally optimal solution. To better understand this, try scattering a handful of pocket change onto a large piece of paper. Next, arbitrarily select two coins on the paper. Then try to draw the shortest possible line that starts at the first selected and goes to every coin on the paper before ending at the second coin. You will find that this is not easily done. This is an example of the traveling salesman problem (TSP) [100], which is a problem initial described in the 19th century by W.R. Hamilton and Thomas Kirkman [101]. TSP is what is known as a nondeterministic polynomial time–complete problem (NP-complete), which means that it is impossible to know exactly how long it will take to solve every time, but it will probably take an infeasibly long time to solve [102].

Additionally, the problem is even more complex, because in navigational problems the points traveled to often do not have the same value. Now imagine that you again have to draw the best line between all of the coins, but now the quality of the line is judged by the value of the coin traveled to, divided by the distance between it and the previous coin in the line. This theoretical study begins to approach the problem complexity of this case study, because we will be looking at navigating through a terrain that is heterogeneous and has variable quantities of risk depending on the grain size of the ground cover [103], the incline of the terrain, and the direction of travel.

However, there are still more layers of complexity to consider. In many cases, while we are navigating to target destinations, we are not traveling through a vacuum, and in this case study specifically, we must consider the path found between points. This time

imagine you scattered your change on a piece of graph paper with a random integer between one and five written in each square. Your task is now to draw the best line connecting all of the change while only being able to move one square at a time. The quality of the total path is determined by the value of each coin reached, divided by the sum of the squares traveled between the destination coin and the previous coin.

The final notable challenge is that under real world conditions there is uncertainty in the environment, and while an autonomous decision maker should have an idea of the direction towards the next point, the exact nature of the terrain is often unknowable until it is in the navigator's immediate proximity. In the coin example, this would be represented by only being able to see the 24 closest squares.

This case study is particularly interested in three questions related to navigational decision making:

1) how sensitive is an autonomous decision maker to the range of information in a high-risk environment?

2) how can decision making behavior be tailored to respond better to risk in navigation and exploration?

3) how does the fidelity of information affect performance?

4.1 Problem Definition

In this case study, I look at how terrain navigation problems are affected by the knowledge of the autonomous decision maker and how performance can be affected by

heuristic biases (risk attitude). I am particularly interested in how the information fidelity gap affects the performance of the autonomous decision maker.

The rover will navigate to target destinations within a one square kilometer area. When needed for calculations, I will assume a rover chassis similar in size, weight, speed, and power generation to Opportunity [82,104].

4.2 Importance of the Work

Previous researchers have performed extensive work exploring effective heuristic path planning methods. Common methods include Ant Colony Optimization (ACO) [105], A* (A-star) [106], simulated slime mold [107], actual living slime mold [108], Markov Decision Processes [109], and Monte Carlo Tree Searches [110]. The goal of this case study is not necessarily to outperform these methods on particular metrics. Instead, I aim to increase understanding of the underlying mechanics of how to make navigational decisions facing risk, and how various factors influence decision making (this is the computational level in Computational Cognition [19]). This shift in perspective away from algorithmic development and towards underlying mechanics allows us to explore how problems of information fidelity and heuristic biases influence decision making. The insights gained are not only useful for the development of new navigational algorithms, but also provide insight into autonomous decision making in general, as shown in the Mission Command and Control Problem discussed in Section 5.4.

4.3 Methodology

For this case study, I randomly generate a terrain inspired by the Mawrth Vallis on Mars (a possible future NASA mission site [111]) and use it as the setting of an automated Martian Exploration Rover (MER) mission.

To quantify the hazards that are faced by the rover, I develop functions that analyze potential paths for the possibility of tipping over on an incline, becoming high-centered and potentially trapping the rover, slipping and getting caught in a sand trap, and loss of power due to excessive work needed to follow the path.

I then design and test several autonomous decision makers, including:

- A rover that is only aware of its immediate surroundings (within one meter of itself)

- A rover that only knows what it can observe (five to seven meters around itself)

- A rover with complete knowledge of the entire map
- A rover with intermediate knowledge of the entire map (as if it were gathered from telemetry)

- A rover with intermediate understanding of the entire map, and also knowledge of its immediate surroundings.

I evaluated all of the rovers on their ability to travel between evenly distributed target destinations, and the hazard rate of the path taken. If a rover becomes stuck and does not complete the path it is only scored for the destinations that it reached. Finally, the expected utility of the path is found by taking the product of the number of destinations reached, and the probability of mission survival.

4.3.1 Terrain Generation

Before autonomous decision-makers can be evaluated on their ability to find a path, an environment to navigate through must be created. For this case study, I base the terrain on the Mawrth Vallis region of Mars. This environment is unknown and hazardous, but would also be inactive and would make sense to be treated as a vacuum. Additionally, I wanted to select an environment where sending a robot explorer instead of a human would make sense. Of several potential candidate sites including inhospitable desert environments, heavily polluted EPA Superfund Sites [112], and various potential space mission sites, Mawrth Vallis was selected because it is topographically interesting, and the generated terrains would also be useful for other potential case studies, including the work explored in Section 5.4.

To generate the topography, a diamond-square algorithm (also known as a plasma fractal was used) [113]. The algorithm involves initiating seed elevation values into an array and then alternating between two steps. During a diamond step, seed values making up the corners of a square are averaged and added to a random value. The resulting value is recorded in the middle of the four points, making a diamond pattern. Then during a square step, the four values making up corners of the newly created diamonds are averaged and added to a random value. This value is recorded in the

middle of the diamond, creating a square pattern. Figure 3 shows an example of how this process works.

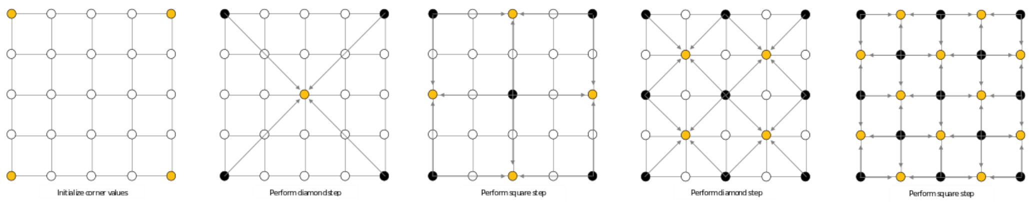


Figure 3: Example of the Diamond-Square Algorithm. Copyright Christopher Ewin [114]

I took seed values from the topographical information of Mawrth Vallis [111]. The values were used to generate a terrain that was 32 km by 32 km with a 10 km wide floodplain cutting through the middle and two 10 km wide 600 m high slopes on either side, from a nine row and nine column seed array, as shown in Figure 4.

600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600
600	400	200	0	0	0	200	400	600

Figure 4: Topographical seed array

To go from a nine by nine array to a 32 km by 32 km terrain with a 1-meter resolution required 12 iterations of the Diamond-Square algorithm. The range of the random number added to the average to create height variation was $\pm$ 100 m on the

first iteration, but was reduced by half after every iteration, giving a final random height variation of 2.44 cm on the final iteration.

To randomly generate the ground cover (dust, sand, rocks, and boulders) a Diamond-Square fractal was used again. This time the seed was randomly generated using a lognormal distribution with a mean grain size of 1.8 mm (sand-sized) and a standard deviation of 2.5 mm (gravel sized) [115]. As a result, the algorithm created a ground cover that was predominately made up of sand and gravel, with occasional patches of dust and larger boulders. For this case study, the rover to explore only a small section of the complete map. A 1 km by 1 km site was selected from the center of the generated site.

Ten Martian valley terrains were generated, and a one square kilometer mission site was taken from each of them, which ensures that the results are not biased due to a specific method performing very well under specific environmental conditions. Figure 5 shows an example of the generated terrain. The map in the top left corner of the figure shows the elevation of the entire valley, the map in the top right is zoomed in on the region outlined in the first map, and the map on the bottom shows the square kilometer highlighted in the second map as a 3D terrain. The green regions are sand and gravel that can easily be traversed by the rover with lighter regions representing finer sand (0.2-0.63 mm diameter) and darker regions representing coarse gravel (20-63 mm diameter). The yellow to red regions are larger stones that are difficult to traverse, with cobble (63-200 mm diameter) shown in yellow and large boulders (greater than 630 mm) shown in dark red. Fine grain sand that could potentially trap the rover (0.06-0.2 mm) is shown in light blue with dark navy representing dust (less than 0.002 mm).

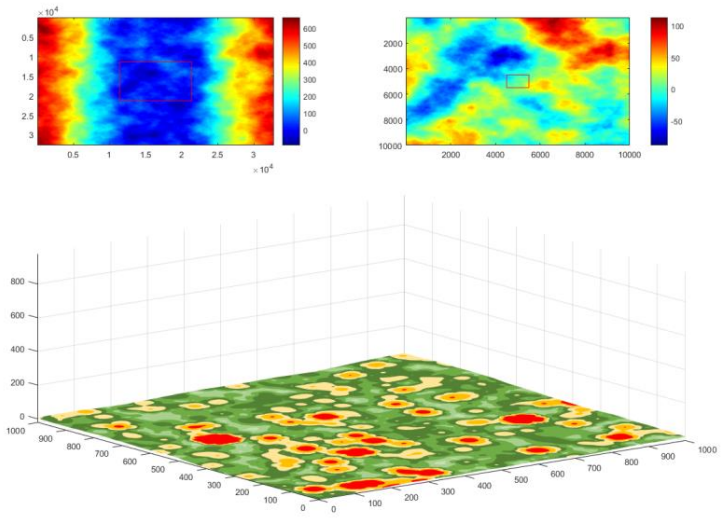


Figure 5: Example of generated Terrain

4.3.2 Failure Modes of Interest

To create a decision maker capable of making informed decisions facing risk, we have to define what the risks are. For the case study, I considered four potential sources of risk:

1. Tipping over while driving on uneven terrain
2. Becoming high centered on unexpected high rocks
3. Losing traction and getting trapped when trying to drive over very fine grain terrain
4. Using more power than can be generated from solar cells and draining the battery

Functions were developed to represent the probability of occurrence for each of these failure scenarios. The probability of tipping, shown in [3, is a cumulative

distribution function of the mean ground cover grain size g, the standard deviation of the grain size$s$, the incline of the terrain ϕ, and the width of the rover w, and it is assumed that the rover will tip over if the angle between the rover and the direction of gravity is greater than 30°.

$$P_t = 1 - \frac{1}{2}\left(1 + \frac{2}{\sqrt{\pi}} \int_0^{\frac{(w*\sin(30-\phi))-g}{s\sqrt{2}}} e^{-x^2} dx\right) \quad [3]$$

The probability of the rover becoming high-centered, shown in [4, is a cumulative distribution function of the mean ground cover grain size g, the standard deviation of the grain size$s$, and the clearance of the rover c.

$$P_h = 1 - \frac{1}{2}\left(1 + \frac{2}{\sqrt{\pi}} \int_0^{\frac{c-g}{s\sqrt{2}}} e^{-x^2} dx\right) \quad [4]$$

The probability of slipping, shown in [5, is a cumulative distribution function of the mean ground cover grain size g, the standard deviation of the grain size s, the maximum expected grain size m, the minimum expected grain size n, the incline of the terrain ϕ, and the width of the rover w. It is assumed that the maximum friction angle of the ground cover is being exceeded and causing the slipping condition of the tires [116–118].

$$P_s = 1 - \frac{1}{2}\left(1 + \frac{2}{\sqrt{\pi}} \int_0^{\frac{\left(\tan^{-1}\frac{g}{w}+\phi\right)-(1.4*\log m-1.4*\log n)/2}{(1.4*\log m+1.4*\log n)\sqrt{2}}} e^{-x^2} dx\right) \quad [5]$$

The probability of power loss, shown in [6, is the probability that the power used traversing the path will exceed the power generation capacity of the rover, and is a function of the mean power consumption p, the standard deviation of the power consumption σ, and the rate of solar power generation for the rover γ.

$$P_p = 1 - \frac{1}{2}\left(1 + \frac{2}{\sqrt{\pi}}\int_0^{\frac{\gamma - p}{\sigma\sqrt{2}}} e^{-x^2} dx\right) \quad [6]$$

The union of these four probabilities is taken and then divided by the time taken to drive the path t, resulting in the expected hazard rate, as shown in [7.

$$\lambda = P_t \cup P_s \cup P_h \cup P_p = \frac{1 - (1 - P_t) * (1 - P_s) * (1 - P_h) * (1 - P_p)}{t} \quad [7]$$

4.4 Autonomous Decision Makers

Over the course of this study, I evaluate five autonomous decision makers on ten randomly generated terrains. I evaluate each decision maker on its ability to reach five-goal locations while minimizing the risk of the path taken. Additionally, for each decision maker, I examine how biasing its decision with a risk attitude affected its performance [38,49,50,54,55].

All four rovers are evaluated on two metrics: probability of mission success, and expected mission completion, which is the product of mission success and target locations reached before stopping divided by the total number of target locations.

4.4.1 One-Meter Information Range

The first autonomous rover is the simplest but also the most limited. This rover can only consider the conditions of the terrain within a 1-m information range of its current location and can only make decisions to navigate to points within that space.

The rover uses an objective function, shown in [8, to evaluate its options. The objective function uses the information range $R\ [m]$, the distance between the current location and the goal $D_L\ [m]$, the distance between the target point (the location being evaluated) and the goal $D_T\ [m]$, the hazard rate between the current location and the target location λ, and the distance between the current location and the target location $\Delta LT\ [m]$. This is biased by the risk aversion factor ζ to give the rover a variable risk attitude. As ζ increases, the rover will weigh the risk as higher compared to the reward, making the rover less risk tolerant and more risk-averse. For this version of the decision maker, $R = 1\ m$ because the information range is limited to one meter.

$$\Omega = \left|\frac{(D_L - R) - D_T}{\big((D_L - R) + D_T\big)/2}\right| + \zeta \frac{\int_L^T \lambda(l)\, dl}{\Delta LT} \qquad [8]$$

The objective function was used along with an A* algorithm (a special kind of greedy best-first algorithm [106]) to perform the navigation. Figure 6 summarizes this process consisting of six steps:

1) Determine the goal location with respect to the rover
2) Examine the points on the map within the information range (the map has a 1-m resolution in the x-y plane) and take note of their elevation, ground cover, and distance from current location

3) Determine the quality of each point using the objective function
4) The decision maker selects the best point as determined by the objective function
5) The rover moves to the selected point
6) The process repeats from the first step until reaching the goal at which point the goal moves to the next goal location.

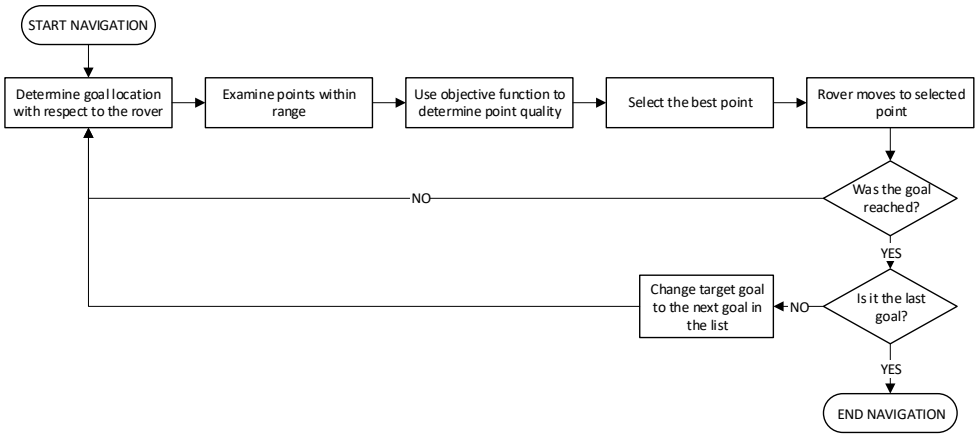
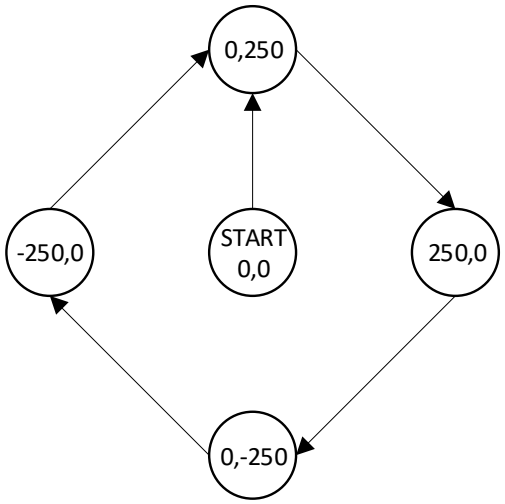


Figure 6: One-Meter Information Range Navigation

The rover will attempt to drive a diamond-shaped route, starting from its center. The rover will first drive north to the first target, 250 meters north of the starting location. The rover will then have to go south-east towards the second location, located 250 meters east of the starting location. Next, the rover will have to drive south-west towards the third goal, located 250 meters south of the starting location. After that, the rover drives north-west towards a location 250 meters west of the starting location. Lastly, the rover drives north-east until it returns to the location 250 meters north of the starting location. Figure 7 shows the direction of travel between goals and lists coordinates in meters east and north of the starting location.



Goal	East (m)	North (m)
1	0	250
2	250	0
3	0	-250
4	-250	0
5	0	250

Figure 7: Diamond Path Goal Locations and Order

If the rover gets stuck and is no longer making progress towards the goal, then the test is stopped, and only the section of path completed is recorded. This rover was evaluated for risk tolerances ranging from $\zeta = 0$ to $\zeta = 1.5$ with a resolution of 0.01.

4.4.2 Five-Meter Information Range

The second autonomous rover is similar to the first, except it has a larger information range. This rover can consider the conditions of the terrain within a 5-m information range of its current location and can make decisions to navigate to any point within that space.

The rover uses the same objective function as its one-meter counterpart, shown in [8, to evaluate its options. The objective function uses the information range R $[m]$, the distance between the current location and the goal D_L $[m]$, the distance between the target point (the location being evaluated) and the goal D_T $[m]$, the hazard rate between the current location and the target location λ, and the distance between the current location and the target location ΔLT $[m]$. The objective function is biased by the risk

aversion factor ζ to give the rover a variable risk attitude. As ζ increases, the rover will weigh the risk as higher compared to the reward making the rover less risk tolerant and more risk-averse. For this version of the decision maker, $R = 5\ m$. Like the one-meter range rover, the objective function was used along with an A* algorithm to perform the navigation. This process is summarized in Figure 6.

This rover will also attempt to drive a diamond-shaped route, starting from its center. The rover will first drive to a goal location 250 meters north of the starting location. The rover will then drive south-east to a goal located 250 meters east of the starting location. Next, the rover will have to drive south-west to a goal 250 meters south of the starting location. After that, the rover drives north-west towards a location 250 meters west of the starting location. Finally, the rover drives north-east until it returns to the location 250 meters north of the starting location. Figure 7 shows the direction of travel between goals and lists coordinates in meters east and north of the starting location.

As with the one-meter rover, if this rover gets stuck and is no longer making progress towards the goal, then the test is stopped, and only the section of path completed is recorded. I also evaluated this rover for risk tolerances ranging from $\zeta = 0$ to $\zeta = 1.5$ with a resolution of 0.01.

4.4.3 Low-Resolution Map with Five-Meter Information Range

The final autonomous rover is the most complicated and is based on insights gained from the previous rover studies. This rover starts with a low-resolution map of the entire

environment and pre-plans a global path using an Ant Colony Optimization (ACO) [105]. The rover then maps those points onto its environment as intermediate goals. The rover then navigates through the environment with an information range of 5 meters (on the full resolution map) and finds a local path between the global planned points.

During the global path planning phase, an ACO is used to find a low-risk path. ACO works by creating a large number of virtual ants and allowing them to wander randomly around the space. The ants that reach the objective lay down a pheromone trail, and the higher the quality of the path the ant took, the stronger the pheromone trail. After all of the ants have explored the space the pheromones evaporate, making the trails slightly weaker. Next the algorithm iterates by releasing another group of ants. These ants wander randomly, but their random wandering is weighted towards strong pheromone paths. These ants also lay down new paths that add to the existing paths. This process repeats for as many iterations as desired or until a halting condition is met. The best way to implement an ACO for this problem is to record the best path found so far which should improve over time as more iterations are run. However, over time, the paths should constantly improve and eventually converge towards the best path, meaning that you can also take the strongest pheromone path and get similar results. While an ACO is a very effective meta-heuristic algorithm for tree searches and navigational problems, it is not very computationally efficient as it requires a large number of iterations. Therefore, an ACO may not be feasible for in-situ navigation problems compared to faster methods. However, it is an effective method for

applications where the planning phase can be any length of time, or computational costs are not relevant, such as if you are preparing for a planetary exploration mission.

The equation for the pheromone path is the hazard rate over the entire path λ (calculated using [7) multiplied by the total length of the path L, as shown in [9.

$$w_p = \lambda \cdot L \quad [9]$$

Additionally, this method needs an equation to weight the random navigational choices of the ants. The weight assigned to each of option available to the ants is found by taking the pheromone weight for each point w_p and adding that to an exploration constant ρ. This is then divided by the sum weight of all options, as shown in [10.

$$W_o = \frac{w_p + \rho}{\sum w_p + \rho} \quad [10]$$

During the local navigation stage. the rover uses the same objective function as the one- and five-meter information range rovers shown in [8, to evaluate its options. For this version of the decision maker, $R = 5\,m$. For the local navigation, the objective function was used along with an A* algorithm to perform the navigation. This process is summarized in Figure 6.

The rover will attempt to drive a diamond-shaped route, starting from its center as shown in Figure 7, except there are intermediate locations along the way, between the five goal locations. If the rover gets stuck and is no longer making progress towards the goal, then the test is stopped and only the section of path completed is recorded. This rover was evaluated for risk tolerances ranging from $\zeta = 0$ to $\zeta = 1.5$ with a resolution of 0.01.

4.4.4 Variable Information Range

This autonomous rover uses the same risk attitude during every run, and instead the information range varies from 1 m to 15 m. However, it is still evaluated on how well it performs in the generated Martian environment, and how well the rover navigates to points within that space.

The rover uses the same objective function and variables as the five-meter information range rover, shown in [8, but one each run it increases the range value R. The objective function was used along with an A* algorithm to perform the navigation as summarized in Figure 6. Like the other rovers, this one will attempt to drive a diamond-shaped route, starting from its center as shown in Figure 7. If the rover gets stuck and is no longer making progress towards the goal, the test stops, and only the section of path completed is recorded. This rover was evaluated for information ranges from $R = 1\ m$ to $R = 15\ m$ with a resolution of 1 m.

4.5 Results

Four experiments were carried out. In each experiment, an autonomous rover navigated through a simulated Martian environment. The first three rovers increased their risk aversion after every run. The fourth rover increased its information range after each run. All rovers were evaluated for the probability of mission success and expected mission completion for each run.

4.5.1 One-Meter Information Range

The first autonomous rover had a one-meter information range and attempted navigation with risk aversion between 0 and 1.5 (at a resolution of 0.01) in ten randomly generated environments.

For the probability of mission success, the general trend was that the probability would increase dramatically as risk aversion began to rise, and then peak at around a risk aversion of 0.1 and a probability of success of 0.65. After that, it would slowly decrease until the rover would become too risk-averse, and unable to complete the mission. The plot in Figure 8 shows the probability of mission success across all ten maps. Excluded are the points where the rover did not complete the mission.

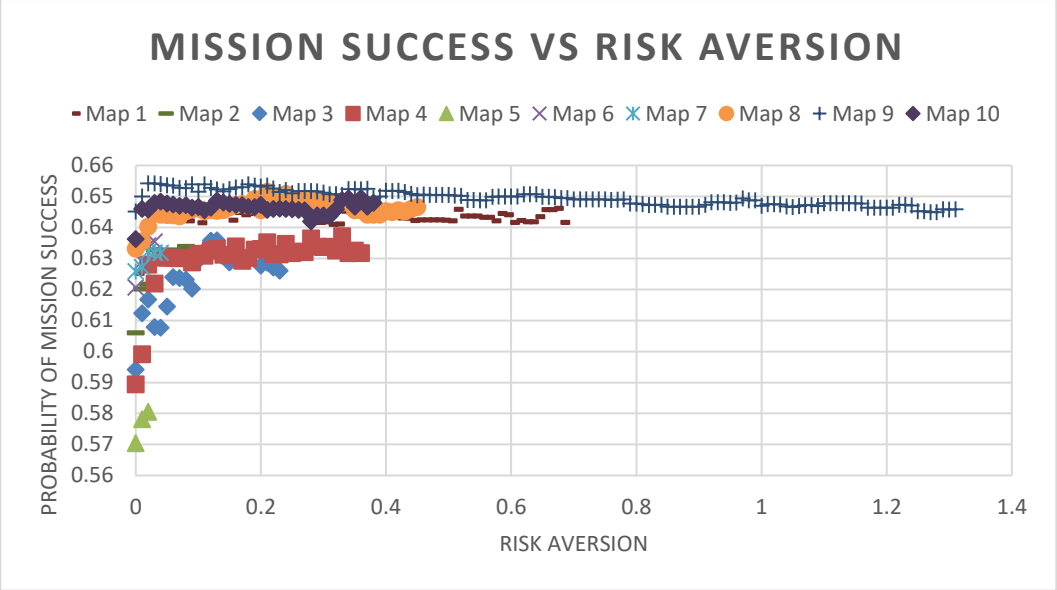


Figure 8: One-meter rover mission success vs. risk aversion on all maps

Expected mission completion is found by taking the product of the probability of mission success and the goals reached before stopping, and then dividing it by the total number of goals.

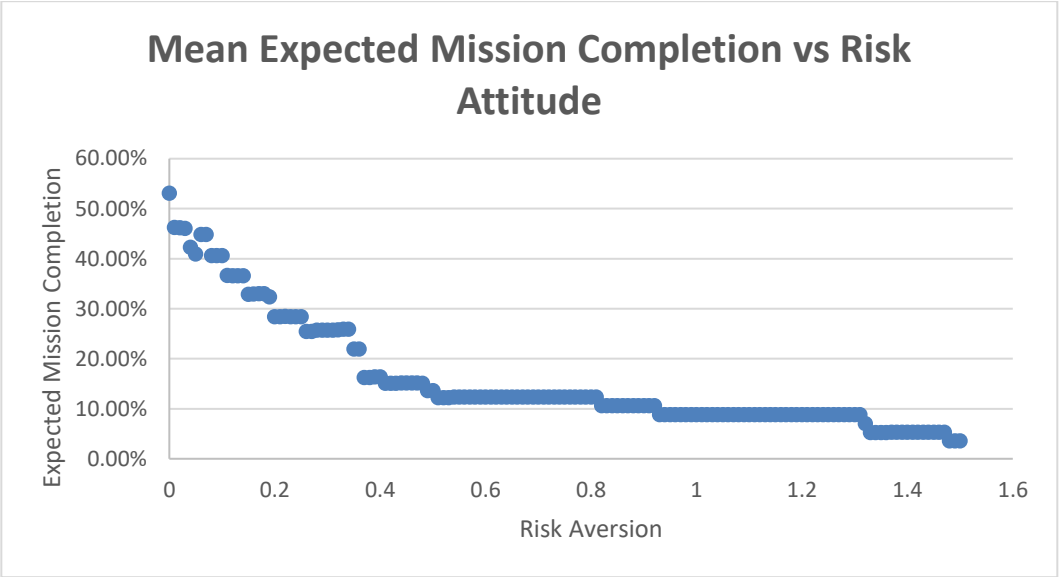


Figure 9: One-meter rover expected mission completion vs. risk attitude

4.5.2 Five-Meter Information Range

The second autonomous rover had a five-meter information range and attempted to navigate between the target points with risk aversions ranging from 0 to 1.5. For the probability of mission success, the five-meter rover is not notably better than the one-meter rover and shares a lot of the same characteristics as the one-meter rover as shown by the graph in Figure 10.

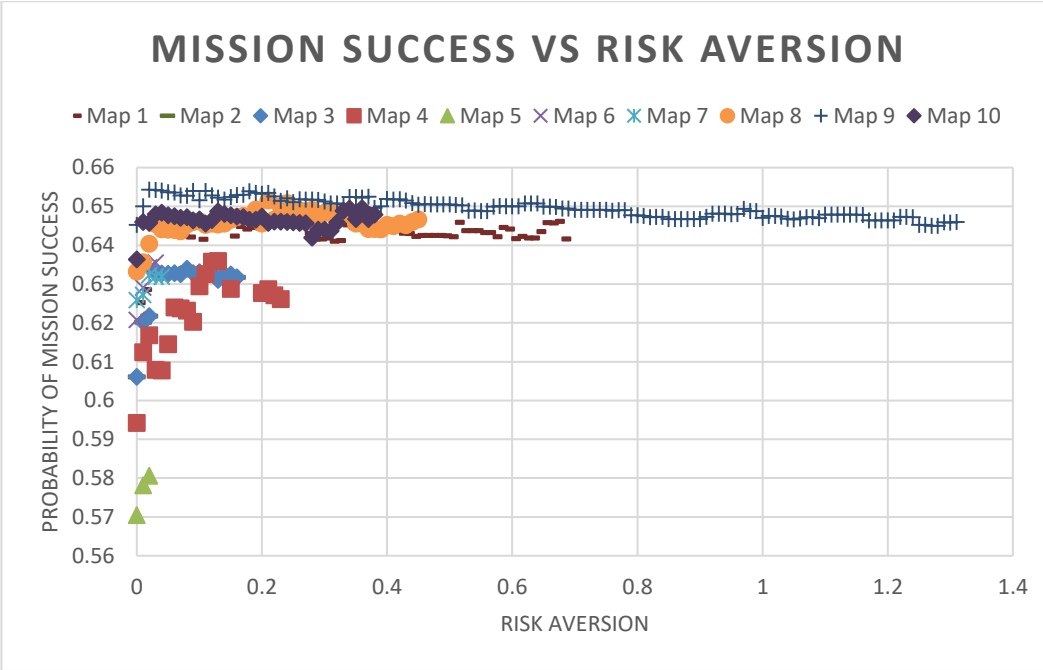


Figure 10: Five-meter rover mission success vs. risk aversion on all maps

At first glance, the mean expected mission completion of the five-meter rover seems to follow the same general trends as the one-meter rover. However, as shown in **Error! Reference source not found.**, the five-meter rover's mean expected mission completion is approximately 10% higher across the entire range of risk attitudes. This means that this rover would be expected to complete 10% more mission objectives before experiencing failure or requiring human intervention.

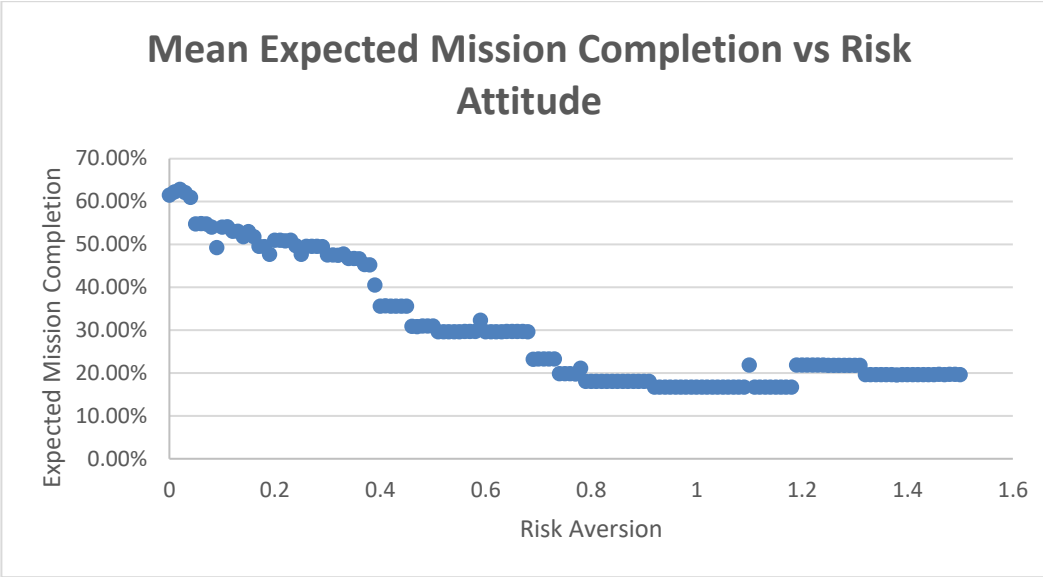


Figure 11: Five-meter rover expected mission completion vs. risk attitude

4.5.3 Low-Resolution Map with Five-Meter Information Range

The third rover starts its navigation with a low-resolution map of the mission environment and then has a five-meter information range at which it could gather higher resolution data. Like the first two rovers, it has a variable risk attitude ranging from 0 to 1.5.

The most notable result here is that as the risk attitude increases, the expected mission completion only reduces slightly. This is a result of the rover being less likely to get stuck, because it started with an initial path that was known to be possible.

Figure 12: Rover with low-resolution map mission success vs. risk aversion on all maps

Figure 13: Rover with low-resolution map expected mission completion vs. risk aversion on all maps

4.5.4 Variable Information Range

The final rover has a constant risk attitude of 0.15 and a variable information range from 1 to 15 meters. As the range increased the mission success initially increased and then leveled out. This same trend was observable in the expected mission completion as information range increased.

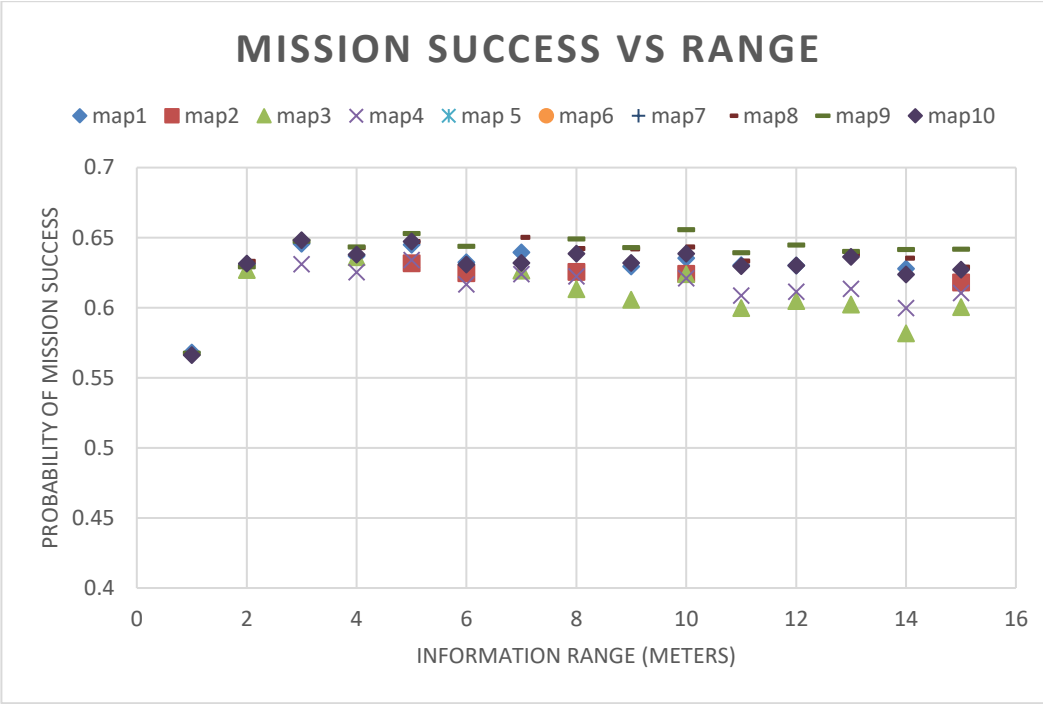


Figure 14: Variable range rover probability of mission success vs. risk information range

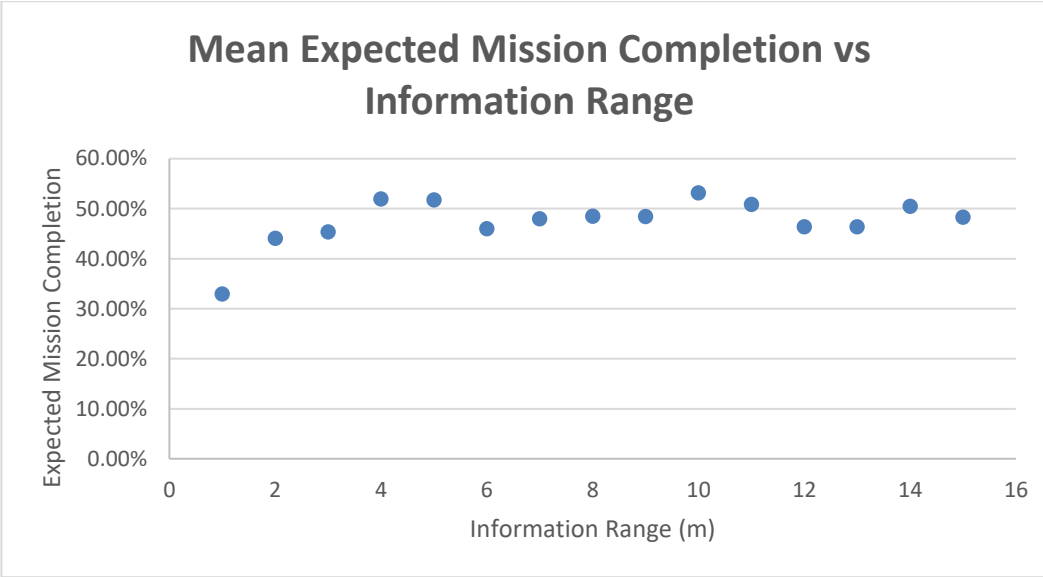


Figure 15: Variable range rover expected mission completion vs risk information range

4.6 Discussion of Results

The work on risk-informed decision making focused on two problems: the first was how risk attitudes play into decision making, and the second was understanding what information is needed to make decisions facing risk. A risk attitude is a bias towards an acceptable level of risk. Generally, a risk attitude can be described on a spectrum between risk aversion and risk tolerance. A risk-averse attitude is a bias against accepting risk for a particular level of reward; for example, a risk-averse attitude towards crossing the street would involve waiting for the appropriate lights and looking both ways before crossing at a crosswalk. A risk-tolerant attitude is a bias towards accepting a reward for increased risk; for example, a risk-tolerant attitude towards

crossing the street could be jaywalking to save time and effort. The study of biases in decision making will enable future work in autonomous decision making.

Risk Attitude Informed Route-planning (RAIR) was a method initially developed to study how various risk attitudes would perform when navigating through a high risk and unknown environment [49]. The case study was formulated as an autonomous Martian rover simulated in the Simulated Physics Environments for Autonomous Rover Studies (SPEARS) [119]. This work concludes that an optimal risk attitude exists that accepts enough risk to complete the mission, without accepting so much risk that the rover would have destroyed itself. A second question also emerged from the work, which was how did the rovers local omniscient view of the environment influence its decisions, as an understanding of the environment not gathered through direct sensing of the rover's systems could be faulty [85]. A modified version of RAIR has also been developed that included multiple risk attitudes for different types of risks [120].

To address the question of information fidelity and the information fidelity gap (the difference between model and reality), two follow-up studies were conducted. The PHM-Informed Damage Aversion Algorithm (PIDAA) was a blind kitten approach to the problem in which all visual sensors were removed from the simulated rover, and its navigation could only consider relative GPS coordinates and the information gathered through PHM models of its sub-systems interacting with the environment. Predictably, the blind kitten approach performed poorly and illustrated how decisions made on different sides of the information fidelity gap are faulty. To directly address the information fidelity gap, Global-to-Local Path Finding and Design Operation Exploration (GLPFDOE) was developed [38]. The global-to-local nature of GLPFDOE

refers to the strategy employed to bridge between the three levels of information fidelity: the low fidelity prior environmental model, the medium fidelity observable environment, and the high fidelity operating environment. GLPFDOE initially creates a mission plan by analyzing a low-fidelity model developed from simulated satellite data. The rover then attempts to execute the plan while gathering both limited visual data and PHM system health data to inform deviation from the plan as necessary. The rover is then judged on its performance based on the high-fidelity simulation model. The published work on GLPFDOE examined both the global-to-local approach, as well as an FFDF-ID design study [38].

5. THE COLOR GRAPH PROBLEM

Design automation is the study of autonomous decision makers performing computational design synthesis [35,121–127]. One of the major hurdles of studying design automation is that most methods are developed—and only ever evaluated—for a single domain-specific case study. This slows the progress of design automation work considerably because it scatters our work into domain-specific publications and conferences. This slows the generation and evaluation of novel methods by only ever testing them on real-world applications that are hard to implement and computationally expensive, and robs us of the ability to easily compare methods on even footing, making it difficult to determine the most useful algorithms for an application.

This case study explores the problem by taking a game theoretic approach and developing an extensive form [59] of a common class of design automation problems called the *design decision tree*. A computationally simple version of this problem is then developed and used to evaluate several common methods applied to design automation problems.

5.1 Problem Definition

The primary focus of this case study is representing the design decision tree in an extensive form that captures the properties of multimodality, opaque intermediate states, unboundedness, and increasing internal locations and then showing that a simple design problem can be developed that captures those properties.

5.2 Importance of the Work

This work has important implications for the field of automated design because it: 1) describes a class of problem that is rarely studied in the abstract, 2) establishes standard evaluation metrics for design decision tree problems, and 3) presents findings that are generalizable to many real-world problems.

Additionally, while the exploration of design decision trees is academically interesting on its own, this work is generalizable to many real-world applications. Automated design of mechanical systems [128–131], trusses [132], architecture [133], Metal-Organic Frameworks [121], and other problems that share the properties of multimodality, unboundedness, opaqueness, and increasing internal locations [122,134] are all enabled to be faster, with a lower computational cost, and higher quality.

5.3 Methodology

In this case study, I derive the design decision tree from a common class of design problems. Next, I develop the Color Graph problem as an example of a design decision tree. Then I evaluate various common methods for performing automated design for their ability to design a Color Graph, including:

- A Monte Carlo Tree Search (MCTS) [66]
- A standard Genetic Algorithm (GA) [135]
- A modified Genetic Algorithm with no Crossover (GAnX)
- An Ant Colony Optimization (ACO) [67]

- A purely random control method was employed and evaluated as a baseline for comparison.

5.3.1 The Design Decision Tree

Design decision trees, like the color graph problem described below, have four unique properties: multimodality, opaque intermediate states, unboundedness, and increasing internal locations.

The design decision tree is *multimodal*, meaning that multiple local maxima and minima exist. The multimodality of the design tree makes finding the global optimum (best possible solution) difficult as local minima may exist at different levels and different branches of the tree. For example, when designing a chair, a seat with a cushion on the floor is a simple but good design that is higher quality than the same seat with a single leg on the bottom of the seat. However, adding a stable number of legs to the bottom of the seat would be preferable to a seat on the ground.

The second property is *opaqueness*. The defining characteristic of opaqueness is that only completed designs can be evaluated and reliable information about the final design cannot be gleaned from intermediate states. For example, attempting to evaluate a potential future chair by how comfortably someone could sit on a single one of its legs is neither useful for a design algorithm nor enjoyable for the person doing the sitting.

The design decision tree is also *unbounded*, meaning that potentially infinite decisions can be made in the design space. Using the chair example, the property of unboundedness means that a chair could be made with armrests, and the armrests could

have cup holders, and there can potentially be an unreasonably large number of cup holders. The unboundedness leads into our final property - increasing internal locations.

Increasing internal locations means that as decisions are made in the design decision tree, it creates new locations from which additional subsequent decisions can be made, resulting in a greater number of potential choices as the candidate solutions increase in complexity. Again, using the chair example, every time an armrest is added the number of current potential design decisions is increased by the new potential of adding a cup holder. To explore this problem without devoting resources unnecessarily to modeling a domain-specific problem, a simplified representative problem is used in initial explorations. Additionally, in this dissertation, it is shown that the design decision tree not only encompasses artifact design but also applies to mission design, and that techniques that work well on the toy problem are easily adaptable to work well in other applications.

5.3.2 Color Graph

The design decision tree differs from typical tree-search spaces commonly explored by computer scientists. Typically trees can be described as game trees [61,65,136] of path-planning problems [137–139]. Game trees can often be non-monotonic and not all the nodes are evaluable (creating opaqueness), but end states are strictly defined, which bounds the tree. Path-planning trees contain only evaluable nodes and are monotonic as the end goal is approached. Another difference is that both the path-planning trees and game trees tend to have constant branching factors. As a result of the design decision tree's increasing number of internal locations, the branching factor is linearly

increasing, and the number of candidates per level of the tree is increasing factorially, making the search space much larger than path-planning or game trees.

5.3.2.1 Constructing a Color Graph

A Color Graph is a directed graph composed of a seed node, n_o, to which red, orange, yellow, green, blue, or violet colored nodes are added. In addition to the seed node, colored nodes can be added to other colored nodes already existing in the graph. Figure 16 shows three examples of Color Graph candidates found in the search tree.

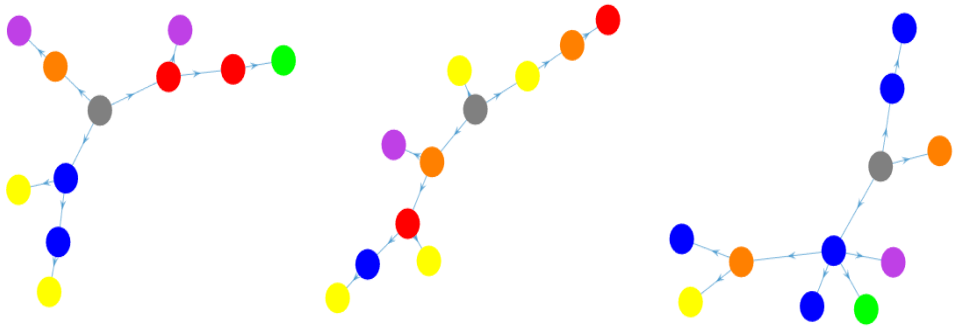


Figure 16 Three examples of a Color Graph

The design decision tree for a Color Graph alternates between location decisions and color decisions. There are six branches off of the tree root representing the six potential colors for the added node, as shown in the first image of Figure 17. From each of those six color options, there are two branches representing potential locations in the Color Graph to add the next node; this can be seen in the second level of the second image in Figure 17. One for the Color Graph seed node, n_o, and one for the first node added, n_1. From each of those location options, the design decision tree branches again with the six color options. The design decision tree continues to repeat this way between color and location, with the number of location options increasing every time

a node is added to the Color Graph. Figure 17 shows the repeating structure of the design decision tree and corresponding Color Graph for the first four nodes added.

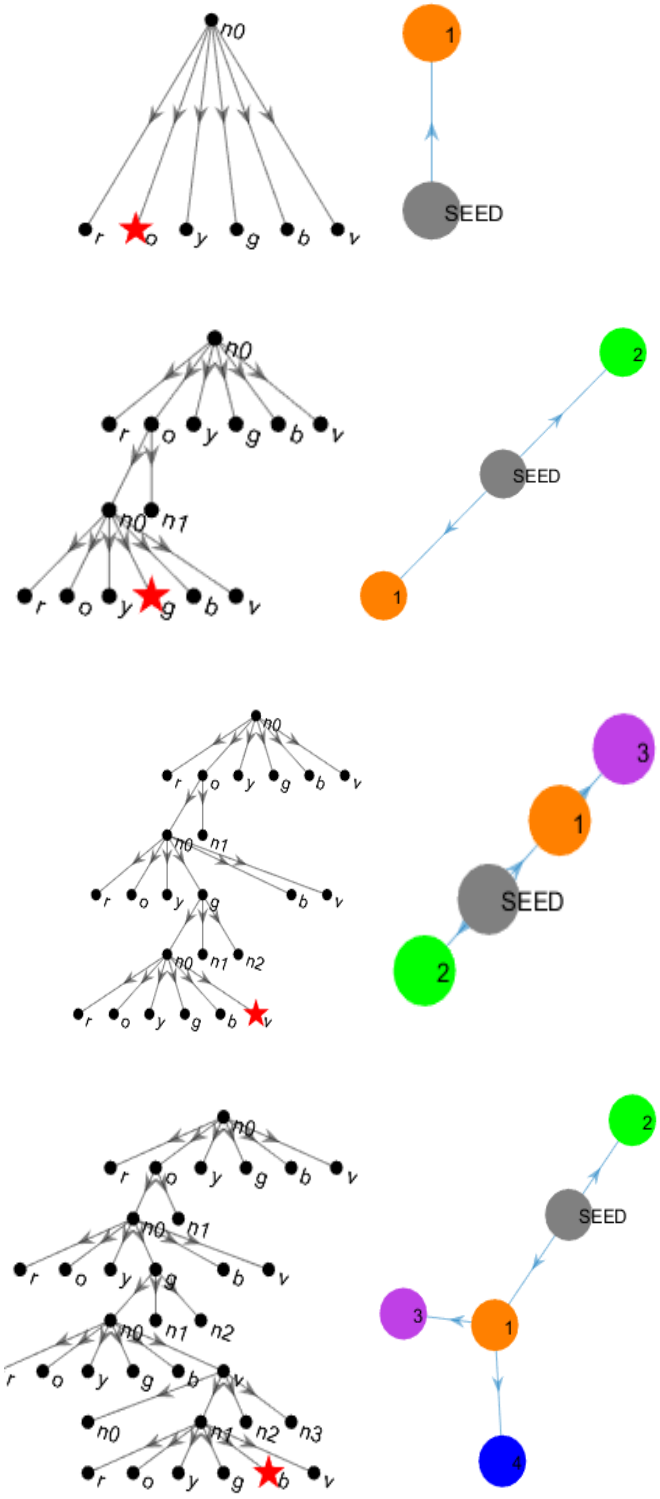


Figure 17 Design Decision Tree and Corresponding Color Graph

5.3.2.2 Evaluating a Color Graph

The edges of a Color Graph design determine its quality. The edge's scores are determined by the source node and target node of each edge. For the case study presented in this chapter, the edges have three arbitrary properties: α, β, and γ, with a range of -5 to 5. These edge properties are independent and are determined by a randomly generated, such as the one shown in Figure 18. When a completed Color Graph is evaluated, the scores for each edge are summed giving a three-dimensional score $[\Sigma\alpha, \Sigma\beta, \Sigma\gamma]$. Then, the design is rated on its proximity to a target score, which was $[0,0,0]$ for this paper. Table 1 shows example edge scores for a Color Graph, and an example Color Graph is scored based on its edges in Figure 18.

Table 1 Edge Properties

		Target Node					
		α					
		Red	**Orange**	**Yellow**	**Green**	**Blue**	**Violet**
Source Node	**Red**	-3.15	3.83	-0.71	3.18	3.27	-3.49
	Orange	3.56	3.45	-4.28	4.09	1.92	-1.10
	Yellow	3.72	-4.54	2.93	-3.12	-4.44	0.85
	Green	1.14	0.76	1.13	-0.04	-3.27	1.43
	Blue	0.30	-4.27	-1.79	2.81	1.42	-4.48
	Violet	0.93	-3.20	-4.42	3.99	4.58	4.47
	Seed	-1.51	-0.22	-0.64	-3.32	-4.89	-2.38

		β					
		Red	**Orange**	**Yellow**	**Green**	**Blue**	**Violet**
Source Node	**Red**	0.38	1.06	-1.94	1.85	-2.20	-0.37
	Orange	1.55	-1.42	0.54	-0.96	4.06	-2.65
	Yellow	0.18	3.95	-4.32	-2.88	-3.94	-3.00
	Green	-2.37	0.66	0.54	2.31	-1.89	4.99
	Blue	4.56	-2.19	-4.55	1.45	2.52	-3.87
	Violet	-0.40	-4.58	-0.39	-0.71	-0.64	1.59
	Seed	1.34	-3.98	3.83	0.47	1.08	4.99

		γ					
		Red	**Orange**	**Yellow**	**Green**	**Blue**	**Violet**
Source Node	**Red**	-0.06	-2.46	2.57	-3.85	-3.62	-2.53
	Orange	-2.82	2.11	0.74	2.83	4.89	1.81
	Yellow	1.88	1.33	-0.24	-4.92	0.98	-3.91
	Green	0.74	2.62	1.65	2.80	1.22	1.29
	Blue	4.15	-3.46	1.03	4.29	3.66	4.23
	Violet	-3.41	0.96	4.89	-1.42	-4.40	-0.16
	Seed	0.09	-4.52	3.58	0.86	3.58	-2.48

The edge between the source n_0 (the seed node) and the target n_1 (the orange node) has an α value of -0.22, a β value of -3.98, and a γ value of -4.52. We sum these with the edge scores from the remaining four edges and get totals of $\alpha = -2.72$, $\beta = -2.10$, and $\gamma = 3.04$ or as a three dimensional coordinate in a design space [-2.72, -2.10, 3.01]. We compare this to our arbitrary target design score of [0,0,0] and determine the quality of the design by its proximity to the target. This can be found by calculating the distance between the two points using Euclidean distance, giving the

design a final quality score of 4.59. A perfect score would be a 0 meaning that the design perfectly recreated the target design.

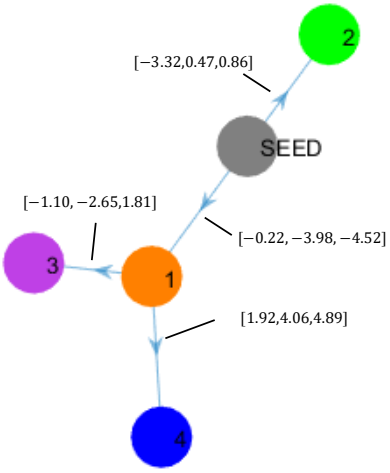


Figure 18 Color Graph Edge Properties

5.3.2.3 Characterizing the Color Graph Problem

The objective of the Color Graph problem is to design an arbitrary directed graph that possesses selected design parameters. A scoring algorithm that evaluates a Color Graph based on its edges and nodes is used to determine the quality of the design. The Color Graph problem captures all four of the properties of interest in the design decision tree. The design objective function for evaluating solution quality is multimodal and non-monotonically related to the number of nodes present in the Color Graph. This is shown by the Color Graph in Figure 17 and the edge properties in Table 1. If these were only the first four nodes of a larger design and a designer added a red node to n_1, then the design quality will improve, but if a designer added a violet node to n_1 the quality of the design will decrease.

This problem is compounded by the opaqueness of the design decision tree. The Color Graph design process is opaque as the final design performance cannot be determined from the performance of an incomplete Color Graph design. In many real-world cases, opaqueness exists in designs that are not immediately or intuitively obvious to a human, due to a large number of sensitive design variables and multimodality of the design. The Color Graph design decision tree is unbounded because it has no set limit on the number of nodes that can be added to the Color Graph.

Lastly, Color Graph possesses an increasing number of internal locations, as the locations where nodes can be placed increases with the number of nodes already present in the Color Graph. This gives the design decision tree for the Color Graph a branching factor of $n \times 6$, making its growth both geometric and factorial. Figure 19 shows a sub-section of the Color Graph design decision tree with 9 levels (selecting node color 5 times and node location 4 times).

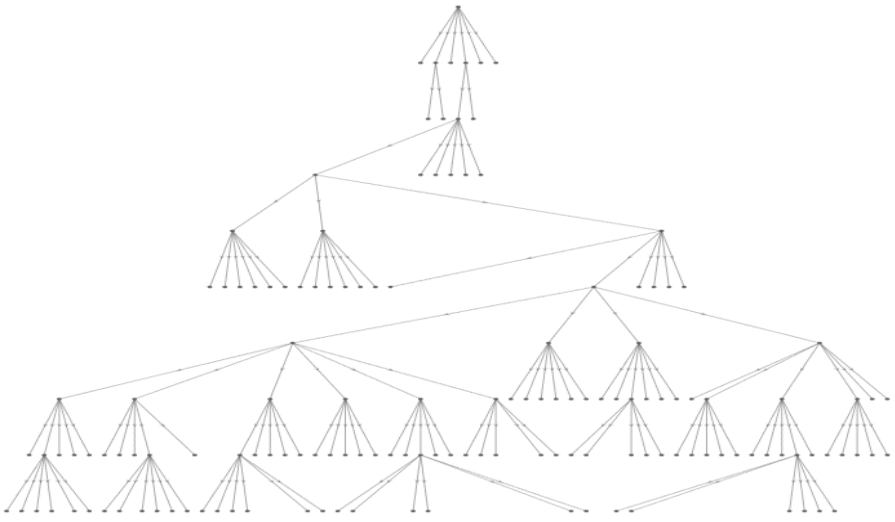


Figure 19: A Section of the Design Decision Tree for a Color Graph

5.3.3 Evaluating Autonomous Design Methods

AI tree-search methods are a common approach to creating generative designs. One advantage of this approach is that representing a design problem as a decision tree allows for exploration of a design space with a greater level of complexity compared to standard numerical optimization methods. However, the problem of searching a design tree for good solutions often necessitates the evaluation of thousands or millions of tree nodes, and the algorithm time and memory requirements are usually a function of the size of the search tree.

This chapter evaluates four algorithms on their ability to generate Color Graph designs: a Monte Carlo Tree Search, a Genetic Algorithm, a Genetic Algorithm with no crossover, and an Ant Colony Optimization. We implemented all of the methods in the 2017 edition of MATLAB [96]. The evaluation consists of algorithms designing Color Graphs where 3 nodes, 5 nodes, and 10 nodes were added to the seed.

Each algorithm attempts graphs of these three sizes to represent the unbounded nature of the design tree, while still setting a consistent end condition and design constraints. Additionally, due to the unboundedness and the expanding branching factor the design decision tree, it is important to examine how algorithms perform as the design complexity increases. For comparison, the 3-node design decision tree has 1296 potential completed solutions, the 5-node design decision tree has 933120 completed solutions, and the 10-node design decision tree has 2.2×10^{14} completed solutions.

Each algorithm attempted was applied to each Color Graph size. Ten trials were performed for each algorithm leading to 150 trials in total for all five algorithms to

attempt all three sizes ten times. Before the experiment, ten sets of random edge properties were generated, one for each trial. This ensures that the results would not be biased due to a single set of properties favoring any one of the algorithms, and allows for a fair comparison to be made across methods and graph sizes.

The primary metrics of interest are the quality score of the designed graphs, the duration of time needed to reach a final design solution, and the number of times Color Graph design evaluations were performed during the process. Additionally, we are interested in the number of iterations before the best solution was discovered for the GAs and ACO, and the depth policy generated by MCTS before it ran out of time. The best Color Graph for each trial was stored in a cell array for later review and verification of results. In addition to the Color Graphs generated by the algorithms, purely random Color Graphs will be generated and analyzed as a baseline for comparison.

5.3.3.1 Monte Carlo Tree Search

A Monte Carlo Tree Search (MCTS) [65,66,140] is a class of heuristic tree search algorithms [141,142] that is commonly used for analysis in game-playing scenarios. The MCTS method consists of four steps: 1) selection, 2) expansion, 3) simulation, and 4) back-propagation.

During the selection step, MCTS uses a policy developed during the previous iteration to select what is currently believed to be the best option. This process starts from the root of the design decision tree (representing the location of the seed node in a Color Graph) and continues until a leaf is reached in the design policy. When a leaf is reached, that means that the policy does not have information for subsequent choices in the design decision tree. This means that expansion is required.

During the expansion step, the algorithm adds new branches to the tree leaf to represent the potential options available for the next decision. At this point, nothing is known about these branches, so simulation is required to gain further information.

During the simulation step, choices are made randomly in the design decision tree from the recently expanded leaf until a design end state is reached, for this paper the end state is determined by the prescribed number of nodes added to the Color Graph. This process is repeated until a large enough sample has been generated, for this paper the sample size was 100 completed designs per simulation. The completed designs are then evaluated, and their scores are then back-propagated through the decision tree.

During the back-propagation step, the score is added to every node in the decision tree between the root node and the leaf node that was expanded upon during this iteration. The sum scores are then divided by the total number of times each node has been used in an evaluated design giving a mean score value. The mean scores make up the policy used in the selection step.

One of the motivations behind examining MCTS is that it performs well when applied to decision tree problems in game theory that are similar to Color Graph [66], however, when applied to automated design problems in the past it has performed poorly [121]. This chapter hopes to understand better why MCTS appears to struggle with this class of problem.

The best implementation found to store the quality scores for the design policy was by storing them directly inside the node and edge values of a MATLAB digraph object (a graph with directed edges) [143] representing the design decision tree. During the

selection stage, the algorithm selects the edge with the best mean score as available in the current policy.

The algorithm started at the decision tree root (representing the seed node location in the Color Graph), the design decision tree was expanded to add the six branches (representing the six potential color choices). The algorithm then takes 100 samples of completed designs propagating out from each of the new branches. A parameter sweep was performed prior to the experiment in order to determine the most appropriate sample size [144,145].

In order to more directly compare MCTS to the other methods parallel processing was not used. This gives a more direct measure of the computational cost and feasibility of the method. To keep the algorithm from running for an excessively long time, a time limit of 15 minutes was imposed. After the 15 minutes elapsed the current policy depth was reported along with the current design score. The MCTS algorithm is shown in Figure 20.

Algorithm 1: Monte Carlo Tree Search
Input: Number of nodes N, and edge properties α, β, and γ, target properties t, sample size s.
Output: A color graph design

INITIALIZATION
Select the root node
SELECTION
Loop Process
While design policy has not reached length n
  While design policy exists
    Select branch with the best score
  End while
EXPAND
  If terminal leaf is a location in the color graph
    Add six new leaves representing color options
  Elseif terminal leaf is a color decision
    Add a number of leaves equal to the number of nodes in the color graph
  End if
SIMULATION
  Loop Process
  While new unexplored leaf exists
    Generate s random completed design branches
    Calculate the average score for decision tree leaf
  End while
BACK PROPAGATION
  While current node≠ root
    Select node back one level in the branch
    Compute the average score for that node based on *SIMULATION*
  End while
End while

Figure 20: The Monte Carlo Tree Search Algorithm

5.3.3.2 Genetic Algorithm

A Genetic Algorithm (GA) [146] is a bio-inspired meta-heuristic design algorithm based on the process of natural selection. GAs are part of a broad class of bioinspired Evolutionary Algorithms (EA) [147]. When applying a GA to the Color Graph problem, a set of chromosomes represent the location and color decisions made.

The algorithm consists of five steps: 1) randomly generate and evaluate the first generation, 2) select parents, 3) crossover parent chromosomes, 4) mutate offspring chromosomes, and 5) evaluate offspring. With the process repeating from step 2 until the desired end conditions are met.

The first generation is randomly generated. The algorithm generates purely random chromosomes representing a completed design solution. The algorithm then evaluates the designs and ranks them based on their performance.

During the second step, the algorithm randomly selects two parents from the previous generation. The selection process is biased towards higher ranking parents.

During the third step, the algorithm generates an offspring from the parent's chromosomes. This is done by randomly selecting each chromosome (representing a single choice) from one of the two parents' corresponding chromosomes. For example, when determining what the offspring's third node added will be, the algorithm looks at both of the parent's third added node and randomly selects between the two. This generates a completed set of chromosomes defining a graph design.

During the fourth step, the offspring's chromosomes are randomly mutated with a small probability that each chromosome will change. This could lead to a location decision changing, causing an entire branch of the Color Graph to be moved to a

different node. Alternatively, the color of a single node may change. During the fifth and final step, the algorithm evaluates and ranks all of the offspring to create the new set of potential parents.

GAs are of particular interest because they generate potential design solutions that automatically abide by formulation constraints, while other search methods require traversing the search tree before generating a full solution. This allows the GA to overcome the problems created by the structure of the search tree. Additionally, compared to MCTS and ACO the GA involves fewer operations being performed and generates only completed solutions (as defined by a user selected end state condition), without needing incomplete intermediate steps, leading to a shorter run time.

The GA implementation is straightforward. Designs were represented as arrays in MATLAB with two columns, representing the location where a node will be added and the color of the added node. Each generation consisted of 100 generated offspring. The algorithm copied the top ten perform parents from the previous generation over to the new generation and then created the remaining 90 offspring through crossover.

The parents were selected using a normal distribution centered at the top performing parent and possessing a standard deviation of 8 places in the ranking. A parameter sweep was performed in order to determine the appropriate standard deviation [144,145].

During crossover, there was an equal chance of any individual offspring chromosome (representing the location and node color decision in the Color Graph) coming from either parent. Each chromosome corresponded to the decision made so that the first chromosome represented the first node added to the Color Graph and so

on. During mutation, there was a 0.05 probability of each chromosome location or color changing.

This process was repeated for 20 generations, but the solution often converged within less than 10 generations. The GA algorithm is shown in Figure 21.

Algorithm 2: Genetic Algorithm
Input: Number of nodes N, and edge properties α, β, and γ, target properties t, generation size b, and number of generations G.
Output: A color graph design

1. Randomly generate b offspring
2. Compute offspring's α, β, γ properties
3. Compute offspring on proximity to t
   Loop Process
4. **for** $g = 2 : G$ do
5. Clone top 10 offspring to new generation
   Loop Process
6. **for** offspring $11 : b$
7. Select random parents biased towards top scoring members of last generation
8. Select chromosomes randomly from parents
9. Randomly mutate chromosomes
10. **end for**
11. **end for**

Figure 21 Pseudocode for a genetic algorithm

5.3.3.3 Genetic Algorithm with No Crossover

The Genetic Algorithm with No Crossover (GAnX) is very similar to the traditional GA except that each offspring only has a single parent that it receives all of its chromosomes from and the only mutation occurs to change them. This leads to slower change than the traditional GA but can lead to better results.

The algorithm consists of four steps: 1) randomly generate and evaluate the first generation, 2) select a parent, 3) mutate offspring chromosomes, and 4) evaluate

offspring. With the process repeating from step 2 until the desired end conditions are met.

The first generation is randomly generated. The algorithm generates purely random chromosomes representing a completed design solution. The algorithm then evaluates the designs and ranks them based on their performance.

During the second step, the algorithm copies the parent's chromosomes to the offspring. This carries over the completed set of chromosomes defining a graph design.

During the third step, the offspring's chromosomes are randomly mutated with a small probability that each chromosome will change. This could lead to a location moving, causing a section of the Color Graph to be moved to a different node or a color changing.

During the fourth and final step, the algorithm evaluates and ranks all of the offspring in order to create the new set of potential parents.

Like the GA, GAnX generates designs using the design decision tree during the first iteration and then improves upon those designs through mutation during subsequent iterations. This again leads to designs that exist in the design decision tree without needing to search through the decision tree itself

The GAnX implementation was similar to the GA. Designs were represented as arrays in MATLAB with two columns, representing the location where a node will be added and the color of the added node. Each generation of design consisted of 100 generated offspring. The algorithm copied the top ten perform parents from the previous generation over to the new generation and then created the remaining 90 offspring through random parent selection.

The parent was selected using a normal distribution centered at the top performing parent and possessing a standard deviation of 8 places in the ranking. The offspring had started with identical chromosomes to its parent.

During mutation, there was a 0.05 probability of each chromosome (representing the location and node color decision in the Color Graph) experiencing a location or color change.

This process was repeated for 20 generations, but the solution often converged within less than 10 generations. The GA algorithm is shown in Figure 22.

Algorithm 3: Genetic Algorithm with No Crossover
Input: Number of nodes N, and edge properties α, β, and γ, target properties t, generation size b, and number of generations G.
Output: A color graph design

```
1.  Randomly generate b offspring
2.  Compute offspring's α, β, γ properties
3.  Compute offspring on proximity to t
    Loop Process
4.  for g = 2 : G do
5.    Clone top 10 offspring to new generation
      Loop Process
6.    for offspring 11 : b
7.      Select random parent biased towards top
        scoring members of last generation
8.      Randomly mutate chromosomes
9.    end for
10. end for
```

Figure 22: Genetic Algorithm with No Crossover

5.3.3.4 Ant Colony Optimization

Ant Colony Optimization (ACO) [67,105,148], is a meta-heuristic approach to optimization that is commonly used in path-planning applications. The ACO method consists of five steps: 1) first generation randomly selects a path, 2) completed designs are evaluated, 3) "pheromones" are laid on the path proportional to design quality, 4)

all pheromone paths evaporate, and 5) a new generation of ant select path biased by pheromones. With the processing repeating from the second step until the desired end conditions are met.

During the first step, a swarm of “ants” randomly travel down the design decision tree until they reach design completion. This creates a random sample of the design space similarly to MCTS.

During the second step, the algorithm evaluates the designs created by the swarm. The scores will be used to calculate the pheromone intensity.

During the third step, the algorithm adds the pheromone trails to the design decision tree. The pheromone is laid along the entire path between the tree root representing the source node and the final leaf representing the completed design. The pheromone is added to existing pheromone trails increasing the weighting towards the specific edge.

During the fourth step, the pheromone trails evaporate, reducing all of their intensities by a consistent amount. This leads to low scoring and unpopular paths fading away while high scoring and popular paths get stronger.

During the fifth and final step, a new swarm of ants follows the previous pheromone trails. At each level in the tree, the ants randomly select a path with a bias towards paths with high pheromone trails. The process then repeats from the second step until the number of desired iterations or another end condition is met.

One motivation for examining an ACO is that they work very well on path-planning applications, but like MCTS, ACO is a direct search of the design decision tree and is likely to struggle with the size of the search space. However, it is expected that the

ACO will still outperform MCTS because it takes a metaheuristic approach to the search, and it only generates completed designs.

Similarly to MCTS, the best implementation found to store the pheromone trails was storing them in the edge weights of a MATLAB digraph object (a graph with directed edges) [143].

The algorithm sends out a swarm of 100 ants to sample from the design decision tree. The ants started at the tree root and traversed the design decision tree branches until they reached a leaf representing a completed design. For the first iteration this was done completely randomly, but for subsequent iterations, ants selected a path by using a random proportional rule, [11, to select paths with a bias towards strong pheromone trails, where T is an individual trail/path, f is the parameter defining how likely an ant is to follow the highest value path, and Q_T is the quality of the completed design at the end of the pheromone trail. This was performed for ten total iterations.

$$T_w = \frac{Q_T + 0.1^f}{\sum_{Tt} Q_T + 0.1^f} \quad [11]$$

To more directly compare ACO to the other methods, parallel processing was not used to increase computational performance, even though it is common practice when implementing an ACO. This gives a more direct measure of the computational cost and feasibility of the method.

To keep the algorithm from running for an excessively long time, a time limit of 15 minutes was imposed. After the 15 minutes elapsed algorithms that had not finished the desired number of iterations were stopped and the design that was generated by

following the strongest pheromone trail so far was evaluated and reported. The ACO algorithm is shown in Figure 23.

Algorithm 4: Ant Colony Optimization
Input: Number of nodes N, edge properties α, β, and γ, target properties t, swarm size s, iterations I
Output: A color graph design

Loop Process

```
For iteration 1:I
  For ant 1:s
    Start at root node
    While current node ≠ completed design
      Randomly select a branch using the random proportional rule
    End while
    Evaluate design
    Lay pheromone along path taken proportional to design quality
  End for
End while
```

Figure 23: Ant Colony Optimization Algorithm

5.3.3.5 Control: The Random Approach

In addition to the Color Graphs generated by the automated design algorithms, a purely random algorithm was developed as a baseline for comparison. This algorithm consisted of three steps 1) select a location to add a node, 2) select the color of the added node, and 3) repeat until the desired number of nodes are added.

5.4 Results

The entire experiment was performed on a computer with a 3.4 GHz Intel Xeon CPU [149] and 16 gigabytes of RAM. It took approximately nine and a half hours to run the experiment. Figure 24 shows a bar graph summarizing the performance of each algorithm. Lower scores are higher quality with zero being a perfect score.

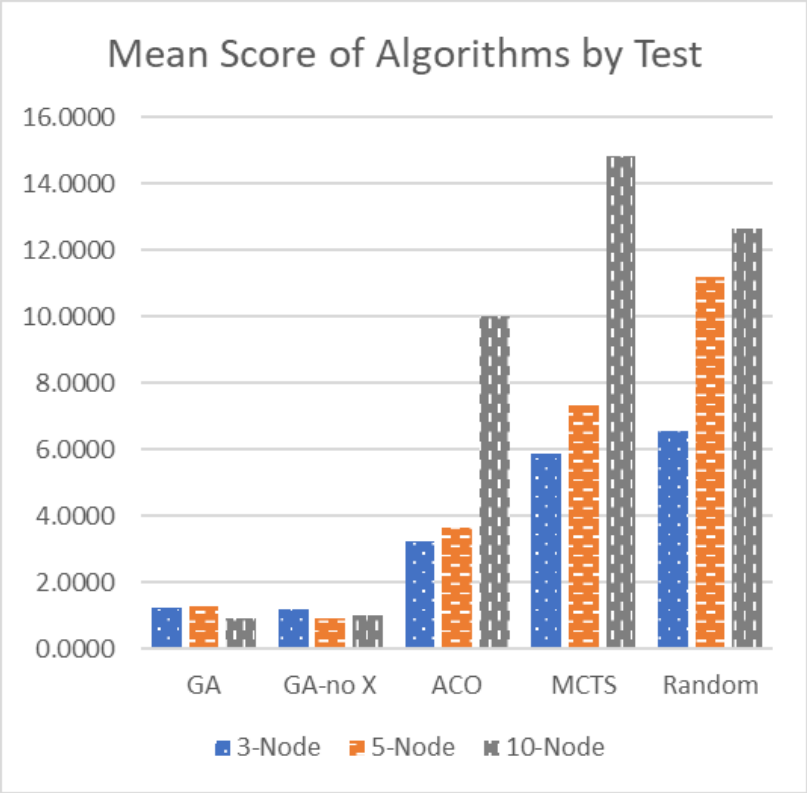


Figure 24: Mean Score of Algorithms by Test

5.4.1 Three-Node Color Graph

In the first test, each algorithm generated a Color Graph with three added nodes. This was repeated across ten trials for each of the methods. Zero was the ideal score which would mean that the method reached the exact target of properties [0,0,0]. Table 2 shows an overview of the results including the p-value for the algorithm performing better than random.

The GA and GAnX approaches performed the best with mean scores of 1.2199 and 1.1957 across the ten trials and standard deviations of 0.3506 and 0.4860 respectively. The ACO algorithm was the next best with a mean score of 3.2099 and a standard

deviation of 0.7331. MCTS performed the worst with a mean score of 5.8465 and a standard deviation of 2.6426.

As a baseline for comparison, random generation of a solution had a mean score of 6.5476 with a standard deviation of 2.1938. At this level, MCTS performing better than random only has a p-value of 0.26, showing that it does not significantly improve upon random. The other algorithms showed much better results, however, with p-values for improving on random of 0.00012 for the ACO, and less than 0.00001 for both GAs.

Table 2: Results of the 3-Node Test

Trial Score	#	GA	GA-no X	ACO	MCTS	Random
	1	1.6924	1.5117	2.7851	4.4494	4.3188
	2	1.3621	1.5765	3.0975	3.3484	7.0521
	3	1.2134	0.7684	3.1074	10.6133	8.1228
	4	0.7677	0.7677	2.7408	6.1370	8.3628
	5	1.3413	1.3413	2.2966	6.6065	4.4198
	6	1.2592	1.7444	3.3733	5.3643	5.9520
	7	0.5662	1.3149	2.7127	3.6929	7.5852
	8	1.5599	0.2488	4.4759	10.2270	2.4918
	9	1.4405	1.6875	3.0104	3.5174	7.6910
	10	0.9959	0.9959	4.4993	4.5093	9.4800
Mean		1.2199	1.1957	3.2099	5.8465	6.5476
std		0.3506	0.4860	0.7331	2.6426	2.1938
p-value		< .00001	< .00001	0.00012	0.26	

5.4.2 Five-Node Color Graph

For the second test, GAnX performed the best again, with a mean score of 0.9109 and a standard deviation of 0.2924. This was better than the GA which had a mean score of 1.2632 and standard deviation of 0.5274. Statistically, this can be called significant with a p-value of 0.04. The ACO scored third best with a mean score of 3.6576 and a standard deviation of 1.7442. Again, MCTS performed worst with a mean score of 7.3315 and a standard deviation of 3.2940. Table 3 shows an overview of the results including the p-value for the algorithm performing better than random.

The random solution had a mean score of 11.1985 and a standard deviation of 4.5135. This time MCTS is significantly better than random with a p-value of 0.02. All of the other algorithms had p-values less than 0.00005

Another notable result is that for the five-node Color Graph, only three of the 10 MCTS trials successfully developed a full five-step design tree policy during the allotted 15-minutes. The other seven trials were only able to develop a three or four-step design tree policy, meaning that the algorithm randomly selected its last several choices. None of the other algorithms failed to finish running within the 15-minute limit.

Table 3: Results of 5-Node Test

	#	GA	GA-no X	ACO	MCTS	Random
Trial Score	1	1.8072	0.7318	3.8408	10.7921	19.5950
	2	1.6685	0.9696	2.1447	3.3922	11.6958
	3	0.5440	1.3916	2.0664	7.9051	11.1203
	4	1.3675	1.4884	4.3730	13.2797	11.2356
	5	1.0449	0.7263	3.2435	10.2693	11.3337
	6	0.7911	0.7911	3.9594	6.0398	12.6152
	7	2.0644	0.7131	2.4318	5.9575	15.0175
	8	1.3451	0.6997	7.9305	4.9321	9.5960
	9	0.5214	0.7211	2.3644	7.4766	2.2646
	10	1.4774	0.8758	4.2211	3.2711	7.5109
Mean		1.2632	0.9109	3.6576	7.3315	11.1985
std		0.5274	0.2924	1.7442	3.2940	4.5135
p-value		< .00001	< .00001	0.00005	0.021	

5.4.3 Ten-Node Color Graph

In the third test, GA and GAnX performed best again with mean scores of 0.8938 and 1.0100 and standard deviations of 0.3024 and 0.3993 respectively and neither can be said to be significantly better than the other. Again, ACO does third best, with a mean of 9.9939 and standard deviation of 6.2229. MCTS performed the worst with a mean score of 14.8139 and standard deviation of 5.7676. Table 4 shows an

overview of the results including the p-value for the algorithm performing better than random.

The random solution scored 12.6358 with a standard deviation of 4.42425. Neither the ACO nor MCTS performed significantly better than random with p-values of 0.14 and 0.17 respectively.

During the 10-node test, the ACO performed poorly. The ACO had very little consistency with one trial scoring a 24.5916 and another scoring a 3.2048. MCTS again failed to finish in the allotted 15 minutes, this time failing during every trial. MCTS failed to generate a completed policy. In three of the trials, MCTS developed a five-node design tree policy, in five trials it developed a four-node design tree policy, and in two of the trials it only developed a three-node design tree policy. For comparison, the mean times the GA and GAnX took to run all 20 iterations were 23.9962 and 22.7598 seconds respectively, and on average they discovered the best design on their 6^{th} and 5^{th} iterations.

Table 4: Results of 10-Node Test

Trial Score	#	GA	GA-no X	ACO	MCTS	Random
	1	0.8883	1.4917	13.9490	5.6568	4.9379
	2	1.7145	1.0220	6.0752	18.5631	11.4240
	3	0.6866	0.9899	5.6296	17.2460	11.9726
	4	0.8503	0.3179	9.0876	22.3351	6.9562
	5	0.9217	1.7720	3.2048	12.0333	14.8488
	6	0.8795	0.8614	11.0807	22.8276	12.9105
	7	0.8393	0.7279	13.3522	8.5696	19.2917
	8	0.6700	1.0083	24.5916	9.7148	15.8583
	9	0.7980	1.0868	7.3123	14.9512	12.5394
	10	0.6901	0.8226	5.6555	16.2417	15.6182
Mean		0.8938	1.0100	9.9939	14.8139	12.6358
std		0.3024	0.3993	6.2229	5.7676	4.2425
p-value		< .00001	< .00001	0.14	0.17	

5.5 Discussion of Results

Several interesting observations can be made from analyzing the results. One clear observation is that generally, the genetic algorithms performed significantly better than all of the other algorithms. Interestingly, it was shown that over the three sizes neither GA nor GAnX performed significantly better than the other, with GAnX having a marginally better mean score of 1.04, but a p-value of only 0.21 when compared to GA with a mean score of 1.13. This implies that crossover does nothing to improve performance within this context.

On the other hand, MCTS performed very poorly and only significantly outperformed the random search during the five-node test. This is likely a result of MCTS attempting to search the entire design decision tree directly and not being exploitative of previous searches, instead of taking an explorative approach.

Additionally, MCTS seemed to hit a wall at a policy depth of approximately five choices, at which point it just kept switching between branches and got stuck.

ACO consistently performed better than MCTS, but worse than the genetic algorithms. This makes sense considering ACO shares some properties with MCTS and some with the genetic algorithms. The ACO's performance dropped off significantly during the 10-node test, even though it completed its full run every time. This is likely due to the 10-node design decision tree having $2.2x10^{14}$ potential completed solutions, so a large amount of luck was involved in finding any good solutions. Future improvements to the approach presented by ACO could involve increasing the swarm size with the design decision tree size or implementing negative pheromones to discourage further exploration of paths that performed poorly.

A general observation is that due to how they are set up, the genetic algorithms required far fewer evaluations to be run than the other methods. Both of the genetic algorithms required 2000 evaluations for every test, though on average they found their final answer in the first 1000 evaluations. The ACO required only 1000 evaluations for each test, but the time between evaluations was much longer and increases with design decision tree size. MCTS required almost 900 evaluations for the 3-node test, over 1300 evaluations for the 5-node test, and over 1500 evaluations on average for the 10-node test (though that was limited by time).

Based on the results of the experiment, it can be concluded that Genetic Algorithms (GA) were the best approach evaluated for solving a design decision tree problem such as Color Graph. It does not appear at this time that including crossover had any significant effect on GA performance.

Ant Colony Optimization (ACO) appeared to perform marginally well on small design decision trees, but its performance dropped off significantly as the number of choices increased. However, this may be addressable by scaling swarm size with the number of choices or implementing negative pheromones to devalue bad paths further.

Monte Carlo Tree Search (MCTS) performed very poorly for all sizes of design decision trees and performed no better than random as the number of choices increased. Additionally, the time needed to run MCTS increases dramatically with the number of choices.

It can generally be concluded that more exploitative methods performed better than explorative methods and that successful methods will likely rely on finding clever solutions to addressing the exceptional size of the search space.

Additionally, it was shown that Color Graph is capable of serving as a benchmark problem for rapidly testing methods for searching design decision trees. The Color Graph problem was capable of providing valuable insight that can lead to the developing of better-automated design methods in the future.

The work presented in this chapter is critical to the development of the methods presented in this dissertation, because not only are multimodality, opaque intermediate states, unboundedness, and increasing internal locations properties of the design decision tree, they are also present in mission command and control problems. By evaluating candidate algorithms on the relatively simple Color Graph problem, I can inform my selection of algorithms for solving task ordering and objective planning problems such as those discussed in the next chapter.

6. MISSION COMMAND AND CONTROL

Mission command and control problems typically involve making complicated logistic decisions facing uncertainty in high-risk conditions that can change over time. Real-world examples of mission command and control problems include planning and executing space missions; project managing the design, manufacture, and distribution of a product; or planning and taking a cross-country road trip. In this context, I define a mission as a series of time discretized objectives that can be completed by agents performing tasks.

We define the structure of a mission command and control problem as consisting of seven steps 1) determining mission objectives, 2) ordering mission objectives, 3) defining tasks needed to complete the objectives, 4) assigning agents to perform the tasks, 5) executing the mission as planned, 6) monitoring the mission's probability of success, and 7) re-planning the mission to respond to unexpected challenges as necessary.

In the past, these types of problems had to be solved by humans; they may have been aided by algorithms for some of the individual steps, but were generally required to make most the decisions independently. This necessity made the process slow and unnecessarily challenging, and while time is not a concern for some steps such as objective and task planning the computational power and time needed make responding to unexpected events during mission execution infeasible.

In this case study, I use a top-down cognitive-computation approach to solve mission command and control problems [19]. First, I develop a computational model of the decision-making space using Active Mission Success Estimation (AMSE) [150]

to determine the probability of mission success, and the Functional Basis for Engineering Design (FBED) [26] to represent systems of interest for the mission. A series of algorithms use the functional representation to solve sub-problems of the mission command and control problem. I then created a cognitive structure that takes the computational probability of mission success and attempts to improve it by applying the appropriate algorithms.

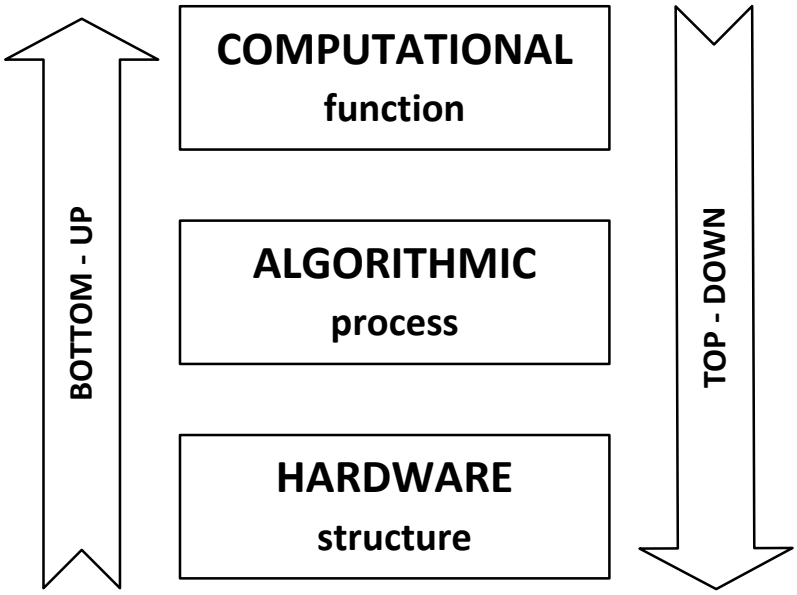


Figure 25: Bottom-Up and Top-Down strategies for studying cognition

6.1 Problem Definition

This work explores ways to improve the reliability and independence of autonomous decision makers facing uncertainty, risk, and complexity through computational cognition. I base many of these processes on observations from a previous study in which a human decision maker made mission command and control decisions [150].

6.2 Importance of the Work

Autonomous systems designed to do jobs that are dangerous, difficult, or generally daunting are highly desirable. Their development is a complex problem in automation because the same factors that make these jobs difficult for humans can make them infeasible or impractical to automate. The most common strategy for addressing this problem is to use substantial and potentially intractable computing power to create a computational representation of the problem [8]. This approach is inherently flawed for decision making in unknown, chaotic, and hazardous environments, because the complexity of a real-world environment is so great that the amount of computational power currently available on Earth is not capable of modeling and predicting it perfectly. This work takes a different approach to automation by focusing on development and application of a top-down cognitive-computation method that is sub-optimal in terms of systems representation, but incredibly fast and efficient. A major inspiration for this work is the way humans rely on heuristics when faced with decisions involving uncertainty, risk, or complexity [5,141].

Additionally, by studying complex autonomous decision making through the lens of computational cognition, we not only learn how to design effective autonomous decision makers, but we also gain insights that can improve our understanding of human decision making.

6.3 Methodology

The mission command and control problem has seven steps, as depicted in the flow diagram in Figure 26: 1) determine mission objectives, 2) objective planning, 3) order

tasks, 4) assign agents and equipment, 5) execute the planned mission, 6) monitor mission success, 7) re-plan when necessary. Steps 2 through 7 can be performed multiple times depending on the mission conditions. To perform these tasks, I created a decision making cognitive structure informed by the previous two case studies. This cognitive structure consists of two groups of algorithms, representing fast and slow cognitive operations [151], and a computational component that performs the AMSE analysis. This decision maker is a top-down approach to computational cognition with AMSE serving as the computational basis, the cognitive operations making up the algorithmic layer, and the structural layer being the overall decision maker design. Each of these elements are detailed in Section 6.3.

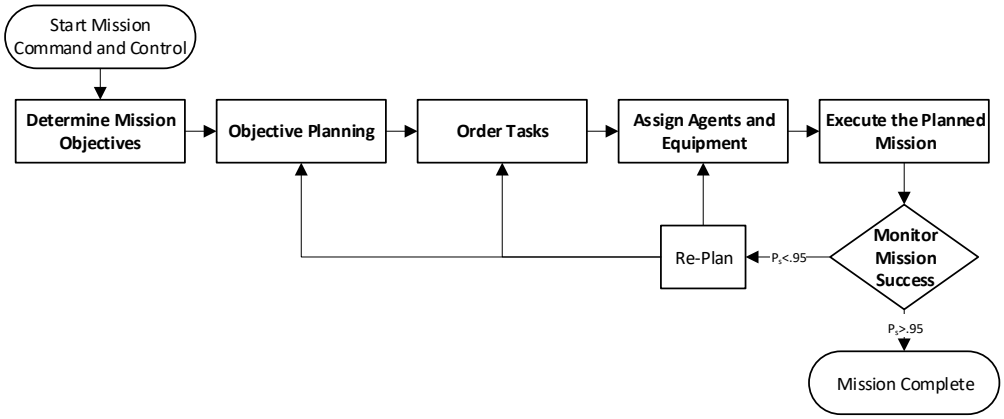


Figure 26: Mission Command and Control Problem Process

Additionally, I create a mission framework to evaluate for the cognitive decision maker: a Martian settlement setup mission [152]. The mission consists of four RoboSimian-inspired robots [153], surveying terrain to select a settlement site, construction of habitable space and a greenhouse, and initialization of an In-Situ

Resource Utilization (ISRU) technology in preparation for eventual human arrival. I use MATLAB digraph objects [143] to represent functional models of systems critical to mission success using the FBED [26] taxonomy. The mission operating environment is a randomly generated Martian terrain inspired by Mawrth Vallis [111].

To better represent risk and uncertainty, I develop a fault injector and a *black swan* generator [154]. The fault injector determines when and how a system fails based on prior established failure distributions. The *black swan* generator randomly inserts scenarios into the mission that are unexpected and unknown to the decision maker before their occurrence.

In this section, I'll describe the theoretical objectives necessary to complete this mission, followed by a description of how the mission environment is generated. Then, I'll discuss how I've used functional modeling to represent the system, and expound on the fault injector and black swan generator. Lastly, I'll discuss the further development and use of the cognitive decision maker, and the fast and slow decision-making construct.

6.3.1 Defining Mission Objectives

The case-study mission consists of 10 objectives. The objectives can be constrained, under-constrained, or unconstrained. A constrained objective has a set time of occurrence or position in the objective. For example, the cargo drop containing the habitat module, greenhouse, and ISRU will arrive three months after the robots. Under-constrained objectives may have prerequisites or other characteristics that set it within a range, without defining an exact time. For example, before clearing the settlement site of debris, a settlement site must be selected; but as this is assessed and

determined during the mission, it can occur at any time within a range. Unconstrained objectives are actions that must take place during the mission, but have no constraints related to other objectives. There are multiple unconstrained objectives; selecting a survey site, deploying robots, exploring the survey site, selecting a settlement site, clearing the settlement site and collecting regolith (the layer of rock covering the Martian surface), directing cargo, unloading cargo, setting up the habitat, setting up the greenhouse, and deploying the ISRU. Figure 27 shows the objectives, their dependencies, on each other, and classifies how they are constrained. In the follow sections I will explain each of these sections individually.

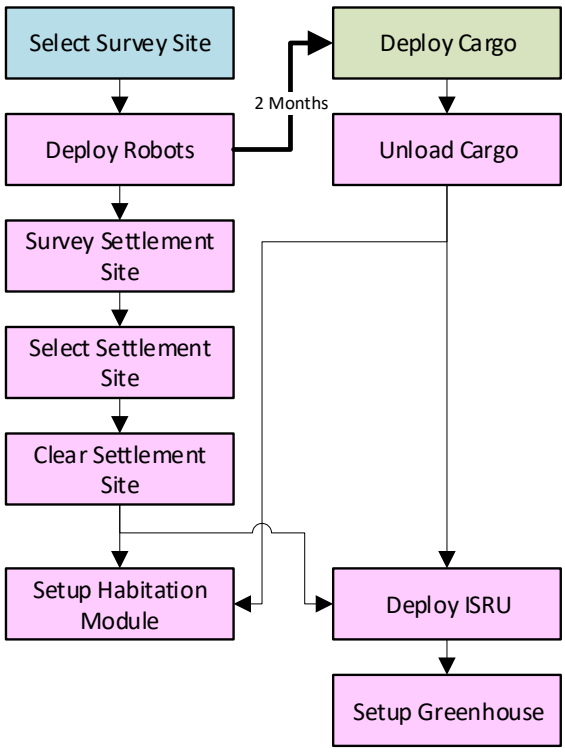


Figure 27: Mission objectives and constraints. Blue objectives are unconstrained, Pink objectives are under-constrained, and Green objectives are constrained

6.3.1.1 *Selecting a Survey Site from Telemetry Data*

The mission takes place in a 32 x 32 km simulated Martian environment with a 1 m resolution. However, the activity will mostly be constrained to 1 x 1 km area called the survey site that the decision maker must select. An ideal survey site will have relatively level terrain and ground cover predominantly consisting of sand and gravel. This objective was treated as unconstrained because it has no prerequisites.

The inputs for this objective are telemetry data, and the output is a survey site location.

6.3.1.2 *Deploying Robots to the Survey Site*

The decision maker deploys the robots to the center of the survey site. This objective is an under-constrained objective that has the selection of the survey site as a prerequisite. The inputs for this objective are survey site coordinates, and the outputs are the robots and the robot base that provides their power being placed at the survey site.

6.3.1.3 *Exploring the Survey Site*

The survey site must be surveyed to create a higher resolution map of the environment. The process of site surveying is an under-constrained objective because it requires a survey site to have been selected and the robots to have been deployed at the survey site. The inputs for this objective are robots and a survey site location. The objective output is a map of the survey site with a one-meter resolution.

6.3.1.4 *Selecting a Settlement Site*

The decision maker uses the high-resolution map (developed by completing the previous objective) to select a settlement site, where the robots will construct the habitat, ISRU, and greenhouse. The best settlement site will be very flat with mostly sand and gravel ground cover and some cobble. Settlement site selection is an under-constrained objective because the decision maker uses the survey data to select the site. The objective input is a high-resolution map of the survey site, and the output is a settlement site location.

6.3.1.5 *Clearing the Settlement Site of Debris*

The robots must clear large stones from a 60 m square at the center of the settlement site for the habitat, ISRU, and greenhouse. Additionally, cobble must be collected from the Martian surface as shielding for the habitat [152]. This objective is under-constrained and requires a settlement site coordinates and robots to do the work. The objective input is a settlement site location and active locating of the robots, and the objective output is stone debris and regolith cobble placed into piles.

6.3.1.6 *Directing Cargo to Settlement Site*

The cargo ship carrying the habitat, greenhouse, and ISRU equipment will arrive two months after the robots have arrived, but it must be given a final location for the settlement site before arriving. The objective inputs are the settlement site location, and the output is the new location of the cargo.

6.3.1.7 Unloading Cargo

The robot must unload the cargo from the lander. The prerequisites are that the cargo and robots to have already arrived. The objective outputs are the habitat, greenhouse, and ISRU.

6.3.1.8 Set Up the Habitation Module

The robots must construct the habitation module where the humans will live. Constructing the module will involve assembling the supports that came in the cargo, attaching foam walls, adding cobble cladding for protection, and connecting the solar panels (as determined by the initial design of the site [152]). The prerequisites of this objective are the unloaded cargo, regolith cobble, and a settlement site clear of debris.

The objective inputs are the cargo and the cobble, and outputs are a constructed habitation module placed at the settlement site

6.3.1.9 Deploying ISRU Water Collection Tent

A transparent tent is set up over a 25 m diameter area of soil to collect water [155]. ISRU deployment requires unloaded cargo and a clear settlement site. The objective inputs are the cargo, and objective outputs are water placed at the ISRU.

6.3.1.10 Setting Up the Greenhouse

The robots must construct the greenhouse that will grow crops. Robots will assemble the supports that came in the cargo and attach foam walls. The prerequisites of this objective are unloaded cargo and the cleared settlement site. The objective inputs are the cargo, cobble, and electricity, and outputs are a constructed greenhouse placed at the settlement site.

6.3.1.11 Start Growing Crops

The last objective of the mission before humans arrive is to start crops growing in the greenhouse.

Objective inputs are water, electricity, and air. Objective outputs are food.

6.3.2 Generate Martian Mission Environment

Before the mission could be planned or executed, an environment was needed to plan it in. For this case study, I decided to base the terrain on the Mawrth Vallis region of Mars, because it has been named as a possible future site for human exploration [111]. To generate the computational topography, I used a diamond-square algorithm (also known as a plasma fractal) [113]. *Section 4.3.1: Terrain Generation* describes this process at length. Figure 28 shows an example of a generated Martian mission environment.

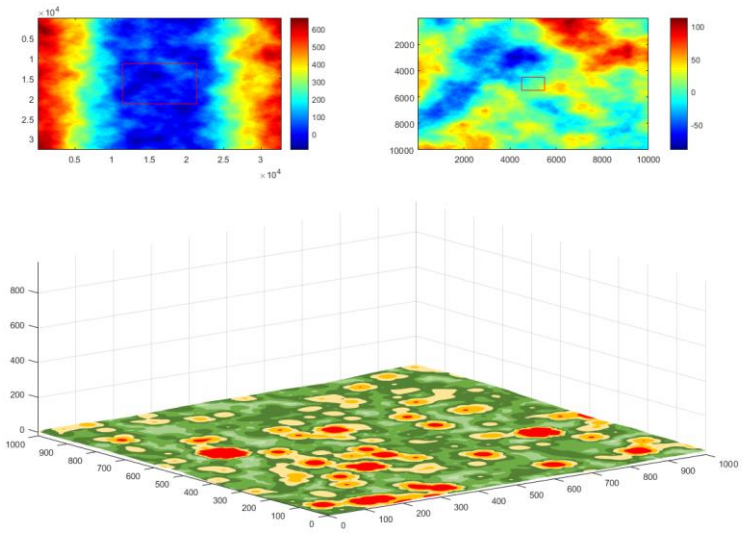


Figure 28: Top left: elevation map of the 30 km x 30 km environment. Top right: selected survey site shown in the red box. Bottom: survey site map, colors represent the grain size of the ground cover.

6.3.3 Create Functional Models of Systems

I developed functional models using the FBED taxonomy for all of the major mission systems of interest. Modelled systems include the robots, the robot base (the robot lander and charging base), and Surface Exploration Vehicles (SEV).

The functional models were developed in MATLAB [96] as directed graph objects [143]. I used directed graphs because they have nodes that can be used to represent functions, directed edges that can be used to represent flows, and edge and node properties can store information such as failure distributions and subsystem wear.

Figure 29 shows a functional model of one of the robot crew members. Appendix A contains the functional models for all of the systems used in this case study along with detailed discussion of their construction.

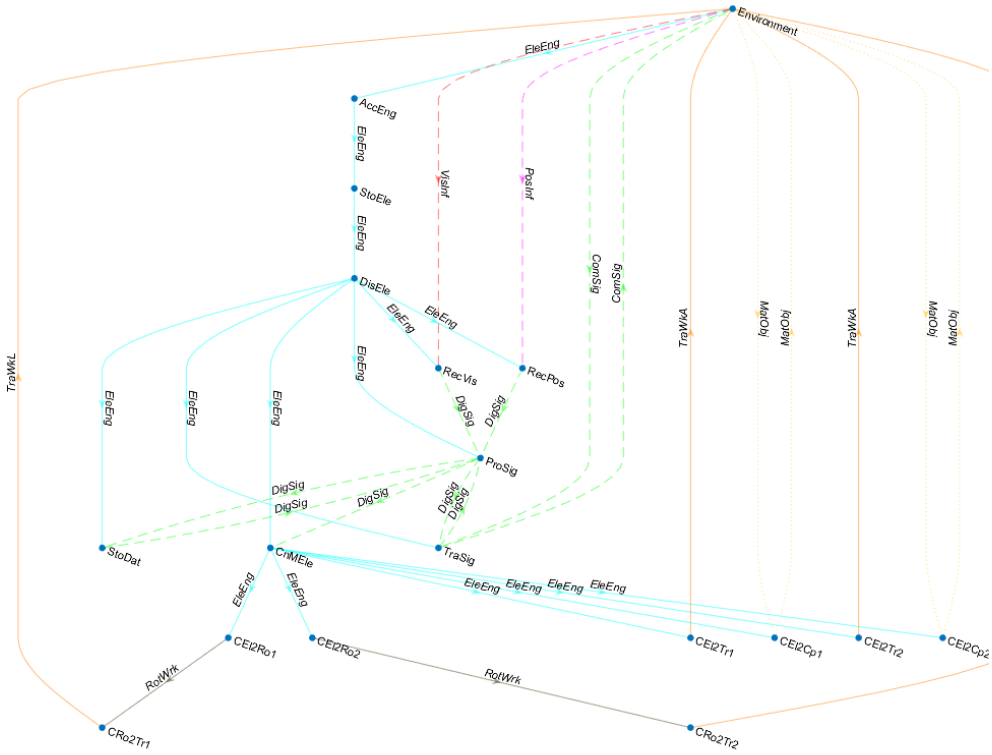


Figure 29: Functional Model of a Mars Exploration Rover, Flows coded by line type and color. Solid lines are energy flows, dotted lines are material flows, and dashed lines are information.

6.3.4 Fault Injector

The fault injector is used to simulate system failures during mission execution. Each of the functional model sub-systems has a failure distribution associated with it that is dependent on how often the sub-system is utilized. Every time a sub-system is used to perform a task, the failure distribution and a random number generator is used to determine if the sub-system fails. If the random number is below the current probability of failure, then the sub-system has failed, and FFIP [39] is used to determine if it leads to the loss of other systems or even total system failure. The decision maker knows the

failure distributions used by the fault detector and can use this information to inform decisions.

6.3.5 Black Swan Generator

The black swan generator inserts black swan events into the mission. A black swan is an event that is so improbable that it was thought to be impossible, but can still occur [154]. The classic example of this are black swans, which were assumed to be almost impossibly rare by Europeans, until they colonized Australia and discover that in Australia they were very common. The black swan generator can be configured to inject one or more unexpected conditions into the mission simulation to which the decision maker must adapt. The black swan events considered include ground cover misidentification which can lead to robot failure [85], telemetry data being incorrect leading to poor site selection, an unexpected sandstorm occurring which delays work and damages systems [156], and decision maker assumptions about failure distributions of sub-systems being inaccurate.

6.3.6 Decision Maker Cognitive Structure

The decision maker cognitive structure informed was informed by the previous two case studies. The makeup of this cognitive structure consists of two groups of algorithms, representing fast and slow cognitive operations [151] and a computational component that performs the AMSE analysis. Figure 30 shows the connections between these processes with fast systems show in orange and slow systems shown in black.

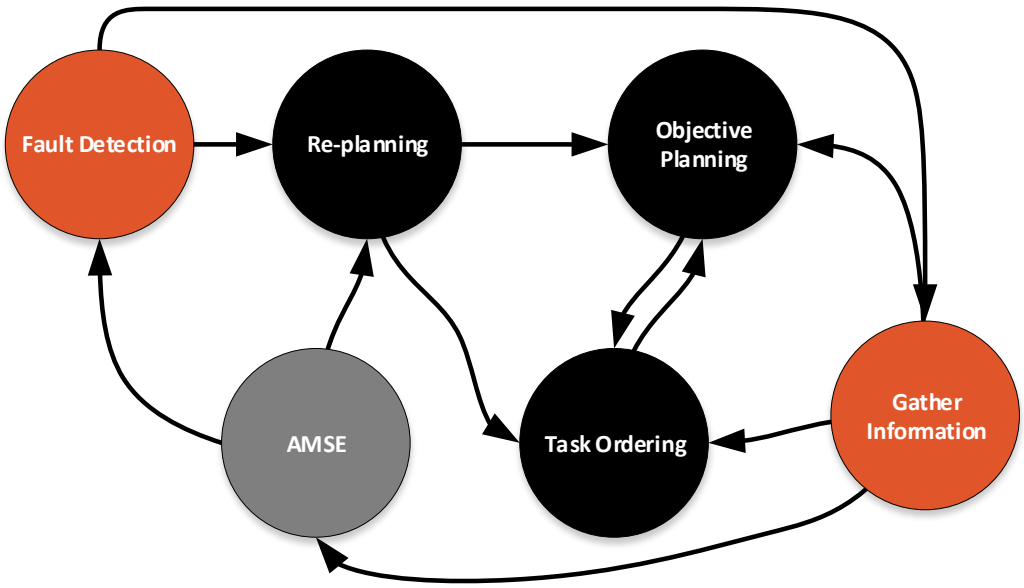


Figure 30: Decision Maker Cognitive Structure

6.3.6.1 *Fast Cognitive Operations*

Fast cognitive operations are automatic processes that happen quickly and often and with very little deliberate effort. An example of a fast operation in humans is our ability to automatically and effectively recognize facial expressions of other humans almost instantaneously [151].

The fast operations are *information gathering* and *fault detection*. Both of these cognitive operations deal with the collection and monitoring of information and signals. *Gather Information* looks outward and takes signals from the environment and uses them to inform decisions automatically. *Fault Detection* looks inward at systems within the mission framework and attempts to detect when something is wrong. A biological analogy to these operations is the ability to see or detect the environment and the ability to feel pain or discomfort [157,158].

6.3.6.2 *Slow Cognitive Operations*

Slow cognitive operations are manual processes that require more time and energy to perform. An example of slow cognitive operation in humans is doing taxes. The slow operations in the cognitive structure are objective planning, ordering tasks, and re-planning when necessary. Objective planning is the process of placing the objectives into an order that will improve the probability of mission success. Objective planning is computationally very similar to a small, constrained Traveling Salesman Problem [100]. While this means that there are a large number of potential solutions to evaluate (making it computationally difficult), it also means that there are many useful heuristic and metaheuristic approaches to the problem [105]. Task ordering is the process of assigning tasks (actions performed by agents to complete objectives) to agents in an order that completes the objectives efficiently while minimizing the risk of mission failure. Task ordering is very similar to the design decision tree problem discussed in the second case study, and cognitive operation algorithms that are well suited to automated design decision tree problems are applied [135]. The third slow cognitive operation is re-planning. Re-planning is necessary when the probability of mission success calculated by AMSE [43] drops below the desired level. The decision maker applies a variety of approaches to the re-planning operation based on the circumstances, including moving around task orders to accommodate system down times, switching out tools or agents assigned to a task, ordering the repair or replacement of failed components, or removing an objective from the mission entirely if necessary.

6.3.6.3 Active Mission Success Estimation

The Active Mission Success Estimation (AMSE) method is used to computationally represent the mission and analyze it to determine the probability of mission success [43]. I developed AMSE to enable the performance of autonomous decision making on mission command and control problems.

AMSE provides timely risk information to inform mission decisions made in crisis during rapidly evolving situations. AMSE calculates the probability of mission success through the adoption of a modular risk-informed object-oriented approach to mission modeling, and system health monitoring and analysis. The analysis enables active recalculation of the risk of mission failure as the mission progresses, and mission-critical decision making with many possible options can be analyzed to help inform mission control decision to increase the probability of total mission success.

The performance of AMSE necessitates that all mission-critical components be modeled thoroughly using risk analysis and prognostic techniques, and that the models are developed for modularity to enable the rapid rearrangement of the model elements to evaluate available decision outcomes and estimate each outcome's mission success probability. To effectively represent a mission framework, I used a functional modeling method in which relevant environments can nest within each other and contain the systems of interest within a super system. This nested super-systems approach to modeling is used to determine what environmental hazards are present and if these hazards can cause damage to the system of interest. I analyzed modeled mission tasks including internal and external system risks, and hazard mitigating factors such as nested functional modeling environments representing protective barriers.

By using techniques derived from functional modeling of systems in conjunction with concepts taken from decision theory, risk analysis, and Prognostics and Health Management (PHM), AMSE is capable of providing useful insights when making mission control decisions by rapidly analyzing potential options when confronted with unanticipated and previously unanalyzed scenarios.

AMSE consists of two pre-steps and three primary phases (modeling, analysis, and interpretation).

Pre-Step 1: Mission Success Definition

We start by establishing a definition of mission success and a quantifiable method for evaluating success. In many cases, mission success can be defined as a primary system (or systems) of interest surviving the length of the mission. One example of a system of interest surviving the length of a mission is a planetary exploration rover remaining functional for the entire duration of the planned mission.

To determine the probability of survival of a primary system of interest and the related probability of mission success, a survival rate must be calculated. A survival rate $S(t)$, often takes the form of a Cumulative Distribution Function (CDF) representing the probability that the system of interest will not have experienced a failure by time, *t*. One common form for a survival rate is the exponential survival rate which is found by subtracting the exponential failure rate, $F(t)$, from 1 as shown in [12. The exponential failure rate is then calculated by taking the integral of the Probability Density Function (PDF) form of the exponential failure rate, $f(t)$, which determines the probability that a failure will occur at the instant, τ, given a hazard rate,

λ, which is the number of expected system failures over time. [12, [13, and [14 define $f(\tau)$, $F(t)$, and $S(t)$ respectively [159]. These and other forms of failure distributions, such as system specific PHM models, are an integral part of the AMSE methodology and necessary for the development of failure models.

$$f(\tau) = \lambda e^{-\lambda\tau} \quad [12]$$

$$F(t) = 1 - e^{-\lambda t} = \int_0^t \lambda e^{-\lambda\tau}\, \mathrm{d}\tau \quad [13]$$

$$F(t) = 1 - e^{-\lambda t} = \int_0^t \lambda e^{-\lambda\tau}\, \mathrm{d}\tau \quad [14]$$

Pre-Step Two: Functional Model Development

The AMSE method requires a series of functional models to represent all major systems involved in the mission, as well as their individual behavioral and system health characteristics. I used the FBED method of functional modeling because it represents energy, material, and data flows. These flows can be used to represent work, which makes them convenient when modeling tasks. PHM information that can be collected from systems in real-time must be identified in this step, and built into the functional model where applicable. This information is encoded into the mathematical models developed below.

Phase 1: Modeling

In Phase 1 of the AMSE method, seven distinct steps are performed to develop the AMSE model.

Step 1: Create a nested functional model of the mission.

The first step consists of creating a metamodel of all major mission systems (previously modeled in the pre-steps above) within a nested super-system framework. This is performed by first modeling each system using traditional FBED methods (Pre-Step 2), before placing the individual systems into a nested super systems structure. This allows for the entire system to be modeled, and to represent environmental hazards and various levels of protection that prevent and mitigate system failure. Additionally, the effects of the current health of each layer of protection on the system of interest can be determined through application of PHM and risk analysis models and information (identified in pre-step 2) for each system.

Step 2: Define Critical System(s) of Interest and Critical Flows

In the case of a functional model of a single system, critical functions and flows are defined as elements of the functional model that must be operational for the system to not be in a failure state [160]. In the context of super systems representing a mission framework, the idea of critical functions and flows is extended from the functional level to the system level, and a critical system of interest is defined. A critical system (or systems) of interest is a system that must be functioning for the mission to be considered not failed. For example, in the case of a rover mission, the critical system of interest is the rover; for the case of a crewed space mission, each member of the crew is considered a critical system of interest; and for the mission presented in this case study, the habitat and the greenhouse are both critical systems. Step 2 concludes once the critical system(s) has been identified and defined.

Step 3: Develop Mathematical Models to Represent Graphical Functional Models, Their Health, Failure Distributions, and How Failures Relate to Each Other

The third step of the AMSE method consists of developing a mathematical model to represent the graphical functional model and risk information developed in the second Pre Step. This mathematical model serves as the computational basis of analysis of the system. Building on my previous work on failure analysis and PHM in functional models, the logic by which failure propagates can be described and analyzed [36,37].

It is important to assign failure distributions to systems that accurately represent how failure is passed between systems [161]. These failure distributions will describe the instantaneous hazard rate of the system. Condition-based failure distributions must be selected that are dependent on the flows passed into and out of the system, and often are dependent on the time over which the system is utilized (though not exclusively, and could be dependent on resources such as the flow of cooling fluid at appropriate levels or available energy).

Once the individual systems have been analyzed in order to determine how failure will propagate [36,39,162], the entire nested super system assembled in Step 1 can be modelled. The super system model is constructed in the same manner as a single functional model, but with systems in the place of sub-systems. The end product is a mathematic representation of a risk-informed functional model that can track the passage of flows between all mission systems and actively reported estimated system health.

Step 4: Define a Mission Plan

A mission plan is used in AMSE to develop future scenarios for mission success probability calculation. In the context of this work, the mission plan is composed of the objectives listed in Section 6.3.1.

For use with AMSE, the mission plan is then broken down further into actionable items that can be completed by agents in the mission. These actionable items are referred to as "tasks." Examples of tasks for a rover mission include driving a specific distance, performing a scientific operation, or performing communication with Earth. For this case study, the autonomous decision-maker selects and orders the tasks.

Step 5: Develop Task Modules

Task modules are important to develop for the AMSE method because they cluster commonly paired tasks together to reduce complexity. Task modules include the duration that a task is to be performed, all systems and resources used during the task, and any fatiguing or consumption of systems affecting the health of systems that may occur during completion of the task. This information will be necessary for analyzing the mission in Phase 2 of AMSE.

Step 6: Organize Tasks into a Task Plan

Using the task modules generated in Step 5, the next step is to organize the task modules into a task plan that defines typical operations or schedules that are to be followed by the mission plan. For example, a task plan can represent all of the tasks to be completed on a particular type of day, such as a day that an EVA is to be performed by a human crew member. Additionally, a typical week can be assembled from task

plans for days and made into a larger meta-task plan. The bundling of task modules into task plans allows for more rapid reconfiguration of the system model for analysis.

Step 7: Arrange Task Plans to Align with the Mission Plan

The general mission plan defined in Step 4 is now filled in with task plans developed in Step 6. This enables the analysis of the mission using AMSE by providing a time-discretized list of all of the actions and systems that are to be used for completion of the mission as a whole. Figure 31 shows how task modules are assembled into task plans and then arranged to align with the mission plan.

While each of the seven steps of Phase 1 must be completed before using AMSE, and the initial modeling can involve a large time investment, once many of these steps have been performed they do not have to be performed again. If the model needs to be reconfigured in order to account for an unforeseen circumstance or to iterate on the mission design (in the case of using AMSE for mission design rather than mission operations), adjustment of the models developed in Step 3 or reconfiguration of the Task Plans in Step 6 can account for the majority of changes that may need to occur to the mission plan and its constituent parts. Due to the ease of configurability enabled by the initial up-front investment of time and resources in model building, AMSE models can be reconfigured rapidly to adjust to unforeseen circumstances or examine a variety of options to inform a mission control decision.

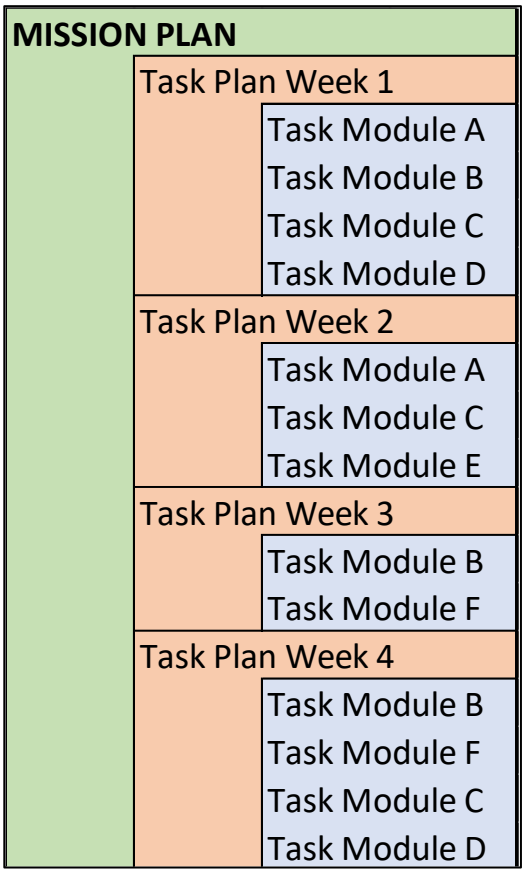


Figure 31: Example of a Mission Plan

Phase 2: AMSE Analysis

As with Phase 1 of AMSE, the second phase, analysis, requires the investment of time and resources to generate the mission models for analysis. Unlike Phase 1, Phase 2 only must only be set up once and will be run whenever the evaluation of a new mission model is desired. The majority of the math necessary for Phase 2 was already developed from Step 3 of Phase 1 where the mathematical representation of the mission was developed. The performance of Phase 2 takes the form of execution of an algorithm consisting of eight individual steps. The eight steps that comprise the Phase 2 algorithm are detailed below. A flowchart of Phase 2 algorithm can be seen in Figure 32.

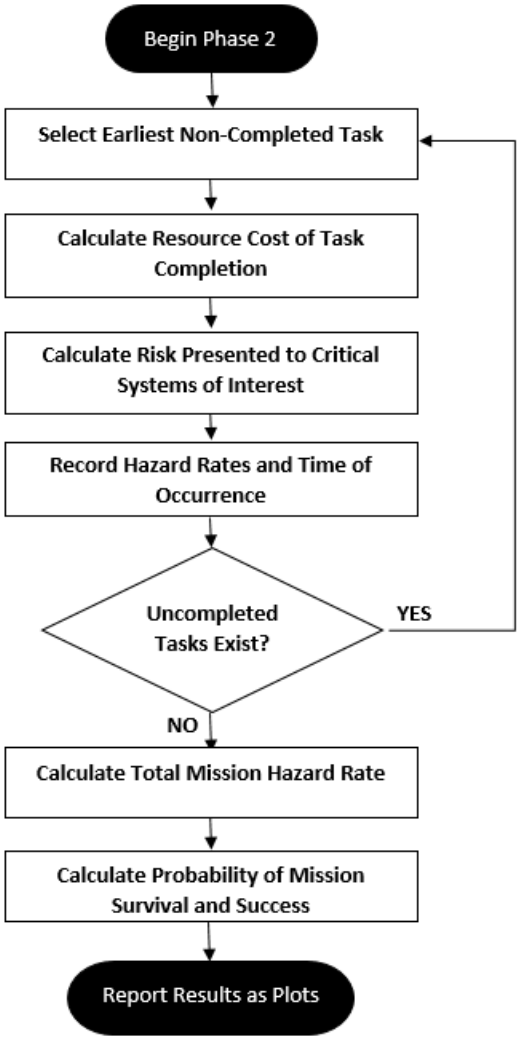


Figure 32: Analysis Process Flow

Step 1: Step through the Mission Plan

Starting with the earliest task that has not yet been analyzed, select each task and then perform Steps 2 through 5 on them. This is necessary to analyze how the success rate of the mission develops over time.

Step 2: Calculate Resource Cost of Task and the System Health Effects

Any resources consumed or systems fatigued by the completion of the task must be clarified. One implementation of this is a resource matrix that contains how much of each resource is available, and subtract from the matrix as resources are consumed. A similar approach can be utilized for the tracking of system health from mechanical wear, environmental conditions, or energy usage.

Step 3: Calculate Hazard Rates Presented to Critical System of Interest

Utilizing the mathematical system model with health information developed in Phase 1, calculate what the risk of system failure is for completion of the task. I recommend calculating the risk in the form of an instantaneous hazard rate, $\lambda(\tau)$, representing the number of system failures expected at the instant τ.

Step 4: Record Hazard Rates

A matrix containing hazard rates for the systems of interest and the time at which the hazard rate was reached should be generated. This will be necessary for the calculation of a total mission failure and success rate in later steps. The matrix values for the first three sols spent on Mars for one of the astronauts in the case study presented in this paper is reported in Appendix 2.

Step 5: Repeat Until Complete

If tasks still exist in the mission plan that have not yet been analyzed, return to Step 1 of Phase 2. If all tasks in the mission plan have been completed, then continue to Step 6 of Phase 2.

Step 6: Calculate Total Mission Hazard Rate

The mission hazard rate, $\Lambda(t)$ defines how often failure is to be expected while executing a mission. For the case study presented, failure is defined as the loss of functionality of the greenhouse or the human habitat module. However, for a manufacturing process, it could be shutting down the production line or the generation of product that doesn't meet quality standards.

Taking the instantaneous hazard rates generated from the functional models and real-time PHM information developed in Steps 2 and 3, calculate the total hazard rate for the remainder of the mission time as a function of time over the entire length of the mission. Like the instantaneous rate $\lambda(\tau)$ the total mission hazard rate $\Lambda(t)$, describes the number of expected system failures per unit of time. While this can be found using integration of continuous data, for the purpose of discretized data generated in completing the AMSE method, a weighted average can find the total mission hazard rate. I instead summed the product of the instantaneous hazard rate for a task and the duration of a task, $\Delta\tau$, and then dividing by the total mission length, *T*, minus the current time of the mission ([15).

$$\Lambda(t) = \frac{\sum_{\varepsilon \in E^t} \boldsymbol{\lambda}(\varepsilon) \cdot \Delta\tau_{\varepsilon}}{T - t} \quad \left[\frac{Losses\ of\ System}{Mission}\right] \qquad [15]$$

Step 7: Calculate Probability of Mission Survival over Time

In this step, calculate the probability of mission survival over time, $S_i(t)$, for a critical system of interest, i, using the total mission hazard rate as shown in [16. In the case of a single critical system of interest, $S_i(t)$, is equivalent to the total mission probability of success, $P^{success}$. However, in the case of multiple critical systems of

interest, $P^{success}$ is equivalent to the intersection of the probability of mission survival, $S_i(t)$, for all systems as shown in [17 Formulation 2 below.

$$S_i(t) = e^{-\Lambda_i(t)\cdot(T-t)}\left[\frac{Systems\ Survive\ Mission}{Attempt}\right] \quad [16]$$

The probability of total mission success is calculated for mission time, t

$$P^{success}(t) = \bigcap_i S_i(t) \quad \left[\frac{Successful\ Missions}{Attempt}\right] \quad [17]$$

Step 8: Pass Mission Success to Decision Maker

The autonomous decision maker is given the total probability of mission success as well as time discretized mission hazard rates for the entire mission. The decision-maker will use the probability of mission success to evaluate the general quality of the current mission plan and the hazard rate over time to identify which tasks are most affecting the probability of mission success.

6.3.7 Experimental Setup

I performed three experiments to explore the performance and behavior of the autonomous decision maker. The experiments were designed to look specifically at how well the autonomous decision maker was able to respond to the unexpected and re-plan the mission accordingly.

The first experiment involved the autonomous decision-maker attempting a mission command and control problem in which five robots are directed to establish a Martian settlement site for human occupation. In this experiment, random failure of systems was included through the use of a fault injector as described in Section 6.3.4, and a

black swan generator was used to add one or more unanticipated scenarios to the mission as described in Section 6.3.5. The black swan events occurred at a much higher rate than would be realistic; this was deliberate, to examine how well the decision maker responds to the unexpected. I initiated 50 trials of this experiment.

In the second experiment, the autonomous decision-maker attempted a mission command and control problem with fault injection, but without any black swans. This experiment provides a baseline for comparison to determine how well the autonomous decision-maker responded to unexpected events. I performed only ten trials of this experiment because there was less variation in the mission conditions.

The third experiment looks in detail at what black swan events most affected mission performance individually. This experiment will allow us to see if there are any trends in the type of black swan events that are particularly hard for the autonomous decision maker to handle. For this experiment, I performed 40 total trials, ten trials for each black swan type.

6.4 Mission Command and Control Results

The primary result of the Mission Command and Control experiment is that the autonomous decision maker was able to successfully plan and execute the mission and respond to failures and random unexpected events. This was shown through 30 trials in which the autonomous decision maker had to respond to one or more black swans. In these trials the control with no black swans did not perform significantly better (p-value of 0.13), meaning that the autonomous decision maker was able to complete the mission with black swans at the same rate as the control. Table 5 summarizes the results of the experiment. The P(Success) column was the probability of mission success given

black swan events and robot failures. A low P(Success) average and a high completion rate means that the decision maker was able to respond well to unanticipated events.

Table 5 Autonomous decision maker response to one or more black swans

Trial	Status	Sand Storm	Failure Distribution	Ground Cover	Site Selection	P(Success)
1	Completed					0.984
2	Completed		X	X		1.000
3	Completed				X	1.000
4	Completed	X				1.000
5	Completed	X				1.000
6	Completed				X	1.000
7	Completed		X	X		1.000
8	Completed		X	X		1.000
9	Completed				X	1.000
10	Completed		X	X		1.000
11	Completed		X	X		1.000
12	Completed				X	1.000
13	Completed	X			X	1.000
14	Completed					0.558
15	Completed				X	1.000
16	Completed		X	X		1.000
17	Completed				X	1.000
18	Completed					0.963
19	Completed				X	1.000
20	Completed	X				1.000
21	Completed					0.916
22	Completed	X	X	X		1.000
23	Failed					0.570
24	Completed				X	1.000
25	Completed					0.810
26	Completed	X				1.000
27	Completed	X				0.558
28	Failed		X	X		0.090
29	Completed				X	1.000
30	Completed					0.558
31	Completed					0.570
					Mean	0.890
					STD	0.220

The results from the control experiment are shown below in Table 6.

Table 6: Control- Response to no black swans

Trial	Status	P(Success)
1	Completed	0.999999988
2	Completed	0.999999516
3	Completed	0.999999988
4	Completed	0.999999988
5	Completed	0.999999988
6	Completed	0.999999554
7	Completed	0.999999988
8	Completed	0.999999988
9	Completed	0.999999988
10	Completed	0.999999988
11	Completed	0.999999515
12	Completed	0.999999988
13	Completed	0.999999988
14	Completed	0.999999988
15	Completed	0.999999988
16	Completed	0.999999988
17	Completed	0.999999988
18	Completed	0.999999988
19	Completed	0.999999516
20	Completed	0.999999988
	Mean	0.999999895
STD		1.89985E-07

Additionally, experiments were performed to determine how much inclusion of specific black swan events affected performance.

Table 7: Response to Individual Black Swans

	SANDSTORMS		Incorrect Failure Distribution		Ground Cover		Site Selection	
Trial	Succeed	P(Success)	Succeed	P(Success)	Succeed	P(Success)	Succeed	P(Success)
1	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.570042254
2	Completed	0.999999988	Failed	0.999999988	Completed	0.999999988	Complete	0.983555933
3	Completed	0.999999988	Failed	0.999999988	Completed	0.999999988	Complete	0.96278783
4	Completed	0.999999554	Completed	0.999999554	Completed	0.999999554	Complete	0.915866177
5	Completed	0.999999988	Failed	0.999999988	Completed	0.999999988	Complete	0.96278783
6	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.558099763
7	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.804289021
8	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.81000157
9	Completed	0.999999988	Completed	0.999999988	Failed	0.999999988	Failed	0.557774253
10	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.809830816
11	Completed	0.999999988	Completed	0.999999988	Failed	0.999999988	Failed	0.557702621
12	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.962837202
13	Completed	0.999999988	Failed	0.999999988	Completed	0.999999988	Complete	0.96278783
14	Completed	0.999999988	Failed	0.999999988	Completed	0.999999988	Complete	0.913566395
15	Completed	0.999999988	Completed	0.999999988	Failed	0.999999988	Failed	0.557774253
16	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.915852549
17	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.81000157
18	Completed	0.999999988	Completed	0.999999988	Failed	0.999999988	Failed	0.558099763
19	Completed	0.999999988	Completed	0.999999988	Completed	0.999999988	Complete	0.558099763
20	Completed	0.999999554	Completed	0.999999554	Completed	0.999999554	Complete	0.983555933
	Mean	**0.999999944**	Mean	**0.999999944**	Mean	**0.999999944**	Mean	**0.785765666**
	STD	1.33417E-07	STD	1.33417E-07	STD	1.33417E-07	STD	0.179749888

6.5 Discussion of Mission Command and Control

There are multiple significant contributions stemming from this work. It is the first time a mission command and control problem has been solved using computational cognition. Additionally, it is the first time that top-down computational cognition has ever been used to construct an autonomous decision maker. As a result, the discussion of this work extends beyond the analysis of the experimental results to include discussion of the effectiveness of implementation, observations on notable emergent behavior that may require further exploration, discussion of how heuristics and other simplifying assumptions influenced performance, and disclosure of the flaws and shortcomings of the work.

6.5.1 Discussion of Results

I performed three experiments to evaluate the autonomous decision maker's ability to solve mission command and control problems. In the first experiment, one or more black swan events occurred randomly in 30 trials. In the second experiment, no black swans occurred to provide a baseline for comparison. In the third experiment, each black swan occurred alone in 10 trials each.

6.5.1.1 Experiment 1: Randomized Black Swans

As shown in Table 5, the autonomous decision-maker performed very well at planning missions and responding to system failures and multiple black swan events. In the 30 trials, there were only two situations in which the mission failed.

In the first failure, a failure rate distribution that did not accurately reflect the mission parameters was used (a bathtub curve instead of a traditional Weibull

distribution), and all five robots failed shortly after starting. This lead to an accelerated failure rate that was unknown to the decision maker, meaning that robots were more likely to fail and the decision maker did not know why.

In the second failure, both the failure distribution the autonomous decision-maker assumed, and the site selection were affected by black swans. In this scenario the Weibull distribution was replaced with a bathtub curve, leading to more early failures and the mission site was rockier. This lead to a compounding effect where robots failed at a higher rate and the amount of work needed to clear the mission site of debris was significantly higher, and as a result, all of the robots ended up failing, making it impossible to complete the mission.

Another notable result is that the instantaneous probability of mission success calculated by AMSE always decreases over time, but rarely dipped below 95% for before the mission was re-planned, and the probability of success would increase. This means that the ability to re-plan a mission is critical when performing mission command and control problems. This finding supports the use of a heuristic trial and error approach that allowed for rapid mission planning.

6.5.1.2 Experiment 2: Control

During this experiment, the mission simulation included fault injection but there were no black swan events. This was done to provide a baseline of comparison to see how the decision maker would perform under ideal conditions.

There was not a single trial that ever dipped below a 99.9% success rate, and the only re-planning needed was in response to the occasional failure of a single robot.

6.5.1.3 Experiment 3: Individual Black Swans

In experiment 3 I explored how the individual black swan events affected performance. It is notable that by definition a black swan event occurring during a mission is incredibly unlikely, and for this experiment, we artificially included them to evaluate the decision maker's response.

The sandstorm was not likely to cause mission failure. In the simulation, this forced the robots to start working and added some fatigue to physical systems, but it was comparatively minor.

Incorrect failure distribution assumptions and ground cover assumptions and poor site selection, however, had significant effects on mission success.

The incorrect assumptions both lead to the decision maker attempting to make decisions given a model of the mission that was incorrect. This is incredibly difficult to respond to and is analogous to a human assuming the floor is made of lava. As a result, the decision maker made a variety of choices that lead to accelerated fatigue of the robots leading to premature failure. One way to overcome this in future iterations of the autonomous decision maker would be to allow online learning so that the decision maker can more easily adjust its models of the mission as time goes on.

Poor site selection leads to the decision maker picking a site that was much rockier than anticipated for the initial survey. This leads to increased wear on the robots and meant that the settlement site would require far more work to clear after selection. The combined effects of increased wear and additional work lead to many more robots failing and the decision maker to have to perform much more re-planning operations.

However, despite this, the decision maker only failed to complete the mission in four of the 20 trials.

6.5.2 Coding and Implementation of Mission Command and Control

Coding the computational cognition-based autonomous decision-maker presented unique challenges that are not present in traditional approaches to autonomous decision making.

Traditional autonomous decision-making methods are linear; generally, their structure follows a formula of problem definition and modeling, analysis of the problem, and the selection of the best options. In many cases, the time and effort needed for the problem definition and modeling are greater than the time needed to code the analysis. These methods are limited, however, because they only apply to a narrow set of simplified problems, and they cannot adapt to decision spaces that change over time. By applying a cognitive computation approach, we can better adapt to changing decision spaces. However, this leads to a decision maker that involves multiple decision-making processes without a predefined order.

The code was implemented in MATLAB [96] as a series of script files. I started by creating the appropriate script files and populating them with pseudo code which introduced the challenge of planning something that changes order over time. The strategy that I adopted was to first go through the entire process flow assuming that no replanning was needed and define all of the code that would be involved in planning and implementation. Then I went through the code a second time assuming that the fault injector had caused a system to fail.

In future recreations of the work something that may have helped simplify this would have been starting with a large process flow diagram that exhaustively described the code and then restructuring the code to make implementing the nonlinear process of planning analyzing and re-planning much smoother. However, it is likely that developing an autonomous decision maker of this type might always require a large degree of nonlinearity.

Exploring best practices for designing and coding autonomous decision makers through computational cognition is a critical piece of future work for this method to become more generally applicable.

6.5.3 Notable Emergent Behavior

When working with autonomous systems and uncertainty, a common occurrence in the development of emergent behavior. Emergent behavior is the result of many components with simple functions operating together and creating something more complex than the sum of their parts. In many applications, emergent behavior in an autonomous system is undesirable and can lead to unpredictable results, such as those discussed in Section 4.5. In this work, the goal was to create complex emergent behavior that was able to respond to problems by throwing a bunch of smaller processes together. As shown by the results this was largely successful, and the autonomous decision maker was able to respond rapidly and effectively to unexpected situations.

However, there were also some undesirable and strange emergent behaviors. For example, for tasks involving moving objects from one place to another, the autonomous decision-maker could either assign the task to be performed by a Surface Exploration Vehicle (SEV) and a robot, or just a robot member on its own. The inflection point of

this turned out to be very sensitive, and the decision maker ended up either giving all of the tasks to the robots or would do everything using the SEV, even if it involved moving something only a few meters. One insight that can be gained by this is that there seems to be a gap in the tools available to the autonomous decision-maker, and that it would benefit by adding additional systems (such as a self-driving robotic SEV) into the mission environment that combines the elements that are useful from both systems. In future work, this could be explored further by allowing the autonomous decision maker to have more control over the setup and problem definition steps, and enabling it to determine what systems are in the mission.

Another problematic emergent behavior that developed was that the autonomous decision-maker initially attempted to get "creative" with task assignments and functional models. For example, if the robot's locomotion systems had more wear on them than the robot's arms, then the decision maker would tell the robots to use their hands to walk somewhere. While this behavior is impractical and unrealistic, unexpected solutions like this show that the autonomous decision maker is capable of finding novel approaches to problems that may occur. However, in this particular case, the behavior was highly undesirable because it did not consider the practicallity of supporting the weight of the robot on its arms while attempting to move rocks with its feet, so I changed the flows for translational work in the functional model to differentiate between work done by arms and legs. In future work, this behavior should be further examined by creating mission scenarios where more "creative" solutions are possible, and developing a decision maker that is better at differentiating good ideas from bad ideas.

6.5.4 Reliance on Heuristics and Mildly Irrational Decision Making

One of the most common but flawed assumptions made in decision making is that all agents will and should make rational and informed decisions. Daniel Kahneman and Amos Tversky rejected this notion and presented empirical evidence that human decision making is not always rational [11,12]. They also showed that irrational heuristic-based approaches to decision making are very effective and efficient [5]. To a small degree, many computational approaches to decision making have built on this through the use of computational heuristics and metaheuristics [88,139,148]. However, these approaches often build heavily on expected utility theory and other hyper-rational approaches.

In this work, there were many cases where a quick decision was needed, but there were many unknown variables that could influence the outcome. The heuristics I employed were closer to the original definition of heuristic meaning a hands-on trial and error approach. The autonomous decision maker made quick decisions based on factors such as the amount of prior use of a system or proximity to a location so that it could make a very rapid decision, and then re-plan if more problems arose.

The trial and error heuristics approach reduced the time needed to plan the mission allowing for a mission consisting of approximately 50 sub-objectives and five agents to be planned in approximately 50 seconds with a high degree of success. It is notable however that this is a very risk-tolerant approach to the problem because, we did not care if we lost one or two of the robots. However, the decision maker was managing humans and loss of life was a real possibility, then more risk-averse approaches should be explored. In most contexts, however, the ability to be risk tolerant and efficiently

generate complicated mission plans is incredibly valuable and could greatly reduce labor to manage large complicated systems.

6.5.5 Flaws and Shortcomings of Mission Command and Control

One of the major shortcomings of the current autonomous decision maker for mission command and control is that it involves a lot of effort to initially set up the problem. For this case study, I defined tasks and objectives, create functional models, and develop a simulated mission environment. However, for many real world applications the comparative cost of employing an engineer for a few weeks to model the problem is worth significantly reducing management costs. One way to improve this process would be through the creation of an exhaustive repository of tasks and functional models, so that the user can import them as needed. Additionally, as this method matures a software package could be developed that could be used by non-engineers to model, simulate, and plan a mission or process.

7. DISCUSSION

Throughout this dissertation I have explored characteristics of the information fidelity gap, design decision trees, and mission decision problems and I have evaluated a variety of techniques on their ability to be applied to these problems. This section discusses general observations made over the course of the work, and discusses what I have learned about the information fidelity gap, design decision trees, and mission decisions.

7.1 Notable Observations

7.1.1 Problem Space Representation

One of the first observations I made while undertaking this work is that the size and complexity of a problem space representation can always be increased, and will never have the full fidelity of reality. Additionally, when it comes to designing an autonomous decision maker that operates in a real world environment the accuracy of the problem space representation greatly affects performance. This is the problem that I ended up calling the information fidelity gap.

I explored various ways to bridge the information fidelity gap, but the most useful that I found was a technique I call global to local planning. The general approach involves starting with a low fidelity representation of the operating environment, and taking time to make the best plan possible given the information available. Then when executing the plan, update the model of the operating environment through locally observable information and adjust the plan accordingly. This allows a decision maker to make a series of small easy decisions in order to accomplish a larger more difficult

task. This approach was used successfully in the first case study by a rover that started the navigational problem by creating a planned route with a low fidelity map, and then deviating from the plan as necessary based on locally observable conditions in the high fidelity operating environment.

7.1.2 Unknown information

The second notable observation is that decisions made in ignorance tend to be poor decisions. While this seems obvious, it is notable because it came up multiple times throughout this dissertation in various forms. In the rover navigational problem, rovers with complete environmental information tended to perform better than their less informed counterparts (though these effects reduce as the amount of information increases past the current decision horizon).

The Color Graph problem also illustrates the importance of accurate information by showing how difficult it is to design something when no information is available until the design is complete. Interestingly the best approaches to the Color Graph problem tend to take a very exploitative approach by randomly generating design and evaluating it on quality, and then randomly changing parts until a good solution was found.

In the mission command and control problem, the black swans that most hindered decision maker performance were events that separated the decision maker's assumptions about the environment with reality. For example, when the survey site did not look like the telemetry data used to select the survey, the mission was significantly more likely to fail due to increased wear on the robots and greater amounts of labor needed to clear the settlement site of debris.

7.1.3 Problem Evaluation Time and Heuristics

In both the rover navigation case study and the mission command and control case study very efficient heuristics were able to be used because if something went wrong additional decisions could easily be made to fix the problem. However, for the Color Graph design problem much more effort needed to be put into which decision to make or the design quality would suffer. I believe that this is related to how the problems are evaluated and what actions can be taken after evaluation. Both the rover and mission command problem allowed for the decision maker to make changes after getting a solution evaluated, but when designing Color Graphs the designs can only be evaluated when completed, and changes cannot be made without creating a completely different design.

Knowing when it is and isn't appropriate to use a simple heuristic instead of a more complicated form of analysis is important because by understanding this dynamic we can learn how to make better decisions with less effort. In future work, I would like to study this dynamic in more detail and try to characterize better when heuristics are most useful. This is important both in the design of an autonomous decision-maker, and to better understand how humans are such good decision makers.

7.1.4 Top-Down Computational Cognition Structures

In this dissertation, I developed a top-down approach to building an autonomous decision maker based in computational cognition. This approach starts with a top-level functional representation of the decision-making problem and then develops

algorithmic processes informed by the functional level that are managed by a lower structural level that determines which process to run when.

The top-down approach worked very well when applied to decision-making problems where there was an accurate functional representation of the problem space. In these cases, many complex decisions were able to be made incredibly quickly and effectively as shown in Chapter 6. However, when the functional representation of the problem space breaks down and is no longer accurate the quality of the decisions made reduced dramatically.

One way this could be addressed in future work, would by adding additional learning processes to the cognitive structure that allow the decision maker to update its functional representation of the problem space and improve decision making over time. However, even without this capability, the autonomous decision-maker evaluated in Chapter 6 was able to respond very well to many situations that were not included in its functional representation of the system.

7.2 Design Decision Tree Problems

Design decision trees were the primary focus of the Color Graph study, but also came up repeatedly in the Martian settlement Mission Command and Control case study. In the Color Graph case study, I explored what methods are most effective for approaching design decision tree problems through the use of a toy problem. Then, in the Martian settlement case study, I applied the insights I gained to create a cognitive operation capable of efficiently solving design decision tree problems in the form of planning objectives and ordering tasks.

The major insight gained was that methods that focus on improving a good early design by making small changes to a path were preferable to methods that continually searched for the best possible node and were better suited to solving design decision tree problems. Additionally, that methods like the genetic algorithms that can take multiple high performing paths from the tree and combine them or modify them directly converged very quickly to a high performing design.

One notable observation of this work was that the effectiveness of the algorithm for the Color Graph case study was more important than for the Martian settlement case study. This was because, in the Martian settlement case study, poor design choices were evaluated a second time in a simulation and the decision maker would go back and re-design the mission plan as needed. However, for the Color Graph study, this was not performed. In future work, a top-down cognitive decision maker could be developed that builds on this insight by using simulation to evaluate designs generated by an automated design process.

7.3 Mission Command and Control Decisions

Both the autonomous rover case study and the Mission Command and Control for the establishment of a Martian settlement case study explored aspects of mission command and control decision problems.

In the rover case study, I saw how preplanning the mission plan with even low fidelity information about the mission environment could greatly improve the performance of the autonomous rovers. By starting with a rough path plan, the rovers were much more likely to complete the mission, because they had a better

understanding of how individual decisions made given high fidelity local information would affect future mission performance.

In the Mission Command and Control for the establishment of a Martian settlement case study, the autonomous decision maker needed to make a large number of mission command and control decisions. By starting with a low fidelity map of the environment and then improving the quality of the relevant section of the map through exploration the autonomous decision maker was able to make very effective decisions about path planning, settlement placement, and task allocation.

The top-down computationally cognitive decision maker was very effective at making mission command and control decisions given this mixed fidelity model of the mission environment and was able to complete 100% of the missions in which no black swans were present. When I added random black swans that the decision maker's did not have knowledge of in their mission model this rate only dropped to 93% with only two cases of mission failure. This is incredibly promising, because it shows that top-down computationally cognitive decision maker is capable of recognizing when something is wrong, and through trial and error is able to come up with a plan that still completes the mission despite not being able to identify exactly what is wrong.

However, black swans that increased the information fidelity gap signicantly reduced the decision maker's ability to complete the mission, with an average mission completion of 77%. This is because the decision maker could replan the mission all it wanted, but it would always make flawed decisions because the model that it used to inform its decisions was flawed. However, the fact that it was still able to complete the majority of the mission is very promising, and in future work inclusion of learning

cognitive operations that allow the decision maker to update its model could dramatically improve performance.

8. CONCLUSION

In this dissertation, I explored why traditional autonomous decision-making techniques struggle to perform under conditions of uncertainty, risk, and extreme complexity, and how performance can be significantly improved through the application and synthesis of techniques from decision theory, risk analysis, game theory, and computational cognition.

Through the three case studies presented I identified and characterized the information fidelity gap, design decision trees, and mission command and control decisions. I explored various methods for addressing these problems in different contexts, and I showed how to improve the performance of autonomous decision makers facing uncertainty, risk, and extreme complexity through the application of techniques from decision theory, risk analysis, game theory, and computational cognition.

8.1 Overview

The three case studies I performed were a terrain navigation problem performed by risk-informed autonomous rovers, an automated Color Graph design problem, and a Martian settlement mission command and control problem.

The risk-informed rovers case study presented in Chapter 4 explores how decision making is affected by information fidelity when facing risk. Through multiple experiments, it was shown that as information available to the decision maker increases, the quality of the decisions also increases. However, this effect diminishes greatly as the level of information approaches problem fidelity.

The rover that performed best had a low-quality map of the entire environment in addition to complete information about the immediate environment. The map allowed the rover first to develop a rough navigational plan based on the incomplete information, and then improve the plan based on higher fidelity local information. This global to local approach allows for more complex problems to be broken down into two easier problems where the decision maker first solves a simplified version of a big difficult problem and then solves a bunch of smaller easier problems to improve the quality of the initial solution. This demonstrates how the decision theory concept of a heuristic can be used to substitute easier problems for difficult problems when facing uncertainty and risk.

I used this same approach for the structure of the mission command and control decision maker explored in Chapter 6, where an initial mission plan was made using a small amount of information, and then revised as needed.

The second case study explored an automated design decision tree problem for the design of a Color Graph. This research asked what the best approach to finding a solution when the design space is uncertain and complex. For this problem, I applied concepts from game theory to develop a game tree representing the color graph problem. This game tree was unusual because its branching factor increased after every decision, making the design space intractably large after very few decisions.

I found that methods that were more exploitative than explorative were much more effective at searching design decision trees such as the Color Graph problem. This is because they were less likely to get lost in the design space and were able to better leverage knowledge gained through previous attempts at the problem. I applied this

knowledge to the development of the autonomous decision maker for mission command and control decisions, by treating the objective planning and task ordering problems as design decision tree problems and using a genetic algorithm (GA) to solve them.

The Color Graph case study shows how applying concepts and techniques from game theory can be used to effectively model complex problems and enable better autonomous decision making.

In the third case study, a mission command and control problem was attempted. The goal of the mission was to command five RoboSimian inspired robots [153] to set up a Martian settlement site for later human occupation. For this problem, I used concepts from computational cognition and decision making to develop an autonomous decision maker that could solve the problem.

The primary question of this case study was what structures are most critical to the development of an autonomous decision maker, and what external factors most influence the decision makers ability to function.

Through three experiments I found that the functional representation of the problem is the most critical to making good decisions, because if the functional representation is wrong, then the decision maker will be much more likely to make poor decisions. This was demonstrated by how much some of the black swans affected system performance. However, I also found that a low-level structure that identifies problems as they occur and applies the correct processes to solve them made the decision maker very robust and allowed it to rapidly respond to changing circumstances.

8.1.1 Impact of the Work

The work presented in this dissertation breaks new ground in the fields of design, decision theory, and automation. The autonomous decision maker based on top-down computational cognition is the first of its kind and lays a foundation that will allow for future development of autonomous decision-makers that excel when faced with uncertainty, risk, and extreme complexity.

Further development of this technology could enable autonomous mobile robots that are more capable of working in cooperation with humans, air and spacecraft that perform their own navigational calculations while also managing system maintenance and adapting to new environments, automation of the construction industry and manufacturing processes, and safer self driving cars that are better equipped to handle the unexpected.

Additionally, within this work, I have defined and characterized the design decision tree and developed the Color Graph problem to study methods for searching design decision trees. Establishing Color Graph as a baseline problem for comparison will allow automated design techniques to be compared more directly, cutting down on the unnecessary recreation of domain-specific work and increasing the pace of development of new automated design methods.

Development of improved automated design methods will increase the quality of designs while decreasing the labor required. The technology could enable a future where a client dictates product needs to a computer, which then designs the product, develops its manufacturing process, and solves logistical problems such as shipping and inventory management without any human intervention.

One of the most promising aspects of this work is that it focused heavily on development of techniques that could be run with very limited computational power. This is critical, because it means that it could more feasibly be implemented on an autonomous mobile robot that doesn't rely on any other systems for decision making or management. However, as shown in the risk informed rover study, pre-planning can greatly improve mission performance. In many cases this could be performed as an ordinary slow cognitive operation, where the autonomous system just stops and performs the calculation before continuing. However for particularly large problems, performing pre-planning on a machine with more computational power and then giving the rough mission plan to a small less computationally powerful robot could also be a very effective approach.

8.1.1.1 Example Application

One of the most exciting potential applications of this work is the complete automation of the design and manufacturing process. In this section I discuss how this could be done by applying insights gained throughout this dissertation using the example of designing and manufacturing an office chair. This could be done through nine steps.

First, determine the client needs for the office chair. Perform interviews and other traditional design practices to determine what sort of forces and use cases the chair is likely to experience. Additionally determine what physical design characteristics the chair needs or should draw on. What materials, shapes, and features are desirable?

Second, develop an objective function that captures the client's desires, and potentially develop a low fidelity model of the chair operating environment. The

objective function (or functions) should include attributes like expected loads, product life, cost, materials, and any other quantifiable metrics of interest. The objective function will be used in a design decision tree problem (as described in Chapter 5). Additionally, a low-fidelity physics model of the chairs operating environment could be developed to further evaluate design quality, as this would likely lead to improved behavior as indicated by the results of the risk averse rovers case study.

Third, develop a design decision tree representing the design process for the chair. The design decision tree should start with a seed (for a chair this should probably be a seat, but it could also be the floor, or even a sitting human) and then the decision tree should consist of selecting a current location on the design and then a second decision of determining what feature should be added there. Additionally, other choices such as material or feature size could also be included in the tree, but at a minimum it much include a location and an added feature decision. This design decision tree will be multimodal, opaque, unbounded, and have increasing internal locations.

Fourth, select an appropriate algorithm for searching the design decision tree. As shown by the Color Graph case study, the most effective algorithm may not be the one you expect. The findings of this research indicate that selecting a method that rapidly selects a single adequate design tree path and then iterates on it is the best approach, but further exploration is highly recommended, especially in the case where additional types of steps have been added (such as a secondary material selection). Setting up a Color Graph problem with additional steps would be a great way to validate algorithm selection, and will also allow the engineer to evaluate methods that have not yet been considered. Additionally, as indicated by the Mission Command and Control for a

Martian settlement case study in Chapter 6, inclusion of learning to better account for unexpected scenarios will likely increase design quality. One way to do this would be to develop a metaheuristic that fine toons the weightings of the design parameters by presenting several design alternatives to the client early in the process, and comparing how the clients ranking of the designs compares to the algorithms ranking.

Fifth, perform the automated design process using the selected algorithm. As shown by the results of Chapter 5, as the problem gets larger, finding a quality design becomes more difficult, so this may take some time. However, selecting an efficient algorithm in the previous step should lead finding a quality design much faster.

Sixth, develop a mission model of the manufacturing process. This is the top level in the top-down approach to computational cognition. This should consist of functional models of all of the systems involved in the manufacturing, an environmental model of the manufacturing environment, important market metrics such as cost and lead times on components, and a mathematical representation of the manufacturing process [163]. These models should be used to develop a success estimator that focusses on the outcome of the design process, which in most cases can be found by taking the product of the probability of profit and quantity of potential profit. This model should also impose an additional penalty on losses in order to include insights from prospect theory [11], which will increase the probability of mission success as shown by the rovers in Chapter 4. Additionally, having an engineer in the loop that it can call when it gets stuck due to risk averse behavior would likely improve performance while still dramatically reducing labor. However through learning techniques (as described in step four), the need for human intervention could be reduced over time.

Seventh, develop slow cognitive operations to solve all of the individual logistical problems. This is the first part of the middle level in the top-down approach to computational cognition. For the development of a chair this will include selection of materials and processes, and assigning those tasks to human or robot workers. Most of these processes will be similar to design decision trees as shown by the objective and task ordering problems in the Mission Command and Control case study, but they may also include path planning problems or analysis of completed chairs.

Eighth, develop fast cognitive operations for fault detection and data collection. This is the second part of the middle level of the top-down decision maker. This could involve using techniques from prognostics and health management to anticipate failure [164] and using sensors to determine critical component health. Additionally, metrics like chair completed by the production line per hour could be very useful for identifying potential issues as they arise. These operations need to be run constantly and passively and should require as little computational power as possible.

Ninth, develop the cognitive structure. This is the bottom level of the top-down cognitive structure. In this stage the autonomous decision maker is assembled so that fast cognitive operations feed into the manufacturing success model. When the desired level of success (in this case the number of profitable chairs being produced) drops below an acceptable level, then either the manufacturing process or the chair design is partially or entirely redesigned as needed. Figure 33 shows an example cognitive structure.

Tenth, test the decision maker through simulation. This step is optional, but highly recommended as autonomous systems often demonstrate unexpected emergent

behavior as shown in Chapters 4 and 6. Additionally, this will allow for the engineer designing the decision maker to inject black swans as shown the Mission Command and Control case study in Chapter 6. Evaluating how well the current decision maker design responds to black swans will enable the engineer to fine tune the design.

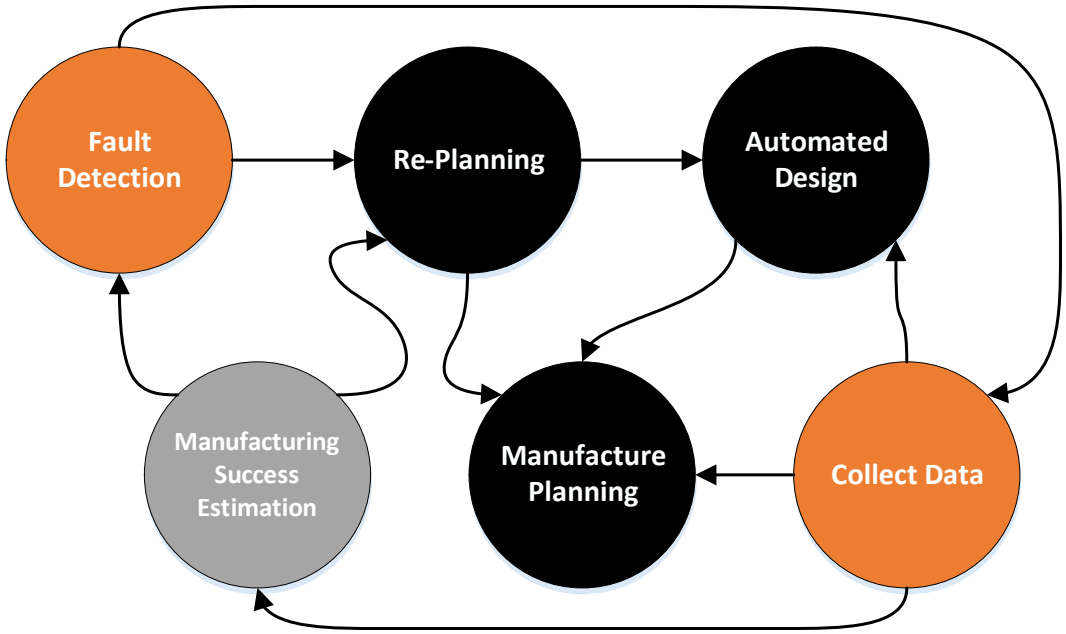


Figure 33: Example Cognitive Structure

8.1.2 Future Work

While the work presented in this dissertation is groundbreaking and is the result of approximately five years of continuous research at three different universities, the concepts present are in many ways still in their infancy. The next step for this work is the get a decision maker based on top-down computational cognition running on a simple mobile robot. This will allow us to explore how well these concepts transfer from simulation into reality. Additionally, exploring how to adapt the decision maker

to other domains, human-machine communication or financial decisions could be very interesting.

One potential avenue for future work that particularly excites me is attempting to introduce biases into the decision making process and seeing how it influences behavior. Extending this work further could allow us to recreate more human-like decision-making behavior which would benefit human-machine teaming problems as it could allow humans to understand better and "empathize" with machine collaborators.

Another interesting line of future work would be the development of wild robots that are designed to perform a function in an uncontrolled environment without any human supervision or input. These robots could perform maintenance tasks, destroy invasive plants, pick up litter or perform other prosocial tasks.

It is my sincere hope that this work will continue, and that these ideas will get more time to mature so that we can all see where they might lead.

9. **BIBLIOGRAPHY**

[1] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D., 2011, *Introduction to Autonomous Mobile Robots*, MIT press.

[2] Imogen, 2016, "7 Popular Wardrobe and Outfit Planning Apps Reviewed," Style.

[3] Van Bossuyt, D. L., Dong, A., Tumer, I. Y., and Carvalho, L., 2015, "On Measuring Engineering Risk Attitudes."

[4] Kochenderfer, M. J., 2015, *Decision Making under Uncertainty: Theory and Application*, MIT press.

[5] Tversky, A., and Kahneman, D., 1975, "Judgment under Uncertainty: Heuristics and Biases," *Utility, Probability, and Human Decision Making*, Springer, pp. 141–162.

[6] Finucane, M. L., Alhakami, A., Slovic, P., and Johnson, S. M., 2000, "The Affect Heuristic in Judgments of Risks and Benefits," J. Behav. Decis. Mak., **13**(1), p. 1.

[7] Kahneman, D., and Tversky, A., 1984, "Choices, Values, and Frames.," Am. Psychol., **39**(4), p. 341.

[8] Wang, F.-Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., Zhang, J., and Yang, L., 2016, "Where Does AlphaGo Go: From Church-Turing Thesis to AlphaGo Thesis and Beyond," IEEECAA J. Autom. Sin., **3**(2), pp. 113–120.

[9] "After Math: Robot Revolutionaries" [Online]. Available: https://www.engadget.com/2018/05/06/after-math-robot-revolutionaries/. [Accessed: 03-Jun-2018].

[10] Alligood, K. T., Sauer, T. D., and Yorke, J. A., 1996, *Chaos*, Springer.

[11] Kahneman, D., and Tversky, A., 1979, "Prospect Theory: An Analysis of Decision under Risk," Econom. J. Econom. Soc., pp. 263–291.

[12] Tversky, A., and Kahneman, D., 1992, "Advances in Prospect Theory: Cumulative Representation of Uncertainty," J. Risk Uncertain., **5**(4), pp. 297–323.

[13] Mongin, P., 1997, "Expected Utility Theory," Handb. Econ. Methodol., pp. 342–350.

[14] Rabin, M., 2000, "Risk Aversion and Expected-Utility Theory: A Calibration Theorem," Econometrica, **68**(5), pp. 1281–1292.

[15] Tenenbaum, J. B., and Griffiths, T. L., 2001, "Generalization, Similarity, and Bayesian Inference," Behav. Brain Sci., **24**(04), pp. 629–640.

[16] Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D., 2011, "How to Grow a Mind: Statistics, Structure, and Abstraction," science, **331**(6022), pp. 1279–1285.

[17] Tenenbaum, J. B., Griffiths, T. L., and Kemp, C., 2006, "Theory-Based Bayesian Models of Inductive Learning and Reasoning," Trends Cogn. Sci., **10**(7), pp. 309–318.

[18] Griffiths, T. L., Kalish, M. L., and Lewandowsky, S., 2008, "Theoretical and Empirical Evidence for the Impact of Inductive Biases on Cultural Evolution," Philos. Trans. R. Soc. Lond. B Biol. Sci., **363**(1509), pp. 3503–3514.

[19] Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., and Tenenbaum, J. B., 2010, "Probabilistic Models of Cognition: Exploring Representations and Inductive Biases," Trends Cogn. Sci., **14**(8), pp. 357–364.
[20] Vityaev, E. E., Perlovsky, L. I., Kovalerchuk, B. Y., and Speransky, S. O., 2013, "Probabilistic Dynamic Logic of Cognition," Biol. Inspired Cogn. Archit., **6**, pp. 159–168.
[21] Fu, X., Cai, L., Liu, Y., Jia, J., Chen, W., Yi, Z., Zhao, G., Liu, Y., and Wu, C., 2014, "A Computational Cognition Model of Perception, Memory, and Judgment," Sci. China Inf. Sci., **57**(3), pp. 1–15.
[22] Robert, C., and Cummins, D. D., 2000, *Minds, Brains, and Computers: The Foundations of Cognitive Science*, Blackwell Publishers.
[23] Hirtz, J., Stone, R. B., McAdams, D. A., Szykman, S., and Wood, K. L., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Res. Eng. Des., **13**(2), pp. 65–82.
[24] Bryant, C. R., Stone, R. B., McAdams, D. A., Kurtoglu, T., Campbell, M. I., and others, 2005, "Concept Generation from the Functional Basis of Design," *ICED 05: 15th International Conference on Engineering Design: Engineering Design and the Global Economy*, Engineers Australia, p. 1702.
[25] Kurtoglu, T., Campbell, M. I., Bryant, C. R., Stone, R. B., McAdams, D. A., and others, 2005, "Deriving a Component Basis for Computational Functional Synthesis," *ICED 05: 15th International Conference on Engineering Design: Engineering Design and the Global Economy*, Engineers Australia, p. 1687.
[26] Stone, R. B., and Wood, K. L., 2000, "Development of a Functional Basis for Design," J. Mech. Des., **122**(4), pp. 359–370.
[27] Coatanéa, E., Roca, R., Mokhtarian, H., Mokammel, F., and Ikkala, K., 2016, "A Conceptual Modeling and Simulation Framework for System Design," Comput. Sci. Eng., **18**(4), pp. 42–52.
[28] Blanchard, B. S., and Fabrycky, W. J., 1990, *Systems Engineering and Analysis*, Prentice Hall Englewood Cliffs, New Jersey.
[29] Bohm, M. R., Stone, R. B., and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," J. Comput. Inf. Sci. Eng., **5**(4), pp. 360–372.
[30] Jensen, D. C., Tumer, I. Y., and Kurtoglu, T., 2008, "Modeling the Propagation of Failures in Software Driven Hardware Systems to Enable Risk-Informed Design," *ASME 2008 International Mechanical Engineering Congress and Exposition*, American Society of Mechanical Engineers, pp. 283–293.
[31] Kurtoglu, T., Tumer, I. Y., and Jensen, D. C., 2010, "A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures," Res. Eng. Des., **21**(4), pp. 209–234.
[32] O'Halloran, B. M., Papakonstantinou, N., and Van Bossuyt, D. L., 2015, "Modeling of Function Failure Propagation across Uncoupled Systems," *Reliability and Maintainability Symposium (RAMS), 2015 Annual*, IEEE, pp. 1–6.
[33] Ramp, I. J., and Van Bossuyt, D. L., 2014, "Toward an Automated Model-Based Geometric Method of Representing Function Failure Propagation across

Uncoupled Systems," *ASME 2014 International Mechanical Engineering Congress and Exposition*, American Society of Mechanical Engineers, p. V011T14A007–V011T14A007.

[34] Stone, R. B., Tumer, I. Y., and Van Wie, M., 2005, "The Function-Failure Design Method," J. Mech. Des., **127**(3), pp. 397–407.

[35] Kurtoglu, T., and Campbell, M. I., 2009, "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping," J. Eng. Des., **20**(1), pp. 83–104.

[36] Short, A. R., Lai, A. D., and Van Bossuyt, D. L., 2017, "Conceptual Design of Sacrificial Sub-Systems: Failure Flow Decision Functions," Res. Eng. Des., pp. 1–16.

[37] Short, A. R., Van Bossuyt, D. L., and others, 2015, "Rerouting Failure Flows Using Logic Blocks in Functional Models for Improved System Robustness: Failure Flow Decision Functions," *DS 80-6 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 6: Design Methods and Tools-Part 2 Milan, Italy, 27-30.07. 15*.

[38] Short, A. R., Mimlitz, and Van Bossuyt, 2016, "Autonomous System Design and Controls Design for Operations in High Risk Environments," *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

[39] Kurtoglu, T., and Tumer, I. Y., 2007, "Ffip: A Framework for Early Assessment of Functional Failures in Complex Systems," *The International Conference on Engineering Design, ICED*.

[40] Tumer, I. Y., Stone, R. B., Bell, D. G., and others, 2003, "Requirements for a Failure Mode Taxonomy for Use in Conceptual Design," *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*.

[41] Hutcheson, R. S., McAdams, D. A., Stone, R. B., and Tumer, I. Y., 2015, "A FUNCTION-BASED METHODOLOGY FOR ANALYZING CRITICAL EVENTS."

[42] Van Bossuyt, D. L., and O'Halloran, B., 2015, "Modeling of Function Failure Propagation Across Uncoupled Systems."

[43] Short, A. R., Van Bossuyt, D. L., Hodge, Robert, and DuPont, Bryony, "ACTIVE MISSION SUCCESS ESTIMATION THROUGH FUNCTIONAL MODELING," Rev. Res. Eng. Des.

[44] Mohr, R. R., 2002, "Failure Modes and Effects Analysis," JE Jacobs Sverdrup.

[45] Distefano, S., and Puliafito, A., 2007, "Dynamic Reliability Block Diagrams: Overview of a Methodology," *ESREL*, pp. 1059–1068.

[46] Kumamoto, H., and Henley, E. J., 1996, *Probabilistic Risk Assessment and Management for Engineers and Scientists*, Institute of Electrical & Electronics Engineers (IEEE Press).

[47] Kurtoglu, T., and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," J. Mech. Des., **130**(5), p. 051401.

[48] Jensen, D. C., Tumer, I. Y., and Kurtoglu, T., 2009, "Flow State Logic (FSL) for Analysis of Failure Propagation in Early Design," *ASME 2009 International*

Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1033–1043.

[49] Short, A. R., and Van Bossuyt, D. L., 2015, "Risk Attitude Informed Route Planning in a Simulated Planetary Rover," *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, p. V01BT02A048–V01BT02A048.

[50] Mimlitz, Z., Short, A. R., and Van Bossuyt, D. L., 2016, "Towards Risk-Informed Operation of Autonomous Vehicles to Increase Resilience in Unknown and Dangerous Environments," *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

[51] Friedenthal, S., Moore, A., and Steiner, R., 2014, *A Practical Guide to SysML: The Systems Modeling Language*, Morgan Kaufmann.

[52] Mohaghegh, Z., Kazemi, R., and Mosleh, A., 2009, "Incorporating Organizational Factors into Probabilistic Risk Assessment (PRA) of Complex Socio-Technical Systems: A Hybrid Technique Formalization," Reliab. Eng. Syst. Saf., **94**(5), pp. 1000–1018.

[53] Wertz, J. R., Everett, D. F., and Puschell, J. J., 2011, "Risk and Reliability," *Space Mission Engineering: The New SMAD*, Microcosm Press.

[54] Douglas L Van Bossuyt, A. D., 2013, "On Measuring Engineering Risk Attitudes," J. Mech. Des., **135**(12).

[55] Douglas Van Bossuyt, C. H., 2012, "Risk Attitudes in Risk-Based Design: Considering Risk Attitude Using Utility Theory in Risk-Based Design," Artif. Intell. Eng. Des. Anal. Manuf., **26**(04).

[56] Pratt, J. W., 1964, "Risk Aversion in the Small and in the Large," Econometrica, **32**(1/2), pp. 122–136.

[57] Beck, J., 2008, *Combinatorial Games: Tic-Tac-Toe Theory*, Cambridge University Press.

[58] O'Neill, B., 1994, "Game Theory Models of Peace and War," Handb. Game Theory Econ. Appl., **2**, pp. 995–1053.

[59] Tadelis, S., 2013, *Game Theory: An Introduction*, Princeton University Press.

[60] Fudenberg, D., and Tirole, J., 1991, "Game Theory, 1991," Camb. Mass., **393**(12), p. 80.

[61] Nash, J., 1953, "Two-Person Cooperative Games," Econom. J. Econom. Soc., pp. 128–140.

[62] Nash, J. F., 1950, "Equilibrium Points in N-Person Games," Proc. Natl. Acad. Sci., **36**(1), pp. 48–49.

[63] Nash, J., 1951, "Non-Cooperative Games," Ann. Math., pp. 286–295.

[64] Gibson, R. G., and Szafron, D., 2011, "On Strategy Stitching in Large Extensive Form Multiplayer Games," *Advances in Neural Information Processing Systems*, pp. 100–108.

[65] Chaslot, G., Bakkes, S., Szita, I., and Spronck, P., 2008, "Monte-Carlo Tree Search: A New Framework for Game AI.," *AIIDE*.

[66] Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S., 2012, "A Survey of Monte Carlo Tree Search Methods," IEEE Trans. Comput. Intell. AI Games, **4**(1), pp. 1–43.

[67] Dorigo, M., and Di Caro, G., 1999, "Ant Colony Optimization: A New Meta-Heuristic," *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress On*, IEEE, pp. 1470–1477.

[68] Tarjan, R., 1972, "Depth-First Search and Linear Graph Algorithms," SIAM J. Comput., **1**(2), pp. 146–160.

[69] Bundy, A., and Wallen, L., 1984, "Breadth-First Search," *Catalogue of Artificial Intelligence Tools*, Springer, pp. 13–13.

[70] Dechter, R., and Pearl, J., 1985, "Generalized Best-First Search Strategies and the Optimality of A," J. ACM JACM, **32**(3), pp. 505–536.

[71] Haouari, M., and Chaouachi, J. S., 2002, "A Probabilistic Greedy Search Algorithm for Combinatorial Optimisation with Application to the Set Covering Problem," J. Oper. Res. Soc., **53**(7), pp. 792–799.

[72] Guizzo, E., 2011, "How Google's Self-Driving Car Works," IEEE Spectr. Online, **18**(7), pp. 1132–1141.

[73] Bregu, E., Casamassima, N., Cantoni, D., Mottola, L., and Whitehouse, K., 2016, "Reactive Control of Autonomous Drones," *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, ACM, pp. 207–219.

[74] Gill, K., Yang, S.-H., Yao, F., and Lu, X., 2009, "A Zigbee-Based Home Automation System," IEEE Trans. Consum. Electron., **55**(2).

[75] Park, M. H., 2006, "Personal Digital Assistant."

[76] Walter, W. G., 1951, "A Machine That Learns," Sci. Am., **185**(2), pp. 60–63.

[77] Walter, W. G., "An Imitation of Life!"

[78] Bhattacharya, S., and Agrawal, S. K., 2015, "Design, Experiments and Motion Planning of a Spherical Rolling Robot."

[79] Bruce, J., Sabelhaus, A., Chen, Y., Lu, D., Morse, K., Milam, S., Caluwaerts, K., Agogino, A., and SunSpiral, V., 2014, "SUPERball: Exploring Tensegrities for Planetary Probes," *12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.

[80] Ducatelle, F., Di Caro, G. A., and Gambardella, L. M., 2010, "Cooperative Self-Organization in a Heterogeneous Swarm Robotic System."

[81] Lin, H.-T., Leisk, G. G., and Trimmer, B., 2011, "GoQBot: A Caterpillar-Inspired Soft-Bodied Rolling Robot," Bioinspir. Biomim., **6**(2), p. 026007.

[82] Squyres, S. W., Knoll, A. H., Arvidson, R. E., Clark, B. C., Grotzinger, J. P., Jolliff, B. L., McLennan, S. M., Tosca, N., Bell, J. F., Calvin, W. M., and others, 2006, "Two Years at Meridiani Planum: Results from the Opportunity Rover," Science, **313**(5792), pp. 1403–1407.

[83] Kornfeld, R. P., Prakash, R., Devereaux, A. S., Greco, M. E., Harmon, C. C., and Kipp, D. M., 2014, "Verification and Validation of the Mars Science Laboratory/Curiosity Rover Entry, Descent, and Landing System," J. Spacecr. Rockets, pp. 1–19.

[84] Yoshida, K., and Hamano, H., 2002, "Motion Dynamics of a Rover with Slip-Based Traction Model," *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference On*, IEEE, pp. 3155–3160.

[85] Lakdawalla, E., 2015, "Curiosity Wheel Damage: The Problem and Solutions" [Online]. Available: http://www.planetary.org/blogs/emily-lakdawalla/2014/08190630-curiosity-wheel-damage.html. [Accessed: 04-May-2015].

[86] Hart, P. E., Nilsson, N. J., and Raphael, B., 1968, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," Syst. Sci. Cybern. IEEE Trans. On, **4**(2), pp. 100–107.

[87] Szczerba, R. J., Galkowski, P., Glicktein, I. S., and Ternullo, N., 2000, "Robust Algorithm for Real-Time Route Planning," Aerosp. Electron. Syst. IEEE Trans. On, **36**(3), pp. 869–878.

[88] Delling, D., Sanders, P., Schultes, D., and Wagner, D., 2009, "Engineering Route Planning Algorithms," *Algorithmics of Large and Complex Networks*, Springer, pp. 117–139.

[89] Doi, R. M., Ghaem, S., and Kirson, A. M., 1992, *Vehicle Route Planning System*, Google Patents.

[90] Gazis, D. C., Jaffe, R. S., and Pope, W. G., 1997, *Optimal and Stable Route Planning System*, Google Patents.

[91] Estlin, T., Gaines, D., Chouinard, C., Castano, R., Bornstein, B., Judd, M., Nesnas, I., and Anderson, R., "Increased Mars Rover Autonomy Using AI Planning, Scheduling and Execution."

[92] Castillo, O., Trujillo, L., and Melin, P., 2007, "Multiple Objective Genetic Algorithms for Path-Planning Optimization in Autonomous Mobile Robots," Soft Comput., **11**(3), pp. 269–279.

[93] Rardin, R. L., 1998, *Optimization in Operations Research*, Prentice Hall New Jersey.

[94] Sweet, A., Gorospe, G., Daigle, M., Celaya, J. R., Balaban, E., Roychoudhury, I., and Narasimhan, S., 2014, "Demonstration of Prognostics-Enabled Decision Making Algorithms on a Hardware Mobile Robot Test Platform."

[95] Byington, C. S., Watson, M., Edwards, D., and Stoelting, P., 2004, "A Model-Based Approach to Prognostics and Health Management for Flight Control Actuators," *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, IEEE, pp. 3551–3562.

[96] 2015, "MATLAB - The Language of Technical Computing" [Online]. Available: http://www.mathworks.com/products/matlab/. [Accessed: 09-Dec-2015].

[97] "Welcome to Python.Org," Python.org [Online]. Available: https://www.python.org/. [Accessed: 07-Apr-2018].

[98] "Java Software | Oracle" [Online]. Available: https://www.oracle.com/java/index.html. [Accessed: 02-Mar-2017].

[99] BillWagner, "C# Programming Guide" [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/. [Accessed: 07-Apr-2018].

[100] Bellmore, M., and Nemhauser, G. L., 1968, "The Traveling Salesman Problem: A Survey," Oper. Res., **16**(3), pp. 538–558.
[101] Applegate, D. L., 2006, *The Traveling Salesman Problem: A Computational Study*, Princeton university press.
[102] Lewis, H. R., 1983, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, JSTOR.
[103] 2000, "ASTM Standard D2487 'Standard Practice for Classification of Soils for Engineering Purposes (Unified Soil Classification System).'"
[104] "Mars Exploration Rover - Opportunity" [Online]. Available: http://www.jpl.nasa.gov/missions/mars-exploration-rover-opportunity-mer/. [Accessed: 08-Apr-2018].
[105] Dorigo, M., and Gambardella, L. M., 1997, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Trans. Evol. Comput., **1**(1), pp. 53–66.
[106] Horacek, H., and Kaindl, H., 1989, "An Outline of a New Algorithm for Game Tree Search," *5. Österreichische Artificial-Intelligence-Tagung*, Springer, pp. 171–176.
[107] Tero, A., Kobayashi, R., and Nakagaki, T., 2007, "A Mathematical Model for Adaptive Transport Network in Path Finding by True Slime Mold," J. Theor. Biol., **244**(4), pp. 553–564.
[108] Adamatzky, A., 2012, "Slime Mold Solves Maze in One Pass, Assisted by Gradient of Chemo-Attractants," IEEE Trans. Nanobioscience, **11**(2), pp. 131–134.
[109] Miller, B. L., 1967, *Finite State Continuous Time Markov Decision Processes with a Finite Planning Horizon.*, RAND CORP SANTA MONICA CALIF.
[110] McDermott, D., and Davis, E., 1984, "Planning Routes through Uncertain Territory," Artif. Intell., **22**(2), pp. 107–156.
[111] NASA, J., "Possible MSL Landing Site: Mawrth Vallis - Mars Science Laboratory" [Online]. Available: https://mars.nasa.gov/msl/mission/timeline/prelaunch/landingsiteselection/mawrthvallis2/. [Accessed: 07-Apr-2018].
[112] "Search for Superfund Sites Where You Live | Superfund | US EPA" [Online]. Available: https://www.epa.gov/superfund/search-superfund-sites-where-you-live. [Accessed: 07-Apr-2018].
[113] Miller, G. S., 1986, "The Definition and Rendering of Terrain Maps," *ACM SIGGRAPH Computer Graphics*, ACM, pp. 39–48.
[114] 2018, "Diamond-Square Algorithm," Wikipedia.
[115] "Lognormal Random Numbers - MATLAB Lognrnd" [Online]. Available: https://www.mathworks.com/help/stats/lognrnd.html. [Accessed: 08-Apr-2018].
[116] Sullivan, R., Anderson, R., Biesiadecki, J., Bond, T., and Stewart, H., 2011, "Cohesions, Friction Angles, and Other Physical Properties of Martian Regolith from Mars Exploration Rover Wheel Trenches and Wheel Scuffs," J. Geophys. Res. Planets, **116**(E2).

[117] Vangla, P., and Latha, G. M., 2015, "Influence of Particle Size on the Friction and Interfacial Shear Strength of Sands of Similar Morphology," Int. J. Geosynth. Ground Eng., **1**(1), p. 6.
[118] Atman, A. P. F., Claudin, P., Combe, G., and Martins, G. H. B., 2014, "Mechanical Properties of Inclined Frictional Granular Layers," Granul. Matter, **16**(2), pp. 193–201.
[119] 2017, *SPEARS_Simulator: The Simulated Physics and Environment for Autonomous Risk Studies*, Van Bossuyt Research Lab.
[120] Mimlitz, Short, A. R., and Van Bossuyt, D. L., 2016, "Towards Risk-Informed Operation of Autonomous Vehicles to Increase Resilience in Unknown and Dangerous Environments," *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.
[121] Manion, C. A., Arlitt, R., Tumer, I., Campbell, M. I., and Greaney, P. A., 2015, "Towards Automated Design of Mechanically Functional Molecules," *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, p. V02AT03A004–V02AT03A004.
[122] Patterson, W. R. J., and Campbell, M. I., 2011, "PipeSynth: An Algorithm for Automated Topological and Parametric Design and Optimization of Pipe Networks," *ASME Conference Proceedings*, ASME, pp. 13–23.
[123] Bryant, C. R., McAdams, D. A., Stone, R. B., Kurtoglu, T., and Campbell, M. I., 2006, "A Validation Study of an Automated Concept Generator Design Tool," *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 283–294.
[124] Campbell, M. I., Cagan, J., and Kotovsky, K., 2003, "The A-Design Approach to Managing Automated Design Synthesis," Res. Eng. Des., **14**(1), pp. 12–24.
[125] Dyck, D. N., and Lowther, D. A., 1996, "Automated Design of Magnetic Devices by Optimizing Material Distribution," IEEE Trans. Magn., **32**(3), pp. 1188–1193.
[126] Manion, C., Arlitt, R., Campbell, M. I., Tumer, I., Stone, R., and Greaney, P. A., 2016, "Automated Design of Flexible Linkers," Dalton Trans., **45**(10), pp. 4338–4345.
[127] Vanderplaats, G. N., and Moses, F., 1972, "Automated Design of Trusses for Optimum Geometry," J. Struct. Div., **98**(3), pp. 671–690.
[128] Patel, J., and Campbell, M. I., 2008, "An Approach to Automate Concept Generation of Sheet Metal Parts Based on Manufacturing Operations," *Volume 1: 34th Design Automation Conference, Parts A and B*, ASME, Brooklyn, New York, USA, pp. 133–142.
[129] Patel, J., and Campbell, M. I., 2008, "Topological and Parametric Tune and Prune Synthesis of Sheet Metal Parts Compared to Genetic Algorithm," *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*.
[130] Swantner, A., and Campbell, M. I., 2012, "Topological and Parametric Optimization of Gear Trains," Eng. Optim., **in review**, pp. 1–18.

[131] Radhakrishnan, P., and Campbell, M. I., 2010, "A Graph Grammar Based Scheme for Generating and Evaluating Planar Mechanisms," *Design Computing and Cognition '10*, J.S. Gero, ed., Springer Netherlands, Stuttgart, pp. 663–679.
[132] Hooshmand, A., and Campbell, M. I., 2016, "Truss Layout Design and Optimization Using a Generative Synthesis Approach," Comput. Struct., **163**, pp. 1–28.
[133] Koning, H., and Eizenberg, J., 1981, "The Language of the Prairie: Frank Lloyd Wright's Prairie Houses," Environ. Plan. B Plan. Des., **8**(3), pp. 295–323.
[134] Short, A. R., and Van Bossuyt, Douglas, "Failure Flow Decision Functions for Iterative Design," Prep. J. Eng. Des.
[135] Whitley, D., 1994, "A Genetic Algorithm Tutorial," Stat. Comput., **4**(2), pp. 65–85.
[136] Feldmann, R., 1993, "Game Tree Search," PhD Thesis, University of Paderborn.
[137] Hart, P. E., Nilsson, N. J., and Raphael, B., 1968, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," Syst. Sci. Cybern. IEEE Trans. On, **4**(2), pp. 100–107.
[138] Šišlák, D., Volf, P., and Pěchouček, M., 2009, "Accelerated A* Path Planning," *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1133–1134.
[139] Castillo, O., Trujillo, L., and Melin, P., 2007, "Multiple Objective Genetic Algorithms for Path-Planning Optimization in Autonomous Mobile Robots," Soft Comput., **11**(3), pp. 269–279.
[140] Chaslot, G., 2010, "Monte-Carlo Tree Search," Maastricht Univ. Maastricht.
[141] Simon, H. A., and Newell, A., 1958, "Heuristic Problem Solving: The next Advance in Operations Research," Oper. Res., **6**(1), pp. 1–10.
[142] Bresina, J. L., 1996, "Heuristic-Biased Stochastic Sampling," *AAAI/IAAI, Vol. 1*, pp. 271–278.
[143] "Graph with Directed Edges - MATLAB" [Online]. Available: https://www.mathworks.com/help/matlab/ref/digraph.html. [Accessed: 15-Dec-2017].
[144] Eiben, A. E., and Smit, S. K., 2011, "Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms," Swarm Evol. Comput., **1**(1), pp. 19–31.
[145] Pinel, F., Danoy, G., and Bouvry, P., 2012, "Evolutionary Algorithm Parameter Tuning with Sensitivity Analysis," *Security and Intelligent Information Systems*, Springer, pp. 204–216.
[146] Koza, J. R., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT press.
[147] Coello, C. A. C., Lamont, G. B., and Van Veldhuizen, D. A., 2007, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer.
[148] Dorigo, M., Birattari, M., and Stutzle, T., 2006, "Ant Colony Optimization," IEEE Comput. Intell. Mag., **1**(4), pp. 28–39.
[149] "Intel® Xeon® Processor E3-1240 v2 (8M Cache, 3.40 GHz) Product Specifications," Intel® ARK Prod. Specs [Online]. Available:

https://ark.intel.com/products/65730/Intel-Xeon-Processor-E3-1240-v2-8M-Cache-3_40-GHz. [Accessed: 16-Dec-2017].

[150] Short, A.-R., Hodge, R. D., Van Bossuyt, D. L., and DuPont, B., 2018, "Active Mission Success Estimation through Functional Modeling," Res. Eng. Des., pp. 1–24.

[151] Kahneman, D., 2011, *Thinking, Fast and Slow*, Macmillan.

[152] Arnhof, M., 2016, "Design of a Human Settlement on Mars Using In-Situ Resources," 46th International Conference on Environmental Systems.

[153] Karumanchi, S., Edelberg, K., Baldwin, I., Nash, J., Reid, J., Bergh, C., Leichty, J., Carpenter, K., Shekels, M., Gildner, M., and others, 2017, "Team RoboSimian: Semi-Autonomous Mobile Manipulation at the 2015 DARPA Robotics Challenge Finals," J. Field Robot., **34**(2), pp. 305–332.

[154] Taleb, N. N., 2007, *The Black Swan: The Impact of the Highly Improbable*, Random house.

[155] Allen, C. C., and Zubrin, R., 1999, "In-Situ Resources," Hum. Space Flight Mission Anal. Des. N. Y. NY McGraw-Hill Space Technol. Ser.

[156] Maegley, W. J., and Diederich, D. P., 1975, "Martian Sandstorms and Their Effects on the 1975 Viking Lander System," J. Test. Eval., **3**(5), pp. 380–388.

[157] Broom, D. M., 2001, "Evolution of Pain," Vlaams Diergeneeskd. Tijdschr., **70**(1), pp. 17–21.

[158] Gehring, W. J., 2014, "The Evolution of Vision," Wiley Interdiscip. Rev. Dev. Biol., **3**(1), pp. 1–40.

[159] Pinto, C. A., and Garvey, P. R., 2012, *Advanced Risk Analysis in Engineering Enterprise Systems*, CRC Press.

[160] Lucero, B., Viswanathan, V. K., Linsey, J. S., and Turner, C. J., 2014, "Identifying Critical Functions for Use across Engineering Design Domains," J. Mech. Des., **136**(12), p. 121101.

[161] Upadhyay, S. K., 2010, "Common Failure Distributions," Wiley Encycl. Oper. Res. Manag. Sci.

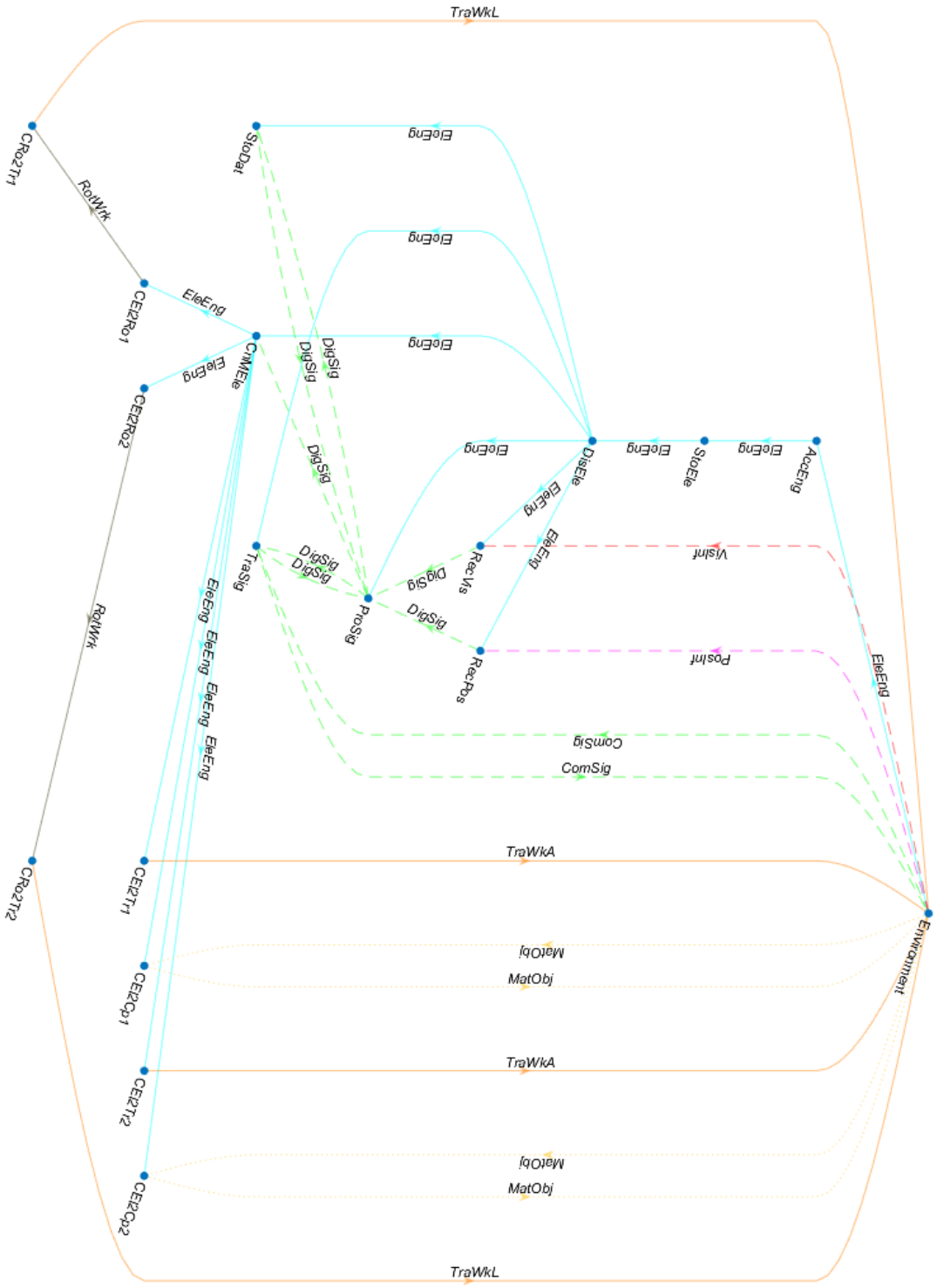
[162] Jensen, D. C., Tumer, I. Y., and Kurtoglu, T., 2015, "FLOW STATE LOGIC (FSL) FOR ANALYSIS OF FAILURE PROPAGATION IN EARLY DESIGN."

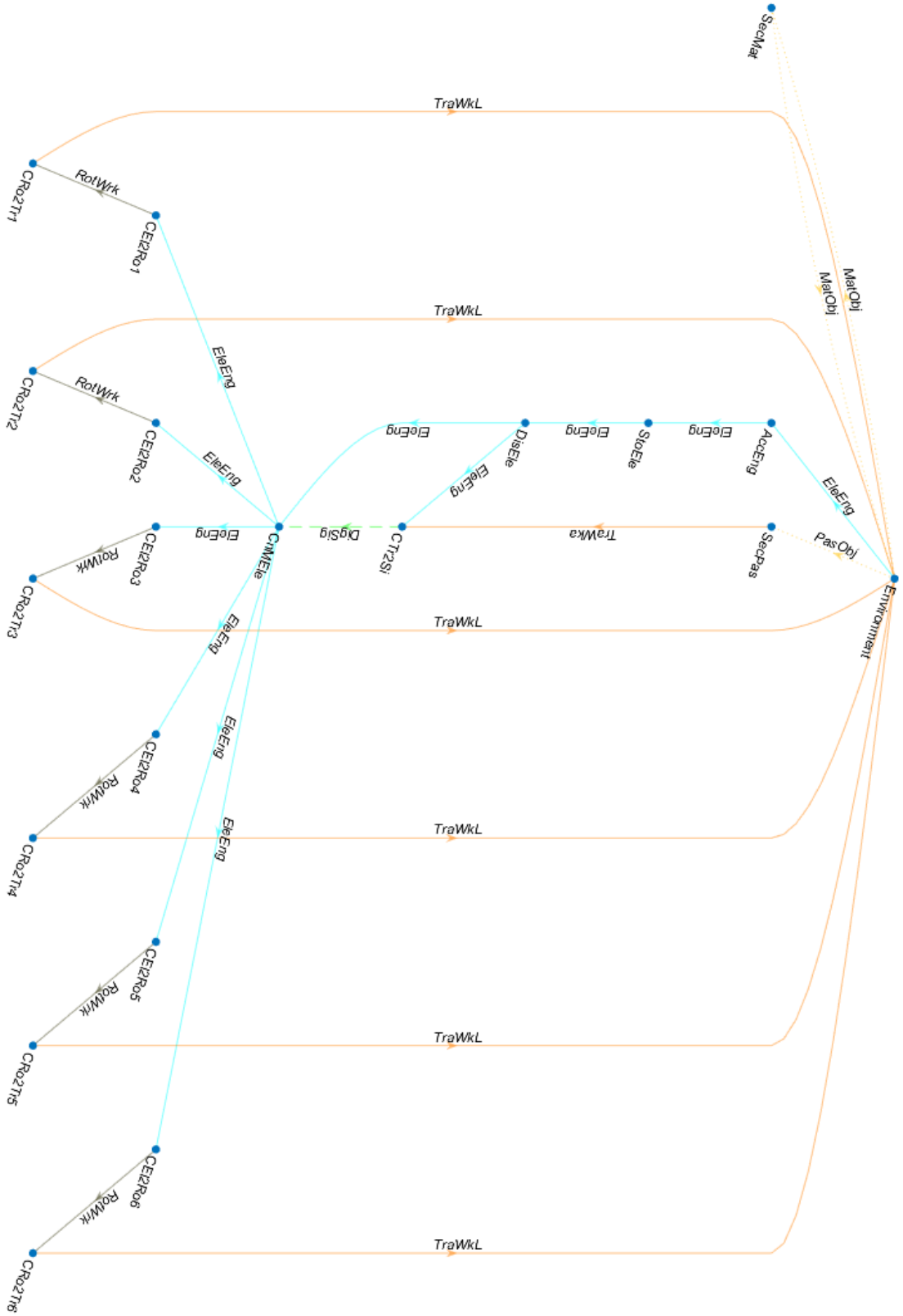
[163] Goldratt, E. M., and Cox, J., 2016, *The Goal: A Process of Ongoing Improvement*, Routledge.

[164] Sheppard, J. W., Kaufman, M. A., and Wilmering, T. J., 2014, "IEEE Standards for Prognostics and Health Management."

APPENDICES

APPENDIX A: DIGRAPH BASED FUNCTIONAL MODELS




SecMat
TraWkL
CRa2T1
RotWrk
CEE2Ro1
MatObj
MatObj
TraWkL
CRa2T2
RotWrk
CEE2Ro2
EleEng
EleEng
EloEng
DisEle
EloEng
StoEle
EloEng
AcqEng
EleEng
CEE2Ro3
RotWrk
CRa2T3
EloEng
CnMEle
DgSig
CTr2Si
TraWka
SecPas
PasObj
Environment
TraWkL
EleEng
CEE2Ro4
RotWrk
CRa2T4
EleEng
TraWkL
CEE2Ro5
RotWrk
CRa2T5
TraWkL
CEE2Ro6
RotWrk
CRa2T6
TraWkL


ConNuc
ThrEng
CTh2El
EleEng
AccEng
EleEng
StoEle
EleEng
DisEle
EleEng
Environment