Tradespace Exploration for Space System Architectures: A Weighted Graph Framework

by

Peter Leslie Davison

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014

Author .....
Department of Aeronautics and Astronautics
May 22,2014

Certified by .....
Prof. Edward F. Crawley
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Prof. Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Tradespace Exploration for Space System Architectures: A Weighted Graph Framework

by

Peter Leslie Davison

Submitted to the Department of Aeronautics and Astronautics
on May 22,2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Many systems undergo significant architecture-level change during their lifecycles as a result of exogenous system disturbances (e.g. budget reduction or changes in stakeholder requirements), failure to develop critical technologies, or planned evolution of the system over time. Given the high cost in terms of resources, schedule, and performance of making these changes during system development or operations, it is essential to make these decisions with a thorough understanding of the available options and costs associated with different architecture changes.

The analysis of such decisions stems from an understanding of the relationships between architectures in the tradespace, however most architecture tradespace generation tools focus on the modeling and evaluation of individual architectures rather than on modeling how different architectures are related to one another. In this thesis we propose a framework for modeling these relationships based on the evaluation of a pre-existing tradespace for the purpose of analyzing architecture change decisions.

This modeling framework is used to discover and evaluate evolutionary pathways through a disorganized tradespace: a process we call tradespace exploration. These pathways can be used to assess architecture selection decisions and to quantitatively compare architecture change decisions against one another, providing a decision analysis tool for system architects. At the core of the framework is the generation of a directed, weighted 'tradespace graph' that serves as a model of the architecture decision making process.

Vertices in the tradespace graph are defined by pairings of architectures from the tradespace with asset portfolios, which are the sets of the common elements shared between multiple architectures. The existence of an edge in the graph, which represents a feasible decision to transform from one architecture to another, is determined by the relationship between the asset portfolios of the two vertices. The weight of an edge represents the cost of the corresponding architecture change, with the sum of edge weights along a path through the tradespace representing the total development cost of the architecture evolution.

We apply this tradespace exploration framework to two pre-existing architecture

tradespaces: the first being the 'HEXANE' tradespace of in-space transportation infrastructures for human exploration beyond low Earth orbit, and the second being the 'SCaN' tradespace of space-based communications network architectures for the relay of data between ground stations and user spacecraft. Using a variety of domain-independent analysis tools and graph search algorithms, we generate several results of potential value to system architects for both applications.

Thesis Supervisor: Prof. Edward F. Crawley
Title: Professor of Aeronautics and Astronautics

Acknowledgments

Like almost everything else I have done in my life, this thesis and my time at MIT would not have been possible without the help and support of many people.

Since they have had to deal with me five days a week for the last two years, I think it is best to start with my lab-mates who have helped make 33-409 our own little slice of paradise. Morgan, thank you for always being a beacon of optimism and for teaching me that research crises are normal. Iñigo, although you joined the team late, it has been a pleasure working with you - I look forward to seeing you carry the tradespace exploration torch when I am gone. Dani, you deserve your own acknowledgments section in everyone's thesis that goes through this lab - you have been an amazing colleague and mentor, and an even better friend. Marc, from Russia to basketball to Zac Brown it has been a lot of fun with some learning thrown in - thanks for all of your help over the last two years and sorry I still can't speak Spanish.

To my advisor, Prof. Ed Crawley, although I wish we had the chance to work more extensively together, I appreciate all of the opportunities you have provided me during my time here and the freedom you have given me to pursue the problems that I find interesting.

Bruce, you have played a huge part in getting this research off the ground and in helping me to structure my ideas and analysis in ways that are interesting and relevant to the people that matter. Thank you for your role in my research, for your mentoring over the course of the last two years, and for being a great friend in the process.

To former and honorary system architects: Alexander, Jonathan, Emily, and Alessandro - you guys are awesome - thanks for your feedback on my work while we overlapped. And of course thank you Ana, a system architect in spirit if not by title, for letting the lab routinely crash your quiet study time, watch basketball on your TV, and eat all of your food.

To my MIT friends and fellow intramural superstars: Bryan, Mark B., Mark C., Chris, Torin, Pratik, and Raquel - thanks for helping make my time here so much

fun, and remember that if this whole ‘aerospace’ thing doesn’t work out, we’ve always got pro sports to fall back on.

Finally, thank you to my family for always being there to support me and believing that I could do it even when they didn’t know what it was I wanted to do. To my ‘extended’ family, especially my grandparents, I feel incredibly lucky to have your support and encouragement in all that I do, even if you don’t know or care what a tradespace is. To my brother and sister, I couldn’t do it without you guys, thanks for helping me to always keep things in perspective. And to my parents, I owe everything to you guys for all that you have done and sacrificed for me over the years - thank you for encouraging me to set the bar high and teaching me how to work hard to get over it.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Defining the design of a system is a decision making process that is centered on the evaluation of both quantitative and qualitative system characteristics. Many of these - performance, recurring and non-recurring cost, and schedule - are characteristics of a single design that can be assessed by analyzing each design in isolation. However, many other system characteristics - system flexibility, robustness to external change, and evolvability - are by their nature concerned with relationships between two or more designs, and consequently the assessment of their impact on design decisions requires a different set of tools that looks beyond the analysis of single point designs.

The term 'Tradespace Exploration' has been used in a variety of contexts to describe both sets of tools and the related models for the enumeration, evaluation, and analysis of a tradespace of design options. 'Tradespace' itself is an ambiguous term that has taken on a number of meanings as its use has evolved over time, however most generally it is taken to mean a collection of objects or concepts that can be compared to, or traded against, one another. In this thesis and the references cited within a tradespace is a set of candidate high-level designs for a technical system. We call this high-level design the system architecture.

1.1 Motivation

The importance of 'exploring' a system architecture tradespace stems from the strong influence that decisions regarding the system architecture have on the performance and lifecycle cost of the realized system. It is well documented that system architecture decisions, which are made very early in the design lifecycle, commit the majority of the system cost and thus are disproportionately expensive to change later in the lifecycle [25, 35, 59]. This simple observation forms the central assumption of this thesis and will serve to motivate the tradespace exploration framework we develop and apply in the following chapters.

The fundamental assertion that the system architecture is costly to change once it has been set leads decision makers to place a great deal of emphasis on the system architecting phase of the design lifecycle. A significant amount of research has been conducted in the area of system architecture modeling and decision analysis to ensure that the full tradespace of options is explored and assessed (see Section 1.4 for relevant literature).

Taken on its own this type of decision analysis would lead the system architect to select the highest performing architecture that fits within the cost and schedule constraints imposed by the project budget and time line. In other words, the decision process is to find the Pareto optimal set of architectures in the tradespace and then pick the highest performing one that fits within project constraints. However, this type of analysis ignores one essential fact of system lifecycle design, which is that in some cases changes to the architecture are required after the initial selection has been made.

The cause of changes to the system architecture after initial selection falls into two categories: planned changes to 'evolve' the system over time in order to improve the system performance, reduce recurring costs, or introduce new system features; and unplanned changes induced by exogenous impacts such as changes to the budget, shifting stakeholder needs, or failure to mature critical technologies during development. Both types of changes are common in complex systems and can cause major

disruptions to system development and operation if the initial architecture selection is not made intelligently [38].

1.2 Thesis Objective

General Objective

The combination of the fact that in many cases the system architecture must be changed after it has been defined with the fact that architecture changes are costly to make forms the fundamental motivation for this thesis. We wish to define a framework in which these types of changes can be modeled, and from which information regarding architecture selection and architecture change decisions can be extracted to aid the system architect.

The development of such a framework will require a model of the relationships between architectures so that the impact of architecture change on system cost and performance can be reliably assessed. Such a model must be general enough to be applied to a variety of different architecture tradespaces, which may span many diverse types of systems, yet it must not be so abstract as to require a unique set of modeling assumptions and analysis tools for each type of tradespace. In other words we wish to develop a modeling framework for architecture change and the related set of analysis tools that can be applied to a variety of system architecture tradespaces with only minor model extensions and specializations required for each application.

Specific Objective

Our starting point will be a fully enumerated and evaluated tradespace of architectures that contains the feasible options for the architecture of the system and a simple set of metrics on which to compare them. With the modeling of the architecture itself defined in this preexisting tradespaces, the primary modeling effort in this thesis will be focused on understanding how to model the cost of changing an architecture and the process by which architectural change occurs. Given a variety of possible scenarios we wish to analyze these relationships to provide the decision maker with

possible options for: i) maintaining flexibility in the architecture in the concept selection phase of the project; ii) mitigating the cost of changing the architecture when one or more exogenous factors makes the current architecture infeasible or sub-optimal; or iii) developing new, planned iterations of the system over time to improve system performance or reduce recurring cost.

The approach we have chosen to analyze this tradespace exploration problem is to transform the tradespace of individual architectures into a weighted directed graph that models the potential decision making options available to the system architect. The tradespace graph provides a simple but powerful tool to encode feasible architecture change decisions and to transform ambiguous tradespace exploration problems into rigorous, quantitative ones. Namely, we will enumerate several well-defined tradespace exploration problems on the graph that can be analyzed using powerful domain-independent methods and algorithms developed over the last several decades in the field of graph theory, most notably Dijkstra's algorithm for solving the shortest-path problem.

1.3 Thesis Overview

The remainder of this chapter will be devoted to a review of the relevant literature and background as it relates to system architecture modeling, tradespace exploration, and several of the computational tools implemented in this thesis. In Chapter 2 we will introduce the generic tradespace graph framework that we have developed to analyze the tradespace exploration problem. In this chapter we also clarify the language and notation used throughout the thesis and define the tradespace exploration problem statements used as a starting point for our analysis.

Chapters 3, 4, and 5 apply this generic framework to tradespaces of in-space transportation infrastructures for human exploration and in-space communications networks. Chapter 3 studies the first of these tradespaces, while Chapters 4 and 5 both focus on the second. These two tradespaces provide interesting case studies in terms of the value of results as they both explore systems with a number of major

architecture decisions yet to be made that will have significant influence on the future direction of their respective application fields. At the same time, together this trio of case studies offers a compelling test of our framework as the two tradespaces studied are vastly different when it comes to basic functions (human transportation versus communications), requirements (ambiguous versus well-defined), performance metrics (abstract versus concrete), and basic architectural model (procedural-based versus rule-based).

Chapter 3 begins with a review of the basics of the HEXANE tradespace enumeration tool and the underlying models and assumptions that define an architecture in the tradespace. Next the framework developed in the previous chapter is applied to this tradespace, with a number of application-specific assumptions and problem statements added to the generic tradespace graph model. In the final section of this chapter several tradespace exploration scenarios are described and analyzed using the HEXANE tradespace graph.

Chapter 4 follows a similar pattern with an explanation of the SCaN tradespace model followed by the development of the application-specific tradespace graph. In this chapter a time-independent tradespace graph, which we call the *static* graph, is developed and presented. The second half of this chapter covers the presentation and analysis of several sets of tradespace exploration results using this static tradespace graph.

In Chapter 5 the same SCaN tradespace model is used to generate the tradespace graph, so this chapter skips to the development of the graph itself. The model of the decision making process used to generate this graph is time-dependent, so we call this application of tradespace graph concept the *time-expanded* graph. This time-expanded graph allows the aging of in-space assets with a limited lifetime to be modeled across steps in time, which supplements the analysis provided by the static graph in Chapter 4. Following the explanation of the modeling assumptions and development of the rules that generate this graph, a number of decision making scenarios are presented and analyzed using this framework.

Finally, in Chapter 6 we present our conclusions from the research presented in

this thesis, and discuss a number of directions for future research in this and other related areas of work.

1.4 Literature Review and Background

1.4.1 System Architecture Models

At the core of our framework is the use of a system architecture model as a representation of the system being analyzed. A variety of modeling approaches are used in system architecture analysis depending on the level of fidelity required and the type of analysis required. In [56] Selva groups system architecting problems into five broad classes: down-selecting, set partitioning, permuting, connecting and assigning problems. The class of problem determines the characteristics that differentiate architectures from one another and informs the type of model that can be used to encode an architecture. Since the concept of architecture is by nature an abstract one, there exists a number of general models that can be applied in a wide variety of domains.

The design structure matrix (DSM) [10] is a flexible tool used to model the relationships between different system elements and system functions using a matrix representation - see for example [21] for organizational architecture optimization. The DSM is a simple and powerful tool, but it lacks the ability to efficiently model constraints and is not an effective way to view complex systems with many elements. Object Process Methodology (OPM) [18] is a graphical modeling language that describes the relationships between system entities and functions, and if organized correctly can provide an elegant view of a complex system . OPM is flexible in that it can represent a system architecture from a variety of perspectives such as operational sequence [31] or functional decomposition [49], however it is not an effective way to enumerate a large tradespace of architectures since system constraints are difficult to encode automatically.

Other specialized architecture models have been developed to aid in the rapid enumeration of architectures across an entire tradespace. Architectures modeled as

sets of decisions assert that each aspect of the architecture can be traced to a decision among multiple options, with each decision narrowing the design space until a complete architecture is reached. Decision trees [13] are a general tool that explicitly represent each distinct architecture as a 'leaf' on the tree, which is reached by making a decision at each sequential branch in the tree. As such they are graphically intuitive but increase exponentially in size with the number of decisions and cannot explicitly model non-sequential decisions. Simmons proposes a more scalable method, the architecture decision graph [60], which represents the tradespace of options as a bipartite graph with decisions as nodes. Architectures are represented by pathways through the decision graph and are enumerated using a constraint satisfaction algorithm.

Other methods also use a path through a network or graph to model an architecture. Koo, Simmons and Crawley present Object Process Network in [31], which connects high level system function or operation to changes in state. The path taken by different system elements (represented by tokens) through the network determines the architecture. Arney and Wilhite [2, 3] develop a tradespace network based on graph theory for in-space transportation in which nodes represent steady states of system operations and edges encode the cost or requirements or moving systems from state to state. An architecture is determined by the path that each constituent system takes through the network from initial to final state.

Simple mathematical structures can also be used to represent an architecture. A partition, which groups a finite set of entities into larger chunks, is used in [52] to define an architecture as the assignment of essential system functions to elements of form. Architectures are enumerated by taking all possible function partitions, checking feasibility using logic constraints and then using a parametric solver to calculate output metrics

Perhaps the most widely used architecture representation is a design vector model, which relates input design parameters to output metrics such as cost, mass or performance through scientific or engineering models of constituent systems. An architecture becomes a collection of input parameters, often called the design vector, that is fed into a model that simulates how these inputs map to the output metrics of

interest. These models are usually customized for each application and can be used to investigate a system at a variety of levels of fidelity. At one end of the spectrum are deep point designs such as that presented in NASA's Design Reference Architecture 5.0 Mars Exploration study [41], which enumerates a single architecture at a fidelity down to the subsystem level. In these cases the 'model' used to map design parameters to architecture metrics is really a collection of domain-specific models developed for the variety of subsystems, combined with the opinions of experienced systems engineers who attempt to integrate these subsystem models to determine system-level metrics.

Ross and Hastings [50] argue that this method is not an effective way to evaluate a tradespace early in the design process since it prematurely constrains the tradespace and drives requirements to a solution specific form that hampers creativity [50]. When used correctly however, the goal of such efforts is not to select an architecture to pursue, but to identify the challenges that must be overcome and the technologies that must be invested in to make such architectures feasible and to give future mission designers a baseline from which to begin their analysis.

At the opposite end of the spectrum from deep point designs, lower fidelity design models allow outputs to be automatically calculated from design parameters, which means large tradespaces can be enumerated simply by varying components of the design vector. These models can contain just a few design parameters as in [16], where five design inputs are used to model a tradespace of 600 unique architectures for communications satellite constellations, or they can be highly complex as in Multidisciplinary Design Optimization (MDO) [36] studies, which often consist of tradespaces that are too large to be fully enumerated.

1.4.2 Architecture Tradespace Exploration

Once a tradespace has been defined, the next challenge often faced is how to find the optimal architecture, either as an initial selection or as a step along a development pathway. This process of searching through the available options is often termed tradespace exploration. Both Koo et al. [31] and Arney [1] start with a baseline

architecture and make incremental changes to the architecture in order to find the optimal configuration. In [31] an Algebra of Systems is used to make these permutations on a moon exploration mission architecture, while an Ant Colony Optimization algorithm is used to hunt for the optimum space transportation architecture for a variety of exploration destinations in [1].

A Genetic Algorithm (GA) is an optimization tool modeled after natural evolution processes ,and is used in [9] and [62] to search the tradespace for optimal architectures. In [9], Brown and Thomas explore the space of naval ship architectures to find the set of non-dominated designs, while in [62] Singh and Dagli develop a method focused on network-based systems of systems, using a smart power grid as a case study. Many other nonlinear programming algorithms are implemented in MDO methods to perform the core function of searching the design space for global optima by iteratively adjusting the design vector [36].

In many cases, multiple evaluation metrics are calculated for each architecture, so rather than defining a single 'best' architecture, the set of non-dominated architectures is found and evaluated further [51, 37]. This simple Pareto front analysis can give the system architect insight into the major trades as well as determine common system characteristics that are present in many or all non-dominated architectures.

Ross presents a review of several tradespace exploration methods in [50] that expand this basic Pareto front analysis to take into account performance or cost uncertainty, changing system requirements, and emergent system properties such as flexibility and robustness. In [70] Walton develops a framework to supplement the classic decision making metrics - cost and performance - with uncertainty to find groups of architectures that can be pursued concurrently in the concept exploration phase. By calculating the uncertainty embedded in the characteristics of each architecture, a portfolio of architectures is defined which matches a decision maker's sensitivity to risk with an uncertain utility per unit cost.

If an optimal architecture exists, exogenous changes may disrupt either the calculation of architecture fitness (e.g. performance, cost) or feasibility of the current optimal selection. Maher and Poon study this phenomenon by allowing both the

solution space and the problem statement to 'co-evolve' using a GA [34]. In [26] a similar evolutionary approach is taken, except in this case requirements are explicitly changed once an optimal architecture has been found to simulate an exogenous disruption and the GA is used to adapt the old architecture to these new fitness requirements.

A decision maker might also wish to understand how an architecture can be extended after selection so that performance can be improved or cost contracted to respond to changing external conditions. This design for flexibility or extensibility is different from searching for a shifting optimum as in [34] and [26], since now there is assumed to be some cost for changing the architecture or adding additional capabilities.

De Weck, de Neufville and Chaize study the staged deployment of a constellation of communications satellites in response to uncertain demand [16]. A real options approach is used to select an initial architecture with modest capacity that can later be supplemented with additional assets and reconfigured to respond to increases in demand. The initial architecture may not be optimal in terms of immediate performance for cost, but the option to upgrade the system at a later date when demand is known outweighs this loss in performance.

A retrospective analysis of architecture development is performed in [40], to chart the evolution of software versions from initial deployment to its final release, with a focus on the magnitude of change made at each intermediate step rather than on changes in performance. In [1] Arney qualitatively analyzes how systems within a particular architecture might be reused between missions to different destinations and uses this analysis to assess the viability of a 'flexible path' through the tradespace.

Silver and de Weck develop a quantitative tool to perform automatic exploration of development pathways through a design space in [59], which they call Time-Expanded Decision Networks. Starting with a tradespace of architectures and a detailed cost model of system development, operation, and retirement, a dynamic network is formulated which encodes the possible sequences of active architectures over several time steps. The problem of finding the progression of architectures among the possi-

ble options for a given demand is then restated as a shortest path problem through the network. The effect of altering the switching costs of moving between any two architectures is also examined as a way to introduce additional flexibility into the tradespace [59]. The framework proposed in this thesis is a closely related to the analysis tools developed in [59], with a number of terms and basic concepts reused in the following chapters.

1.4.3 Architecture Distance

As we will see in Chapter 2, a concept of distance between architectures is essential to the framework we propose as a way to encode the cost of switching from one architecture to another. This architecture distance will clearly be dependent on how each architecture is encoded. In [40], software architectures are represented as graphs of software elements with edges connecting interfacing elements. A distance metric is defined that measures the number of similar elements and connections over all the different substructures. A similar approach is used in [1] with the number of common systems between architectures used as a proxy for the ease of developing follow-on systems.

In [30] Hofstetter defines a measure between two architectures that counts the number of overlapping technologies and use environments to determine the feasibility of commonality for two products. The *delta* DSM is introduced in [64], which is a matrix representation of the difference between architectures. A weighted sum of all elements can be used to define a scalar value for the distance between architectures. The switching cost defined in [59] is a much more detailed definition of architecture distance, since the cost of decommissioning retiring systems and developing new systems must be calculated from the bottom up. This approach is not feasible for tradespaces with hundreds or thousands of architectures however since it is too labor intensive.

For cases where the architecture is modeled as an abstract entity such as a partition (which will be used to model architecture in Chapter 3), it is more difficult to parametrically estimate the cost of moving from one architecture to another. How-

ever, there do exist well known tools for determining the distance between these abstract representations. If this distance and the difficulty of making the architectural change that this distance measures can be linked to cost, then distance can be used as a switching cost proxy.

The literature regarding metrics on partition spaces is quite mature, with a number of proven metrics available for a variety of applications. A set of partition metrics is presented by Simovici and Jaroszewicz using the concept of generalized entropy between all pairwise combinations of partition blocks [61]. Using a much simpler approach, Day [15] characterizes six fundamental transformations for partitions on discrete sets - removal, augmentation, mutation, division, mergence and transfer - that are used to define several simple metrics on a partition space. One such metric is used in [68] to quantify the difference between clusters of software resources for reverse engineering of complex software systems. In the application of our tradespace exploration framework we will make use of a similar distance measure, which will be explained in Chapter 3.

1.4.4 Technology Portfolio Selection

Closely related to the problem of architecture modeling and selection is that of technology portfolio selection. The development of large, complex systems often requires large technology investments, the study of which has created a distinct field of research in the literature. The standard approach for NASA and many other organizations responsible for the management of complex technology portfolios is to create broad technology roadmaps based on the input of subject matter experts and current organizational priorities; see for example [65, 47, 20]. The purpose of roadmaps is usually not to evaluate the impact of technologies on a specific architecture or chart the detailed development of a technology, but rather to connect strategic goals and broad organizational needs to technology development. Their form is usually that of an enumeration of potential technologies, an evaluation of the projected impact of these technologies on strategic goals over time, and a prioritized subset of technologies to carry forward.

Next these broad technologies are assigned to specific projects, with the goal of increasing the maturity of a particular technology either by performing basic research or implementing the technology in a functioning system. The problem of allocating resources to the technology development process has been shown to be highly uncertain since the cost of maturing a technology along with its final impact on system performance is difficult to predict [27, 66]. Technology development for large government portfolios adds additional layers of complexity and uncertainty to the process as discussed in [66].

In [28] Heidenberger and Stummer describe several methods to quantitatively evaluate the process of selecting and allocating resources to technology projects. They divide these approaches into six categories: benefit measurement techniques, mathematical programming, decision and game theory models, simulation models, heuristic methods, and cognitive emulation. Within the class of benefit measurement techniques a commonly used method is real options analysis, which uses options pricing models to value technology investment decisions. Real options analysis has been used to evaluate investment decisions for NASA [27, 58, 53] as well as for information technology project investments [7] and automotive industry research and development [4] among many other applications. Other technology portfolio decision aids studied in the literature include multi-attribute utility theory [39], financial portfolio theory [32] and knowledge-based expert systems [33].

These methods share the common trait that the benefit of a technology is calculated relative to a baseline architecture or parametrically defined model of a project portfolio. Battat et al. take a different approach by considering the effect of a particular technology on many architectures spanning the tradespace [5], a perspective that is useful if no baseline architecture exists or the set of projects to which the technology will be applied is highly uncertain. Starting with a large tradespace of feasible architectures for manned space exploration to Mars, the required set of technologies is extracted for each architecture. The impact of a technology is evaluated by taking the difference in total mass to orbit required for those architectures that use the technology versus those that do not. Using these global measures of technology impacts,

a decision maker can then prioritize the investment of technologies for situations in which the final architecture is highly uncertain.

1.4.5 Rule-Based Systems and Jess

A Rule-Based Expert System (RBES) is computational tool that imitates the decision making process of human experts by modeling expert knowledge as logical rules. Several rule-based expert systems will be used extensively in Chapters 4 and 5. As opposed to conventional procedural programming approaches that specifies the sequence of computational steps taken by the program, rule-based programs are made up of collections of '*if-then*' statements with order of execution determined at run-time. Refer to [24] for an in-depth introduction to rule-based expert systems.

A rule-based system is composed of three primary components: a set of rules, a set of facts, and an inference engine. Rules are statements which relate a set of preconditions to a set of actions to be executed. Rules encode the expert knowledge that defines the behavior of the program. The set of facts, also called the working memory, consists of information about the problem being studied such as initial conditions, conclusions or objects for information storage. Facts can be created, removed, or modified by rules, and generally correspond to objects in the procedural programming sense of the word. The inference engine, which is the computational core of the RBES, searches for rules whose preconditions are met and then executes the actions associated with those rule, altering the fact set and potentially activating additional rules [56, 23].

The rule-based systems implemented in this thesis are based on the *Jess* (Java Expert System Shell) rule language and engine. The *Jess* rule engine uses an implementation of the Rete algorithm [23] to efficiently match facts to rules and execute the actions of the rule-based system. We refer the reader to [23] for an overview of the *Jess* language, syntax, and execution engine.

Chapter 2

A Framework for Tradespace Exploration with Weighted Graphs

Chapter 1 has highlighted the utility of a rigorous framework in which to analyze and explore large architecture tradespaces. In Chapter 2 we will present such a framework based on the organizing principle that tradespaces can be more clearly understood and analyzed as graphs than as a collection of individual point designs. The key 'value added' by a graph representation of the tradespace is that it allows the *relationships* between architectures to be modeled in the decision making process, rather than just the architectures themselves.

The core of this chapter will be devoted to the conversion of an architecture tradespace into its graph representation and an explanation of the general problem statements we have chosen to guide our subsequent analysis. As a well developed field of study with a wide range of applications, the study of graph theory provides a variety of domain-independent concepts and analysis tools that we can apply 'as is' to our tradespace graph. In particular, we will rely on well understood and highly efficient algorithms, such as Dijkstra's Shortest Path Algorithm, to focus more narrowly on the decision modeling problems that are specific to the study of tradespace exploration.

We will begin in Section 2.1 with an explanation of the terms and notation that we will use throughout this and subsequent chapters to more clearly communicate our framework and analysis. In Section 2.2 we will present and discuss the fundamen-

tal problem statement and general sub-problems that guide our development of the tradespace graph and the methods to analyze it. Section 2.3 outlines the structure of the tradespace graph model.

2.1 Vocabulary and Notation

2.1.1 Vocabulary

To provide clarity for future discussion, it will be useful to define several terms to which we will assign specific meaning in the context of this thesis.

- A **system architecture** (also referred to simply as an architecture) is the central unit of analysis of this thesis. Although system architecture is defined by a number of sources at varying degrees of abstraction, we will use the general definition offered by Crawley *et al.* of an architecture as "an abstract description of the entities of a system and the relationships between those entities" [14]. For the systems studied in this thesis, an architecture is defined at a level of fidelity that abstracts out the detailed characteristics and decisions that are typically associated with the design of a system. Instead, the focus of the architecture is on the high level mapping of function to form along with both upstream and downstream influences that drive overall system cost, benefit, and performance.

- A **tradespace** of system architectures is a collection of architectures that can be readily compared to one another. A tradespace might consist of many architectures that span all feasible configurations of the system, or it might consist of architectures which vary only across a subset of the characteristics that define the architecture model. The framework presented in this thesis assumes that the tradespace under analysis has already been enumerated and the architectures within it have been evaluated.

- The **distance** between two architectures in the tradespace is a proxy for the cost of changing from one to the other (also referred to as the switching cost).

Ideally, this distance is stated using a cost basis that is easily transformed into a dollar-figure, however depending on the architecture model used and the cost models available, the distance may be expressed in a less concrete basis. Intuitively two architectures which are very similar to one another should have a low switching cost and therefore should be separated by a small distance.

- An **asset** in the context of this thesis is a technical capability or subsystem that is common to two or more architectures. If the system architecture is changed, assets are the 'non-perishable' capabilities or elements developed for the original architecture which can be transferred to and used as part of the new architecture. One example of an asset, analyzed in the HEXANE application in Chapter 3, is a technology such as methane-based in-space propulsion which is utilized in a number of different architectures. Another example from the SCaN application in Chapters 4 and 5 is a constellation of communications satellites that is common to two or more distinct architectures.

- The **asset portfolio** is the collection of all assets available to the decision maker to be used in the architecture. Assets may be added to the portfolio by making an investment in that asset (e.g. investing in a technology or developing and deploying a new satellite constellation), or they may be removed from the portfolio if the asset has a limited lifetime or quantity that is exhausted.

- In this thesis we are concerned with modeling how decisions might be made to transform a system from one architecture to another. Therefore, we need a way to describe the architecture at each step in the transformation. We define the **stage** of the transformation as the way to index the 'snapshots' of the architecture along the transformation. For example we might say that at Stage 1 the system architecture was A, while at Stage 2 it was B. Presumably, some decision must have been made from Stage 1 to Stage 2 to change the architecture in some way.

  From a mental model perspective, it is easiest to link the stage index directly to time by saying for example that two consecutive stages are separated by

one year, such that the architecture at Stage 1 is a snapshot of the system at year 1, and the architecture at Stage 2 is a snapshot at year 2. We use this type of model in Chapter 5, however more generally the stages of a transformation can be independent of time. For applications with highly uncertain decision schedules (e.g. the HEXANE tradespace), stages simply communicate the *ordering* of snapshots along the transformation. In this sense, the system will not necessarily be manifested (i.e. built and operated) at each stage along the transformation, rather these stages represent the sequence of incremental architectural changes that constitute a larger transformation. As we will see in Chapter 3, this decomposition of large architecture changes into incremental ones that might not be realized individually can provide novel value to the decision maker in terms of assessing architecture robustness and flexibility.

- The **active architecture** is the architecture in the tradespace which characterizes the system at a particular stage. From the perspective of the primary decision maker (i.e. system architect), the active architecture at a given step is the architecture which embodies the system. All other architectures in the tradespace are **candidate architectures**. The **initial architecture** is the active architecture at the first stage in the tradespace, while the **final architecture** is the active architecture at the last stage.

- The **evolution** of architectures is the sequence or architectures at all stages through the tradespace from the initial architecture to the final architecture. The evolution can also be thought of as an incremental pathway through the tradespace from initial to final architecture. Analogous to the architecture evolution is the asset portfolio **progression**, which tracks how the asset portfolio changes from one stage to another along the path through the tradespace. Finding the evolution of architectures and progression of the asset portfolio that satisfy a number of different optimality conditions will be central to this work.

- Finally, it will be useful to clarify the language used when referring to the tradespace graph itself. The graph is made up of **vertices** and **edges**, which

connect two vertices to one another (or in some cases a vertex to itself). The graphs we will be studying are directed and weighted. The **source** of an edge is the vertex from which it originates, while the **sink** of the edge is the vertex at which it terminates. The **weight** of an edge is a proxy for the cost of traversing an edge and will be discussed in detail in subsequent sections.

2.1.2 Notation

The notation in this section is presented to aid in the discussion of the generic tradespace exploration problem statements in the following section and the development of the tradespace graph in the remainder of the chapter. Nowhere in this thesis do we perform rigorous mathematical analysis or derivation using this notation, so we would like to make it clear that the intended value of the notation presented is as a communication aid rather than a rigid part of the tradespace graph framework.

Let the indexed variable a_i represent a distinct architecture defined during the enumeration of the tradespace. Note that a_i can be encoded in any way we choose (e.g. function to form mapping, form DSM, decision sequence, etc.). The whole tradespace is the set A of N distinct architectures

$$A = \{a_1, a_2, ..., a_N\}.$$

We denote the distance between two architectures as

$$d_{i,j} = d(a_i, a_j).$$

Each architecture will also be evaluated according to a scalar performance metric $F_i = F(a_i)$ which will be used to compare two architectures to one another in terms of the value they provide to stakeholders. The calculation of the performance metric is an integral part of the tradespace models used for each case study, and will be discussed in detail in the corresponding application section.

When referring to any quantity with respect to a stage in the system transfor-

mation, the convention we have chosen is to put the stage index in parentheses as a superscript. Thus, an evolution E of architectures is the ordered sequence

$$E = \{e^{(1)}, e^{(2)}, ..., e^{(D)}\} \quad e^{(k)} \in A,$$

where D is the number of architectures in the evolution from initial architecture to final architecture, which is the same as the number of stages in the transformation. Each element of E is an architecture within the tradespace, with the ordering corresponding to the sequence of the evolution such that $e^{(1)}$ is the initial architecture and $e^{(D)}$ is the final architecture. Note once again that the elements of E do not necessarily all represent desired physical manifestations of the system architecture, rather they more generally describe the *incremental* pathway through the tradespace.

Let M be the number of unique assets present in the enumerated tradespace. The value of M might represent the number of unique technologies modeled in the tradespace (e.g. in the HEXANE study), or it might represent the number of unique satellite constellations present across all architectures (e.g. in the SCaN study). In performing our analysis we do not necessarily need to know the value of M, though it is a useful notation aid here.

For each of the M possible assets in the tradespace, let the variable $p_m^{(k)} \in \{0, 1\}, \quad m = 1, ..., M$ be a binary value that represents the presence (1) or absence (0) of the m^{th} asset in the asset portfolio at the k^{th} stage of the transformation. The asset portfolio at the k^{th} stage of the transformation is the vector of all assets at that stage:

$$P^{(k)} = \left[p_1^{(k)}, p_2^{(k)}, ..., p_M^{(k)}\right].$$

The progression of the asset portfolio $\mathbf{P}$ is then the $D \times M$ matrix

$$\mathbf{P} = \begin{bmatrix} P^{(1)} \\ P^{(2)} \\ ... \\ P^{(D)} \end{bmatrix}.$$

Turning now to the notation used to represent the tradespace graph in this chapter, the K vertices in the graph are represented by the vector

$$V = \begin{bmatrix} v_1 & v_2 & \dots & v_K \end{bmatrix}$$

The graph edges, $U = \{u_{i,j}\}$ are encoded in the edge weight matrix $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} W_{1,1} & W_{1,2} & \dots & W_{1,K} \\ W_{2,1} & W_{2,2} & \dots & W_{2,K} \\ \vdots & \ddots & \ddots & \vdots \\ W_{K,1} & W_{K,2} & \dots & W_{K,K} \end{bmatrix}$$

where $W_{i,j}$ is the weight of the directed edge $u_{i,j}$ from v_i to v_j. To encode the absence of an edge between two vertices in the weight matrix, we set the corresponding element of $\mathbf{W}$ to infinity.

2.2 Problem Statement

The purpose of the tradespace exploration effort, as discussed in Section 1.1, is to give decision makers an analytic tool to understand how system architecture decisions affect both short term and long term performance and cost in complex systems. The fundamental reason behind the need for such careful analysis of these types of decisions is that the cost of changing the system architecture is high, however often architecture change is required or even desired.

As discussed in Section 1.1, we have identified three broad use cases of the tradespace exploration effort: maintaining architectural flexibility to respond to changes in system goals; maintaining architectural robustness to exogenous system disturbances or failures that make the current architecture infeasible; and developing new, planned iterations of the system over time to improve system performance or reduce recurring cost. In an abstract sense these three cases define the tradespace exploration problem, however in order to develop a quantitative tool we must frame the

problem in more concrete and simpler terms.

All three of the use cases above are centered on the architecture changing in some way, so having introduced the *evolution* as a way to characterize architecture transformation we frame the tradespace exploration problem around finding the optimal evolution of architectures through the tradespace. Depending on the particular type of change being analyzed, we impose constraints on how the architecture can vary from stage to stage in the evolution and what types of decisions can be made. Independent of these constraints on the evolution, we can decompose the process of defining an evolution into three separate tasks:

- Finding the initial architecture: $e^{(1)}$
- Finding the final architecture: $e^{(D)}$
- Finding the incremental path from initial to final: $e^{(2)}$ to $e^{(D-1)}$

We then define the problem statements for this framework based on which of these tasks are assigned to the tradespace exploration algorithm and which are constrained *a priori*. We have identified four general problems that serve to guide our approach to developing the tradespace graph and subsequent analysis tools. These problems, in order of increasing complexity in terms of the graph framework to be presented, are:

Problem 1: Find the initial architecture $e^{(1)}$.

Problem 2: Given $e^{(1)}$ and $e^{(D)}$, find the architecture evolution E.

Problem 3: Given $e^{(1)}$, find the architecture evolution E.

Problem 4: Find the architecture evolution E with no predefined initial or final architecture.

It is important to note that these generic problem statements must be augmented with additional decision constraints and problem attributes that are specific to the decision problem and tradespace under analysis, which is a task left to Chapters 3, 4, and 5.

Each of the four problems above is based on realistic decision making scenarios that system architects might face depending on the phase of the system in the lifecycle and stakeholder goals. Scenarios in which Problem 1 might be applicable include complex systems in the Pre-Phase A stage of the design cycle that are projected to have only a single iteration - perhaps due to the length of the lifecycle or unacceptably large switching cost for even small architecture changes. Another class of applicable systems are those for which too much uncertainty exists regarding the future of the system (perhaps in terms of future stakeholder needs or future funding) to reliably plan the evolution of the architecture.

Problem 2 is applicable to systems such as commercial software products or consumer electronics with multiple planned iterations towards a well-defined final architecture. Time constraints, cost constraints, or risk reduction strategies might limit the magnitude of system change that can be realized at each iteration and require a phased development approach. An application from the space systems domain is the historic case of the selection of the evolutionary sequence of the Gemini and Apollo mission architectures once Lunar Orbit Rendezvous was chosen in 1962 as the architecture for the lunar landing. This same type of problem may appear again in the future once NASA selects an initial architecture for deep space human exploration and defines a mission architecture for the stated end goal of human exploration of Mars.

Problem 3 is relevant to systems for which a current architecture exists (either as a current manifestation of the system or as the active architecture under development) that decision makers wish to augment to improve system performance, meet changing stakeholder needs, or reduce long-term costs. An example of this type of decision problem is studied in Chapters 4 and 5 in terms of the question of how to evolve the current Space Network communications architecture (i.e. the TDRS network) into a future space-based network that meets changing stakeholder needs and incorporates modern technologies.

Problem 4 is the most general of the problems we have identified, and consequently has a wide variety of applications. One application is a system for which

no current architecture has been defined, but for which goals for the initial and final state of the system have been set. As an example, in Chapter 3 we study a system - the in-space transportation infrastructure for human exploration - for which no current architecture exists, but for which definite long-term goals have been defined by the administration (i.e. transport a human crew to Mars for a surface exploration mission). Selecting an initial architecture that allows for near-term exploration but also allows for evolution to an architecture that fulfills this goal is a challenging and relevant problem for current NASA decision makers.

Another example of Problem 4 is a scenario in which the current architecture of a system becomes infeasible due to some exogenous change (e.g. failure in technology development, development of a competing system, or budget reduction). A new initial architecture that takes into account constraints imposed by development of the now-infeasible architecture must be selected and presumably a path to a more suitable architecture planned. An example of this type of scenario can be found in the major restructuring of NASA's human spaceflight program with the cancellation of Constellation and the introduction of the Space Launch System (SLS) and Multi Purpose Crew Vehicle (MPCV) [69, 42]. Following a restructuring of strategic goals and with a Congressional mandate to "utilize existing contracts, investments...and capabilities from the Space Shuttle and Orion and Ares I projects" [69], NASA decision makers had to select a new system architecture that met stakeholder needs while making use of as much of the previous project architecture as possible. In addition, such an architecture was required to "incorporate capabilities for evolutionary growth" [69] to higher performance missions along an uncertain time line.

2.3 Graph Definition

Each of the problems described above is based on the analysis of relationships between architectures rather than simply the analysis of architectures in isolation. Transforming the tradespace into a graph allows us to model these relationships in a simple and intuitive way, and to reformulate the problems above as well-studied problems in

graph theory.

A natural candidate for the vertices of the tradespace graph is the set of architectures that define the tradespace. In fact, in order to model the progression of the asset portfolio, which is key to much of the analysis we wish to perform, we define a vertex as an architecture *paired* with a feasible asset portfolio.

$$v_i = (a_j, P) \quad a_j \in A, \quad P \in \{1, 0\}^M$$

where P must be a feasible asset portfolio for a_j. The definition of a feasible asset portfolio is part of the application-specific problem and can vary based on the particular constraints imposed by the decision analysis desired. In order to populate the vertices in the graph, we take each architecture under consideration and enumerate all the feasible asset portfolios for that architecture. Thus a single architecture might map to many vertices in the graph.

The key to modeling the relationships between vertices is defining the existence and weight of edges. In our framework we equate the existence of an edge to the feasibility of a decision to transform the architecture from one vertex to another along the direction of the edge. As we traverse an edge, jumping from one vertex to another, we assume that changes are made to the architecture and asset portfolio of the source vertex to transform them into the architecture and portfolio of the sink vertex. This transformation might be the addition or deletion of technologies or subsystems from the portfolio, it might be the reassignment of functions to different elements of form, or it might be a change in the structure of components of the architecture.

We define the existence of an edge based on the relationship between the asset portfolios of the two candidate vertices. This modeling decision was made under the assumption that changes to the asset portfolio are the 'big ticket' investments that a project makes, so basing the feasibility of jumps between vertices on these large investments allows us to examine the effects of incremental investment decisions on the system. For each application of the tradespace graph framework, we define the *edge existence rule*, which contains the criteria for an edge to exist between two vertices.

When we define the edge existence rule, we make explicit the types of decisions that are allowable when making asset investments to change the architecture. For both the SCaN and HEXANE tradespaces studied in this thesis, the edge existence rule is essentially that at most a single asset can be added to the portfolio across an edge.

Once the existence of an edge has been determined, the final task in building the graph is defining the weight of an edge. One interpretation of edge weight, which we use here, is as an indication of the difficulty, or cost, of moving from one vertex to another. As discussed above, there is presumably some estimated cost of changing the architecture (or at least a proxy for this switching cost), which is a natural candidate for the edge weight. In the notation section, we defined the distance metric d based on this same switching cost, so we set the edge weight equal to the distance from the architecture of the source vertex to the architecture of the sink vertex:

$$W_{i,j} = d(a_i, a_j).$$

Clearly this will require the explicit definition of the architecture distance metric as well, but like the edge existence rule we will leave this task to the individual application chapters. Figure 2-1 shows an example tradespace graph with three architectures. There are four possible assets ($M = 4$) in the asset portfolio, with a 1 encoding the presence of the asset at that index and a 0 encoding its absence as described in Section 2.1. Note that the edge existence rule is that either one or zero assets is added across the edge. Thus there is an edge from v_1 to v_2 since the asset at index 2 is added, but there is no edge from v_1 to v_3 since the assets at indices 1 and 2 would need to be added.

2.4 Graph Analysis

We now have all the pieces in place to describe how the tradespace graph can aid in solving the tradespace exploration problems described in this chapter. Three of the four problems are concerned solely with the evolution of the architecture from one

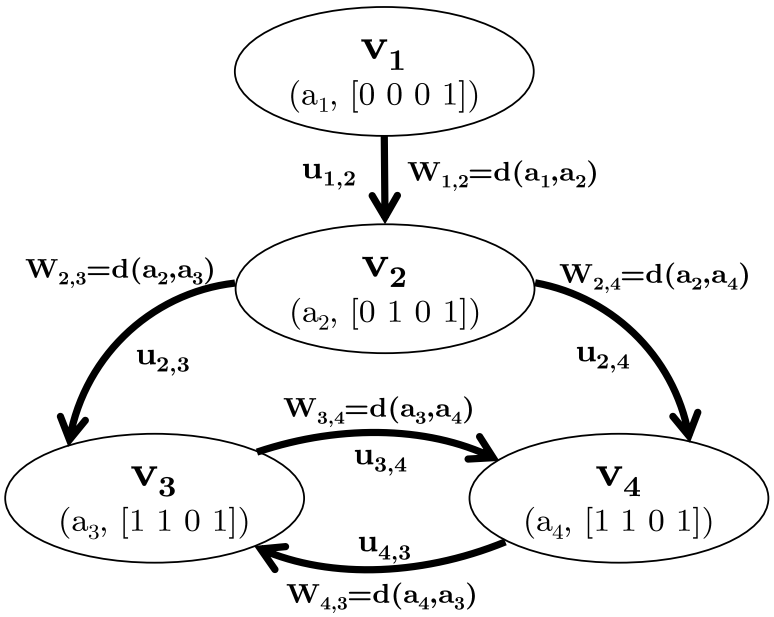


Figure 2-1: An example of the tradespace graph framework with an edge existence rule that either one or zero assets can be added across an edge.

point in the tradespace to another. Similar to the approach taken by Silver and de Weck in [59], we restate an architecture evolution - which itself is simply a sequence of related architectures - as a path through the tradespace graph. Each step in the path from one vertex to another represents a decision to change both the architecture and asset portfolio across the edge, with the edge weight serving as an estimate of the cost of the decision. The rules and constraints used to generate the graph itself encode the set of feasible decisions available at each vertex. The concept of a stage as defined in Section 2.1 is also now much clearer, as the stage of the evolution is equivalent to the index along the path.

This relatively simple conceptual jump from a sequence of architectures to a path through a graph allows us to implement a variety of domain-independent tools and algorithms tailored to analyze paths through a graph. Consider first the context of Problem 2, where both the initial and final architectures of the evolution are known. Here the decision maker wants to know the intermediate architecture and asset portfolio decisions that minimize the cost of the transformation. This is the rigorously studied and widely applicable *shortest path problem* in graph theory. A

number of optimal and heuristic algorithms to the solve the shortest path problem exist [11].

For the applications presented in this paper, the size of the tradespace is small relative to the size of many graphs studied in graph theory and edge weights are non-negative so we implement an optimal algorithm known as Dijkstra's algorithm [17] to solve the shortest path problem when it arises. The implementation of Dijkstra's algorithm used in this thesis uses a Fibonacci heap to reduce the time complexity of the algorithm from $\mathcal{O}\left(|V|^2\right)$ to $\mathcal{O}\left(|U| + |V| \log |V|\right)$ and is developed by Fredman and Tarjan [22]. Dijkstra's algorithm returns the set of vertices along the shortest path from start vertex to end vertex, which defines the architecture evolution and asset portfolio progression for our problem.

Dijkstra's algorithm on its own finds only the single shortest (i.e. minimum cost) path through the graph, and for the decision problem described above there is often an additional consideration that the decision maker might wish to trade a higher cost of transformation for higher performance in the intermediate architectures. Such a case might arise during the development of an operational system such as a communications network through many iterations, where the gradual improvement of performance over time is desired. In such a scenario, we might wish to find the set of paths which comprise the K-shortest paths from the initial to final architecture, and allow the decision maker to trade among this set.

Yen's K-shortest paths algorithm [71] is implemented in this thesis for this purpose, and works by repeatedly applying Dijkstra's algorithm to perturbations of the original graph with edges deleted along the original shortest path. We can then compare different paths to one another by trading the total path length against a path performance measure such as the sum of the architecture performance metrics along the path.

Finding the end point(s) of the architecture evolution as applicable to Problems 1, 3 and 4 is a less straightforward problem, but can still be aided by simple analysis of the tradespace graph. The simplest approach is to select the subset of vertices that define the Pareto front of architecture lifecycle cost and benefit, and then to allow the

decision maker to trade between the possible endpoint decisions. Another approach is to define a subset of candidate initial or final architectures, and then to find the shortest path from (to) a defined endpoint to (from) each of these candidates. Plotting each of these candidates as the candidate performance metric against the path length allows us to define another Pareto optimal front for decision makers to trade among further.

If the goal of the endpoint selection is flexibility, we can analyze the structure of the graph itself rather than pathways through it. The concept of vertex centrality is one which can be applied in this context, as we wish to select an initial or final architecture that is near to many other architectures - especially those with high performance scores. A relevant centrality metric is defined in [48] that allows us to prioritize connections differently based on whether they lead to high performing or low performing architectures. Finally, we can also simply count the number of architectures within a given distance neighborhood of a candidate endpoint to determine its relative flexibility.

Chapter 3

In-Space Transportation Infrastructure: HEXANE Tradespace Graph

In this chapter, we apply the tradespace exploration framework of Chapter 2 to a tradespace of system architectures for in-space transportation infrastructures as required for human exploration missions beyond low Earth orbit. In the first section of the chapter we will present an overview of the HEXANE tradespace enumeration tool, which is used to generate the tradespace used in our analysis, as well as discuss in detail the system architecture model used for this particular case study. We will then define the tradespace graph for this application based on the framework in Chapter 2, as well as make explicit the assumptions made and specific problem statements analyzed in subsequent results. Finally, we will present the results of our tradespace exploration analysis for several analysis scenarios.

3.1 Background

The Human Exploration Architecture Network Evaluator (HEXANE) tool is a computational model that allows users to enumerate and evaluate an exhaustive set of feasible architectures for transporting humans and their cargo beyond low Earth or-

bit for the purpose of scientific exploration. HEXANE was developed at MIT by Rudat et al. [51, 52] to analyze the fundamental architectural decisions that drive the high-level cost, performance, and functions of a human exploration infrastructure. The user specifies basic exploration mission parameters such as mission destination, number of crew, duration of exploration mission, and science payload, and HEXANE outputs the full set of transportation infrastructure architectures that could feasibly realize this mission.

Although we will not discuss the implementation of the HEXANE tool in this thesis (see [51] for reference), an understanding of how an architecture is modeled and how performance and cost metrics are calculated is essential to our definition and analysis of the tradespace graph.

3.1.1 The HEXANE System Architecture Model

The complexity and uncertainty surrounding a system as vast and unprecedented as the future transportation infrastructure for sending humans to the surface of Mars, the Moon, or an asteroid make modeling the full mission architecture down to the subsystem level both unrealistic and of limited use. Since mission requirements are vaguely defined and currently in flux, and critical technologies have yet to mature, the architecture model used is necessarily a low fidelity one and abstracts components of the total mission architecture that will play a critical role in eventual system realization.

An architecture in the HEXANE tradespace is modeled by the primary supporting systems required to transport humans and their cargo from low Earth orbit to their destination (e.g. Mars surface, Moon surface) and back to the Earth's surface. These supporting systems consist of the propulsion elements required to execute the necessary velocity changes to change orbits and the habitation elements necessary to keep the crew alive throughout the mission. Importantly, the HEXANE model does not include the launch vehicle to reach low Earth orbit, the mission-specific systems such as surface transportation or science payload beyond a user-specified 'mission payload' mass contribution, or operations-centric aspects of the architecture such as

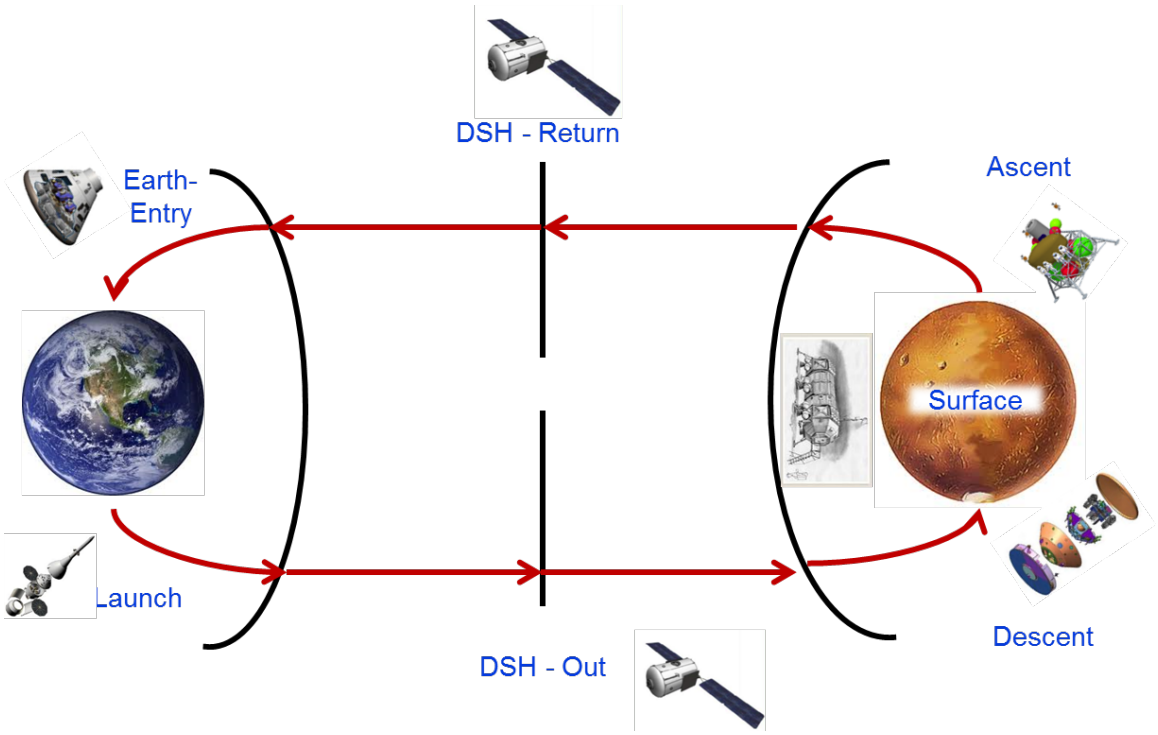


Figure 3-1: The seven top-level habitation functions in the HEXANE model shown for a Mars surface mission.

mission control or contingency planning.

The core of the tool is based on the assertion made by Rudat et al. that any manned exploration mission must execute seven top-level habitation functions and ten top-level propulsion functions regardless of the destination or mission architecture. These functions are shown in Figures 3-1 and 3-2 and are listed in Table 3.1. Each of these functions must be assigned to an element of form (i.e. a propulsion stage or pressurized habitat), and the nature of this assignment determines the basis of the architecture.

Multiple functions can be assigned to a single element of form (e.g. a single engine and tank can execute multiple maneuvers), however each function can only be assigned to one element. This mapping of function to form defines a partition of the set of 17 top-level functions and encodes the *functional architecture*, a component of the total architecture.

Other components of the architecture include the propellant used for each maneuver, the deployment time line of different elements and the required technologies.

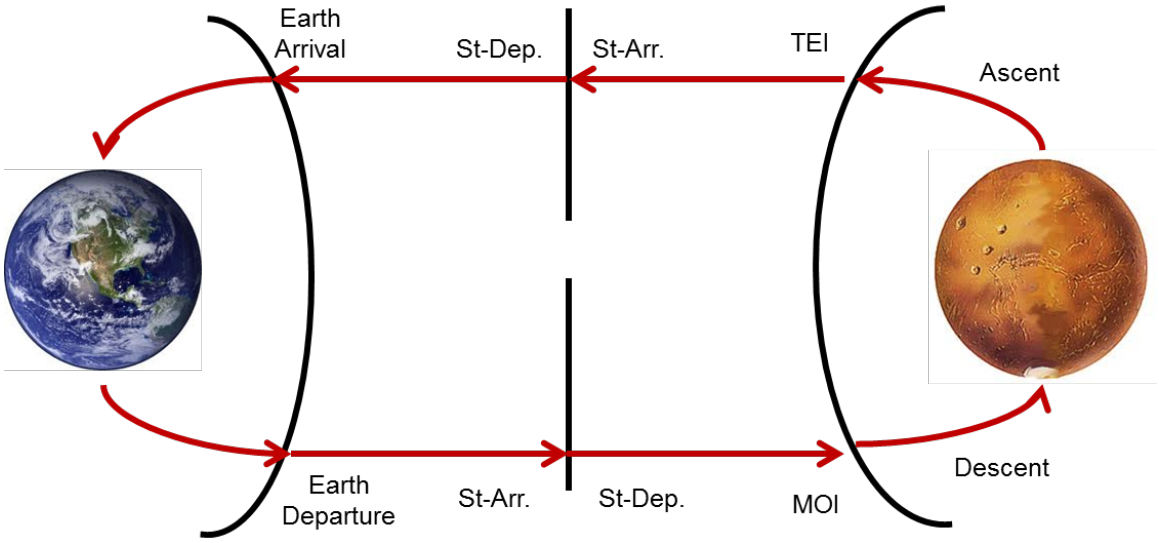


Figure 3-2: The ten top-level propulsion functions in the HEXANE model shown for a Mars surface mission.

Propulsion Functions	**Habitation Functions**
1. Earth Departure	
2. Staging Arrival	1. Launch
3. Staging Departure	2. Deep Space Out
4. Destination Arrival	3. Descent
5. Descent	4. Surface
6. Ascent	5. Ascent
7. Destination Departure	6. Deep Space Return
8. Staging Arrival	7. Earth Re-entry
9. Staging Departure	
10. Earth Arrival	

Table 3.1: List of propulsion and habitat functions common to all HEXANE architectures.

Technology Description	Score
In-Space Nuclear Thermal Propulsion	1
In-Space LOX-Hydrogen Propulsion	$\frac{1}{6}$
In-Space LOX-Methane Propulsion	$\frac{1}{2}$
Descent LOX-Hydrogen Propulsion	1
Descent LOX-Methane Propulsion	1
Descent Hypergol Propulsion	1
Ascent LOX-Hydrogen Propulsion	1
Ascent LOX-Methane Propulsion	1
Ascent Hypergol Propulsion	1
Solar Electric Propulsion	$\frac{1}{3}$
Propellant Boil-Off Control	$\frac{1}{2}$
Aerocapture	$\frac{1}{3}$
Surface In-Situ Resource Utilization	1

Table 3.2: List of HEXANE technologies and relative development costs.

The maneuver propellant determines the performance characteristics of the maneuver (i.e. specific impulse) and the sizing of the tank and propellant required. The deployment time line reflect how elements of form are transported to their destination. One option is to send an element (i.e. propulsion stage or habitat) on the 'high-energy' stack with the crew, while the other is to pre-deploy an element by placing it on a 'low-energy' stack that uses low thrust, high efficiency electric propulsion to place elements to be used later in the mission into their desired positions.

The final component of the architecture is the set of technologies that must exist for the architecture to be feasible. Battat et al. [5] define 13 major technological capabilities that are modeled in the HEXANE framework. Using a simple analysis of the estimated difficulty of technology development and the utility of the technology to other stakeholders, each technology has a score to give it a relative 'price' for development. The 13 technologies and their development scores are shown in Table 3.2. Each architecture is analyzed to determine which of these 13 must be available for the architecture to function, which is stored as the required set for that architecture. This required set is then combined with the development scores to determine a rough technology development cost (TDC) for each architecture.

Enumeration of the tradespace is executed combinatorially by varying each archi-

tecture modeling parameter over allowable values. A number of logical constraints are checked during the enumeration stage to ensure that logically infeasible architectures, such as an architecture that has two different propellants assigned to the same element, are filtered out. After all feasible alternatives are enumerated each architecture is evaluated by sizing each habitat and propulsion element using a number of parametric sub-models explained in detail in [51]. After sizing each element, incorporating mass contributions from user defined payload, and iteratively solving for the propellant mass required, the initial mass in low Earth orbit (IMLEO) can be calculated. Since mass to orbit has proven to be a good indicator of recurring mission cost, IMLEO is used as the performance metric in our analysis of relative architecture fitness.

The result of this evaluation is a tradespace of several thousand architectures that can be compared along two simple metrics: the initial mass of the mission to low Earth orbit (IMLEO), and the technology development cost (TDC) of the technologies required for that architecture. By plotting the tradespace according to these two metrics, we generate the scatter plot shown in Figure 3-3 for a 500 day Mars surface mission. This tradespace is made up of $75,351$ unique architectures, each represented by a blue or green point in the plot. The red line represents the Pareto optimal frontier of this tradespace, which is the set of non-dominated architectures for which neither IMLEO nor TDC can be decreased without increasing the other. The green points are those architectures in the *Fuzzy Pareto set* [64], which is determined by finding successive Pareto fronts up to a certain percentage of the tradespace.

3.2 Graph Development

In this section we will transform the HEXANE tradespace model into a tradespace graph using the basic principles outlined in Chapter 2. The HEXANE tradespace graph will be used to analyze a variety of architecture selection decision problems currently facing NASA system architects as well as a number of potential future problems that highlight the utility and flexibility of the tradespace graph as a decision

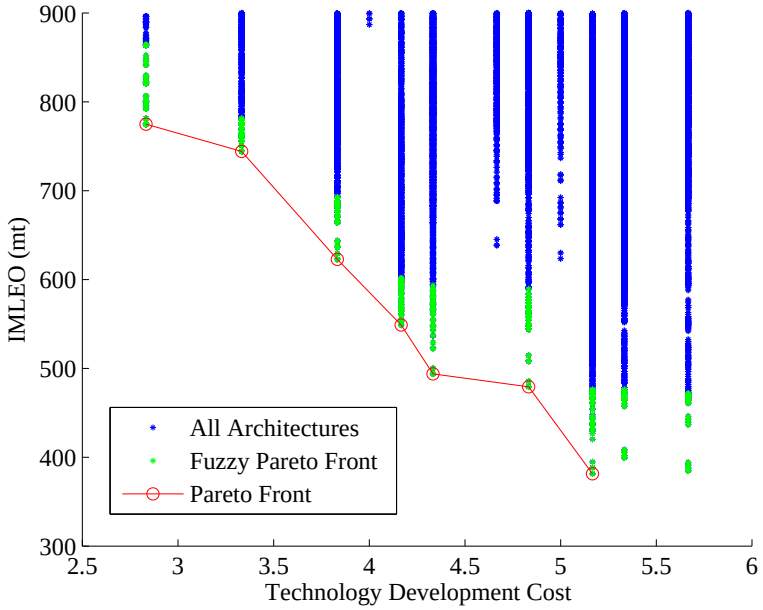


Figure 3-3: The evaluated in-space transportation infrastructure tradespace with architectures plotted according to IMLEO and Technology Development Cost. Pareto optimal architectures and Fuzzy Pareto Front shown.

analysis tool.

To develop the HEXANE graph we first define the components that make up a vertex in the graph, and then enumerate the edge existence criteria that are an application-specific extension of those in Chapter 2. Next we define and explain the distance metric used for this tradespace as a proxy for the architecture switching cost, and finish with a discussion of the various scenarios and decision constraints we wish to analyze as well as the assumptions underlying this model of the decision making process.

Before focusing on the definition of the graph, it is essential to have a clear understanding of what the output of the tradespace graph framework for the HEXANE application means to the decision maker. Namely, we wish to clarify the meaning of the stages of the evolution of architectures and progression of technology portfolios. Since any in-space transportation infrastructure is necessarily an extraordinarily complex and expensive system, it is highly unlikely that a decision maker would plan to take an iterative approach of changing the architecture over time. Much more likely

would be an approach whereby the decision maker would select a single design point that would serve as the dominant architecture model for all missions for which the infrastructure is designed, with a degree of planned sub-architecture upgrades over later missions.

This fact clearly highlights the need to make this initial selection to maximize performance, flexibility, and robustness to exogenous changes, however it also suggests that viewing an evolution as a series of iterative changes made over time is not useful for this application. Instead, stages along the pathway through the tradespace serve as incremental checkpoints, or intermediate steps that describe the atomic changes that compose the much larger changes in which the decision maker is interested.

Therefore, each stage of the evolution or technology portfolio does not necessarily represent a recommended manifestation of the system, but rather a potential intermediate point of the system transformation, which we argue in the results section contains significant information value for the decision maker.

3.2.1 Vertex Definition

A vertex in the HEXANE tradespace graph models a feasible state of the system in existence or under development. In Chapter 2 we define a vertex as an architecture paired with an asset portfolio. For the current application, an architecture is one of the points enumerated and evaluated using the HEXANE tool, while we define the asset portfolio as the set of technologies available to be deployed in the system (i.e. the technology has been matured or an investment has been made to ensure maturity by system realization), which we call the technology portfolio. The components of a HEXANE tradespace vertex are shown in Figure 3-4 in graphic form, with the decomposition of the HEXANE architecture as described above.

Note that the technology portfolio is different from the set of technologies required by the architecture. At a given vertex, the portfolio must contain all the architecture's required technologies, but it may contain additional 'surplus' technologies that are not used by the architecture. The reason for including the possibility of these surplus technologies is that they allow technologies that might be critical to the final mani-

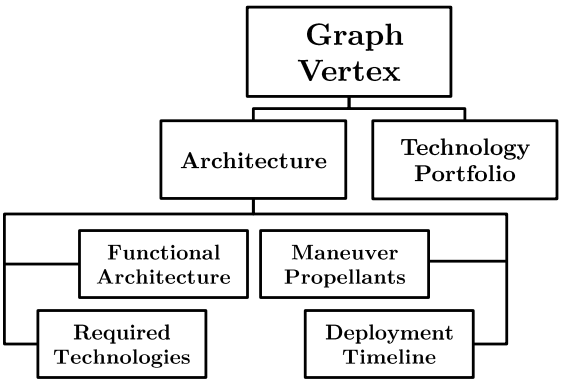


Figure 3-4: Hierarchy of components that define a vertex in the HEXANE tradespace graph.

festation of the system but not used in one or all of the intermediate architectures to be 'carried' down the evolutionary pathway.

The enumeration of graph vertices is the first step in the creation of the tradespace graph and is executed simply by enumerating all possible technology portfolios, determining which portfolios contain the required technologies for each architecture, and creating a vertex for each of these matches. Since there are $2^{13} = 8,192$ possible technology portfolios, for a tradespace of more than a few architectures, the number of vertices would increase beyond the computational capabilities of our analysis model, so to allow for many architectures to be studied we enforce a decision-making constraint that limits the number of surplus technologies in any vertex portfolio to two. This constraint is grounded in reality as decisions to invest in multiple technologies at one point in time and to leave them unused in subsequent systems are wasteful, unpopular, and unlikely to be consciously made. However, the ability to model a few such decisions is a useful facet of this model and helps to make the connectivity of our tradespace graph more robust.

3.2.2 Edge Definition

An edge in the graph represents a feasible incremental development to transform one architecture into another. The existence of an edge between two vertices is determined by the relationship between technology portfolios with the edge weight determined

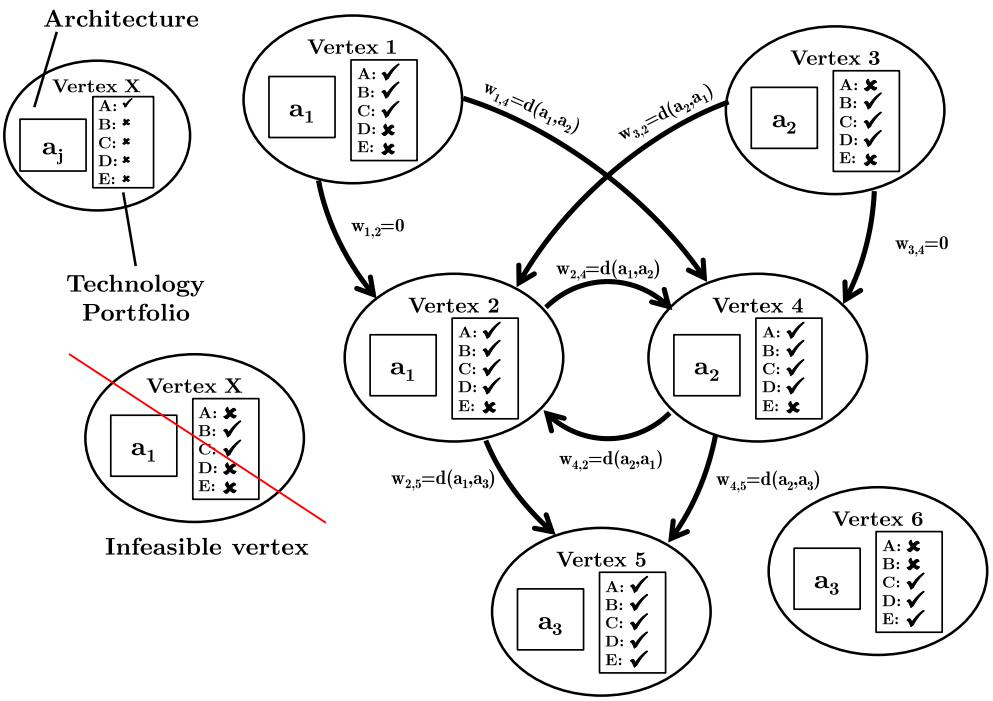


Figure 3-5: An example HEXANE tradespace graph with six vertices made up of three architectures and five possible technologies.

by architecture distance.

If two portfolios contain all the same technologies, then we place edges between the two vertices in both directions. If the two portfolios differ by a single technology, then we place a directed edge going from the vertex without the technology to the one with it. This single-technology-added criterion was selected as it models incremental decisions in the tradespace and allows the system architect to isolate and analyze the impact of a single technology on the architecture.

Figure 3-5 shows an example tradespace graph with three unique architectures: a_1, a_2, and a_3. Rather than the full suite of 13 technologies, we only include five 'dummy' technologies for illustration purposes, with a_1 requiring technologies A, B and C, a_2 requiring B, C and D, and a_3 requiring C, D and E.

In reality, for highly complex, low volume systems such as those modeled by the HEXANE framework, any changes to the architecture or technology portfolio would likely be made in large chunks or all at once for precisely the reason that large

system changes are so costly at the architectural level. However, the incremental development view afforded by decomposing this large jumps through the tradespace into single-technology jumps gives the decision maker a way to analyze the effects of each technology or architecture change individually and to prioritize them accordingly. In addition, an analysis of the incremental decision path allows the decision maker to gain an understanding of the contingency options available for responding to failures or system changes during 'long jumps' through the tradespace.

3.2.3 Architecture Distance

Once edges have been defined, the final step in the creation of the graph is the assignment of edge weight using the architecture distance metric. Since the cost of architecture transformation for the in-space transportation infrastructure is plagued by a number of modeling challenges, we have decided to use a low-fidelity proxy metric that aligns with the fidelity of the HEXANE architecture model to represent the cost of architecture transformation. This proxy focuses on the functional architecture, ignoring the other architecture components that we assume to have a negligible impact on architecture transformation cost relative to the functional architecture.

Though there are a number of options available, the distance metric used for this application is similar to the partition metric Day terms $B(Q, P)$ in [15]. A partition consists of a set of *objects* grouped into *blocks* such that each object is assigned to a block and each block has at least one object in it. If we define each of the habitation and propulsion functions as objects and elements of form as blocks then our definition of the functional architecture can be thought of as a partition of the set of system functions.

The metric that Day defines is the minimum number of *divisions*, *mergences* and *transfers* needed to transform one partition into another. Although these three operations have rigorous definitions given in [15], their intuitive meaning is simple. A division is the creation of a new partition block (element of form) by splitting out one set object (function) into its own block. A mergence is the deletion of a partition block by merging a block that contains one object with another block. Finally,

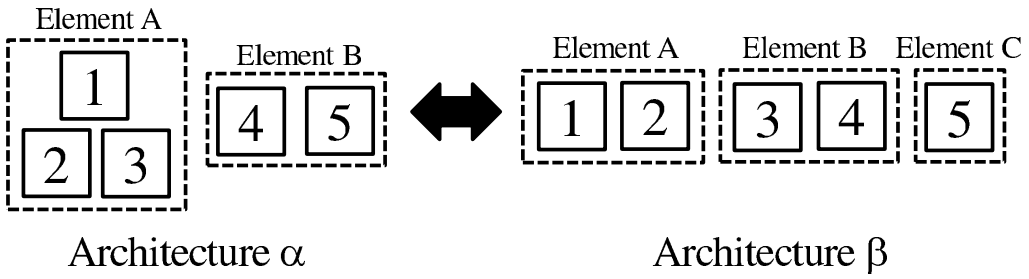


Figure 3-6: Two possible architectures α and β for five functions. Each solid box labeled with a number represents a function while each dashed box is an element of form, which is assigned one or more functions. To transform α to β Function 5 is split off to create Element C. Function 3 is then transferred to Element B from Element A. Since there are two operations the distance between α and β is two.

a transfer is the removal of an object from one partition block combined with its addition to a different block.

In terms of our present application this metric counts the number of functions that must be moved between elements of form in order to make two architectures identical. We refer the reader to Figure 3-6 for a graphical representation of the metric. This particular metric was chosen as a simple proxy for the cost of changing the functional architecture due to its simplicity and first order approximation of growth in technical complexity of the system. The other characteristics of the architecture were left out of the distance metric as their cost is already captured in the TDC metric (required technologies and propellant), or changing the characteristic affects the operation of the system much more than the development (deployment time line).

The action of creating additional elements of form or adding functions to existing elements can be viewed as growth in both formal complexity as defined by Boothroyd and Dewhurst [8] and structural complexity as defined by Sinha and de Weck [63]. Alternatively this growth in complexity can be viewed as large, tightly coupled and disruptive engineering changes [19]. When a function such as deep space habitation is transferred to a different element both the old and new element must undergo a redesign of interfaces and the addition or deletion of existing components and modules, driving up the technical complexity of the project. It has been shown that technical complexity is a major source of both cost and schedule overruns for space

systems [6, 12], which justifies our use of the present metric as a proxy for architecture switching cost.

The choice of using technology portfolio relationships to determine edge existence and functional architecture distance to define edge weight was made to separate these two sources of switching cost from one another, thereby avoiding the need to estimate a relative weighting factor to compare them directly. Ideally, such a weighting factor would be available to convert them into a 'dollar figure', however at this stage and fidelity of analysis such a conversion is beyond the scope and accuracy of the model, and we have chosen instead to separate these two cost components to the extent possible with the proposed framework.

3.2.4 Problem Types and Assumptions

The four basic problem statements discussed in Section 2.2 form the basis for our analysis of the tradespace graph applied to the in-space transportation infrastructure tradespace, however there are two additional constraints that we can place on the tradespace graph to enrich the information this framework can provide.

As discussed above, each vertex in the graph is made up of an architecture and a technology portfolio (see Figure 3-4) - with the only constraint between these two components being that the technology portfolio must contain the architecture's required technologies. Since these two components are separate, we can place constraints on either and analyze the architecture selection process when only one component is allowed to vary. Namely, we can hold the technology portfolio fixed across all vertices in the graph and allow only the architecture to change across edges, or we can hold the architecture fixed and allow the technology portfolio to vary from vertex to vertex. In fact, this second constraint will not provide any useful information since IMLEO is a function of the architecture only, so instead we will hold the functional architecture fixed and allow other architecture components to vary. We say that when we fix the functional architecture we are fixing the architecture family, since the functional architecture is the primary differentiating feature between architectures.

These problem constraints are motivated by Henderson and Clark's [29] frame-

work for defining innovation. Henderson and Clark characterize a system along two dimensions: core concepts, which we call technologies; and linkages between concepts and components, which we call the system architecture. If neither of these system dimensions change, only *incremental innovation* can be realized. If the core concepts change while the linkages remain constant (technological change without architecture change) they term this change *modular innovation*. If the opposite is true, technology is constant across a changing architecture, they use the term *architectural innovation*. Finally, if both core concepts and linkages are in flux there is *radical innovation*.

These problem constraints decompose each of the tradespace exploration problems of Chapter 2 into three distinct types based on which component, if either, is fixed across vertices in the HEXANE tradespace graph.

Graph Constraint Type 1: Both technology portfolio and functional architecture are free to vary across vertices in the graph.

Graph Constraint Type 2: The functional architecture (i.e. architecture family) is fixed for all vertices in the graph.

Graph Type 3: The technology portfolio is fixed for all vertices in the graph.

Each of these constraint types corresponds to different contexts in which a decision maker would wish to discover possible pathways through the tradespace. Graph Constraint Type 1 might apply in a context in which decision makers are looking towards long term planning and system evolution through multiple technology cycles. Here a decision maker would want to know what architecture changes need to be made at different stages to enable prospective technologies to be incorporated to maximum effect, and what initial architecture and technology decisions should be made to provide a number of flexible responses to uncertain future developments.

For Type 2, the system might have a mature, stable architecture when demands for improved performance or planned system evolution allows additional technology investments to be made and incorporated without major changes made to the archi-

tecture. One example from the space systems domain is the modernization of the Global Positioning System over the last two decades, with improved communications and processing technologies incorporated in all three segments without changing the underlying architecture [46].

A sample context for Type 3 is a system with an active architecture in development optimized for a particular technology portfolio. If this technology portfolio is externally disrupted or the architecture otherwise becomes infeasible, understanding how to modify the architecture without being able to add new technologies would be valuable to decision makers. The transition from Constellation to Orion could be thought of in this context, as decision makers had to create a new architecture using the same set of broad technologies, with tight constraints on heritage asset reuse [42].

Graph Considerations for Fixed Architecture Family Constraint

The edge existence and weight rules proposed above for the HEXANE tradespace graph assume that both the technology portfolio and functional architecture are free to vary from vertex to vertex. If we apply the edge existence and weight rules to Type 2 problems, then every edge that exists will have weight equal to zero since the distance $d(a_i, a_j)$ between any two architecture in the same architecture family is zero. With zero edge weight, every path through the tradespace would have the same total weight and could not be differentiated from any other.

For the special case of Type 2 graph constraints we retain the single-technology-added edge existence criterion and change only the edge weight rule. Our new rule is that if an edge exists its weight is equal to the cost of developing the technology that is added to the portfolio along the edge in terms of the TDC discussed above. Since the technology portfolio is the only component that changes across an edge, the technology development cost is the only switching cost incurred by the project (according to the model of the architecture change process we are using, in reality there would be a non-negligible cost to implement this technology into the architecture). Therefore, we assume that the cost of developing this new technology is an acceptable proxy for the cost of moving from one vertex to another.

Graph Constraint Type	Edge Existence	Edge Weight
Type 1	Single Technology Added	$d(a_i, a_j)$
Type 2	Single Technology Added	TDC of Technology Added
Type 3	$d(a_i, a_j) \leq \delta$	$d(a_i, a_j)$

Table 3.3: A summary of the HEXANE tradespace graph edge existence and edge weight rules for the three graph constraint types.

Graph Considerations for Fixed Technology Portfolio Constraint

If we apply the general edge existence rules to Type 3 constraints in which the technology portfolio is the same for all vertices, we can see that every pair of vertices in the graph would have edges between them in both directions. Such a graph would be meaningless, since the shortest path between two vertices would always be the path connecting them directly and the incremental development view would be lost.

For this special case, we change the edge existence rule to depend on the relationship between functional architectures rather than technology portfolios. Our rule is that an edge exists between two architectures a_i and a_j if the distance $d(a_i, a_j)$ between them is less than or equal to some cutoff value δ. The weight is then equal to the value of d. Note that if $d(a_i, a_j) = d(a_j, a_i)$, which is true for our proposed metric, then each edge is bi-directional.

This rule encodes the desire to view evolutions of the tradespace incrementally rather than in large 'jumps'. By limiting the distance between two connected architectures, we limit the magnitude of architectural change that can take place across any edge, thereby breaking up large architecture transformations into small steps.

Table 3.3 summarizes the edge existence and weight rules for each of the three problem types.

Problem Assumptions

It is important to note that several key assumptions in addition to those stated above have been made in order to reduce the complexity of the HEXANE tradespace exploration problem. First, although the development of a system from an initial state to a goal state would presumably take place on a temporal scale, the concept of

time and along with it the time value of assets has been deliberately left out of our analysis since in many cases the amount of time between events is highly uncertain. Additionally, by isolating steps through the tradespace from instants in time, we can analyze all the potential architecture and technology portfolio increments along a path rather than just those that would represent the physical manifestation of the system over time.

Second, we make the assumption that the development of technologies is deterministic. In other words, we assume that when a project invests in a technology it becomes available to be incorporated into the architecture. Viewing technologies as 'switches' rather than probabilistic events allows us to more closely examine the coupling between architecture and technology at the expense of abstracting the risk inherent in technology development.

A final assumption is that technology development and architecture development can be decomposed into two distinct processes. This decomposition allows us to isolate architectural change from technological change and to analyze the effects of each type of change separately.

3.3 Tradespace Exploration Results

In this section we present the results of the application of our tradespace exploration framework to the in-space transportation infrastructure for human exploration. We will analyze the tradespace of a 500-day exploration mission to the surface of Mars, shown in Figure 3-3. This tradespace is well suited to the application of our framework since architectures in it require substantial technology investment, architecture and technology decisions are tightly coupled, and the tradespace spans a wide variety of concepts, all of which enrich the analysis that the proposed framework can provide.

We will present four separate problem scenarios which span the range of tradespace exploration problem statements and graph constraint types discussed above. For the first two sets of results, we assume an existing initial architecture and seek to identify the optimal or set of optimal final architectures and intermediate evolutions along

with progressions of the technology portfolio (Problem Type 3 from Chapter 2). We analyze this scenario from the perspectives of both a fixed architecture family and a fixed technology portfolio in Sections 3.3.1 and 3.3.2 respectively.

The next two sets of results analyze the problem of selecting an initial architecture. In Section 3.3.3 we analyze the selection of the initial architecture alone without considering the future evolution (Problem Type 1), while in Section 3.3.4 we combine initial architecture selection with analyzing the evolution of the architecture and progression of the technology portfolio through the tradespace.

3.3.1 Fixed Architecture Family

For this analysis case, we consider a scenario in which an initial architecture has already been defined and the system is far enough along in the design lifecycle such that changes to the functional architecture are not feasible. The decision maker in this case might wish to analyze how future technology investment decisions can be made to improve system performance without changing the functional architecture. In addition to simply knowing what technologies can improve system performance, the decision maker would also be interested in how these technologies can be prioritized to ensure that limited resources are allocated to maximize IMLEO savings.

To begin we arbitrarily assign our initial architecture, $e^{(1)}$, to be the minimum TDC architecture along the TDC-IMLEO Pareto frontier. This architecture, which is similar to many architectures found in the low-TDC region of the tradespace splits habitation functions into many separate elements. Launch, deep space out and deep space in are assigned to one element; descent and surface to another; ascent to its own element; and then Earth re-entry to a fourth element. From a propulsion perspective each maneuver has its own separate stage. The technologies required for this architecture, which define the initial technology portfolio, $P^{(1)}$ are: in-space LOX-hydrogen, a Mars descent hypergol-fueled stage, a Mars ascent hypergol stage, solar electric propulsion for cargo predeployent, and aerocapture.

We then search the tradespace for other architectures with the same functional architecture as $e^{(1)}$, which defines the fixed architecture family that composes the

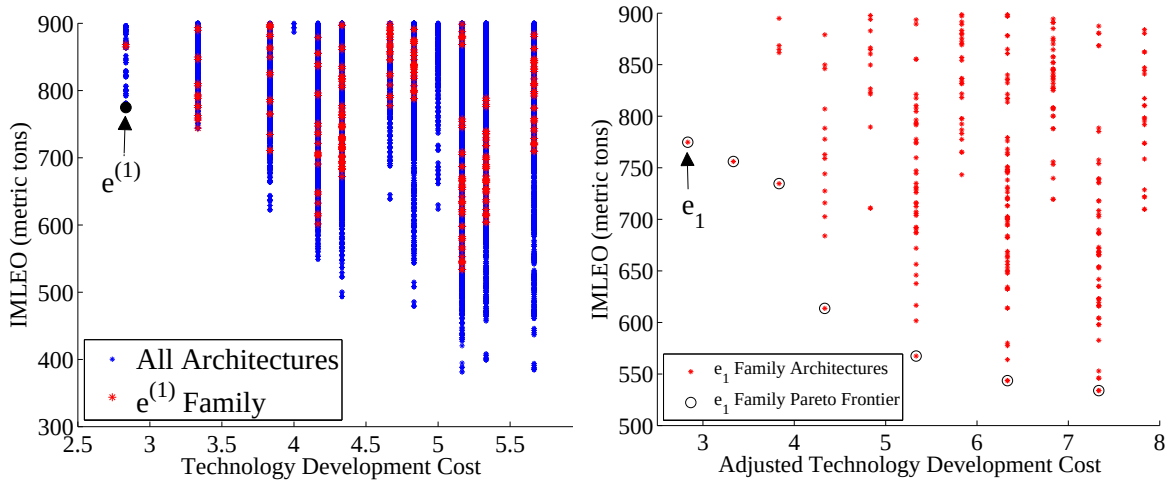


Figure 3-7: Left: The Mars surface 500 day tradespace with $e^{(1)}$ and its family shown. Right: The 288 $e^{(1)}$ family architectures shown with Adjusted TDC on x-axis.

tradespace graph. There are 288 such architectures in the tradespace, shown as the red markers in the left half of Figure 3-7. This family of architectures spans the range of TDC values but moves farther away from the Pareto front as more technologies are added. This trend indicates that the particular functional architecture present in $e^{(1)}$ is not well-suited to extract the maximum benefit out of more advanced technology portfolios, but that it is flexible enough to realize modest improvements in IMLEO with these added technologies. It is clear that the quality of options available to evolve the system is sensitive to the selection of the initial architecture, which highlights the need to make the initial selection with evolvability in mind.

Presumably, once the selection of $e^{(1)}$ has been made, an investment is made in all of its required technologies. We wish to reflect this fact in the tradespace analysis by favoring architectures that require technologies already in the portfolio. We achieve this bias by defining the *Adjusted TDC* of an architecture as the TDC of the union of that architecture's required technologies with $e^{(1)}$'s required technologies. In other words, we add a penalty to a_i for any technology that is required by $e^{(1)}$ but not by a_i since we assume this technology investment has already been made.

Thinking about this TDC adjustment graphically, we can see that its effect is to favor architectures that utilize technologies in which investments have already been

	Technology Added	**ΔIMLEO [mt]**
$e_1 \rightarrow e_2$	Boil-Off Control	19
$e_2 \rightarrow e_3$	Nuclear Thermal Propulsion	143
$e_3 \rightarrow e_4$	In-Situ Resource Utilization	45
$e_4 \rightarrow e_5$	Methane Ascent Engine	24
$e_5 \rightarrow e_6$	Hydrogen Descent Engine	10

Table 3.4: Optimal technology portfolio progression and IMLEO savings for the fixed functional architecture problem.

made by moving architectures which would carry these as surplus technologies farther along the TDC axis, thereby altering the Pareto frontier of this family. A view of the 288 architecture family is shown in the right half of Figure 3-7 with Adjusted TDC on the x-axis and IMLEO as usual on the y-axis.

At this point we build the tradespace graph as defined in the previous sections, ensuring that all technology portfolios in the graph contain the initial technology portfolio (i.e. $e^{(1)}$'s required technologies). Setting the constraint that no surplus technologies are allowed in the portfolio beyond those that would be required to contain the initial portfolio, we find the tradespace graph contains 288 vertices and $5,644$ edges.

Each of the architectures along the IMLEO-Adjusted TDC Pareto front shown in the right of Figure 3-7 is a candidate final architecture, $e^{(D)}$, since each represents a non-dominated trade between IMLEO savings and additional technology development cost. Rather than analyze each path here, we assume the decision maker wishes to find the optimal progression of the technology portfolio to minimize IMLEO for the selected architecture family. In this case the minimum IMLEO point in the $e^{(1)}$ family becomes the final vertex.

The optimal pathway from initial vertex to final vertex is shown in Figure 3-8 as the series of circle markers connected by a solid line. The technologies added to the portfolio at each step on the path are shown in Table 3.4. The fact that the pathway remains along the Pareto front of the tradespace is an ideal result since it means that all intermediate architectures maximize returns on technology investment, however this result is not general.

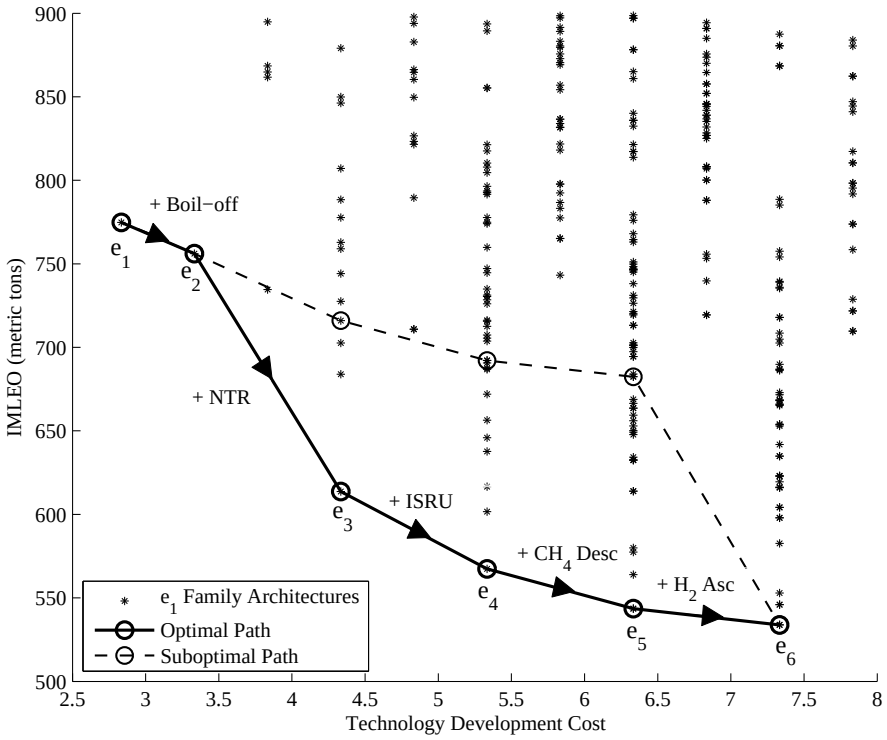


Figure 3-8: Evolution pathways with fixed functional architecture constraint from initial architecture to minimum IMLEO architecture in family.

We can contrast this optimal path with a different path shown as the dashed line in Figure 3-8. This second path enforces the constraint that Nuclear Thermal Propulsion is the final technology added to the portfolio (i.e. during the $e_5 \rightarrow e_6$ step). As is clear from the figure, this restriction moves the path far away from the Pareto front even though the starting and ending points are the same. This comparison suggests that the value of this type of analysis as applied to the fixed architecture family constraint is to inform the decision maker as to the best way to prioritize investments in technology in order to ensure that even if low priority technologies cannot be developed in time the resulting architecture-technology combination will retain the largest possible performance improvement.

Regarding the optimal path, it is clear from both Figure 3-8 and Table 3.4 that the majority of the improvements in IMLEO occur with the introduction of both boil-off control and nuclear thermal propulsion during the first two steps of the pathway, while the latter three technologies provide much smaller improvements. To get a clearer idea of how these incremental improvements contrast with the full potential of the technology, we can overlay the pathway on the tradespace of all possible architectures. This plot, shown in Figure 3-9, tells us that the functional architecture selected in $e^{(1)}$ is well suited to extract most of the benefit out of the inclusion of boil-off control and NTR since these architectures are close to the global Pareto front, but is a poor architecture with which to pursue the last three technologies.

Among these three technologies, the one with the largest impact on IMLEO across the whole tradespace is ISRU, which allows the mission to produce fuel and resources at the destination surface (rather than having to include them as payload in LEO) at the expense of a large fixed mass to account for the infrastructure necessary for this production capacity. This technology then favors architectures that reuse some fixed propulsion assets that have already been brought to Mars' orbit or surface for the return journey. The current functional architecture does the opposite by assigning the ascent, Mars departure and Earth arrival maneuvers to individual elements, each of which has a fixed engine and tank mass that offsets the benefits of ISRU propellant production.

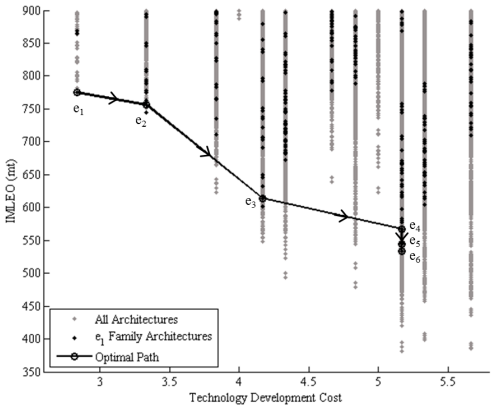


Figure 3-9: The optimal path shown in Figure 3-8 overlaid on the whole Mars surface 500 day mission tradespace.

3.3.2 Fixed Technology Portfolio

In this second set of results, we use the tradespace graph framework to analyze architecture flexibility for a case in which an exogenous project change disrupts system development. Let us consider a scenario in which an initial architecture selection already exists with a corresponding investment made in that architecture's required technologies. After some period of development, we will assume that an exogenous event, such as technology development failure or shifts in stakeholder goals, causes the technology portfolio to change, making the current functional architecture either infeasible or suboptimal. We wish to use the tradespace graph framework to determine how the architecture can be incrementally transformed to maximize performance from this disrupted technology portfolio.

For this scenario we set our initial architecture to be the highest-performing architecture in the model, which is the minimum IMLEO point on the Pareto frontier in Figure 3-3. The functional architecture of this selection from a habitation perspective is 'monolithic', that is all functions except Earth re-entry are assigned to a single habitat. From a propulsion perspective, Mars descent and ascent burns share a single stage, with the same true for Mars arrival and departure maneuvers. This architecture requires both NTR and ISRU as well as Boil-off control, Solar Electric Propulsion and Aerocapture. The descent and ascent propellant technologies are both

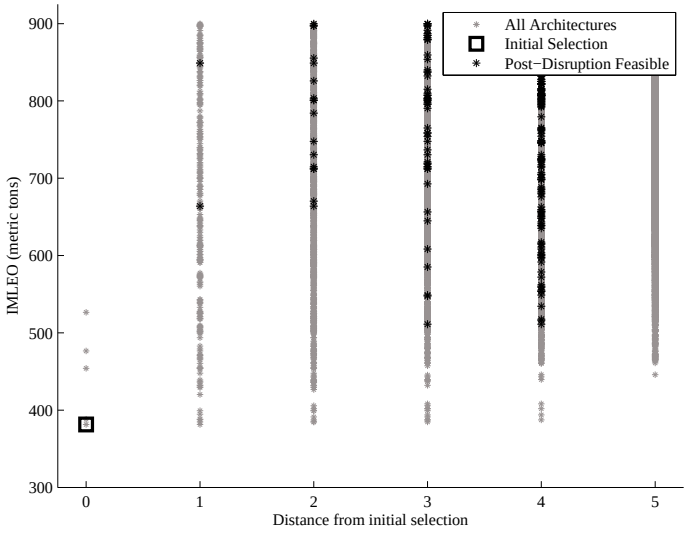


Figure 3-10: The HEXANE 500 days Mars surface tradespace plotted with IMLEO on the y-axis and distance from the initial functional architecture on the x-axis. Black points are feasible architectures with post-disruption technologies.

methane-based.

As part of our problem scenario we assume that the exogenous technology disruption causes methane to be replaced by hydrogen as the descent propellant technology and by hypergolic propellant for the ascent technology. This post-disruption set of technologies becomes the fixed technology portfolio, $P^{(k)}$ for all k in the evolution.

Figure 3-10 shows the entire Mars surface 500 day tradespace, with the black points representing those architectures with that are feasible given this new disrupted technology portfolio. This view of the tradespace is plotted with the distance of each architecture from the initial selection on the x-axis, since architecture distance is the only source of switching cost for this scenario. The Pareto front of the feasible points in this plot trades IMLEO savings against the cost of changing the architecture in terms of the magnitude of change required in the functional architecture.

The disruption to the technology portfolio described above will force the architecture to change, as the required technologies for the initial selection are no longer available. Ideally the functional architecture would remain constant across this change. In fact technology and functional architecture are so tightly coupled in this region of

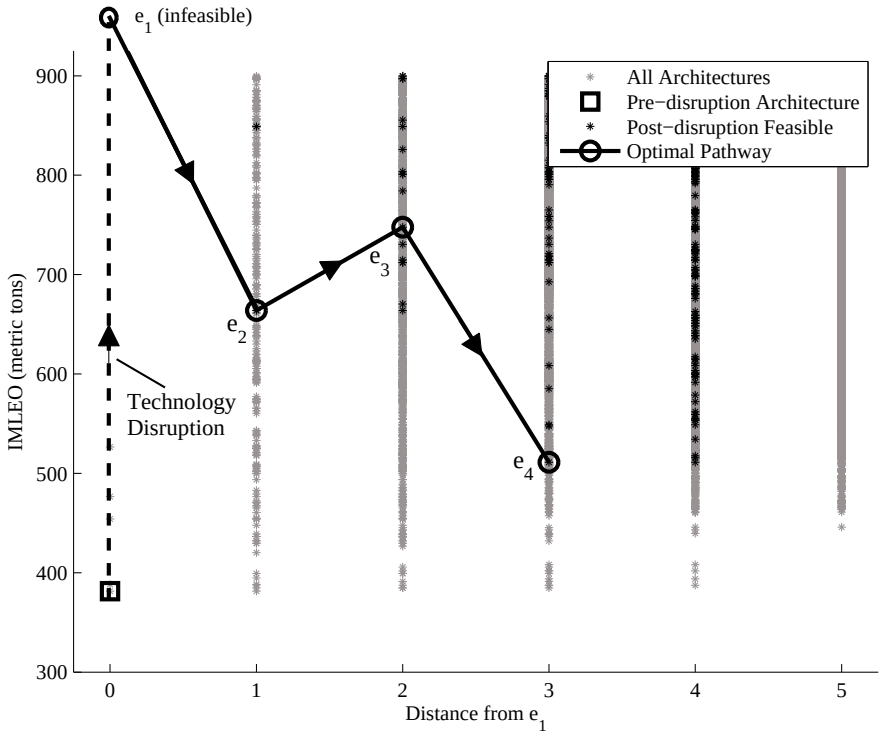


Figure 3-11: Architecture evolution after a technology disruption resulting in an infeasible initial architecture. There are 255 feasible architectures and 5,137 edges in the graph representation of the tradespace.

the tradespace that the shift in technology makes the initial functional architecture infeasible. Therefore, for this case we assign $e^{(1)}$ to be an unspecified infeasible architecture with a functional architecture that is the same as that of the disrupted initial selection.

To define the tradespace graph we need to specify the edge existence cut-off, δ, as described in the section above since the technology portfolio is fixed for all vertices. For this case we use a cut-off of $\delta = 1$, which means that two vertices are connected only if the distance between their functional architectures is zero or one. We then select candidate final architectures based on location on or proximity to the Pareto frontier in Figure 3-10. Again, we will only show one example and will select the minimum IMLEO point in the feasible set, as this is the point a decision maker would analyze to determine how to recover the maximum possible performance following the technology disruption.

Step	Change to Functional Architecture
$e_1 \rightarrow e_2$	Divide ascent/descent propulsion into separate stages.
$e_2 \rightarrow e_3$	Divide deep-space return habitation into separate element.
$e_3 \rightarrow e_4$	Divide ascent habitation into separate element.

Table 3.5: Changes to the functional architecture at each step in the evolution for fixed technology portfolio disruption.

Figure 3-11 shows the technology disruption process as well as the pathway that is found to change the functional architecture in order to reduce IMLEO. The tradespace shown is plotted with respect to IMLEO and distance from the functional architecture of e_1. The gray markers represent all architectures in the tradespace, while the black markers are the ones that are feasible with the post-disruption technologies, which comprise the 255 vertices in the tradespace graph. The final vertex is selected to be the minimum IMLEO architecture within this new feasible set, and is labeled e_4 in Figure 3-11.

The changes to the functional architecture made at each step in the evolution are shown in Table 3.5. We see that at each step functions are broken off from monolithic elements and assigned to specialized elements. This disaggregation of the architecture occurs because the coupling between propellant and ISRU is no longer strong enough to efficiently support monolithic elements after the technology disruption.

This type of information delivers value to a decision maker who is analyzing the robustness of his or her architecture to disruptions during the system development stage. The analysis above indicates that in this case technology and architecture are very tightly coupled and the system is not robust to technology disruption. However, should a disruption occur, the indicated path tells the decision maker what his or her contingency options could be and what the relationship is between spending additional resources to change the architecture and improving performance.

3.3.3 Initial Architecture Selection

In this third analysis case, we consider a scenario in which an initial architecture selection has not been made for the in-space transportation system, and the decision

maker wishes to analyze how different options compare against one another to aid in the initial selection decision.

Ideally, the decision maker would like to select a high-performing initial architecture that is flexible to both exogenous changes to system requirements, budget, and operating context and to internal issues such as technology development failure or discovery of design flaws. In the context of the transportation infrastructure for Mars exploration, such negative 'development disturbances' are not only possible but also likely given the long development period of such large, complex systems that could overlap multiple administrations, see drastic changes in national economic priorities, and be exposed to a changing geopolitical environment. Also possible are positive disturbances such as the unexpected development of a useful technology or increases in project budget that could push the system in the direction of lower cost or higher performance. In either case decision makers would want the ability to make incremental, relatively inexpensive changes to the architecture to respond to these disturbances in a flexible manner.

As a first approximation for the flexibility or robustness of an architecture, we can try to understand how many other architectures are in the immediate neighborhood of the architecture candidate. Although this concept does not take into account the type of disturbances that the architecture is flexible to (e.g. technology failure, budget changes), it does give a rough estimate for how large a region of the tradespace the architecture can access by making relatively small changes.

To define the flexibility of an architecture we will use the tradespace graph, but rather than relying on the shortest path formulation we will simply count the number of neighbors for each candidate architecture and use that as a proxy for flexibility. For this problem we are working with an unconstrained graph (Type 1) with edges and weights defined according to Table 3.3. For this particular graph we will ensure a one-to-one mapping of architectures to vertices by constraining the technology to have no unused technologies.

In this study, we say that a_j is a neighbor of a_i if there is an edge from the a_i vertex to the a_j vertex with weight less than or equal to one. However, the decision maker

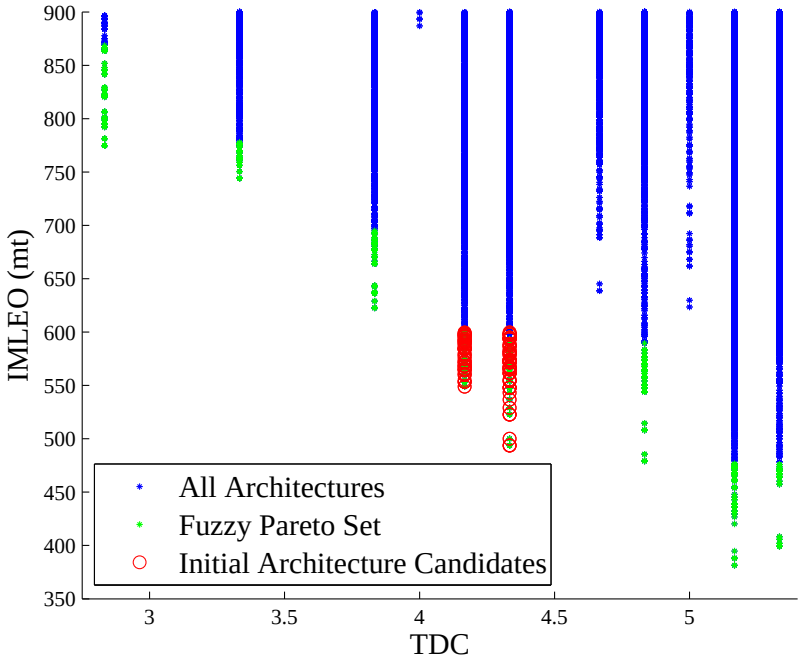


Figure 3-12: Candidate initial architectures for initial architecture selection problem.

is likely not interested only in the raw number of neighbors architecture a_i has, but rather the number of 'good' neighbors that are easily accessible. We define a 'good' neighbor then as one that is contained in the 1% Fuzzy Pareto set (i.e. one that is close to the performance-cost Pareto frontier). Further, the *Normalized Flexibility* of an architecture is the number of 'good' neighbors divided by the maximum number of 'good' neighbors of any candidate initial architecture.

For this scenario, we will assume that decision makers are budget-constrained to select an initial architecture in the mid-TDC range of $4.0 \leq TDC \leq 4.5$. To ensure that the architecture maximizes performance, we also only look at candidate initial architectures with $IMLEO < 600$. Figure 3-12 shows the Mars surface 500-day tradespace with candidate initial architectures shown in red and neighbor candidates (i.e. the 1% Fuzzy Pareto set) shown in green.

For each of these initial architecture candidates, we want to understand the trade between the performance of the architecture and its flexibility to future changes. Analyzing this trade in detail and selecting a 'best' initial architecture is beyond

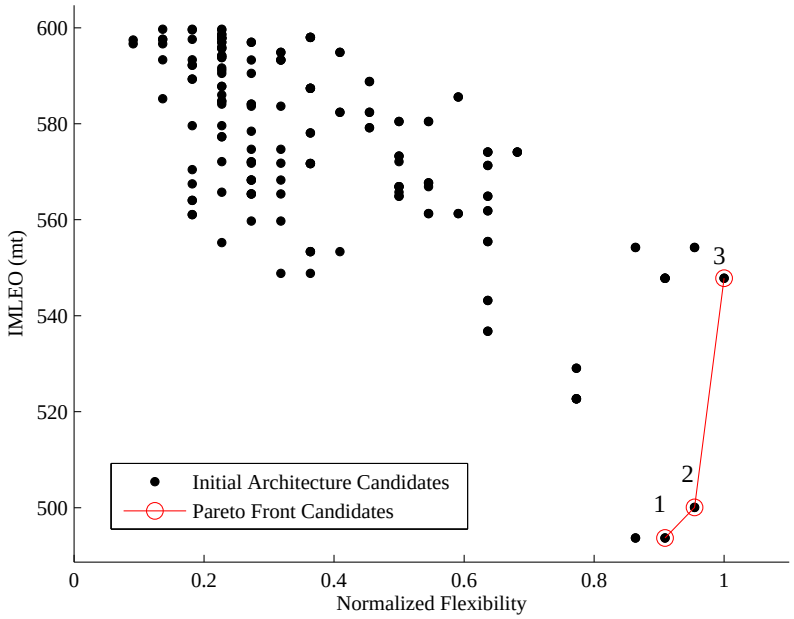


Figure 3-13: Initial architecture candidates plotted with IMLEO performance on the y-axis and normalized flexibility on the x-axis.

the scope of this work and of the fidelity of this model, however if we scatter plot architecture IMLEO against normalized flexibility we can visualize the rough trends in this trade.

Figure 3-13 shows this plot of the initial architecture candidates according to IMLEO and flexibility. The Pareto front of this set is also plotted, indicating that there are three Pareto optimal architectures with similar flexibility scores. As a reference, the maximum number of connections (i.e. the number of neighbors of the architecture with normalized flexibility equal to one) is 22.

Under the assumptions of our architecture and tradespace models, the three Pareto optimal architectures labeled 1 through 3 are the initial feasible options for trading between performance and development flexibility as we have defined both metrics. As an example of how these initial architectures are connected to other architectures in the fuzzy Pareto set, we plot the connections of Pareto architecture 1 on the original TDC-IMLEO axes in Figure 3-14. We can see that this initial architecture (labeled with the blue circl)e is connected to both higher performance architectures

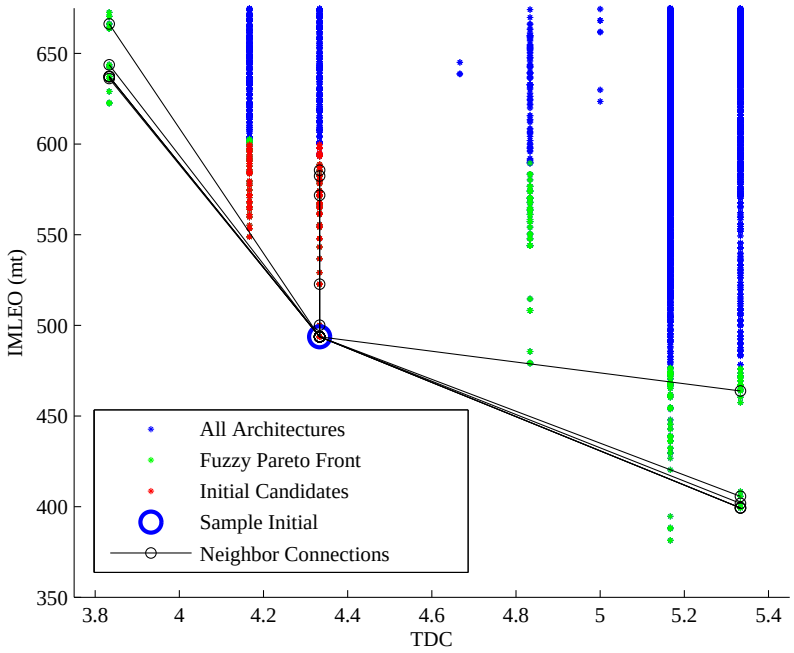


Figure 3-14: Neighbors of IMLEO-Flexibility Pareto frontier architecture 1 plotted on IMLEO-TDC tradespace.

(indicating extensibility under positive exogenous impacts) and to lower development cost architectures (indicating flexibility to some types of negative impacts).

3.3.4 Initial Architecture and Evolution Selection

In this final analysis case, we will analyze a similar initial architecture selection scenario as Section 3.3.3, but in this case will make the selection with a known final architecture in mind. The primary assumption underlying such a scenario is that the system designer must pick an architecture for the system under development, but that this architecture is not necessarily the ideal architecture in the long run. This discrepancy could be a result of initial development cost constraints which force the system designer to select a lower-performance architecture that can be modified over time to the ideal high-performance architecture.

An example of such a decision scenario can be seen in NASA's plan for the evolution of the Space Launch System (SLS) [42]. The selection of the current 70 metric

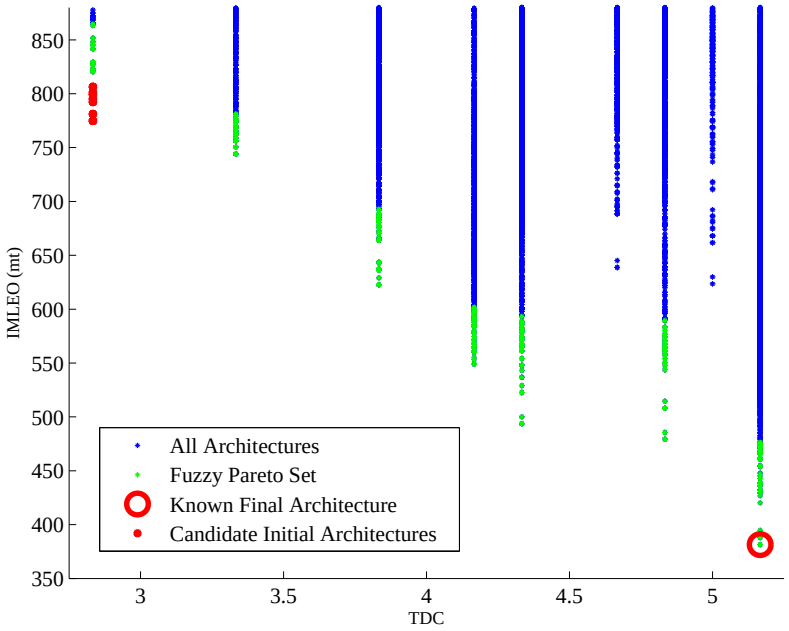


Figure 3-15: Candidate initial architectures and known final architecture plotted along IMLEO and TDC.

ton launch vehicle as the initial architecture is a function of budget, reuse, and schedule constraints. NASA's plan calls for the evolution of this initial architecture into a 130 metric ton launch vehicle after initial development, which represents stakeholders' ideal performance preferences.

Ideally the highest performing initial architecture candidate would also be the one closest in terms of switching cost to the goal final architecture, however this is not always the case. For example NASA may have had a choice between a 70 mt vehicle with a projected $2B development cost to reach the 130 mt version, and a 75 mt vehicle with a $4B cost to develop the follow-on 130 mt vehicle. With this example in mind we can see that the need to select an architecture that can most easily evolve into the final architecture must be traded against the performance of the initial architecture itself. Since this trade for the in-space transportation infrastructure would take into account a variety of technical, social, and political factors far beyond the scope of our tradespace model, our goal here is simply to present the best options to be traded against one another as input for the decision maker.

To find these options, we once again define a subset of candidate architectures for the initial selection. For the results presented in this section we define the set as all architectures with $TDC < 3$ and $IMLEO < 810mt$, which is the set of 36 architectures shown as red points in Figure 3-15. Although these architectures span a variety of required technology sets and function to form mappings, in general we see that they all share basic technologies such as aerocapture and solar electric propulsion, and that the habitation and propulsion functions are distributed across many separate elements of form.

To reduce the size of the graph and computational time, we consider only architectures in the Fuzzy Pareto Front to be feasible. For our application our fuzzy set consists of the 2% of the tradespace, shown as the green points in Figure 3-15. This reduction of the tradespace also enforces an implicit constraint that the sequence of architectures which define the path through the tradespace should themselves be acceptable architectures, though not necessarily independently optimal.

The known final architecture, shown as the black circle in Figure 3-15 is set to be the minimum IMLEO architecture in the tradespace. This selection is a technology rich architecture incorporating nuclear thermal propulsion, boil-off control, and in-situ utilization. In terms of the functional architecture, we note that it is a monolithic mapping with many functions assigned to a single element of form, especially with respect to habitation.

The tradespace graph is built using the unconstrained (Type 1) edge rules with at most two unused technologies allowed in the technology portfolio of each vertex. There are 723 unique architectures in the analysis set, which map to 4991 vertices in the graph when combined with feasible technology portfolios. These vertices are connected by 1.56 million edges, which must be searched using the shortest path algorithm.

Using Yen's algorithm, we find the minimum average IMLEO shortest path from each of the 36 initial candidate architectures to the fixed final architecture. The length of the path serves as a proxy for the total cost of transforming the candidate initial architecture to the desired architecture in incremental steps, which can be traded

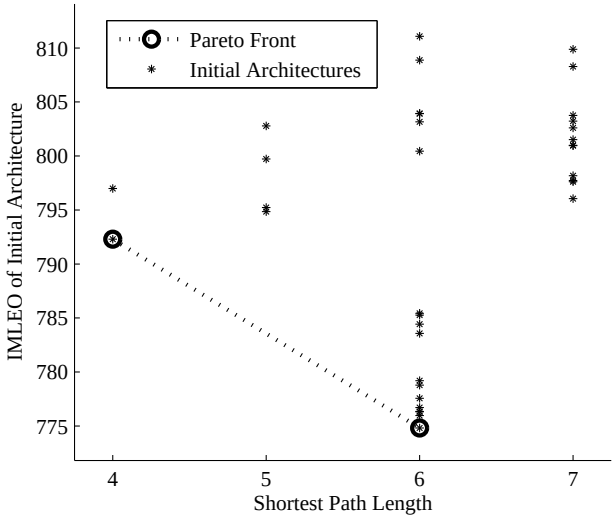


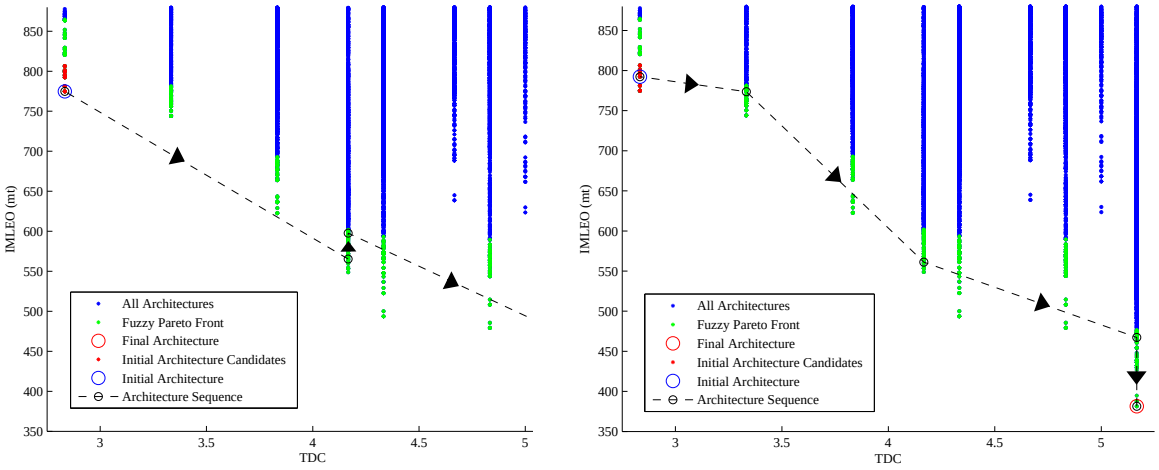
Figure 3-16: Plot of the shortest path length from each candidate architecture to the final architecture versus IMLEO of the initial candidate.

against the immediate performance of the initial architecture itself.

The results of these calculations are shown in Figure 3-16 as the length of each path plotted against the IMLEO of the initial architecture. From this figure we see that there are two Pareto optimal choices for the initial architecture in terms of these two metrics, which represent the non-dominated options that can be compared to one another within the bounds of this architecture and decision model.

To examine each of these Pareto frontier choices more closely we examine the path through the tradespace from the two initial architectures. Figure 3-17 shows plots of each path on the familiar IMLEO-TDC axes. The candidate initial architectures are shown as the red points in both figures, with the selected initial architecture marked by the blue circle. Figure 3-17(a) shows the path for the minimum IMLEO initial architecture, while 3-17(b) shows the path for the minimum length initial architecture. Visually these two plots indicate that the minimum IMLEO initial architecture takes an inconsistent route through the tradespace, with two very 'small' hops along the path (with one actually increasing IMLEO), while the minimum path length initial architecture takes a more direct and gradual route.

We can also note that the path in (a) has one additional hop relative to the path



(a) Minimum IMLEO initial

(b) Minimum path length initial

Figure 3-17: Plots of the optimal paths to the final architecture for two Pareto optimal initial candidate architectures.

in (b), which causes the length of (a) to be two units greater than the length of (b). This difference in path length can be traced to the habitation functional architecture of each of the initial candidates. For the minimum IMLEO candidate, the habitation architecture is fairly distributed with Mars descent, Mars surface, and deep space return split into three separate habitats. For the minimum path architecture, all three of these functions are assigned to the same habitat. Since the final functional architecture assigns all habitat elements reentry to the same monolithic habitat, this means that more architecture modifications are required for the minimum IMLEO architecture, corresponding to a higher cumulative switching cost and a longer path through the tradespace.

With this type of analysis available, the decision maker could then determine if the sacrifice of initial architecture performance by deciding to more closely modeling the relevant difference between the two architectures. Once again, this type of analysis would be unavailable using the traditional view of the architecture tradespace as a collection of singular points. It is only when these points are connected to one another based on the relationships modeled in this chapter that the analysis of architecture change and decision flexibility can be realized.

Chapter 4

Space Communications Networks: SCaN Static Tradespace Graph

In this chapter, we will apply the generic tradespace exploration framework presented in Chapter 2 to a tradespace of space-based communications networks. We will begin with a short overview of the SCaN program and the model used to define the tradespace that we will be studying. In the following section we will present a time-independent application of our exploration framework to the SCaN tradespace to define the static tradespace graph. In the final section of this chapter we will describe and analyze in detail several sets of results drawn from this framework.

4.1 Background

The tradespace that forms the core data set in this chapter and the next is a tradespace of architectures for NASA's Space Communication and Navigation (SCaN) program. The SCaN program is a large program office within NASA that manages and coordinates several NASA communications networks that play a critical role in the operation of a variety of space-based systems. The SCaN tradespace model is being developed as part of an ongoing project at MIT with the goal of enumerating and evaluating potential future SCaN architectures.

The goal in applying our exploration model to the SCaN tradespace is to organize

the large amount of information contained in the tradespace into a traceable set of trades and recommendations that can be used as analysis support tools in the process of determining NASA's future SCaN architecture. In order to give some context to our analysis, as well as to present the fundamental aspects of the tradespace model that are key to our development of the tradespace graph, we present an overview of the SCaN program and a basic introduction to the SCaN tradespace model below.

4.1.1 SCaN Program Overview

NASA's Space Communication and Navigation (SCaN) program was created in 2006 to centralize the management and systems engineering responsibilities of NASA's communications networks. The component networks of SCaN have evolved from a long series of both ground-based and space-based networks as communications technologies and user needs have developed over time. For a thorough historical treatment of NASA's spaceflight communications networks, refer to [67].

The SCaN program is currently comprised of three independent networks: the Near-Earth Network (NEN), the Space Network (SN), and the Deep Space Network (DSN)[44]. The NEN is a collection of 12 ground stations which communicate directly with users on orbit. Ground stations are comprised of both NASA-owned and commercial facilities, with services provided to a variety of NASA and non-NASA users. The SN consists of a constellation of Tracking Data Relay Satellites (TDRS) in geosynchronous orbit (GEO) and two dedicated ground stations. The TDRS constellation, with satellites in three orbital slots, acts to relay data between the ground and customers in low Earth orbit (LEO) with 100% coverage. Finally, the DSN is a network of three ground stations that communicates with missions beyond Earth orbit.

The primary goal of the SCaN program is to develop these three communications networks over time to ensure that the communications infrastructure exists to meet both the current and future communications needs of human and robotic missions across the Solar System [54]. As new communications technologies mature, as user needs change with the launch or retirement of missions, and as existing SCaN assets

reach their end of life or become obsolete, the existing networks must evolve in synergy with one another into a unified network that provides the highest feasible data rates with modern technologies and protocols to all users [55, 43].

To understand the options available for the future of NASA's communications infrastructure and to plan the evolution of these networks, architectural studies are already under way to model and explore the tradespace of potential future architectures. One such study is a cooperative effort between Goddard Space Flight Center (GSFC) and MIT to develop a tradespace enumeration and evaluation tool for the future Space Network architecture [54, 55]. As the SN is currently being upgraded with the launch of the third generation of TDRS satellites (TDRS K, L and M), the scope of this study is concerned with the mid-to-long-term evolution of the network to reduce cost, improve performance, and maintain flexibility for continued development. From this point forward, we will refer to this study and architecture enumeration tool as the *SCaN Tradespace Model*.

4.1.2 SCaN Tradespace Model Background

The SCaN Tradespace Model developed in [54, 55] is a computational tool based on the VASSAR (Value Assessment of System Architectures using Rules) framework developed by Dr. Daniel Selva at MIT [57, 56]. The core of this framework is the implementation of a rule-based expert system (RBES) to assess the value of an architecture based on the matching of capabilities with stakeholder requirements. As discussed in Section 1.4.5, the primary strength of an RBES framework is the ability to encode and apply domain-specific expert knowledge in an easy-to-use and computationally efficient manner.

In addition to the assessment of stakeholder satisfaction, the SCaN Tradespace Model also uses an RBES to evaluate architecture cost and to enumerate the tradespace of feasible Space Network architectures. However, before discussing the tradespace enumeration and architecture evaluation stages it is important to develop an understanding of the model used to represent an individual architecture in the tradespace.

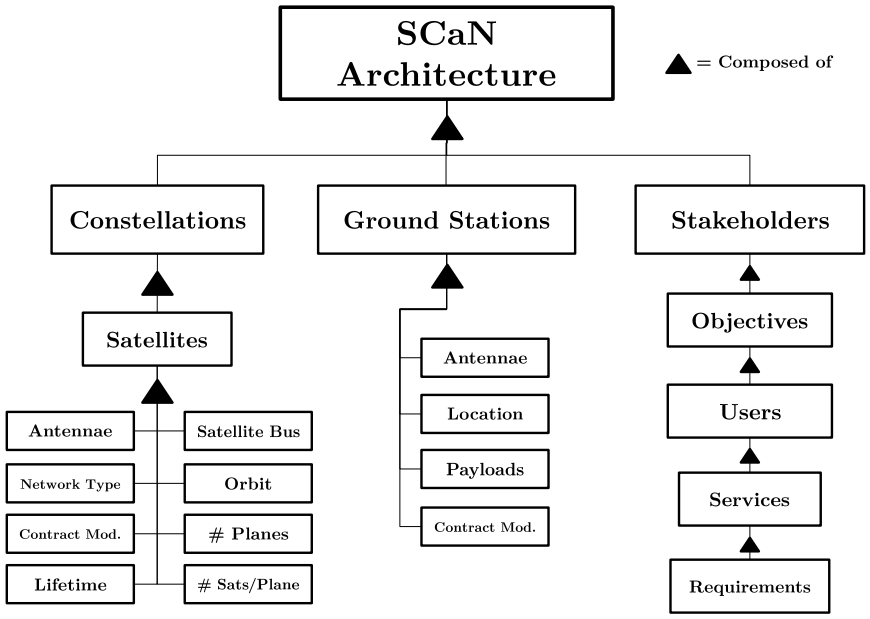


Figure 4-1: Hierarchy of elements in a SCaN Model architecture.

Architecture Model

An architecture in the SCaN tradespace is modeled as a collection of space assets, ground assets, and stakeholders that together determine the capabilities, cost, and benefit of the architecture. The graphical decomposition of elements that define an architecture is shown in Figure 4-1.

Space assets consist of **constellations** at the first level of decomposition. A constellation is a collection of one or more satellites that execute the core data communications function of the Space Network by relaying data between in-space users and ground stations. An important assumption made in this architecture model is that all satellites in a given constellation are identical, both in terms of design and orbit. For this reason, we characterize a constellation using properties normally used to characterize an individual spacecraft, and make the constellation, rather than the satellite, the fundamental building block of our tradespace exploration framework.

At the next level of decomposition, a constellation itself is characterized by a multitude of properties that are used within the RBES to calculate performance and cost. There are too many properties to describe in detail here, so we have highlighted

the essential ones that play a major role in our comparison of one architecture to another in building the tradespace graph. These key constellation properties are:

- *Communications Payloads* - The primary value function of a constellation is the relaying of data through the communications network, which is realized via the transmission of electromagnetic waves over a variety of frequency bands. A payload is characterized by the frequency band in which it operates, and consists of the transponder and supporting electronics to execute satellite-to-satellite or satellite-to-ground communication. Possible payloads include S-band, X-band, Ka-band, Ku-band, optical, and combinations of those frequencies packaged into a single payload.

- *Antennae* - Provides the means by which signals are received and transmitted by the spacecraft. Antennae may be single access or multiple access (able to service one or multiple users at a given time, respectively), and are sized during the architecture evaluation stage to close the communications link budget.

- *Network Type* - Characterizes the network architecture of the constellation, or the way in which data is sent from one node in the network to another. Possible network types include bent-pipe (the current TDRS network architecture), circuit-switched, and packet-switched.

- *Contract Modality* - Characterizes the owner of the payloads and the owner of the satellites in the constellation. In the traditional procurement contract the entire satellite is purchased, launched, and operated by NASA. In a hosted-payload contract, only the payload is owned by NASA. The payload and required antennae are 'hosted', in return for a service fee, on a commercially-owned satellite bus which contains other value-delivering payloads. A third option is a completely commercial contract, in which NASA 'rents' capacity on communications links provided by commercially owned payloads on a commercially owned bus.

- *Lifetime* - The planned time period for which satellites in the constellation can

perform their communications functions. After this time period, the satellites lose their operational capabilities and are either decommissioned or deorbited.

- *Satellite Bus* - The underlying infrastructure of each satellite in the constellation. The bus includes that non-payload subsystems, such as power and attitude control, that provide a platform for the communications payload and antennae.

- *Orbit* - The constellation orbit is characterized by its semi-major axis, inclination, and eccentricity. Broadly speaking, orbits are divided into orbit types of geostationary orbit (GEO), medium earth orbit (MEO), low earth orbits (LEO), and highly elliptical orbit (HEO). Satellites in the constellation have additional properties - right ascension of the ascending node and argument of perigee - to specify their location in the constellation for contact purposes.

- *Number of Planes* - The number of different orbital planes in which a constellation has satellites.

- *Satellites per Plane* - The number of satellites a constellation has in each of the orbital planes.

Ground assets consist of **ground stations** which provide the means through which data is passed from the space segment to terrestrial users and vice versa. Ground stations are characterized by their payloads, which determine the frequency bands over which the ground station can communicate with on-orbit assets; their location, which determines when and how often data can be passed between the space and ground segments; their antennae, which determines the gain required on the space asset side to close the link budget; and the contract modality, which characterizes how the cost of construction and operations is contracted.

It is important to note that although the SCaN Model has been created so that ground stations are a potential architecture variable, in the analysis that follows in this chapter we will use a fixed set of ground stations (White Sands and Guam) to represent the ground assets of all architectures being analyzed since these ground sta-

tions are predicted to remain the backbone of the ground segment for the foreseeable future.

Finally, the set of **stakeholders** of a particular architecture is what drives the performance, or benefit, of an architecture. Stakeholders such as NASA, NOAA, and USGS are given relative weights in the architecture, with their individual satisfactions computed in the evaluation stage and combined to determine the architecture benefit. To compute satisfaction, stakeholders are modeled as having a number of high-level objectives (e.g. earth observation, weather, and human spaceflight) that they wish to achieve.

Going a level further, objectives are defined by a set of users, which are on-orbit missions that wish to make use of the Space Network to move data between the ground and the spacecraft. Each user requires one or more services from the network, which characterize the desired type of data, data rate, and quality of service metrics of the communications link. Finally, these services are divided into requirements that the architecture attempts to meet. In the evaluation stage, the architecture is optimized so that as many requirements as possible can be met and stakeholder satisfaction can be maximized.

The inclusion of stakeholders, users, and requirements allows the evolution of stakeholder needs over time to be modeled in the tradespace. Since the analysis time frame of this model is 10 to 20 years in the future, the ability to capture how differing sets of stakeholder needs affects optimal architecture decisions is a major strength of the SCaN Tradespace Model. In order to compare two architecture directly to one another they should have the same set of requirements and relative weights, however it is a strength of the proposed tradespace exploration framework that the evolution of stakeholder needs can be encoded in the criteria that determines edges between architectures in the tradespace graph.

Architecture Enumeration

Enumeration of the feasible tradespace is controlled by a set of *decisions*, which determine an architecture by specifying each of the modeled properties described above. A

Table 4.1: Primary tradespace enumeration decisions

Decision	**SAP Class**	**Explanation**
Payloads	Down-Selecting	Which payloads are present in the architecture.
Payload Allocation	Partitioning	How payloads are assigned to constellations.
Contract Modality	Assigning	The contract modality of each constellation.
Orbit	Assigning	The orbit of each constellation.
Network Type	Assigning	The network type of each constellation.
Lifetime	Assigning	The lifetime of each constellation.
Number of Planes	Assigning	The number of planes in each constellation.
Number of Satellites per Plane	Assigning	The number of satellites per plane in each constellation.

rule-based engine is then used to permute each of these decisions and prune infeasible sets of decisions based on logical and encoded knowledge-based constraints. Each of the decisions that define an architecture falls into one of the five classes of System Architecting Problems discussed in Section 1.4. The abstraction of these decisions into System Architecting Problem classes allows the model to use domain-independent knowledge developed in [56] and already encoded in an RBES to efficiently enumerate the possible decision options.

Given the large number of properties mentioned above and the many characteristics not discussed here, the set of potential decisions is large, and thus it is computationally infeasible to enumerate the entire tradespace of architectures across all decisions. For this reason, and to make the tradespace accessible to human analysis, we only consider a subset of possible decisions for each tradespace analyzed, leaving the reminder of the decisions constrained as fixed parameters across all architectures.

The focus of our analysis is on this relatively small number of decisions that drive the dominant architectural features of the communications network. These decisions are given in Table 4.1, with their System Architecting Problem class and a short explanation also provided.

Architecture Evaluation Metrics

Once the set of architecture in the tradespace has been enumerated, the next step is to evaluate each architecture to compute metrics by which architectures can be compared against one another. As mentioned above, the two primary metrics of concern to our analysis are benefit and cost.

The evaluation of stakeholder benefit is computed using a detailed performance model as described in [54]. In short, this performance model first determines the topology of the network as a function of time by propagating the locations of relay spacecraft, user spacecraft (which are contained in a predefined database) and ground station, and determining at what times viable communications links exist between nodes.

Next, the ideal allocation of relay satellite resources is determined based on a prioritized list of users and services. For this purpose, a rule-based scheduling algorithm is used to schedule contacts between relay satellites and user spacecraft given the known user data transfer requirements. Based on the number of successfully scheduled contacts relative to the needs of each mission, stakeholder satisfaction is determined and aggregated to compute overall architecture benefit.

Whereas benefit is the only performance metric we are concerned with in the analysis, there are several components of cost that will be important to our tradespace exploration model. At the highest level, cost can be divided into *recurring cost* and *non-recurring cost*. As their names suggest, recurring costs are those that must be continually paid over the lifetime of the architecture, while non-recurring costs are 'up front' costs that are paid only a single time.

As mentioned previously, our analysis is concerned with the evolution of the in-space segment of the Space Network, so we incorporate only the constellation-related costs into our tradespace exploration model. Relevant costs with recurring and non-recurring components are shown in Table 4.1.2.

The cost model used to compute these and other costs consists of an additional set of rules that encode different parametric cost models and databases to estimate

Table 4.2: Table of relevant constellation-related costs

Non-Recurring Costs	**Recurring Costs**
Antenna Development and Manufacture	Operations
Satellite Bus Development and Manufacture	Service Fee
Payload Development and Manufacture	
Launch	
Service Fee	

the cost of the different subsystems and operations. Although a discussion of the precise cost model used is beyond the scope of this work (see [56] and [54] for a detailed discussion), it is important to mention that the service fee is only calculated for hosted-payload and commercial contract modalities. The service fee contains all the costs associated with the development, launch and operation of the constellation that NASA pays to the service provider. For hosted-payload contracts, NASA still pays the payload cost, while for commercial contracts this cost is included in the service fee. An explanation of how the service fee is separated into non-recurring and recurring cost is given in Section 4.2.3.

4.2 Static Graph Development

In this section we will describe the development of the model used to build the graph representation of the SCAN tradespace. This section focuses on the development of a *time-independent*, or static, formulation of the tradespace graph, while Chapter 5 will build on this formulation to develop a graph that models the passage of time in the decision making process.

In this version of the graph, each vertex is a Space Network architecture consisting of a set of ground stations and one or more constellations of relay satellites. As such there is a one to one mapping from an architecture to a vertex in the tradespace graph.

The fundamental idea behind the generation of edges in the graph is that the addition and/or removal of a constellation changes the system architecture. In our formulation of the graph, we assume that relationships between architectures are de-

termined by the ability to add or remove constellations to transform one architecture into another. This assumption mirrors the process by which NASA has developed the Space Network architecture over its lifetime. Since the launch of the first TDRS satellites, NASA periodically procured and launched new satellites in batches either to replace satellites reaching the end of their lifetime or to expand the capabilities of the network.

With the launch of a new constellation or the decommissioning of an existing one the system architecture changes, however there may be one or more constellations that are retained across the architecture change (i.e. they are common to both the old architecture and the new architecture). In the language of Chapter 2, constellations define the *asset portfolio* of the system, which allows us to model their addition, retention or deletion across architecture changes.

Since the starting point of our analysis is an enumerated tradespace, we need to determine which pairs of architectures in the tradespace we could feasibly jump between by adding or removing constellations. It should be clear that if we define an architecture only by the constellations it contains, we could jump between any two architectures by deleting and adding an arbitrary number of constellations. However, this type of transformation does not characterize a feasible decision available to NASA since the SCaN program has a limited annual budget with which to augment and operate the network.

For this reason, we introduce the constraint that in order for an architecture transformation to be feasible, exactly one constellation must be added. As we will see in the following sections, it is possible to further constrain this assumption by asserting that the cost of adding this constellation must be below some threshold, but for now we assume that the decision maker has the capability to add any constellation in a single step. This 'single added constellation' rule defines our basic edge existence criterion, with the direction of the edge such that the edge sink is the architecture with the added constellation (see Section 4.2.2 for minor clarifications and further explanation).

It is important to note that this edge existence rule does not take into account the

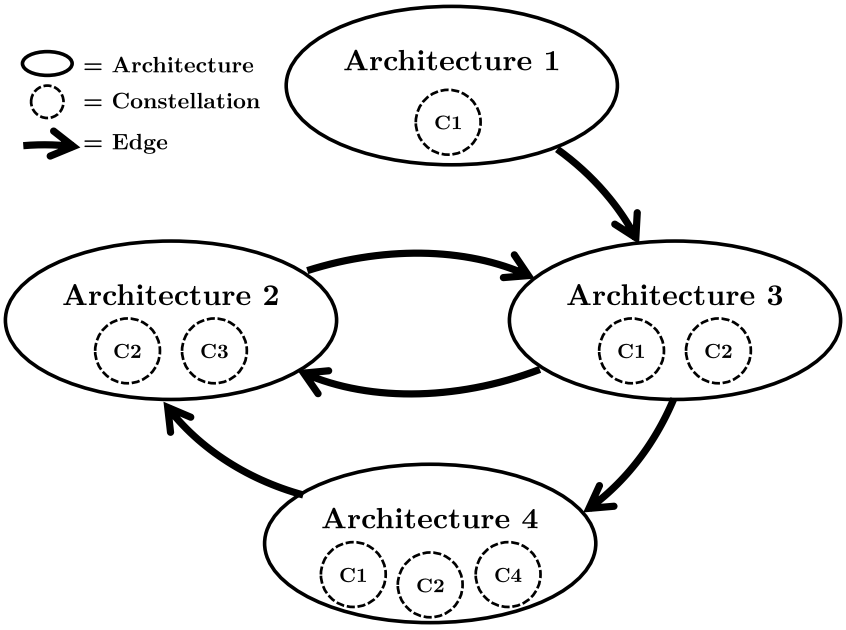


Figure 4-2: A conceptual example of the static graph for the SCaN tradespace.

number of constellations that might be removed from the architecture across the jump since we assume that the cost of decommissioning a constellation is already taken into account in the evaluation of non-recurring cost. Figure 4-2 shows a visualization of a small conceptual tradespace graph to illustrate this simple edge rule.

Looking at the constellations shared between pairs of architectures connected by an edge, we see that exactly one constellation is 'added' in the direction of the arrow. For edge 1-3 (i.e. the directed edge from Architecture 1 to Architecture 3), constellation C2 is added with no constellations removed. The same is true for edge 3-4 with C4 added. Architectures 2 and 3 share constellation C2, with C1 added across edge 2-3 and C3 added across edge 3-2. Finally, edge 4-2 is an edge across which two constellations (C1 and C4) are removed, and a single constellation (C3) is added.

As a small note on implementation, we use a rule-based system written in *Jess* to build and modify the tradespace graph for this tradespace. A rule-based approach is well suited to this particular problem as the process of comparing architectures to one another requires the matching of several numeric and string attributes over all

possible pairs of architectures: a problem which becomes computationally expensive using traditional procedural approaches. In addition, our criteria for the existence of edges between architectures and the calculation of their weights are simple to pose as single rules, with the much more complex ordering and execution of these rules efficiently abstracted into the *Rete* algorithm.

4.2.1 Constellation Matching

In order to determine whether two architectures can be connected, we must first determine which constellations the two architectures share. If the tradespace were enumerated as in Figure 4-2, with two identical constellations having the same label or referencing the same object, this process would be straightforward. However, the tradespace enumeration tool is built such that each architecture possesses its own unique constellation objects and identifiers. For this reason, in order to determine that two constellations are identical we must explicitly match constellation details.

To match constellations we will use the architecturally distinguishing features identified in Section 4.1.2. In other words, in order for two constellations to be considered identical they must have the same:

- Payloads
- Network Type
- Lifetime
- Orbit
- Satellites Per Plane
- Antennae
- Contract Modality
- Satellite Bus
- Number of Planes

For each pair of architectures in the tradespace we track the number of constellations that are shared between them. We then use this number, combined with the number of constellations in each architecture to determine whether to place an edge between the pair.

4.2.2 Edge Existence

The rule for placing an edge between two architectures is straightforward to understand and implement. Our primary criterion is that in order for an edge to exist,

exactly one constellation must be present in the sink architecture that is not present in the source architecture. If we let `num-constel-A1` and `num-constel-A2` be the number of constellation in Architectures A1 and A2 respectively, and `num-shared` be the number of constellations shared by A1 and A2, then this primary criterion becomes:

```
if (num-constel-A2 - num-shared == 1)
then (create edge from A1 to A2)
```

There are two minor changes that we make to this with this simple rule in order to address what we believe are real-world decision making constraints. The first of these is that this rule does not take into account the number of constellations that are 'removed' from the architecture across the edge. We believe there is an implicit constraint on the number of constellations that NASA would decommission as part of any single architecture change, both in terms of the cost of making such a large change at one time and in terms of the performance disruption of decommissioning too many assets simultaneously. However, we do believe there are feasible situations in which NASA might decide to decommission two low performing constellations to replace them with a higher performing one. For this reason, we have decided to allow the removal of at most two constellations from the architecture across an edge.

The second addition we make to the single constellation added criterion is that we dictate that the number of shared constellations must be greater than zero for an edge to exist between two architectures. Essentially, this constraint ensures there are no edges between two architectures for which all the constellations in the source architecture are removed. We believe this graph constraint reflects the real-world decision making constraint that a decision maker would not accept the risk of simultaneously replacing all operational assets with new relay satellites, since the delay, loss or malfunction of the new (likely unproven) constellation would result in a loss of all operational capabilities for the network.

Crucially, both of these augmentations also help to limit the number of edges in the graph, which is the factor that drives the computational times of both building and searching the graph. Taking these two constraints into account, we can now write

the full edge existence rule.

```
if (num-shared &gt; 0)
and if (num-constels-A1 - num-constel-A2 < 2)
and if (num-constel-A2 - num-shared == 1)
then (create edge from A1 to A2)
```

If we enumerate all the possible pairings of architectures with different numbers of shared constellations, we find there are three classes of edges that are generated by this rule. These edge types, shown in Figure 4-3, can be classified according to the number of constellations that are removed across the edge.

4.2.3 Edge Weight

Once all edges between architectures have been enumerated, the final step in the construction of the tradespace graph is the assignment of edge weights. The fundamental idea behind the assignment of weights, as discussed previously, is that the edge weight should indicate the cost of changing the architecture from one vertex to another. In this static representation of the graph, which ignores the passage of time between steps through the tradespace, this switching cost is assumed to be the cost of adding the new constellation to the architecture.

In turn we assume that the cost of adding the constellation to the architecture is well modeled by the non-recurring cost of the constellation itself, which includes the costs associated with designing, manufacturing, and launching the relay satellites that make up the constellation. Therefore, the edge weight is assigned to be the non-recurring cost of the constellation added across the edge.

The manner in which this non-recurring cost is calculated depends on the contract modality of the constellation being added, since this determines the amount that NASA actually pays. For the simplest case of a procurement contract, the edge weight W is simply a sum of the component non-recurring costs for the added constellation, which are listed in Table 4.1.2:

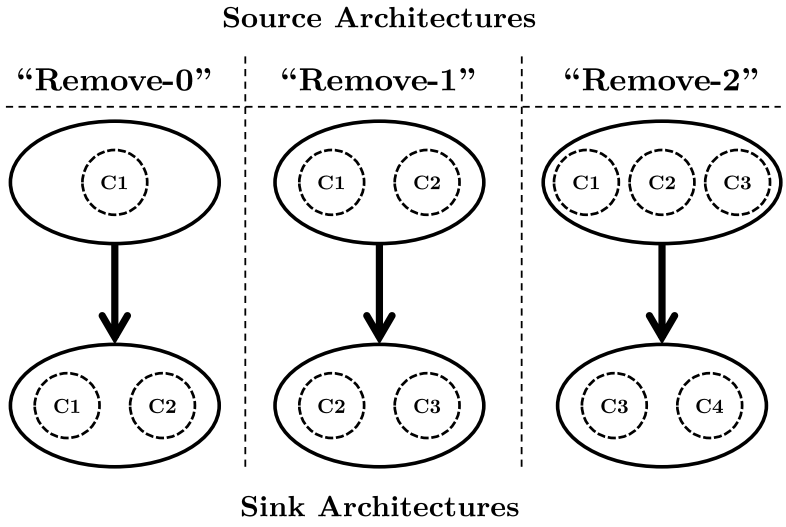


Figure 4-3: A classification of the different types of edges possible in the SCaN tradespace graph.

$$W_{procurement} = \textit{antennae-cost} + \textit{bus-cost} + \textit{payload-cost} + \textit{launch-cost}$$

For a hosted payload contract, in which NASA provides the payload and pays a service fee, we assume that the service fee is paid entirely on a recurring basis over the lifetime of the constellation so it does factor into the static edge weight.

$$W_{hosted} = \textit{payload-cost}$$

Finally, for a commercial contract, all direct recurring and non-recurring constellation costs are paid by the service provider, with NASA paying a service fee for the entire package. Here, we assume that the service fee splits into recurring and non-recurring components based on the ratio of recurring and non-recurring costs paid by the service provider:

$$W_{commercial} = \textit{service-fee} \times \frac{antennae + bus + payload + launch}{operations + antennae + bus + payload + launch}$$

The assignment of weights completes the generation of the static, or time-independent, tradespace graph. A discussion of the analysis that can be performed with this graph will be presented in the following section along with a number of results drawn from SCaN Tradespace Model data.

4.3 Tradespace Exploration Results: Static Graph

In this section we present the results of the application of the static graph to a number of SCaN tradespaces generated by the enumeration and evaluation tool outlined in Section 4.1.2. We will analyze three distinct sets of results for the static graph case to both demonstrate the utility and flexibility of the tradespace graph framework for this particular application as well as produce conclusions of potential value to current NASA decision makers.

In the first result we will demonstrate several of the basic analysis tools we have developed for the SCaN tradespace analysis as we study the problem of trading between several candidate final architectures with differing payloads and payload allocation combinations. Our second result incorporates a number of additional architecture decisions to add variety to the tradespace, and analyzes how the path through the tradespace is influenced by intermediate architecture decisions. In the third set of results, we attempt to answer the question of whether incorporating hosted-payload constellations into the tradespace results in cost savings during architecture evolution, and how cost savings are distributed among different hosted payload types.

Since we will be using three different architecture tradespace for each of the three results in this section, Table 4.3 gives a summary of the decisions that define each tradespace, with decision variables shown in bold. We also give a number of basic properties of the tradespace graph that will be discussed further in each individual

Decision/Property	Result 1	Result 2	Result 3
Payload Selection	**S, KaKu, Tri, Opt**	**S, KaKu, Tri, Opt**	**S, KaKu, Tri, Opt**
Payload Allocation	**All**	**All**	**All**
Satellites per Plane	3	$\mathbf{1, 2, 3}$	$\mathbf{1, 2, 3}$
Contract Modalities	Procurement	Procurement	**Proc, Hosted**
Orbit	GEO	GEO	GEO
Number of Planes	1	1	1
Constel Lifetime	15	15	15
Network Type	Bent-Pipe	Bent-Pipe	Bent-Pipe
# of Vertices	409	365	538
# of Edges	$8,414$	$6,092$	$10,408$
Final Architectures	Benefit > 0.98	Benefit > 0.98	Benefit > 0.95
# Final Architectures	179	39	111

Table 4.3: A summary of the tradespace decisions and tradespace graph properties for each set of static graph results.

result section.

We note that any number of the four options available in the payload selection decision can be assigned to a constellation as this decision is of the down-selecting class (see Table 4.1). The KaKu payload combines a Ka-band and a Ku-band payload into a single package, while the Tri-band payload combines Ka-band, Ku-band, and S-band. The payload allocation decision variable simply tells us that there is no restriction on how payloads in an architecture are assigned to its constellations.

It is important to note that as we have not yet incorporated the concept of time and the aging of in-space assets, the evolution pathways calculated in this section do not necessarily represent a feasible sequence of decisions made over time, they simply make clear the relationships between different architectures in the graph and provide a means to understand how architectures are connected with one another in the tradespace. We argue that although time constraints are not accounted for, the output of our analysis in this section allows the system architect to organize a complex tradespace and provides an initial basis on which to analyze system investments and prioritize different types of decisions.

An important theme that runs throughout this section, and indeed throughout this thesis, is that the questions that motivate each set of results are highly ambiguous

and impossible to answer without analyzing the relationships between architectures rather than just their individual performance. The framework that we apply to these problems allows us to constrain this ambiguity, make explicit the assumptions and decision making model inherent in our analysis, and bring rigorous, computationally powerful tools to bear on these problems to quickly generate decision trades that are understandable to the system architect.

4.3.1 Result 1: Final Architecture Selection

In this first result we seek to demonstrate the generation and visualization of the tradespace graph, the analysis tools that can be developed to extract valuable information from it, and the unique insights that this framework can supply to the system architect.

Graph Preliminaries

The tradespace used here is described in Table 4.3, with the primary decisions between architectures being the selection and allocation of payloads. This tradespace is shown graphically in Figure 4-4 with each point corresponding to one of 409 architectures. Each architecture is plotted according to benefit on the x-axis and lifecycle cost on the y-axis. These architectures map one-to-one with vertices in the static graph, which contains $8,414$ edges after evaluation of the graph generation rules above.

For this scenario, we assume that the fixed initial selection is the lowest benefit architecture on the Pareto frontier, shown as the green circle in Figure 4-4 (note that we do not consider TDRS to be the initial architecture for this scenario, as we will in subsequent results). This architecture is composed of a single constellation with an S-band payload.

When selecting a set of candidate final architectures, we impose the single constraint that a final architecture must have benefit greater than or equal to 0.98. The set of all such candidate final architectures is shown as the green stack of points on the right. This benefit constraint mimics a stakeholder's ambiguous stipulation that

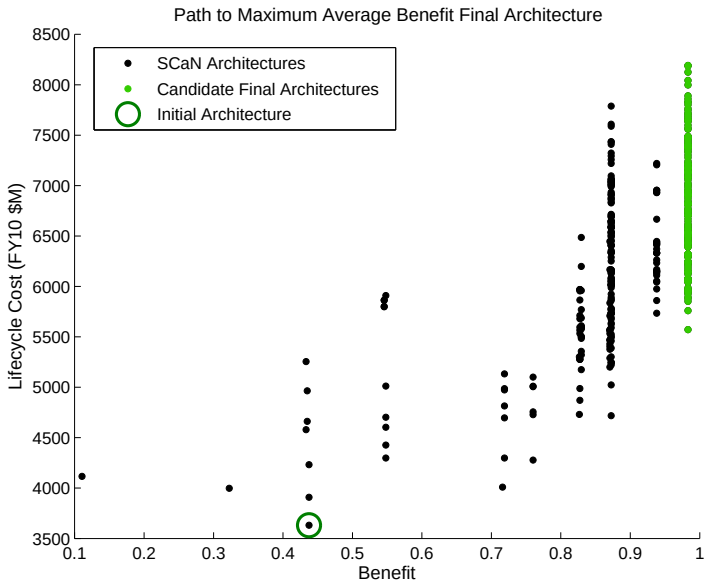


Figure 4-4: An overview of the SCaN tradespace for payload selection and allocation decisions plotted as benefit versus lifecycle cost.

the final architecture must be in the high performance region of the tradespace.

Final Architecture Selection: Minimum Lifecycle Cost

When selecting a final architecture from this set, the most intuitive choice is to select the architecture with the lowest lifecycle cost, represented by the point at the bottom of the candidate stack. Using the static graph and Dijkstra's algorithm, we can easily find the shortest path through the graph, which is the lowest cost sequence of decisions, from the fixed initial architecture to this minimum lifecycle cost architecture. This path through the graph is shown on the original cost-benefit plot in Figure 4-5.

Although this path appears direct, the lifecycle cost metric shown in this plot hides inefficiencies in this sequence of decisions. If we instead look at each step along the path in terms of the changes that take place across each edge we get an idea of why this might be the case. Figure 4-6 shows the path decomposed into steps diagrammatically, with each edge represented by an arrow and the architectures and constellations at each stage represented by the ellipses and circles respectively.

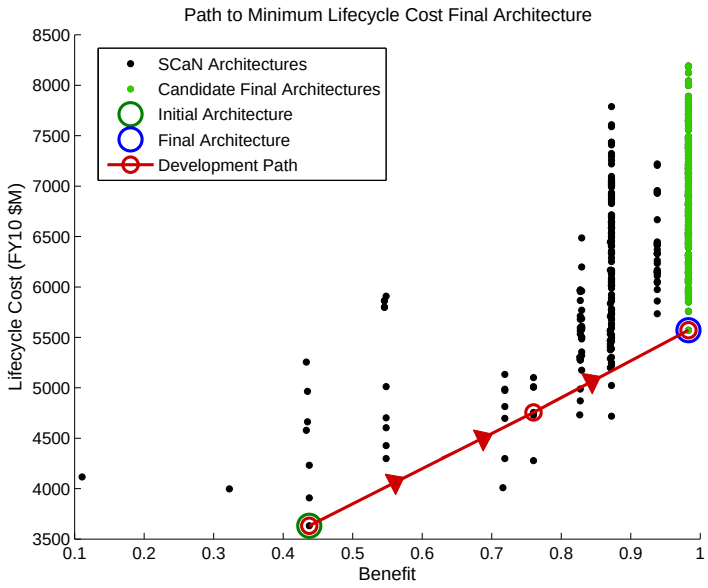


Figure 4-5: The shortest development path from the initial architecture to the minimum lifecycle cost final architecture.

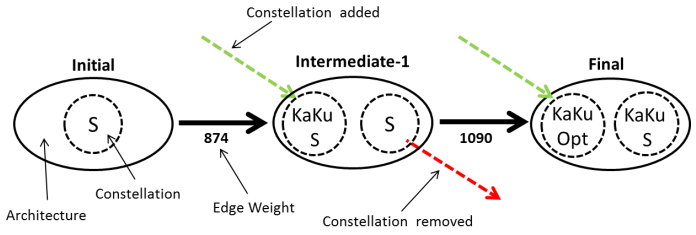


Figure 4-6: Diagram of the development path from initial architecture to the minimum lifecycle cost final architecture. Ellipses and circles represent architectures and constellation respectively, with constellations labeled by the payload(s) they carry. Green arrows show constellations added across the previous edge, with red arrows showing constellations deleted across the next edge.

The S-band constellation in the initial architecture represents an investment that is discarded over the course of the development path as it is removed from the architecture across the edge from the intermediate architecture to the final architecture. Note also that the final architecture contains a two constellations with a KaKu-band payload, which in terms of development costs leads to duplicated costs at different points in the development path.

Paths to All Candidate Final Architectures

A more intelligent selection of the final architecture might allow the initial S-band investment to be utilized throughout the development path or invest in only one constellation with a KaKu band payload to save on payload non-recurring costs. We might also suspect that the final architecture could be chosen such that the intermediate architecture has greater benefit than the one already calculated.

Since the size of our graph is small relative to many other graph theory applications and Dijkstra's algorithm is highly efficient, it is computationally trivial to find the shortest path to all the candidate final architectures (the green points in Figure 4-4) and to examine the final architecture selection problem based on the lowest-cost development path each one allows.

Figure 4-7 shows a plot of the development path to each of the 179 candidate final architectures. In this plot, each point represents a path to a unique final architecture, with the point plotted as the total development cost along the path (i.e. the sum of the weight of the edges traversed) on the y-axis, and the average benefit of the architectures along the path on the x-axis, which represents the overall performance of the intermediate architectures.

First, it is clear that the path to the minimum lifecycle cost final architecture (highlighted with the red circle in Figure 4-7), which was our intuitive first guess for the 'best' final architecture, is a dominated option when we look at the decision from the perspective of development path cost. In other words, we could select a different final architecture that would yield the same or better average benefit at a lower development cost. This cost metric, rather than the lifecycle cost of the final

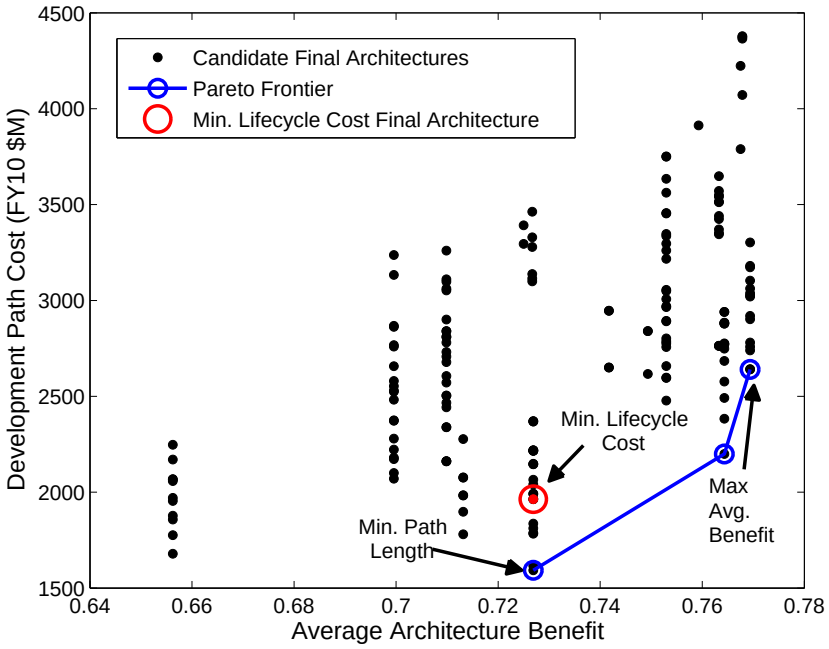


Figure 4-7: The shortest paths to each of the 179 candidate initial architectures plotted along total development path cost and average architecture benefit.

architecture, is the cost metric the decision maker is truly concerned with in assessing architecture evolution since it takes into account the investment already made in the initial architecture.

Figure 4-7 also shows the Pareto frontier of the development path cost and average benefit metrics as the solid blue line. The final architectures that map to the three points on this frontier represent the options that the decision maker would trade between, all other things being equal. We can also look in more detail at one of these development paths, for example the minimum development cost selection to understand how improvements can be made on the previous selection.

Final Architecture Selection: Minimum Development Cost

Figure 4-8 shows the minimum development cost path on the original tradespace axes with the final architecture highlighted in blue. We note that the final architecture is far from the Pareto frontier along the individual architecture cost-benefit metrics (approximately \$1 billion dollar larger lifecycle cost), which further highlights the

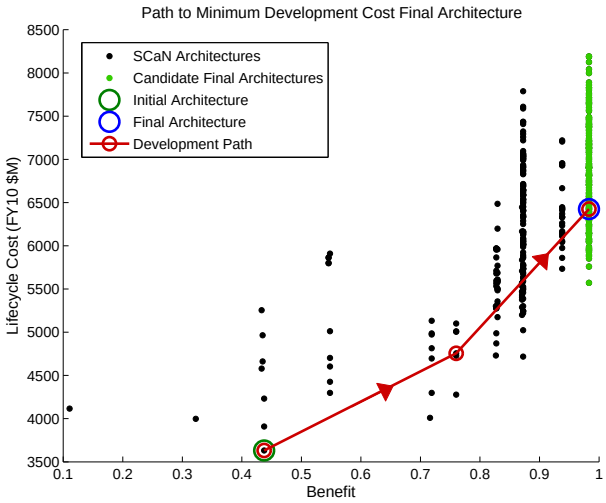


Figure 4-8: The development path to the final architecture that yields the minimum total development cost for the given initial architecture.

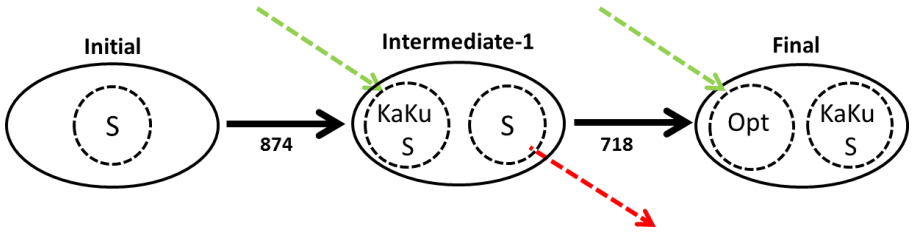


Figure 4-9: Diagram of the development path from initial architecture to the minimum development cost final architecture.

point that the relationship of the final architecture with the initial architecture is difficult to ascertain without the aid of such a graph-based framework which rigorously codifies architecture relationships.

To understand how this development path differs from that of the minimum life-cycle cost final architecture, we turn to Figure 4-9, which shows a detailed view of how the architecture changes at each stage in the evolution. Once again we see that the S-band constellation is discarded across the second edge, which suggests that the S-band constellation on its own either does not appear in the high performance architecture region, or is dominated due to the presence of higher cost constellations in any architecture in which it does exist. We do note however that the constella-

tion added across the second edge contains only a single payload - optical - which accounts for the difference in total development cost relative to the minimum lifecycle cost architecture.

Although this section presents a relatively simple tradespace analysis scenario with only two variable decisions, we believe this analysis supports the assertion that building a framework for the rigorous and exhaustive assessment of architecture relationships and development pathways has the potential to provide significant value to the decision maker.

4.3.2 Result 2: Intermediate Architecture Decisions

In this second set of results we analyze a more complex tradespace with a greater number of architecture variables, and seek to answer the question of how to enumerate and evaluate architecture evolution decisions once an initial architecture and final architecture have been defined. Once again, the purpose of this analysis is to demonstrate the analysis techniques that can be used with the static tradespace graph rather than serve as a source of conclusions and recommendations that are grounded in concrete problems currently faced by SCaN decision makers.

Graph Preliminaries

The tradespace used in this analysis is once again described in Table 4.3, with the additional constellation decision of number of satellites per plane added relative to Result 1. Before generating the graph, we first define the initial architecture that will be used in all subsequent analysis. This initial selection is a 'TDRS-like' architecture with three procured GEO satellites all carrying two tri-band payloads (S-band, Ka-band, and Ku-band). Although this architecture does not match the current architecture of the Space Network exactly with its backup satellites in storage and other low data-rate antennae, it is an adequate approximation of TDRS given the assumptions of the SCaN model.

This tradespace consists of the 365 architectures with benefit greater than or

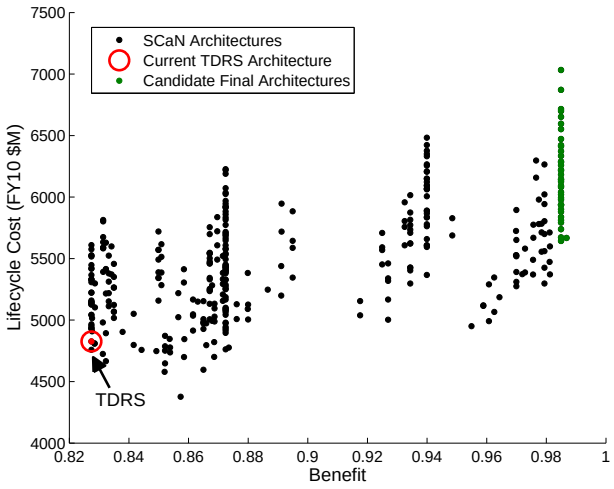


Figure 4-10: Tradespace for the second static result set. TDRS initial architecture and candidate final architectures shown.

equal to the TDRS architecture, which when converted to static form are connected by $6,092$ edges. Figure 4-10 shows this tradespace with the initial TDRS architecture highlighted on the left.

First it is worth noting that this tradespace has a spread distribution of benefit values, rather than the clusters of many architectures at only a few benefit values in the tradespace used in Result 1. This wider spread is caused by the greater number of architecture decisions encoded in the space, resulting in less stratification of performance values. As in the Result 1 we will first determine the shortest development paths to many candidate final architecture, shown as the green points with benefit greater than 0.98.

Paths to All Candidate Final Architectures

The minimum-cost development path to each of these final architectures is shown in Figure 4-11 plotted along development cost and average path benefit as in the previous set of results. Once again, we see a spread of development costs and average benefit metrics along each path, with the Pareto front representing the non-dominated set of final architecture decisions.

To analyze the options available for TDRS evolution in more detail, let us pick

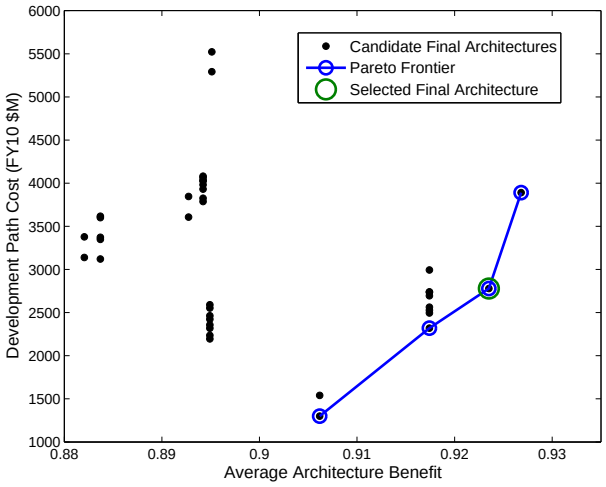


Figure 4-11: Paths of minimum development cost from TDRS to each of the 39 candidate final architectures.

one of these Pareto optimal final architectures to be the notional final architecture selected by the system architect. The architecture we have selected is highlighted by the green circle in Figure 4-11, which for ease of reference we will call *TDRS-Final*.

100 Shortest Paths to TDRS-Final

The path we have found already found from TDRS to TDRS-Final is the path that minimizes the development cost along the evolution without taking into account the benefit provided by the intermediate architectures. However, a decision maker might be interested in understanding whether this intermediate benefit can be improved by increasing the development path cost. In other words, he or she is interested in the trade between the cost and performance of the development path.

To analyze this trade, we must extend beyond the shortest path algorithm used up to this point and instead find other paths from the initial vertex to the final vertex in the graph. For this purpose we use a version of the K-Shortest Paths algorithm, which implements the Bellman-Ford algorithm, to find the 100 shortest paths in terms of development cost through the graph. These paths are shown in Figure 4-12 plotted with total development cost versus average architecture benefit.

From this plot it is clear that the majority of paths are clustered in the domi-

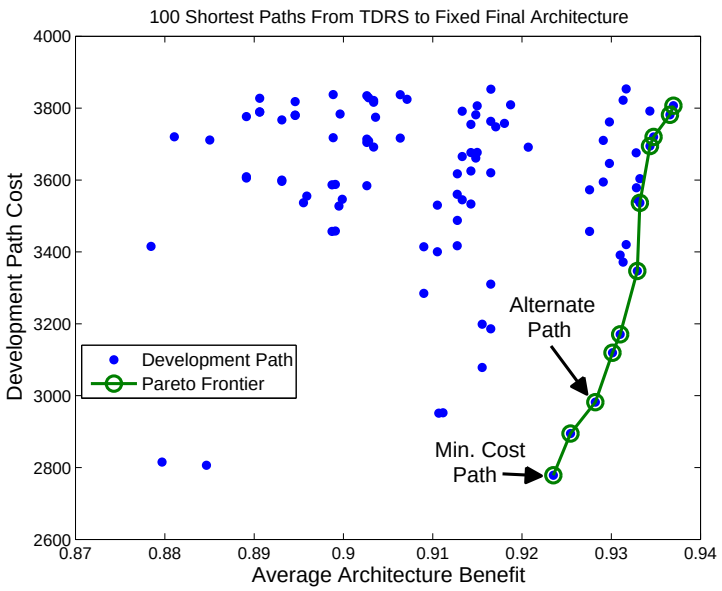


Figure 4-12: The 100 lowest cost paths through the tradespace from TDRS to TDRS-Final.

nated high-cost region of the trade, however we do see a front of increasing average architecture performance with increasing development cost, which means that we cannot unequivocally say that the shortest path found in the first stage of analysis is the best path through the space. If the system architect is willing to invest additional resources in evolving the architecture, increased benefit can be realized in the intermediate architectures.

Two Different Paths to TDRS-Final

To explore this point further, in Figure 4-13 we plot two of the paths that lay on the development cost-average benefit Pareto frontier of Figure 4-12. The minimum cost path is shown as a solid line, while an 'alternate path' with higher development cost and average benefit is shown as a dashed line.

The single difference between these two evolutions is the constellation added to the architecture across the first edge. In the minimum cost path this constellation contains a single satellite with an optical payload, and in the alternate path it contains two satellites with optical payloads. The increased benefit that this additional satellite

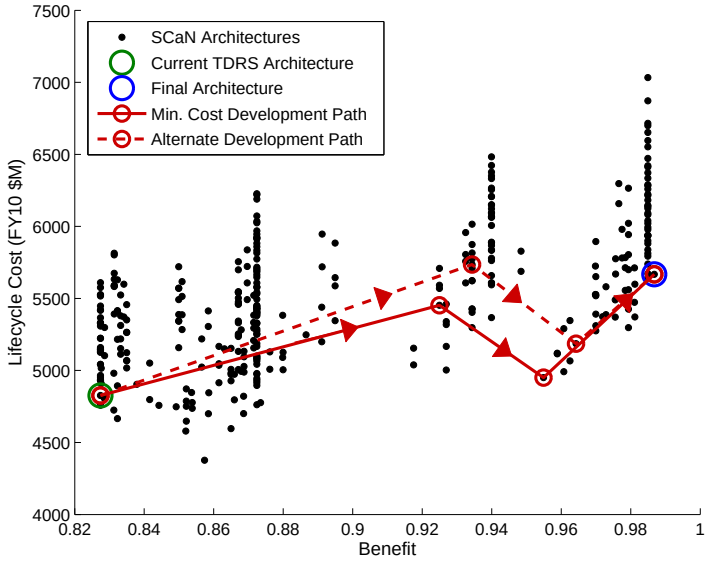


Figure 4-13: Two development paths through the tradespace from TDRS to TDRS-Final.

provides can be seen in Figure 4-13 as the relative shift of the dashed line intermediate points to the right, while the increased cost of this satellite requires can be seen in Figure 4-11 as the vertical distance between the minimum cost point and alternate path point (approximately $210 million).

Depending on stakeholder requirements and projected budgets, this trade and many others like it can be examined to determine what the optimal set of decisions are for the system architect and what options exist for either down-scoping or expanding intermediate architecture capabilities to respond to project changes during the development cycle.

4.3.3 Result 3: Hosted Payload Cost Savings and Prioritization

Since using commercial or government satellite buses to host scientific, military, or commercial payloads has already been proven as a viable and cost-efficient capability, NASA is interested in deploying hosted payloads in future versions of the SCaN architecture to realize cost savings over the traditional procurement strategy. In this

third set of results we seek to analyze a specific question of direct interest to SCaN stakeholders: what is the benefit of introducing hosted payload constellations into the development path, and how does this benefit vary depending on what type of payload is hosted?

Hosted Payload Constraints

Although hosting payloads on commercial buses is a proven capability, it is reasonable to assume that NASA stakeholders would be hesitant to make an immediate and drastic change to the current architecture by immediately switching the entire SCaN architecture to a hosted payload contract structure. Instead, a much more likely approach would be for SCaN system architects to gradually incorporate hosted payload constellations by limiting the role that they play in architecture to limit exposure to the risk of the hosting bus (or hosted payload) failing on orbit or being delayed in development.

To account for this preference in our tradespace exploration model, we introduce the decision to either have a hosted payload or procurement contract as a variable in the tradespace with two constraints. First, any architecture that contains a hosted payload constellation can only contain one such constellation. Second, any constellation with hosted payload contract modality can only have one payload assigned to it.

These two constraints together model the preference that the role of hosted payload constellations is minimized. In other words, if all hosted payloads were to fail the architecture could still maintain an acceptable level of performance. These constraints also carry the added benefit of making the tradespace graph smaller, and consequently run times shorter: on the order of 15 seconds to build the graph and 5 to find the shortest paths.

Graph Preliminaries

The tradespace used in this set of results is described in Table 4.3 and contains all the same decisions as Result 2 plus the contract modality decision with the two

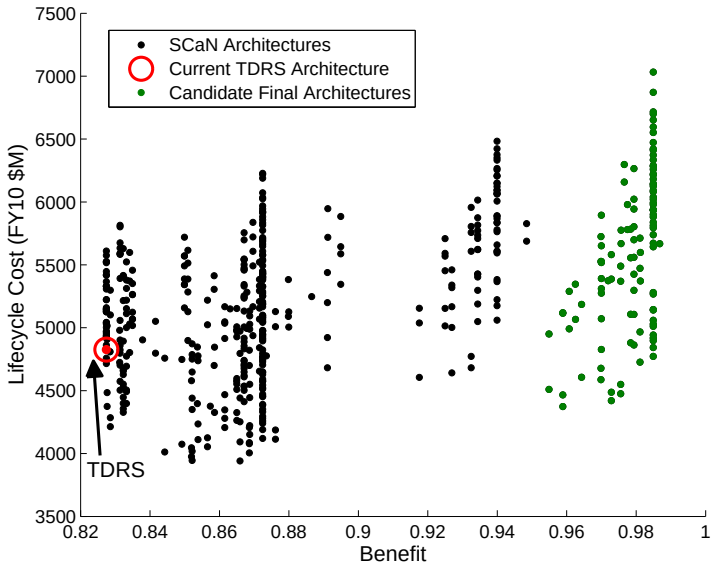


Figure 4-14: The tradespace used for the static graph hosted payload analysis.

constraints above. This tradespace is shown in Figure 4-14 with TDRS again selected as the initial architecture. The tradespace static graph contains 538 vertices and 10,408 edges.

The set of candidate final architectures for this result is the collection of 111 candidate final architecture shown in green on Figure 4-14, which are all architectures with benefit greater than 0.95. This lower performance threshold was chosen for this result to introduce more variability into the final architecture set and to increase the total number of paths under analysis.

Analysis Outline

The analysis carried out for this result is significantly more complex than that in the previous sections, so it will be helpful to refer to a brief outline of the steps involved in this process. The goal of our analysis is two-fold: first determine what types of hosted payloads are used in development paths through the tradespace, and second calculate how much is saved by introducing these hosted payloads to the architecture versus the case where only procurement constellations are permitted.

We can break the analysis performed to answer these questions into five steps:

1. Find the best path from TDRS to each of the 111 final architectures with hosted payloads allowed.

2. Break down how many paths contain each type of hosted payload at any point along the path.

3. Map each of the 111 final architectures to the identical architecture with all procurement constellations.

4. Find the best path from TDRS to each of the 111 procurement-only final architectures, with no hosted payloads allowed along the evolution.

5. Compare the cost of the paths that finish at the same architecture with and without hosted payloads allowed to determine hosted payload savings.

Each of these steps will be discussed in detail below, with figures and analysis provided to visualize and understand how this set of results answers the motivating questions posed above.

Paths with Hosted Payloads - Breakdown by Type

After defining the set of 111 candidate final architectures, we find the shortest path through the tradespace graph from the initial TDRS architecture to each of these final architectures individually. Note that although a final architecture may not contain a hosted payload constellation, such constellations can exist in the intermediate architectures of the evolution. Once these paths are found, we then search through each and determine which ones contain each type of hosted payload.

We can organize these paths through the tradespace based on how close they are to the Pareto frontier of path cost and average architecture benefit. We define the 'X% Fuzzy Pareto set' as the 'X%' of the 111 paths through the tradespace that are closest to the Pareto frontier. Therefore, if we look at the '10%' Fuzzy Pareto set, we will be looking at the 11 'best' paths through the tradespace. This concept is important to our analysis as it allows us to understand how the relative quality of the path relates to the type of hosted payload used.

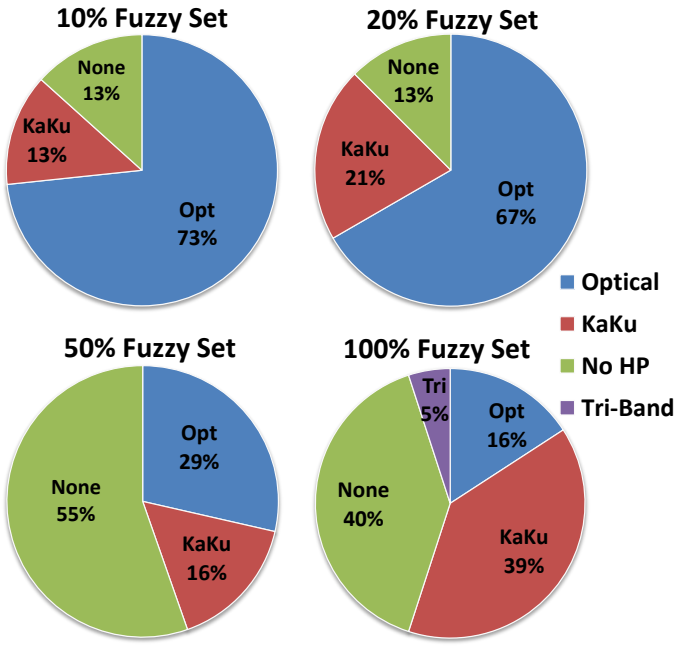


Figure 4-15: The distribution of paths that contain each type of hosted payload for different sizes of the Fuzzy Pareto set.

The breakdown of paths to the 111 final architectures through the hosted payloads tradespace is shown in Figure 4-15 for four sizes of the Fuzzy Pareto set.

First these charts tell us that for the best final architecture selections in terms of total development cost and average benefit (the 10% and 20% Fuzzy Pareto sets), the majority of paths through the tradespace utilize an optical hosted payload. As we start to include progressively worse paths in the analysis, the number of paths that do not utilize hosted payloads rises significantly. Finally, no Tri-band hosted payloads appear in the set of the best development paths until the entire set of paths is considered, indicating that Tri-band is not a good choice for a hosted payload investment in terms of development cost.

In terms of concrete recommendations that might be made from this aggregate data, it is clear that investing in hosting an optical communications payload has the greatest chance of having an impact and being included in an uncertain architecture evolution future since optical is found in most of the best paths through the tradespace.

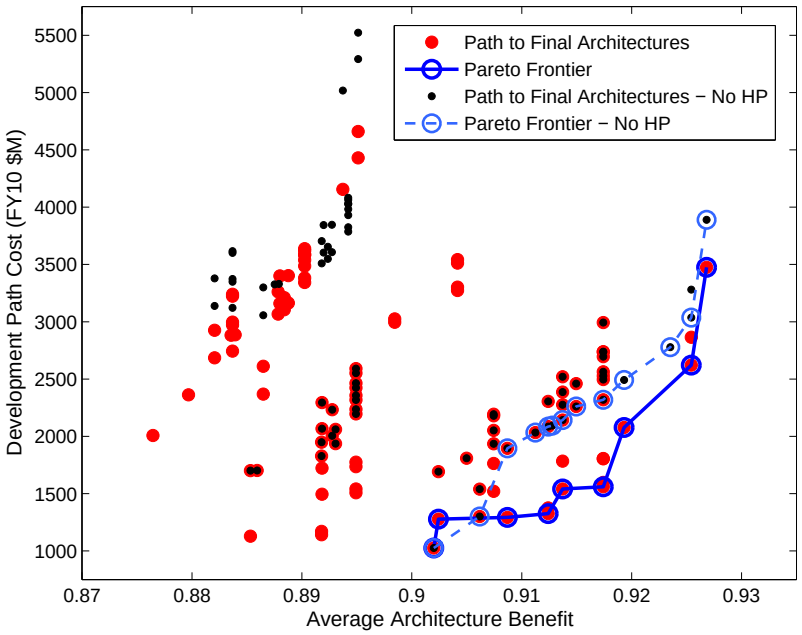


Figure 4-16: A plot of the development paths from TDRS to 111 final architectures in the hosted payload tradespace and the corresponding 81 final architectures in the procurement-only tradespace.

Mapping of Final Architectures to Procurement-Only Set

The next step in our analysis is to match each of the 111 final architectures to its identical architecture with only procurement constellations. In other words, if one of the 111 architectures contains a hosted payload, the matching architecture is the one with all the same decisions except that the hosted payload constellation is now procured.

This architecture matching yields 81 candidate final architectures in the non-HP tradespace. The reason that this number is smaller than the number of HP tradespace candidates is that multiple architectures with hosted payload constellations can map to the same architecture with only procurement constellations. We then find the path from TDRS to each of these 81 candidate architectures, with the constraint that all architectures on the path contain no hosted payload constellations.

Comparison of Paths with and without Hosted Payloads

At this point we have two sets of paths through the tradespace. The first set, which we will call the HP set, consists of 111 paths to the same number of final architectures, where both the final architecture and the intermediate architectures along the evolution may have hosted payload constellations. The second set, which we will call the non-HP set, consists of 81 paths to 81 different final architectures where neither the final architectures nor any intermediate architectures contain any hosted payloads. Note that there is a mapping (not one-to-one) between the two sets.

Figure 4-16 shows a plot of both the HP set and non-HP set of paths according to the total development cost and average architecture benefit or each path, along with the Pareto frontiers for each path set. We can note several interesting trends from this plot alone. First, as expected the inclusion of hosted payloads even given the constraints imposed results in cost savings across the performance spectrum, which is apparent in the vertical shift of the respective Pareto frontiers. Looking broadly across the tradespace as a whole, we also see that the red points are generally shifted down from the black (non-HP) points.

A second observation worth noting is that although there is generally a cost savings shift between the two sets, many non-HP points on this plot overlap points in the HP set. These points represent paths which contain only procurement constellations regardless of whether hosted payload constellations are present in the tradespace. This observation tells us that incorporating hosted payloads into the architecture is in some cases not the optimal decision depending on the final architecture selected.

One example of such a case is the minimum development cost final architecture for each case (i.e. the far left point on both Pareto frontier). This final architecture and path are the same for both the HP set and non-HP set. Looking closer at the final architecture, shown in Figure 4-17, we can see why this is the case.

This final architecture contains the TDRS constellation (the constellation with two Tri-band payloads) in addition to a high-performing KaKu-Optical constellation. The shortest path from TDRS to this architecture need only contain one edge, across

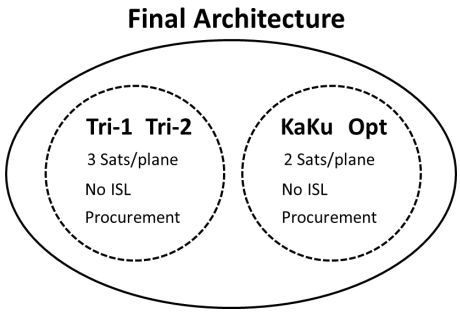


Figure 4-17: A diagram of the minimum development cost final architecture, which is the same for both the HP set and non-HP set. The Tri-band payloads contain S-band, Ka-band, and Ku-band transponders.

which the KaKu-Optical constellation is added. Even if hosted payload constellations are an option, adding the single procurement constellation is still the optimal choice in terms of cost since it must be added to the architecture in any scenario to reach this particular final architecture.

Savings by Hosted Payload Type

Although we have found that introducing hosted payloads into the tradespace leads to development cost savings in many cases (it never leads to cost increases), we still have not answered the questions of how large these savings are and how they vary across different payload types. To perform this analysis, we make use of the fact that each path in the HP set is mapped to a path in the non-HP set by the matching of final architectures in step 3 above.

For each pair of paths, we calculate the difference in the total cost of the development path when hosted payloads are allowed in the tradespace versus when they are not. In other words we find the cost of development from TDRS to an architecture through a tradespace graph that includes hosted payload architectures, and then subtract it from the cost of development to the same architecture through a tradespace graph with only procurement architectures.

We can decompose these cost savings according to what type of hosted payload is deployed in the architecture, and further break down the analysis by looking once again at successive Fuzzy Pareto sets. In Figure 4-18 we show a bar graph of the

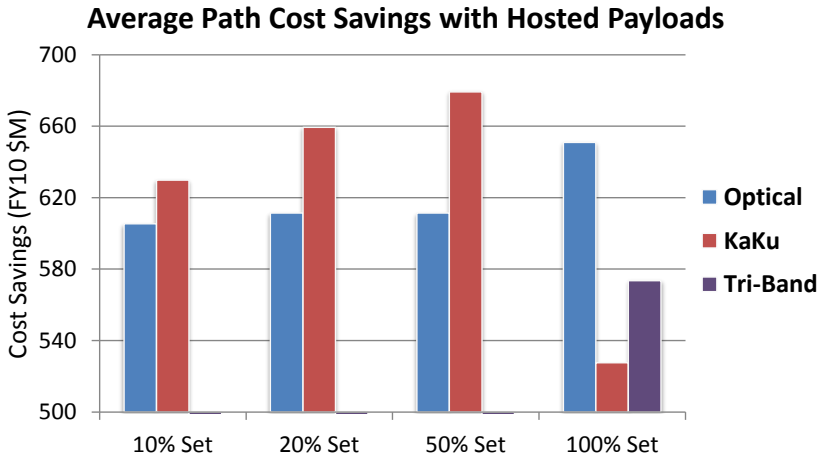


Figure 4-18: The average cost savings along the development path broken down by payload type and Fuzzy Pareto set size.

average savings along the development path according to payload type and Fuzzy Pareto set size.

Once again, several interesting observations can be made about this data. First, average savings across the board is in the \$500-\$700 million range, which, considering that the majority of total development cost is in the 1-\$3 billion range, tells us that incorporating hosted payloads into the architecture results in significant cost savings. Second, we note that in the best paths (small Fuzzy Pareto set size) KaKu payloads provide greater cost savings when hosted versus Optical payloads. Finally, if we compare the 50% and 100% data points in Figure 4-18, we see that there are several highly dominated paths in the tradespace that provide very high savings from optical payload hosting (the average increases substantially) and much smaller savings from KaKu payload hosting (the average decreases substantially).

Following the development of the time-expanded tradespace graph in the next section, we will again examine these questions on the effects of hosted payload constellations, but with natural time constraints added into the equation. Even without these constraints, we have still been able to produce several conclusions regarding the distribution of hosted payload type and savings that are of value to a system architect in the process of deciding whether to risk investing in hosted payload contracts.

Chapter 5

Space Communications Networks: SCaN Time-Expanded Tradespace Graph

We move now to the time-expanded version of the SCaN tradespace graph, which allows the aging of in-space assets with limited lifetimes to be modeled along the evolution of the architecture. The underlying architecture model and tradespace definition is identical to that in Chapter 4, so we begin in the first section with the development of the graph itself. The time-expanded graph shares many characteristics and generation rules with the static SCaN tradespace graph, however the modeling of time along the development path requires the introduction of a number of new concepts and rules. In the second half of this chapter we present four distinct sets of results drawn from the analysis of the time-expanded graph that lead to conclusions of potentially significant value to the system architect.

5.1 Time-Expanded Graph Development

In the previous chapter we defined a tradespace graph that took into account only the addition and removal of constellations to connect vertices in the graph to one another. One major factor missing from this model of the architecture decision making process

is the effect of the limited lifetime of in-space assets on the options available. Since the planning horizon of the SCaN Tradespace Model is 20 to 30 years in the future, and since communications constellations typically have a lifetime of 10 to 15 years, it is desirable to capture how the aging of in-space assets affects the SCaN decision making process.

To incorporate the passage of time into the tradespace graph we will build directly upon the static graph developed in Chapter 4. The key change that we will make is that rather than associating a vertex with an architecture, we will associate a vertex with an architecture and a set of constellation ages. Note that the age of a constellation, which counts the number of years it has been on orbit, is distinct from its lifetime, which is the number of years it was designed to operate on orbit. Whereas in the static graph there is a one-to-one mapping between architectures and vertices, in the time-expanded graph several vertices map to the same architecture with different constellations ages. Edges are then generated based on both the addition of a constellation, as in the static graph, and the feasible progression of the constellations' ages.

We construct the time-expanded graph in three steps. First, we enumerate the possible vertices in the graph by expanding each of the static graph vertices with all feasible age combinations. Next we expand the static graph edges by only adding time-expanded edges that satisfy the feasible progression of the constellation ages. Finally, we augment the edge weight by incorporating the recurring costs of the active constellations.

Comparison to Time Expanded Decision Networks

It is important here to compare our version of time-expansion to that used by Silver and de Weck's *Time Expanded Decision Networks* [59]. In both cases, the purpose of the time-expansion process is to convert a dynamic graph in which the characteristics of vertices vary with time into a time-invariant graph in which vertices and edges remain fixed in order to implement common graph theory analysis tools such as Dijkstra's Shortest Path algorithm.

The difference between our version of Time Expansion and Silver's lays in both what is being modeled in the time expansion process and how the expansion is implemented. In Time Expanded Decision Networks, the variable that changes over time relates to the exogenous factors that determine the performance and cost of the architecture represented at each vertex. Namely, Silver and de Weck expand each architecture vertex to account for its performance and cost under a number of different stakeholder demand snapshots that change along a defined profile as a function of time. Stated differently, each time expanded vertex represents the potential performance of an architecture at a particular time if that architecture were to be active.

On the other hand, our time-expanded graph models the aging of assets in the architecture itself rather than changes in the environment over time. Therefore, each time-expanded vertex does not represent the architecture at a particular time, it simply represents the architecture with a particular set of ages assigned to its constellations.

In terms of implementation, the processes used to determine edges in the two types of time-expanded graphs are considerably different. In Silver and de Weck's implementation, the criterion for placing an edge between two vertices is that their scalar time index differs by a single unit. In our implementation of the graph, we must ensure that constellations which are common to both vertices increase in age by the time step, all removed vertices are assigned the maximum age, and all added vertices are assigned edge zero. The complexity of this process makes our time-expansion step well-suited to a rule-based implementation, which is another point of difference between the two methods.

A major point of commonality between Silver and de Weck's work and the work here is the manner in which edge weight is defined. In both cases the edge weight has an 'up-front' component (we call it non-recurring cost, while Silver calls it switching cost) and a 'repeating' component that accounts for recurring cost of operations.

5.1.1 Vertex Expansion: Assigning Constellations Ages

The first step in creating the time-expanded vertex set is defining the global time step which will define the number of years by which the age of each constellation will be incremented at each step between vertices. The size of the time step determines both the level of fidelity at which we wish to model the decision making process as a function of time and the size of the time-expanded graph relative to the static graph.

The optimal time step is one which approximates the time period of major SCaN architecture decisions. Based on the evolution of the TDRS network, NASA's current decision cycle is approximately 5 to 7 years, with major upgrades to the network occurring more or less with this period [67]. This cycle is determined both by the expiry of previously launched assets and the desire to procure relay satellites in batches to realize cost savings over single satellite contracts. For this reason and the fact that it evenly divides the usual constellation lifetimes of 10, 15 or 20 years, we have chosen 5 years as the time step for our analysis.

The constellation ages of a vertex represent the number of years that the constellation has been on orbit *at the beginning of the time step*. Therefore, the minimum constellation age is zero (constellation just added to architecture), while the maximum age is the lifetime minus the time step. The reason that the maximum value is the lifetime minus the time step is that after we jump from a vertex with the max value, the age will be incremented to its lifetime and must be removed from the architecture.

To generate the time-expanded vertices corresponding to an architecture, we first find all the possible combinations of constellation ages which are feasible given the lifetime of each constellation. We then create a vertex in the graph for each of these age sets, and store both the architecture and constellation ages in the vertex for later reference.

The rules for vertex expansion are best illustrated graphically in Figure 5-1, which shows an architecture with two constellations each with lifetime equal to 15. The static vertex, which consists only of the architecture, is mapped to nine time-expanded

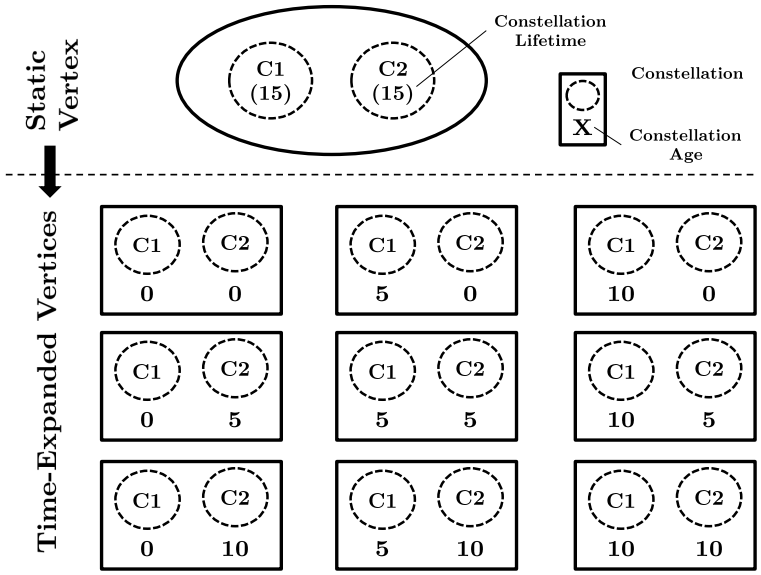


Figure 5-1: Example vertex expansion for the SCaN Time-Expanded Tradespace Graph.

vertices with all the possible age combinations of the two constellations.

5.1.2 Edge Expansion: Incorporating Constellation Age Progression

An edge in the tradespace graph represents a viable step along the evolutionary pathway of an architecture, which in this version of the graph incorporates both the addition of a constellation and the passage of time. Each edge in the time-expanded graph is generated from a static graph edge, which encodes the 'constellation added' relationships between architectures. To determine the time-expanded edges, we simply pick a static edge and look at all the time-expanded vertices whose 'parents' are the source and sink of this static edge. We then place a time-expanded edge between vertices whose constellation ages satisfy three criteria:

1. Each constellation that is common to both architectures has its age incremented by the time step.

2. Any constellation added to the sink architecture has an age equal to zero.

3. Any constellation that is removed from the source architecture has an age equal to its lifetime minus the time step.

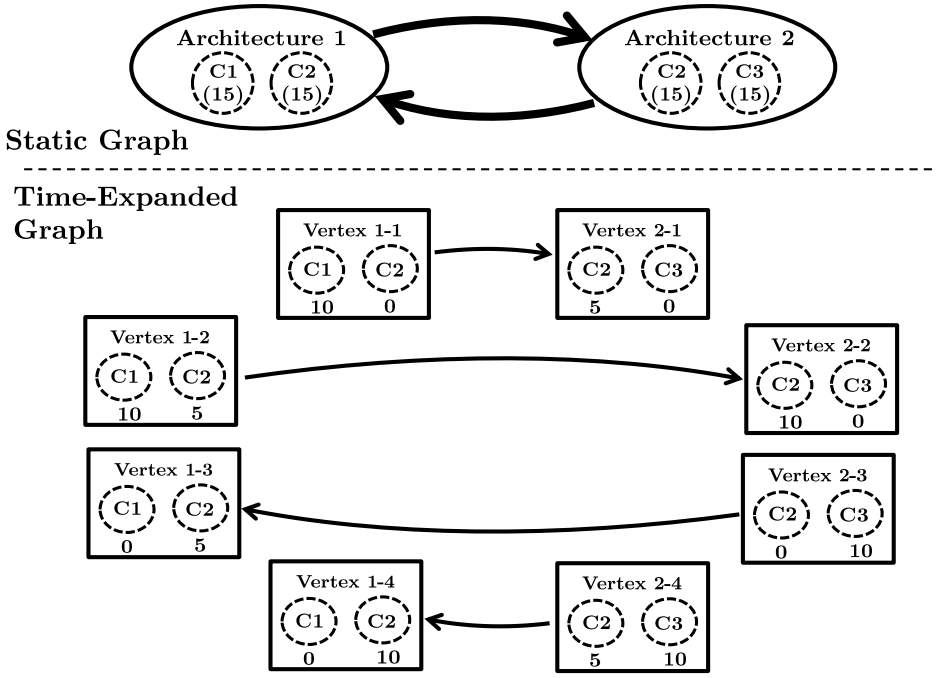


Figure 5-2: Example edge expansion for the SCaN Time-Expanded Tradespace Graph.

Figure 5-2 shows the expansion of edges for two architectures with two constellations each, one of which is common to both architectures. All three constellations have a lifetime of 15 years, with the usual global time step of five years. Note that there is an edge in both directions in the static graph, so we expand each of these individually in the time-expanded graph. It is important to make it clear that in the generation of time-expanded edges we do not reevaluate the matching of constellations and analyze their addition and deletion, rather we expand directly from the static edge that already contains this information.

In addition to deciding to add and/or remove constellations at each time step, a decision maker might also wish to retain the architecture in its current configuration. This type of decision is one that is highly desirable to model since NASA has neither the resources nor the need to add assets to the Space Network every five years, which

we have chosen as our graph time step. Allowing the architecture to go unchanged from one step to another allows us a great amount of modeling flexibility in terms of analyzing decisions in response to changing demand or constraints on non-recurring cost that would not be possible without the inclusion of the passage of time in the graph model.

To generate these constant architecture steps in the time-expanded graph, we simply add self edges (i.e. edges whose source and sink are the same) to each of the vertices in the static graph. We then expand these edges using the same set of rules that we used for all other edges. Since there will be no constellations added or removed across a self edge, the only relevant rule is that the ages of each of the constellations is incremented by the time step across an edge.

Figure 5-3 shows a model tradespace of four architectures with the static graph (including self-edges) shown above and the time-expanded graph shown below. The color of the vertex in the static graph matches the color of the vertices in the time-expanded graph with the same architecture. Each constellation in the tradespace has a lifetime equal to 15 years, except Constellation C3 which has a lifetime of 20 years; and as usual the global time step is 5 years.

Note that the full set of expanded vertices does not appear in the time-expanded graph. Namely, there is no vertex which has two constellations that are the same age. The reason for the absence of such vertices is that by only adding one constellation at each step, we ensure that ages of constellations added to the initial architecture are staggered. If we make the additional assumption that the initial architecture contains constellations that are themselves different ages, then we can remove all vertices with duplicate ages from the graph, which significantly reduces the number of vertices and edges in the time-expanded graph. We believe this is a reasonable assumption since the current Space Network consists of constellations with staggered ages and the SCaN program does not have the resources to launch multiple constellations in a single time period.

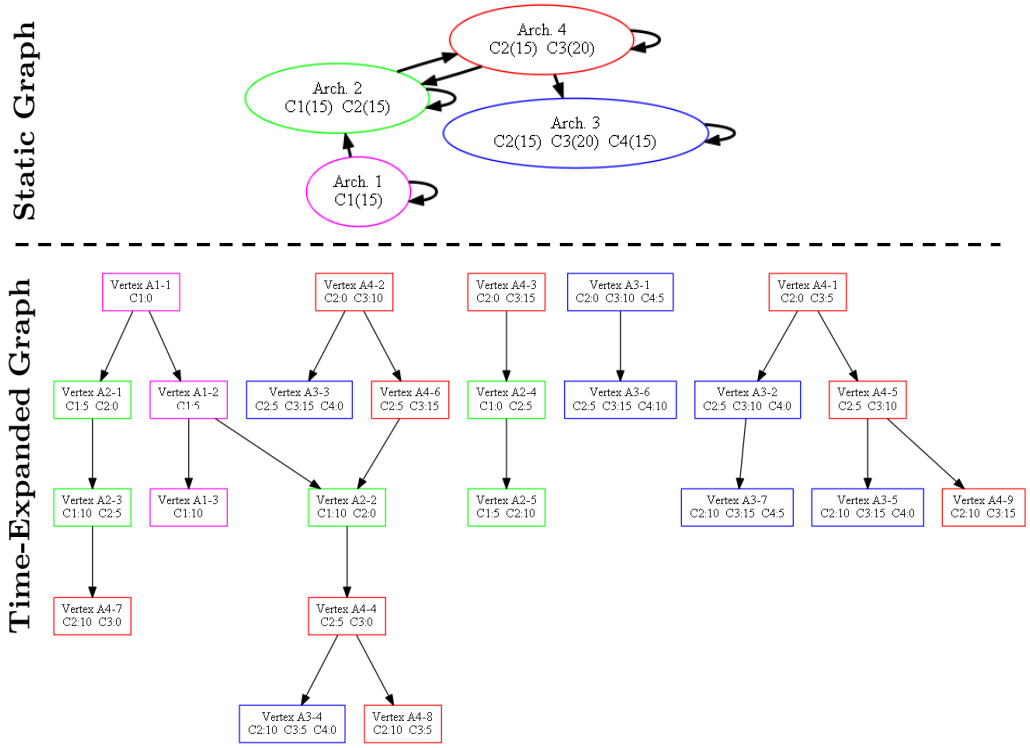


Figure 5-3: Comparison between static and time-expanded SCaN Tradespace graphs for a sample four-architecture tradespace.

5.1.3 Edge Weight: Incorporating Recurring Costs

With the structure of the time-expanded graph defined, the final step is to calculate the edge weights for the newly created edges. Whereas in the static case the only contribution to the edge weight was the non-recurring cost of the added constellation, with the modeling of the passage of time across an edge, we can now incorporate the cost of maintaining and operating the constellations on orbit.

$$C_{rec} = \begin{cases} \textit{operations-cost} & \text{for procurement} \\ \textit{service-fee} & \text{for hosted-payload} \\ \textit{service-fee} \times \frac{ops}{ops+ant+bus+payl+launch} & \text{for commercial} \end{cases}$$

The recurring cost for each constellation is given in the SCaN model as the recurring cost over the entire lifetime of the constellation. Since we model the passage of time in periods equal to the globally defined time step, we need to re-scale the total recurring cost by the time step. Depending on the contract modality, we also need to take into account the recurring component of the service fee, which is calculated using the same strategy as in Section 4.2.3.

With the recurring cost per time period of all constellations calculated, the edge weight is then equal to the recurring cost per time period of all constellations in the source architecture, plus the non-recurring cost of the constellation added across the edge (i.e. the weight of the static edge). Note that if the time-expanded edge points to a vertex with the same architecture, no constellation is added to the architecture so the non-recurring component of the edge weight is equal to zero. The rule for edge weight calculation is shown below, with L^i the lifetime in years of the i^{th} constellation in the source vertex, C^i_{rec} equal to the total recurring cost of the i^{th} constellation, T is the time step length in years and C^*_{nrec} is the non-recurring cost of the constellation added across the edge (i.e. the static graph edge weight).

$$W_{time-expanded} = C^*_{nrec} + T \sum_i \frac{C^i_{rec}}{L^i}$$

This rule for weight calculation makes explicit the assumption that each vertex is a snapshot of the architecture at the *beginning* of the time step. That is to say, when we traverse an edge the architecture during the five year time step is the architecture of the source vertex since we do not include the operating cost of the added constellation in the edge weight. This modeling decision is equivalent to saying that the added architecture is developed during the five year time step and deployed for operation at the end of the step period.

5.2 Tradespace Exploration Results: Time-Expanded Graph

In this section we will introduce and analyze several sets of results drawn from the time-expanded tradespace graph developed above. We will see that this model of the architecture decision process provides us with a more complex view of the development pathway relative to the static graph model by maintaining continuity in architecture assets across time. By modeling the aging of in-space assets explicitly, we will see in all the results that the time-expanded graph allows us to constrain the system evolution to a specific planning 'horizon', creating for example a 30-year evolution plan that supplements the type of long-term planning that NASA is currently undertaking as part of the SCaN project.

In the first set of results we explore the variety of information that can be drawn from the time-expanded tradespace graph using a simple example evolution, and discuss in more detail how time constraints on the evolution can be implemented to construct development pathways of fixed duration.

In the second result we examine in detail the question of how the architecture decision of the number of satellites per plane affects the evolutionary pathways available to the system architect. Namely, we ask how introducing single satellite constellations (i.e. one satellite per plane) to the tradespace changes the minimum-cost evolutions through the tradespace.

Decision/Property	**Result 1**	**Result 2**	**Result 3**	**Result 4**
Payload Selection	**All**	**All**	**All**	**All**
Payload Allocation	**All**	**All**	**All**	
Satellites per Plane	$\mathbf{1, 2, 3}$	$\mathbf{1, 2, 3}$	$\mathbf{1, 2, 3}$	
Contract Modalities	Proc	Proc	**Proc, Hosted**	
Orbit	GEO	GEO	GEO	
Number of Planes	1	1	1	
Constel Lifetime	15	15	15	
Network Type	Bent-Pipe	Bent-Pipe	Bent-Pipe	
# of Architectures	365	365	539	365
# of Vertices	$2,157$	$2,157$	$3,193$	$2,157$
# of Edges	$12,914$	$12,914$	$21,978$	$12,913$
Final Architectures	$B > 0.95$	$B > 0.95$	$B > 0.95$	$B > 0.95$
# Final Architectures	82	72	112	82
# Final Architectures	492	432	672	492

Table 5.1: A summary of the tradespace decisions and tradespace graph properties for each set of time-expanded graph results.

Third, we analyze the same hosted-payload questions studied in Section 4.3.3, this time from the perspective of the time-expanded graph. We wish to understand how introducing the natural constraints of asset aging and replacement affects which hosted payload are present in the best evolutions and how much on average each type of hosted payload reduces development cost over the lifetime of the system.

Finally, our fourth result analyzes a problem of particular interest to current SCaN stakeholders relating to the expected lifetime of TDRS. Since there is a possibility that the current TDRS constellation will outlive its design lifetime, we are interested in how a plan can be formulated which minimizes development cost while maintaining system functionality regardless of whether the TDRS constellation's operational lifetime is extended.

As in Chapter 4, we present in tabular form the decision variables for the tradespaces used in each of the analysis cases described above, and give a few basic statistics for each of the time-expanded tradespace graphs. This information is given in Table 5.1.

5.2.1 Result 1: Exploring the Time-Expanded Graph and Path Constraints

For this first set of results drawn from the time-expanded graph, we present a number of new analysis methods and explore the information contained in the time-expanded graph rather than draw conclusions or present recommendations related to a current decision making problem.

Graph Preliminaries

The tradespace used for this first analysis case is described in the second column of Table 5.1. It is worth noting that this is the same tradespace used in Result 2 of Chapter 4. The tradespace itself is shown in Figure 5-4 plotted along the usual metrics of lifecycle cost and benefit for each architecture.

The initial architecture chosen for this analysis case (and all subsequent analysis cases in this section) is the representative TDRS architecture with three satellites carrying two Tri-band (Ka, Ku, and S band) antennae. Since each vertex in the time-expanded graph is defined by an architecture paired with a set of constellation ages, the initial vertex in our graph must also specify an age of the TDRS constellation. For simplicity, we assume that the age of our starting TDRS vertex is zero, which means that the TDRS constellation remains part of the architecture for three subsequent time steps (or edge hops) through the development path.

The set of final architectures, also shown in Figure 5-4 consists of all architectures with benefit greater than 0.95; once again modeling the stakeholder preference that the final architecture is in the high-performing region of the tradespace. Each of the final vertices in the time-expanded graph to which we will chart a development path must also consist of an architecture paired with a set of constellation ages, so we define the set of final vertices as all the time-expanded vertices in the graph which map to the high-performing final architectures. In other words, we define the final vertices as all the possible age combinations of the final architectures.

The tradespace above is made up 365 unique architectures, 82 of which are in

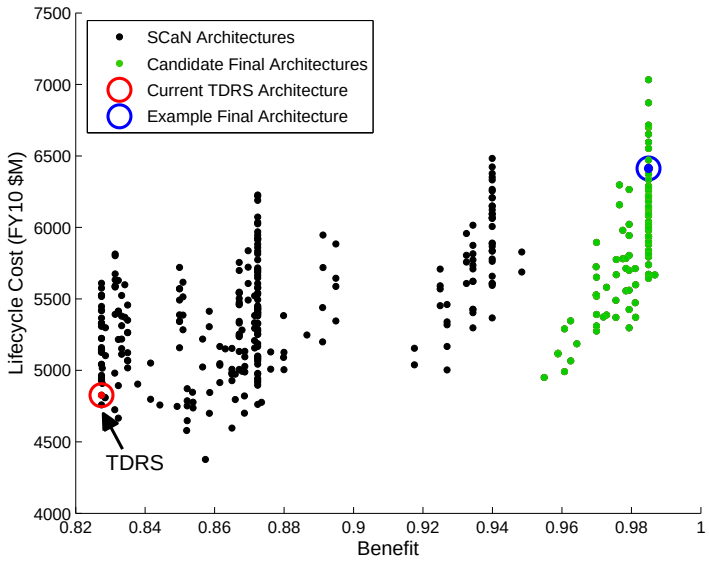


Figure 5-4: The SCaN tradespace showing initial architecture, candidate final architectures with benefit greater than 0.95, and an example final architecture.

the set of candidate final architectures. When we perform the time-expansion of vertices and definition of edges described above these architectures map to $2,157$ graph vertices connected by $12,914$ edges, with 492 vertices in the candidate final set. As in the static graph analysis of the previous chapter we will find the shortest path through the graph from the TDRS vertex to each of the 492 candidate final vertices, however before we analyze this result we first describe in more detail the form of a path through the time-expanded graph.

Example Time-Expanded Development Path

To provide an example of how a path through the time-expanded graph differs from one through the static graph, we select the vertex with constellation ages 10 and 5 corresponding to the architecture highlighted with the blue circle in Figure 5-4. The path from TDRS to this vertex is shown in Figure 5-5 plotted with architecture benefit along the y-axis and time in years on the x-axis.

The path is shown as a series of hops between points in time, with each edge in the graph corresponding to an increment of five years on the x-axis. We can see that

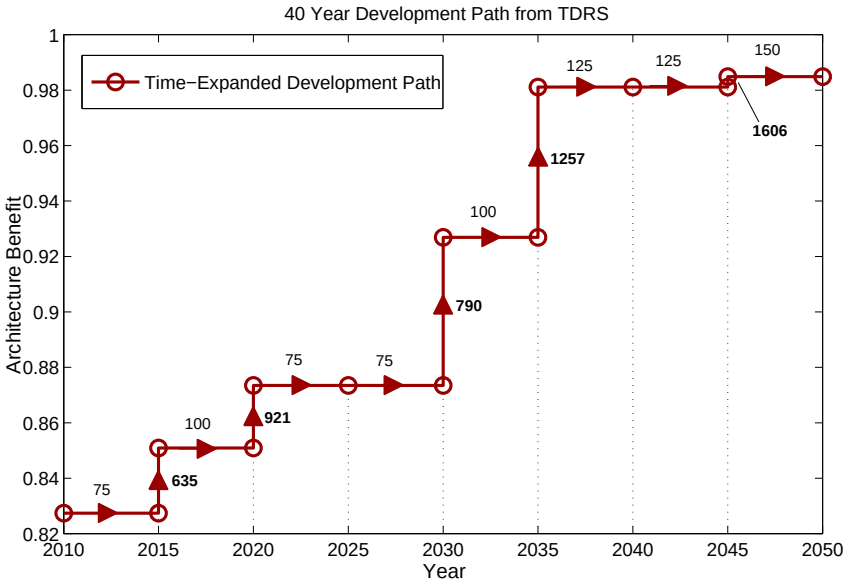


Figure 5-5: An example path through the time-expanded graph with architecture benefit plotted as a function of time. Edge weights are shown broken into recurring operations cost (horizontal steps) and non-recurring development and manufacturing cost (vertical hops).

some increments consist of just a horizontal shift, while others also add a vertical jump in benefit (note that the vertical shifts happen at the *end* of the five year interval as explained in the previous section). The horizontal shifts alone correspond to the traversal of edges across which no constellations are added to the architecture and only the operations cost of the existing assets is paid. An example of such an increment is the period from 2020 to 2025 on the plot in which benefit remains constant and the cost of \$75 million is paid to operate the current constellation for five years.

Increments where there is both a vertical and horizontal shift correspond to edges across which a constellation is added to the architecture. One such example is the interval from 2025 to 2030 where \$75 million is paid to operate the existing constellations and \$790 million is paid to develop, manufacture, and launch a new constellation at the end of the time interval.

This path through the tradespace spans an interval of 40 years, which means that the path in the graph is made up of seven edges of five years each plus an additional five years at the final vertex. Several other paths through the tradespace exist of a

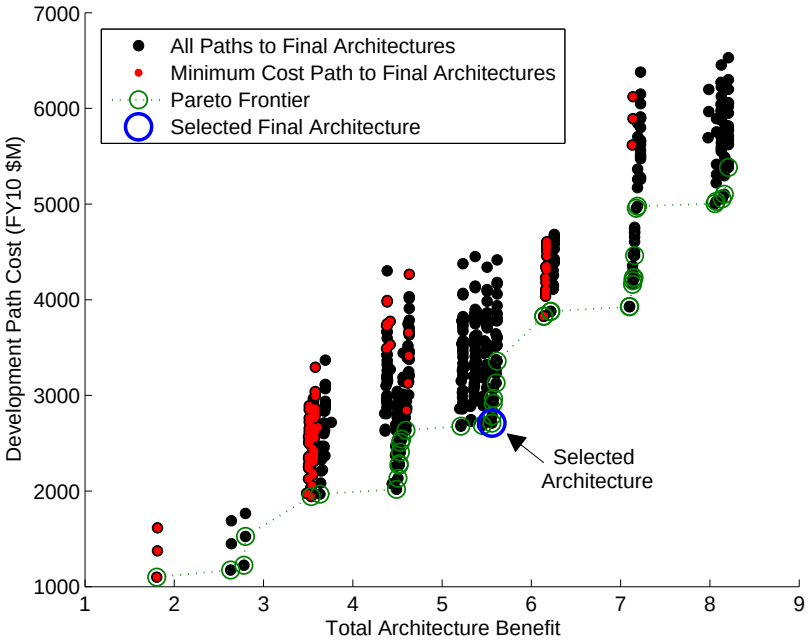


Figure 5-6: The minimum cost path to all of 492 time-expanded vertices corresponding to the 82 final architectures, plotted according to development path cost and the sum of vertex benefits along the path. The minimum development cost to each *architecture* is shown as the set of red points. A sample final vertex corresponding to a Pareto optimal 30-year development path is highlighted in blue.

variety of other lengths, but this one was chosen as an example to illustrate a case where edges are traversed along the path. The absolute dates given on the x-axis are entirely arbitrary, as we could select any year to be our starting point and then each step would be an increment of five on top of that reference date. For this example 2010 was chosen to baseline development costs in 'year zero' value, however in subsequent cases we will choose the reference date to be other points in time.

Paths to All Final Vertices

With a clearer idea of how a path through the time-expanded graph can be described, we move now to a different view of the tradespace analysis be calculating and plotting the minimum cost path to each of the 492 vertices in the final vertex set. This data is shown in Figure 5-6, with each point representing a path from TDRS to a different final vertex, as the total development path cost (which includes operations cost) on

y-axis and the *sum* of the benefits of each architecture along the path on the x-axis for the path to each final vertex. Note that contrary to the static graph analysis, for which we used average architecture benefit, we now plot the *total* benefit accrued along the path. The reason for this change is that plotting the benefit sum allows us to differentiate between paths that have different numbers of hops (and therefore occur over different intervals of time) from one another, and avoid comparing development paths of different lengths to one another.

A visually dominant feature of Figure 5-6 is the grouping of development paths into clusters separated by approximately integer increments along the x-axis. Each of these clusters corresponds to a set of paths with a particular number of edges along the path. Paths that fall in the interval between 4 and 5 on the x-axis are paths with four edges traversed through the graph, which covers an interval of 25 years when including a five year operating period for the final vertex. The reason that the path clusters are separated by an increment of about one on the x-axis is that each architecture added to the path has a benefit of slightly less than one.

Therefore, by finding and plotting all the paths through the tradespace, we can then identify and isolate those paths which correspond to the development time-frame of interest to the system architect. Clearly as we extend the development time-frame the total path cost will increase, so it is essential to only compare paths of the same length when trading cost and benefit to different architectures.

Also plotted in Figure 5-6 is a set of red points, which corresponds to the minimum length paths for each architecture from the final architecture set. In other words, we look at all the vertices which map to a particular architecture, and then plot the minimum length path from among the paths to those vertices as a red point. As we can see in the plot, many of these red points are dominated by black points that correspond to non-minimum length paths to other architectures. This simple conclusion tells us that it is crucial to examine paths to all the vertices that correspond to the selected final architectures rather than just the minimum length paths for each architecture. We can see that the Pareto frontier of cost versus benefit contains almost exclusively black (i.e. non minimum length paths to an architecture) points as the

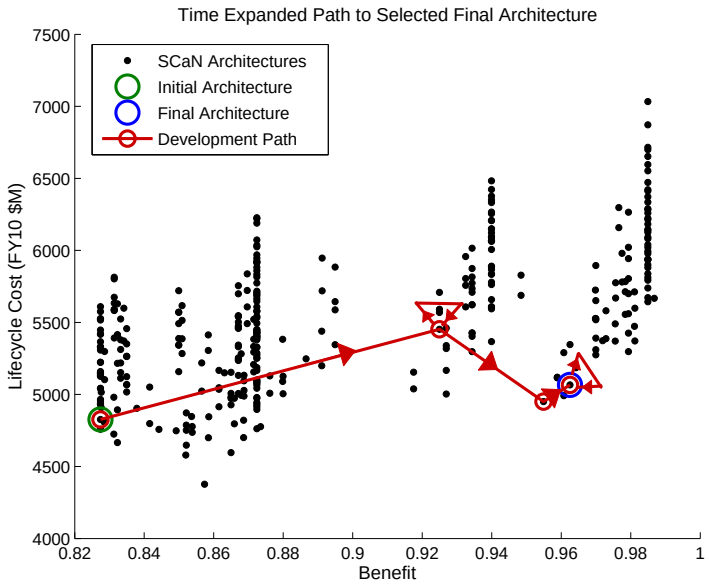


Figure 5-7: The minimum cost path through the time-expanded graph to the selected 30-year final vertex overlaid on the original cost-benefit tradespace.

development 'horizon' extends further into the future (i.e. the path contains more hops).

Example 30-Year Development Path

To further demonstrate the type of information this version of the tradespace graph can be used to analyze, we once again delve deeper into the analysis of an individual path through the tradespace. We pick one path on the Pareto frontier of Figure 5-6 (highlighted by the blue circle) to explain in detail. Figure 5-7 shows the path to this final vertex plotted on the original lifecycle cost versus benefit view of the tradespace. Each link on the path represents the traversal of an edge, with a 'self-link' corresponding to a time interval across which no constellation is added or removed from the architecture. Since there are five total hops, this development path spans a period of 30 years.

This view of the development path allows us to see the progression of architecture benefit over time and generally visualize how the architecture moves around the tradespace as it evolves. In addition, since the path incorporates hops that corre-

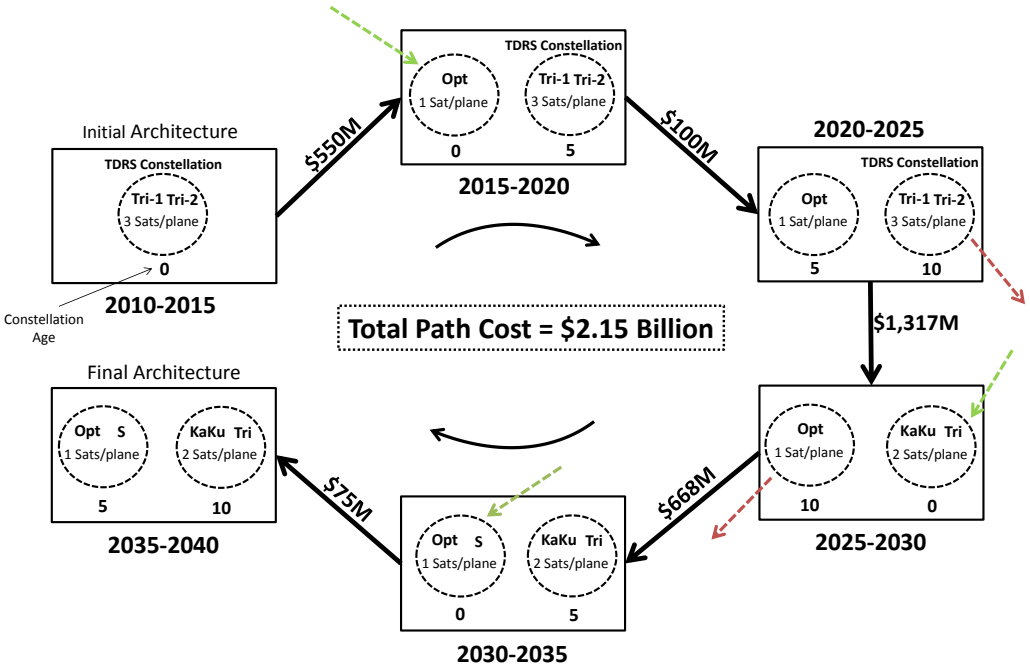


Figure 5-8: The same 30-year development path shown diagrammatically with constellation details and ages at each stage.

spond to increases in individual lifecycle cost, this result once again demonstrates that making architecture evolution decisions based on the lifecycle cost of intermediate architectures is an erroneous decision making criterion as it fails to take into account existing asset investments.

However, the plot in Figure 5-7 does not provide information as to what constellations are added or removed from the architecture at each time step and how these constellations age during the evolution. In Figure 5-8 we describe the same 30 year path diagrammatically, showing what constellations are added and removed from the architecture and the ages of these constellations at each time step.

Each green arrow in this diagram shows which constellation was added to the architecture across the incoming edge, with each red arrow showing which constellation is removed from the architecture across the outgoing edge. Since each constellation has a planned lifetime of 15 years and the time step between vertices is 5 years, note that the age of all constellations with green arrows have an age equal to zero, while all constellations with red arrows have an age equal to ten (since they operate for an

additional five years before removal).

The total development cost of this path is $2.15 billion, which accounts for both the non-recurring cost of adding constellations to the architecture and the recurring costs of operating constellations in the architecture over time. We can see that nearly half of the development cost is tied to the development and launch in 2025 of the KaKu/Tri band constellation of two satellites. The other constellations added to the architecture are single-satellite constellations that consequently require significantly smaller investments.

5.2.2 Result 2: Single-Satellite Constellations

As we saw in Figure 5-8, single satellite constellations have the potential to play a major role in the evolution of an architecture by acting as low-cost additions to the intermediate architecture. In this section we analyze what effect these single satellite constellations have on the tradespace as a whole in terms of their impact on both cost and benefit of development paths through the tradespace. In particular we wish to isolate the effect of single satellite constellations in the intermediate architectures only.

Graph Preliminaries

The tradespace of architectures used for this analysis is shown in the third column of Table 5.1, and is the same as that used in Result 1 of this chapter. As before the time-expanded tradespace graph contains $2,157$ vertices and $12,914$ edges, and the initial vertex is set as the current TDRS architecture with age equal to zero of a fifteen year lifetime.

The set of final architectures for this is slightly different from Result 1. It still consists of architectures with benefit greater than 0.95, however in this case there is an additional constraint that all final architectures must not contain any constellations with one satellite per plane. This additional constraint on the final architecture set reduces it to 72 architectures, which map to 432 final vertices.

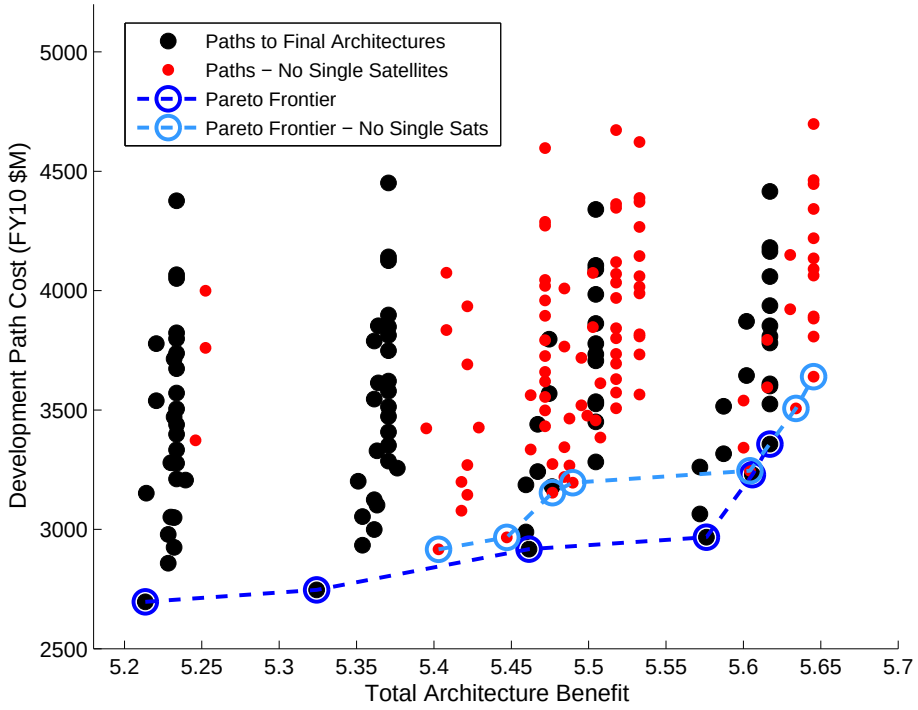


Figure 5-9: All development paths of length 30 years to 94 vertices corresponding to candidate final architectures. Red points show paths with single satellite constellations excluded from the tradespace graph; black points allow single satellite constellations in intermediate architectures.

30-Year Paths - Single Satellites Allowed

After finding the path from TDRS to each of these 432 vertices using Dijkstra's algorithm, we then impose an analysis constraint that each development path under consideration covers a time interval of 30 years, as this is the planning horizon currently under consideration by SCaN system architects. Of the 432 paths from TDRS, only 94 have length equal to 30 years. These 94 'free' paths, which may or may not contain single satellite constellations in the intermediate architectures are plotted as the black points in Figure 5-9 with the dark blue Pareto frontier. As expected there is a positive correlation between increasing development cost and increasing average benefit.

30-Year Paths - No Single Satellites

These black points give us the set of 30-year paths through the tradespace that make use of single satellite constellations, so the next step is to determine how these paths change when single satellite constellations are excluded from the development path. To perform this analysis we generate a new time-expanded graph where all architectures that contain a constellation of a single satellite are excluded. This constrained graph contains a reduced $1,583$ vertices and $8,048$ edges. Note however that all 94 final vertices from above are included in this constrained graph since we stipulated that they contained no single satellite constellations to begin with.

We then calculate the development path from TDRS to each of these 94 final vertices through the constrained tradespace graph, and determine the cost and benefit of the each 30 year path without single satellite constellations. These paths are plotted in Figure 5-9 as the red points, with the Pareto frontier of this data set shown in light blue.

It is clear upon visual inspection that the effect of excluding single satellite constellations from the intermediate architectures is to move development paths up and to the right: that is to increase both their cost and aggregate benefit. The first effect is easily explainable as single satellite constellations are less expensive by virtue of only having one spacecraft to manufacture, launch, and operate. Using these constellations as the transient constellations which are added early in the development path and then removed three steps (15 years) later reduces both the non-recurring and recurring costs paid along the path. Therefore, paths that exclude single satellites will be more costly than those with these constellations.

The second effect, that of increasing aggregate benefit, is slightly less obvious but still relatively easy to explain. All other things being equal (e.g. identical payloads, orbital plane, and antenna sizing) a constellation with two or three satellites will outperform a constellation with a single satellite. This intuitive relationship is based on the fact that two satellites gives a higher level of coverage to a greater number of customers and can deliver a higher data rate to the ground if the excess demand exists on

Fuzzy Set %	10%	20%	50%	100%
Avg. Cost Savings	\$253.8M	\$251.2M	\$253.7M	\$249.0M
% Avg. Cost Savings	7.7%	7.5%	7.2%	6.6%
Avg. Benefit Loss	0.083	0.094	0.094	0.106
% Avg. Benefit Loss	1.5%	1.7%	1.7%	1.9%

Table 5.2: Summary of average cost savings and benefit loss due to the introduction of single satellite constellations into the tradespace graph for intermediate architectures. Data shown for several sizes of the Fuzzy Pareto set from Figure 5-9

orbit, resulting in a higher benefit calculation. When single satellite constellations are excluded from the tradespace, these higher benefit multi-satellite constellations will necessarily be included in the intermediate architectures. However, when the lower cost single satellite constellations are allowed, Dijkstra's algorithm will always choose these to populate the intermediate architectures as the weight used to determine the shortest path is defined by cost alone. Consequently, excluding these single satellites from the tradespace increases aggregate benefit relative to the unconstrained shortest path.

Cost Savings Analysis

We can look quantitatively at the data in Figure 5-9 by comparing pathways through the unconstrained and constrained graphs. Since all 94 final vertices are common to both graphs, we can isolate the effect of allowing single satellite constellations by comparing the path through the unconstrained graph ending at a given final vertex to the path through the constrained graph ending at the same final vertex.

This analysis is shown in Table 5.2 as the cost savings and benefit reduction of allowing single satellite constellations averaged over a varying percentage of the entire set of 94 path pairs. For example, in the 10% Fuzzy set case, we take the 9 black points in Figure 5-9 that are closest to the Pareto front, find the red points that map to each of these points (i.e. has the same final vertex), calculate the difference in development path cost and aggregate benefit for each of these points, and average over these values.

We can see that the values for cost savings and benefit reduction remains relatively

stable across all final vertices, with savings around 7% and benefit reduction between 1% and 2%. In other words, the global effect of introducing constellations with one satellite is to decrease the development cost by 7% while also decreasing the total benefit accrued along the intermediate development path by a couple percent. Whether this analysis indicates that this trade should be made in one direction or the other is beyond the scope of this work since it depends on stakeholder preferences, the risk inherent in such a decision, and a variety of other decision factors not modeled in this framework, however we believe this type of simple aggregate analysis can be used to supplement the high-level problem understanding of the system architect.

5.2.3 Result 3: Hosted Payload Cost Savings and Prioritization

In this section we perform the identical analysis to Result 3 in Chapter 4, except in this case we run the analysis through the time-expanded version of the tradespace graph. Again, we focus on a planning horizon of 30 years, which means we only consider paths through the tradespace that contain five edges from TDRS to the final vertex. This path constraint combined with the modeling differences inherent between the static and time-expanded versions of the tradespace graph produces a new set of decision support results that supplements rather than supersedes the previously presented static hosted payload analysis.

Graph Preliminaries

The description of the tradespace used in this section can be found in the fourth column of Table 5.1, which is the identical to the tradespace used in the hosted payloads result of the last chapter and shown in Figure4-14. We use the same hosted payload constraint that any architecture that contains a hosted payload constellation can only have one such constellation, and that any such constellation can only carry one payload. We refer the reader to Section 4.3.3 for an explanation of this constraint. This tradespace consists of 539 distinct architectures, which map to $3,193$ vertices

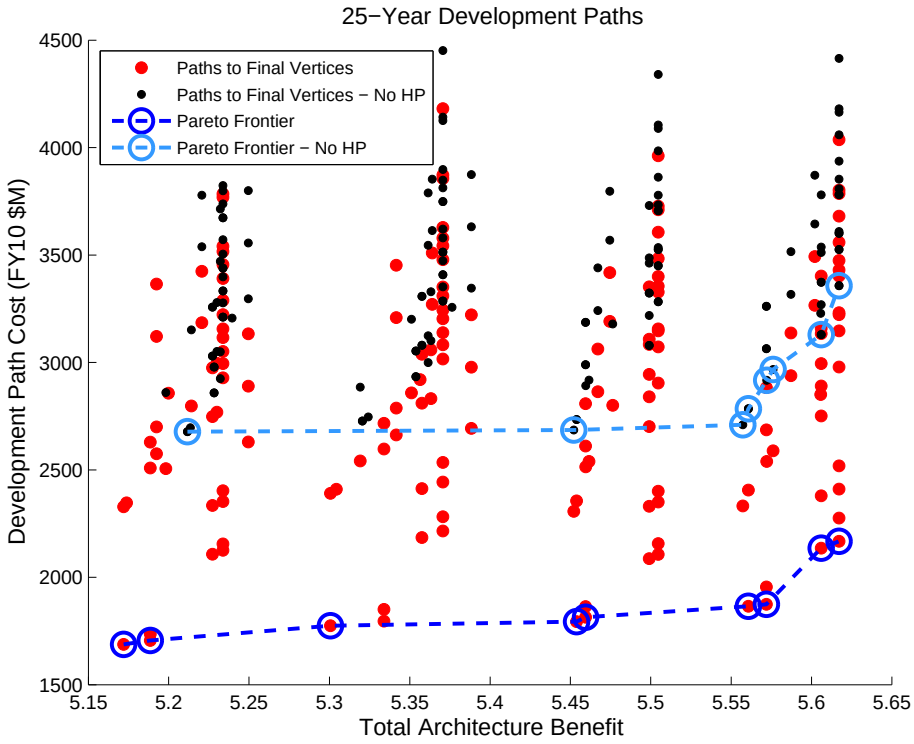


Figure 5-10: Shows all development paths with length equal to 30 years all feasible time-expanded vertices corresponding to candidate final architectures. Red points show path through tradespace with single hosted payload constellations permitted, while black points enforce constraint that all constellations are procured.

connected by 21,978 edges in the time-expanded graph.

Calculation of Paths with and without Hosted Payloads

The set of candidate final architectures is defined by the 112 architectures with benefit greater than 0.95, which map to 672 candidate final vertices. Of the paths to these candidate vertices 162 have the required length of 30 years, which define the paths through the hosted payload tradespace. Many of these 162 final vertices contain constellations with hosted payloads, so the next step as before is to map these vertices to the matching procurement-only vertices.

We now find the shortest path from TDRS to each of these 162 procurement-only vertices while excluding all architectures with hosted payloads from the development path. At this point we have two sets of paths: one set through the procurement-

only tradespace and one through the tradespace with hosted payload constellations. Further there is a one to one mapping from a path in one set to a path in the other. We plot all of these paths in Figure 5-10 along the development path cost and aggregate benefit axes, with the hosted payload paths shown in red and the procurement-only paths shown in black.

It is immediately clear from this plot that there is a much larger gap between the two sets of paths, with a vertical shift of approximately $1 billion between the two Pareto frontiers. This observation tells us that when a 30 year development time line is taken into consideration, the effect of introducing hosted payloads into the tradespace is magnified considerably. The reason for this increased savings relative to the static graph case, is that the 30 year time line dictates that there will be constellations added along the development path that reach the end of their lifetime before the final architecture is reached. These 'transient' constellations will be selected by the shortest path algorithm to be the lowest possible cost constellations that maintain continuity in the development path, which will always be a hosted payload constellation if they are present in the tradespace. Therefore, cost savings is greater because there are more steps required in the development path.

As an aside we also note that there is no observable horizontal shift in benefit between the two sets of paths as there was for the analysis of single satellite constellations. This lack of a shift makes sense since changing the contract modality from procurement to hosted payloads only affects architecture cost and not benefit.

Example Path Comparison - Maximum Savings

Of the many pairs that terminate at the same vertex in the two path sets, we select one pair to describe the details of the evolution diagrammatically in Figure 5-11. Note that the red outline indicates that a constellation contains a hosted payload. The two features that tell us that two paths form a pair (i.e. they map to one another) is that they have same starting vertex and the same final vertex minus any difference in contract modality.

Both paths are on the Pareto frontier of their respective path sets, with this pair

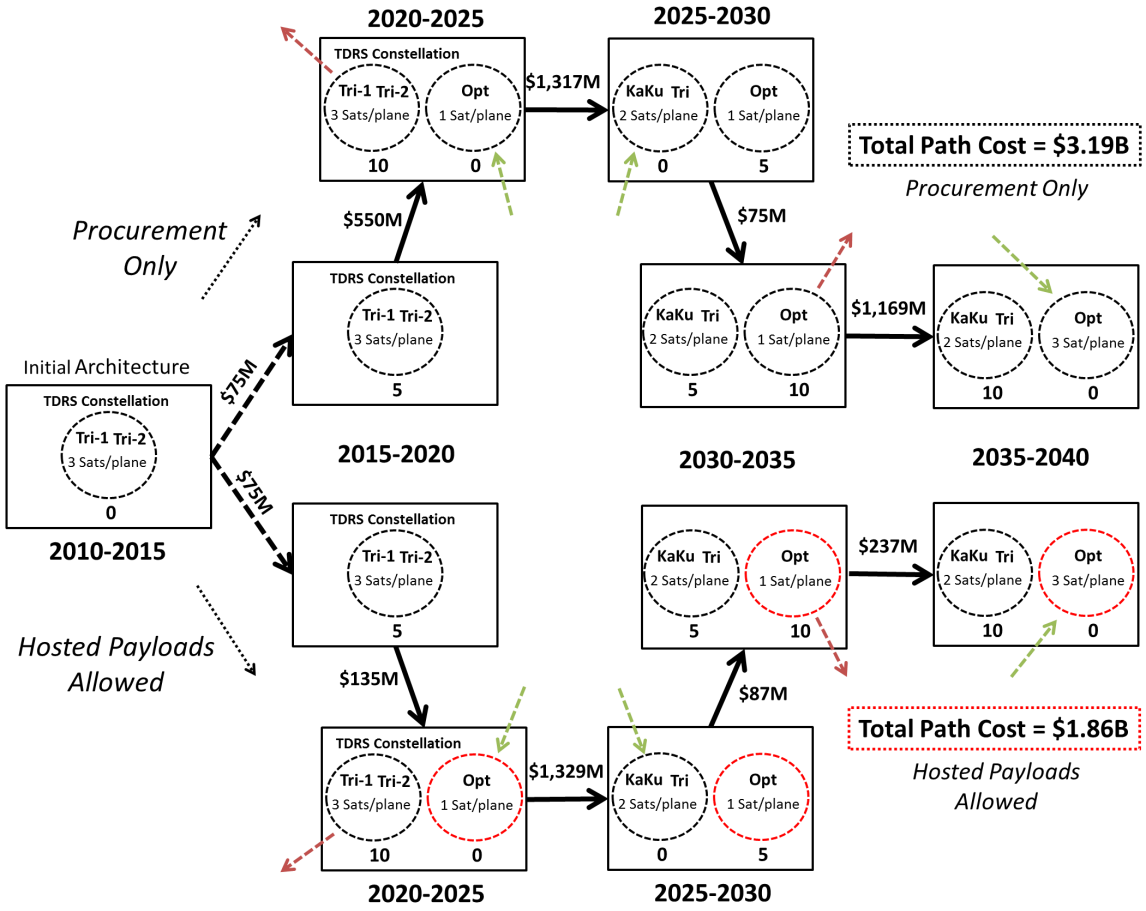


Figure 5-11: Diagrams of the matching HP and non-HP paths through the time-expanded graph that together represent a development path savings of $1.33 billion with the introduction of single hosted payload constellations into the tradespace. Red constellation outline indicates hosted payload contract modality.

also being the one which results in the maximum hosted payload savings. Over the 30 year development interval, this savings is calculated to be $1.33 billion.

Looking through all the steps along the upper evolution (procurement-only path) and the lower evolution (hosted payload path) we see that the only difference between each decision is the contract modality of the constellation added across the edge. We must emphasize that this result is not general, and that in many other path pairs the added constellations differ in both payload and number of satellites per plane as well. If we look at the edge costs next to each arrow, we can see that all the cost savings comes from the non-recurring costs of the added constellations, with cost for hosted payload constellations being about a factor of five less expensive. In fact, the recurring costs of the hosted payload constellations is higher than their corresponding procurement cousins, as we can see in the difference of weight along the 2030 edge.

Breakdown of Hosted Payload Paths by Type

Turning now to the aggregate hosted payload savings data, we can once again break down the paths through the hosted payload tradespace based on the type of payload hosted along each of the 162 paths. These results are shown in Figure 5-12 for the four different sizes of the Fuzzy Pareto set to determine how the relationship changes as we examine paths further away from the Pareto frontier.

As was the case in the static graph analysis, the paths through the tradespace predominantly deploy optical hosted payloads in the best paths, with the proportion of KaKu payloads growing as we include the development paths further away from the Pareto frontier. The primary difference between this set of results and those in the static graph section is that here all the paths through the tradespace deploy a hosted payload, while there were some paths in the static analysis that used only procurement constellations. The reason for this change is the presence of the 'transient' constellations discussed above due to the 30 year length constraint. With hosted payloads available, the transient constellations are always selected to be these lower cost constellations.

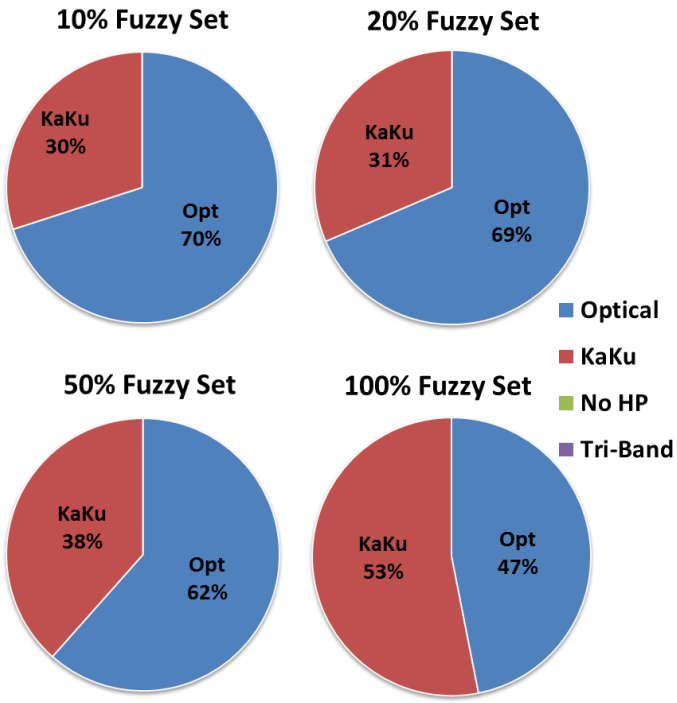


Figure 5-12: The distribution of 30-year time-expanded paths that contain each type of hosted payload for different sizes of the Fuzzy Pareto set.

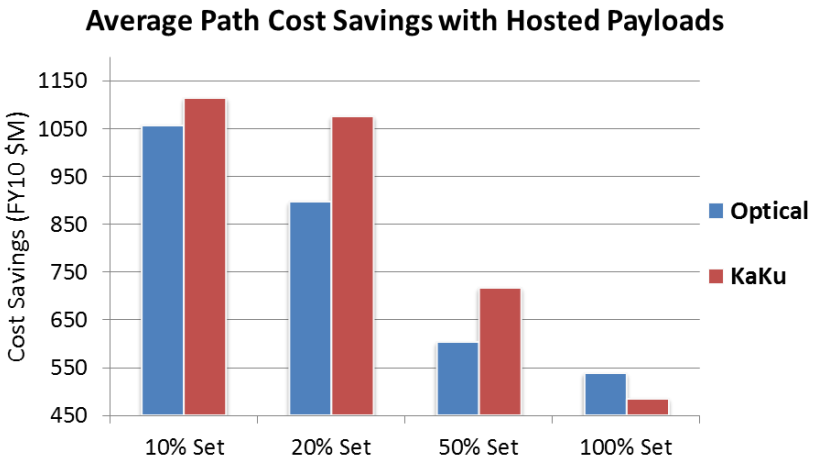


Figure 5-13: The average cost savings along the 30-year time-expanded development path broken down by payload type and Fuzzy Pareto set size.

Savings by Hosted Payload Type

Finally, we can once again calculate the savings for each pair of paths to determine the average savings realized by the introduction of hosted payloads. The plot of savings as a function of payload type and Fuzzy Pareto set size is shown in Figure 5-13. Once again we see that KaKu payloads result in marginally higher savings relative to optical payloads which is likely due to the difference in antenna cost required for the two different payload types. In contrast to the static analysis we see here that the savings for both payload types is monotonically decreasing as the Fuzzy Pareto set size increases. This indicates that the set of the best path pairs are the same whether we consider 'best' to be defined by cost saved or cost-benefit Pareto optimality, which was not the case for the static analysis.

5.2.4 Result 4: TDRS Lifetime Uncertainty Analysis

This final set of results explores a much more specific question of how uncertainty in the lifetime of the current TDRS constellation affects architecture evolution decisions. In the last three sets of results for the time-expanded graph, we have started from the TDRS-like architecture with a constellation age equal to zero in 2010. This means that the projected decommissioning date of the TDRS constellation for these

examples would be 2025. The year of 2010 was chosen as the reference time for these examples as it corresponds to the year in which costs are measured in the SCaN cost model.

TDRS Age Assumptions

However, in this set of results we will use a reference time of 2020 to begin our development path. This reference time was chosen as it is likely the earliest point at which this type of analysis could be applied to SCaN architecture decisions. With this reference date in mind, it is important to understand how the aging of current Space Network assets fits into our analysis.

The current TDRS system consists of seven operational GEO satellites grouped into three generations (what in this thesis would be called constellations). The currently operational first generation spacecraft, launched in the early 90's are far beyond their designed lifetime, so we assume that these will no longer be part of the TDRS system at the start of our development path. Similarly, the second generation of satellites launched between 2000 and 2002 are will reach the end of their projected lifetimes before the beginning of our development path. If history is a reliable indicator, there is a good chance this generation of satellites will also remain operation beyond its planned lifetime, however for the purposes of this model we will assume they will also be retired before 2020.

That leaves the third generation of TDRS satellites, of which two are currently on orbit (only one is operational at this point in time) and a third is planned for launch in 2015 [45]. As a convention, we will call this TDRS generation the *current constellation* when referring to the development path analysis. The first two satellites in the third generation were launched in 2013 and 2014 respectively, which means that the planned retirement date for the third TDRS generation is around 2030. This in turn means that the projected retirement of all current TDRS assets would occur 10 years into the development path given a starting reference date of 2020.

Problem Motivation and Framing

However, like many in-space assets (in particular other TDRS satellites) there is a non-negligible probability that the operational lifetime of the current constellation will be extended beyond its initial lifetime. For this reason a decision maker might be interested in analyzing how an evolution plan can be implemented that allows for a 'split' in the decision tree based on whether the lifetime of the initial constellation is extended.

For our analysis, we make an assumption that there is a known probability, p, that the constellation lifetime is extended by five years beyond the planned retirement date of 2030 to 2035. Starting our analysis in 2020 as mentioned above, we assume that the actual TDRS end of life is not revealed until 2025, which means that in the $2020-25$ interval decision makers must make a flexible decision that keeps good evolution options available for either TDRS retirement date. Ideally, our initial decision would reflect the estimated probability of TDRS being extended the extra fives years.

We start our analysis by defining the tradespace and building the time-expanded graph. The tradespace used is identical to that used in Results 1 and 2 of this chapter, with the decisions shown in the fifth column of Table 5.1, and the tradespace shown in Figure 5-4. The tradespace graph will also be the same with $2,157$ vertices connected by $12,914$ edges, and the final candidate set consists of the 492 vertices with architecture benefit greater than 0.95.

Definition of Path Sets

We will calculate two sets of paths to each of these 492 vertices. The first set will assume that TDRS is retired in 2030 as currently planned after 10 years of operation along the development path. This means that our starting vertex is the TDRS vertex with constellation age equal to five years. We will call this set of paths the *projected* lifetime set.

The second set of paths will assume that TDRS is extended five additional years until 2035 for 15 years of operation along the development path. However, we also

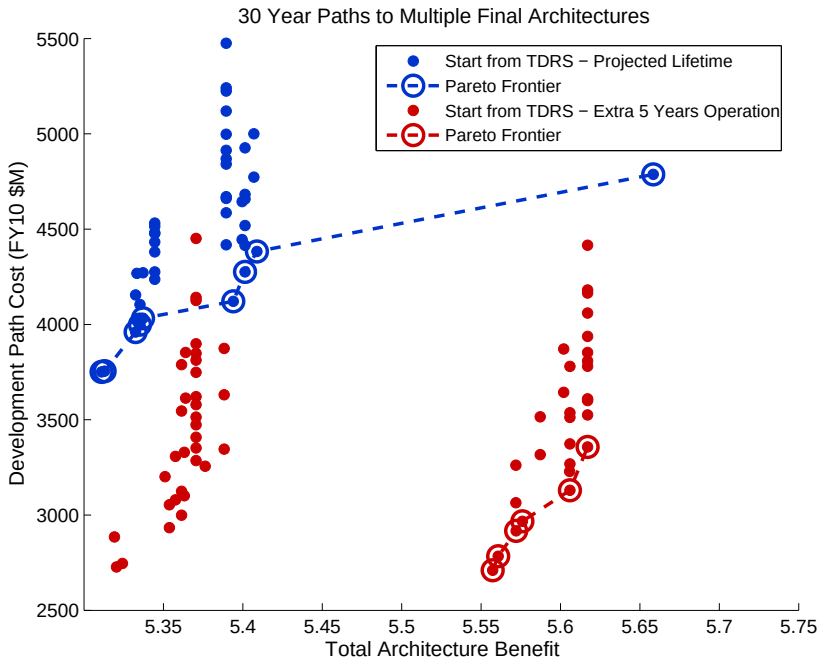


Figure 5-14: The development paths from TDRS to each candidate time-expanded vertex with a 30 year development path. Paths shown for both the planned retirement of the TDRS constellation in 2030 and for the scenario in which TDRS operations are extended five years to 2035.

assume that knowledge of this extension comes after the first time period. What this means in terms of our graph is that we will start at the TDRS vertex with age zero, but that we will not allow the first hop to go to the TDRS vertex of age five since the decision to maintain the architecture as is would never be made with the projected 2030 retirement. To account for this constraint, we simply remove the edge from TDRS age zero to TDRS age five in the graph. We call the set of paths from this 'younger' TDRS to the final vertices the extended lifetime set.

Calculation of Projected and Extended Path Sets

After defining the two sets of paths, we then reduce them both to only those paths which have length equal to 30 years. This step reduces the sets to 43 paths in the projected lifetime set and 63 paths in the extended lifetime set.

Both sets of 30 year paths are shown in Figure 5-14, with the projected lifetime set

in blue and the extended lifetime set in red. It is immediately clear that the extra five years of TDRS lifetime results in significant cost savings and benefit improvements for almost all final vertices. The source of cost savings comes from the fact that extending TDRS by five years reduces the number of constellations that must be developed, since we have five years of additional lifetime from TDRS. The shift to the right of many of the extended lifetime paths results from the presence of TDRS in the architecture for an additional time step, which increases the benefit of that architecture and improves the aggregate value.

Pairing of Extended and Projected Paths

Although this global result is valuable, of true interest to decision makers is what architecture decision should be made during the first time step to ensure flexibility to TDRS either being extended or not in the next time step. To assess the performance and cost of the first evolution decision (i.e. what edge is selected as the first in the evolution) we will pair paths from the projected lifetime set with paths from the extended lifetime set and assess the *expected* development path cost and aggregate benefit based on the probability of the TDRS lifetime being extended.

To be more precise, we find all the pairs of paths (one from each set) that share the same second architecture (not the same vertex, since the TDRS constellation age is different between the two sets). In other words we pair two paths if the same constellation is added to the architecture in the first step. With p equal to the estimated probability that TDRS will be extended for an additional five years (i.e. the extended set path will be taken), we define the expected development cost of the path pair as:

$$E[C] = p \times C_{Extended} + (1 - p) \times C_{Projected}$$

and the expected total benefit of the path pair as:

$$E[B] = p \times B_{Extended} + (1 - p) \times B_{Projected}$$

where C is the total development path cost and B is the aggregate benefit along

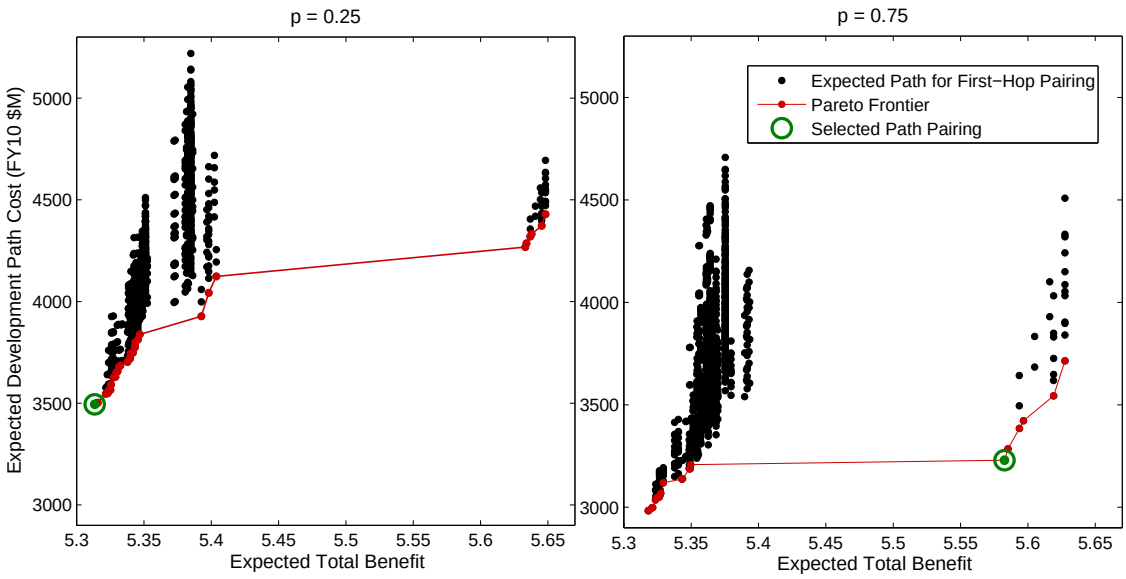


Figure 5-15: Expected development path cost and average architecture benefit for each pair of pathways that share the same initial hop. Two values of p, the probability of TDRS constellation extension until 2035, are shown.

the 30 year development path. Analyzing all the paths in projected and extended sets, we find that there are 875 pairings between projected and extended lifetime paths which share the same initial evolution decision. We plot the expected cost and benefit for all of these pairings in Figure 5-15 for two values of p: $p = 0.25$ on the left and $p = 0.75$ on the right.

As expected, increasing the probability of TDRS being extended to 2035 decreases the expected cost of the development path as we can see in the relative vertical offset between the two plots. However, this change between the two probabilities is not simply a vertical shift or a scaling of the scatter plot, rather the structure of the plot itself changes, causing the pairings on the Pareto frontier to change between the two cases. The grouping of pairings on the right of the plot are predominantly those for which an optical payload is added in the first common step. This plot suggests that this initial optical constellation is much better suited to a high probability of TDRS extension, perhaps since it operates in synergy with the large tri-band TDRS constellation, and the extra five years of TDRS operation extend the benefits of this synergy.

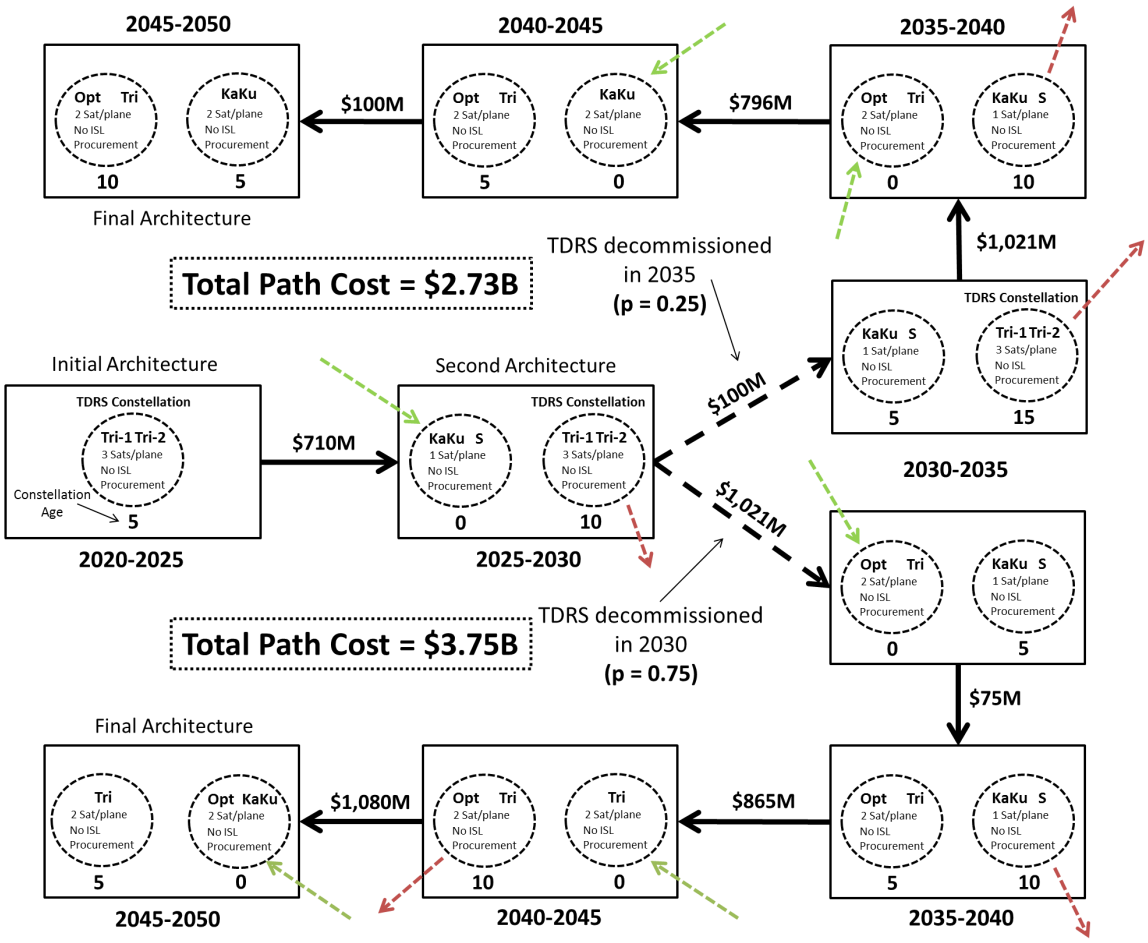


Figure 5-16: Diagram of the two possible evolution branches that are available given the selection of the first hop in the tradespace to minimize expected development cost with $p = 0.25$. This selection is shown as the green circle in the left plot of Figure 5-15

Exploring Paired Paths - $p = 0.25$

To explore some of the Pareto optimal pairings in detail, we will pick one pairing from each of the probability assignments. For the case where $p = 0.25$ we select the minimum expected cost pairing, highlighted with the green circle in the left plot of Figure 5-15. We show both paths in Figure 5-16 as a diagrammatic decomposition of the stages in each evolution. The first step along both paths is by definition the same, and consists of a single satellite carrying a KaKu and S-band payload added to the TDRS constellation in 2025. In the following time step the path splits, with the top branch representing the case in which TDRS is extended to 2035, and the

bottom branch representing the case in which TDRS is decommissioned as planned in 2030.

We see that the two paths follow very different trajectories, and that the final vertices themselves do not even share one constellation in common since we only paired paths based on the first step taken. The total development cost path is also shown for each branch, and we see that the extended lifetime path is approximately $1 billion less than the projected lifetime path. The cause of this cost difference can be traced to the number of constellations that must be added along each branch. The extended lifetime path develops and launches three constellations over the course of the 30 year path, while the projected lifetime path adds four constellations. The recurring cost of this additional constellation accounts for the majority of cost difference.

It is important to note that although a portion of this cost difference can be accounted as direct savings from the TDRS extension, a portion of it is also not actually savings, since the constellation ages in the final vertices differs. For the extended lifetime path, our final vertex has constellations of ages 5 and 10, which indicates that there are 15 'satellite-years' remaining on orbit at the end of the path. For the projected lifetime path, the final constellations have ages of 0 and 5, which means that there are 25 satellite-years remaining. In other words, at the end of both development paths, the amount of 'value' present on orbit differs between the two cases, which makes a direct development cost comparison erroneous. An analysis of this remaining value has been left to future work.

Exploring Paired Paths - $p = 0.75$

As a final example, we look in detail at a pair of development paths that is Pareto optimal for the case in which $p = 0.75$. Rather than select the minimum expected cost path, we choose a Pareto optimal pairing further along the frontier, shown as the green circle in the right plot of Figure 5-15.

Figure 5-17 details the steps along this path diagrammatically, again with the two paths shown as the two branches from the second path vertex. In this case note that the first constellation added to the existing TDRS assets is an optical payload,

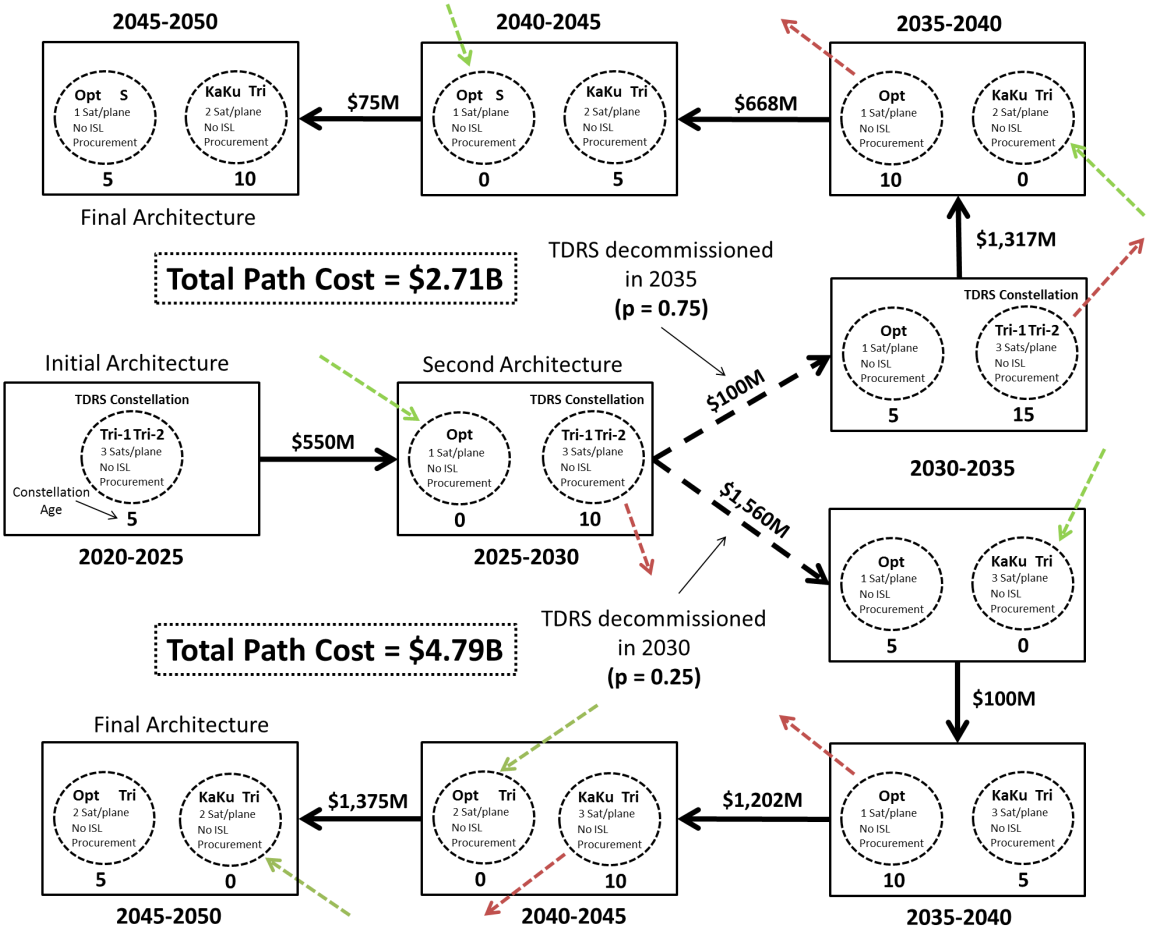


Figure 5-17: Diagram of the two possible evolution branches that are available given the selection of the first hop in the tradespace to lay on the expected development cost - expected benefit Pareto front with $p = 0.75$. This selection is shown as the green circle in right plot of Figure 5-15.

which agrees with our previous conclusion that high values of p favor optical payloads as the common constellation added. We note that the development path along the lower (projected lifetime) branch is significantly more costly than the lower branch from the $p = 0.25$ example, but that this high cost plays a much smaller role in the calculation of expected cost and benefit. We can see then that this selection of the first decision step heavily favors the extended lifetime branch and would be a poor fit for the scenario in which TDRS was decommissioned in 2030 as initially planned.

Chapter 6

Conclusions

6.1 Thesis Overview

This thesis presents a framework for tradespace exploration based on a weighted graph representation of a tradespace of system architectures. The motivation behind the creation of this framework is the fundamental assertion that architecture-level changes in complex systems are exceedingly costly, but that in many cases these changes are made necessary by external events or are desirable in order to improve system performance or reduce recurring costs.

At the heart of the tradespace graph framework is the modeling of relationships between architectures, specifically the feasibility and cost of transforming from one architecture to another. Encoding architecture relationships in a graph gives structure to a set of highly ambiguous tradespace exploration problems, and allows us to implement a variety of domain-independent quantitative tools developed over the past several decades in the field of graph theory to solve them.

The tradespace graph is introduced in Chapter 2 in an application-independent framework that is flexible enough to be applied in a variety of complex systems domains, but is not so abstract as to require the user to rebuild the graph model from the ground up for each application. Besides developing the generic rules used to transform a tradespace of point architectures into a graph, this chapter also introduces a number of vocabulary and notation conventions that are useful for clearly and

concisely discussing the framework. The final contribution of this chapter is the statement and grounding of four problem statements that help provide structure to the highly ambiguous tradespace exploration problems that decision makers might wish to analyze within this framework.

This tradespace graph framework is applied to two tradespaces: the 'HEXANE' tradespace of in-space transportation infrastructure architectures, and the 'SCaN' tradespace of space communications networks architectures. Each of these tradespaces presents unique challenges and requires a unique implementation of the tradespace graph, however the underlying structure of the graph and the framing of decision problems remain true to the generic framework presented in Chapter 2. In Chapter 3 we introduce a number of application-specific graph constraints and problem scenarios that lend additional structure to the graph and help to produce more informative and better grounded results. We present four separate sets of results drawn from the HEXANE tradespace graph, which will be summarized and discussed below.

For the SCaN tradespace application we produce two distinct manifestations of the tradespace graph: the static graph and time-expanded graph, developed and analyzed in Chapters 4 and 5 respectively. The static graph is a time-independent model of the system development process, while the time-expanded graph takes into account the aging of in-space assets and constraints on the length of the development time line as the system evolves. Each perspective provides unique views of the decision making process that together and separately produce valuable information for the system architect.

6.2 Summary of Tradespace Graph Framework

The tradespace graph framework presented in Chapter 2 transforms a tradespace of individual architectures into a directed, weighted graph that can be used to analyze a variety of architecture selection and evolution problems. Each vertex in the graph is defined by an architecture in combination with an asset portfolio, which is a collection of assets that are common to multiple architectures. Edges in the graph are

determined by relationships between the asset portfolios of two vertices. The weight of each edge, which serves as a proxy for the cost of switching the system from one vertex to another is equal to the distance between the architectures of the two vertices.

To give structure to the types of problems that this graph can be used to analyze, we define four generic problem statements. These problem statements all relate to finding the optimal evolution of architectures through the tradespace, which is a sequence of architectures between an initial and final point. We can rephrase each of these problems in terms of a shortest path problem on the graph, which allows us to implement a variety of efficient graph analysis tools such as Dijkstra's Shortest Path algorithm. We further develop these analysis tools in the individual application chapters where more explicit tradespace exploration scenarios can be defined.

6.3 Summary of Tradespace Exploration Results

6.3.1 HEXANE Tradespace

In Chapter 3 we examine four unique tradespace exploration problems both to illustrate the capabilities of the tradespace graph framework and to produce grounded decision support analysis for future system architecture decisions. The first set of results is drawn from the analysis of a decision scenario in which the functional architecture of all candidate architectures is fixed by a previous architecture selection. In this problem, we ask the question of how technology investments can be made to incrementally evolve the architecture to a higher-performing region of the tradespace. Our analysis indicates that different orderings of technology developments results in different pathways through the tradespace of varying degrees of optimality, indicating that the tradespace graph analysis can be used to prioritize technology investments.

Our second result from this chapter explores the inverse case of imposing a fixed technology portfolio constraint on the architecture evolution and analyzing how the architecture can be transformed to maximize performance from this technology portfolio. The primary conclusion from this analysis is that tight couplings exist between

technology and functional architecture in certain regions of the tradespace, which makes the system architecture very sensitive to technology or architecture disturbances.

The third and fourth results focus on the problem of selecting an initial architecture. The third set of results explores the case in which a high degree of uncertainty exists in the decision making process, relating either to the development process or the preferences for future evolution of the system. We analyze the flexibility of a set of candidate initial architectures based on their connectivity to efficient architectures in the rest of the tradespace. In the fourth result, an initial architecture is chosen based on a desired final architecture. Paths through the tradespace or plotted from a number of candidate initial architectures, with the Pareto front of development path weight vs. IMLEO performance used to analyze this decision.

6.3.2 SCaN Tradespace

The SCaN tradespace is analyzed in Chapters 4 and 5 from the perspectives of the static graph and time-expanded graph respectively. For the static graph, we find that the development paths calculated contain a variety of information related to how decisions can be sequenced to most efficiently move through the tradespace. These paths tell the decision maker what constellations need to be added in what order to make a connection in the tradespace from one architecture to another. In turn this constellation information can be decomposed to calculate global graph statistics that could not be generated without an explicit model for relationships between architectures.

The first two sets of results in Chapter 4 are analyzed more in terms of a demonstration of the capabilities of the tradespace graph as applied to the SCaN tradespace rather than in terms of a set of recommendations or decision support. The third result in this chapter seeks to answer the question of how hosted payload contracts affect the decision making process and the benefit they produce in terms of savings along the development path. We find that the best paths from the current TDRS architecture to high-performing architectures are dominated by hosted optical payloads along the development paths, and that switching to a hosted contract modality for

these payloads results in an average savings of 20%-30% of the entire development cost.

In Chapter 5 we analyze the SCaN tradespace from the perspective of the time-expanded graph in four separate sets of results. Once again, the first result in this chapter is designed as a demonstration of the information that can be drawn from this version of the graph and the unique analysis tools that can be deployed in this context - most importantly the ability to constrain the length of the development path under analysis in terms of number of years.

Our second result in this chapter focuses on the question of how constellations with single satellites impact the development path from TDRS to high-performing architectures. We find that introducing single satellite constellations results in an average savings of around 7% of the total development cost at a reduction in the aggregate benefit along the path of 1%-2%. In the third set of results we analyze the same hosted payload questions as in the previous chapter, except in this case constrain the development path to be 30 years long. This time constraint magnifies the savings achieved with the introduction of hosted payloads, and changes the breakdown of what hosted payloads are deployed along the development timeline.

Finally, our fourth result in this chapter studies the complex problem of how extending the current TDRS constellation by an extra five years impacts possible development paths through the tradespace, and what decisions should be made in the short-term to ensure flexibility to TDRS extension in the long term. We find that extending TDRS five years, results in a cost and benefit improvement on average over the 30 year development cycle. Based on the expected probability of TDRS being decommissioned as planned or extended an extra five years, we also determine the initial architecture decisions which place the expected cost and performance of the development path on the Pareto frontier.

6.4 Future Work

6.4.1 Generic Framework Extensions

A number of future directions of research exist to extend and improve the tradespace graph framework as it is presented in this thesis. Chief among these is the development of additional generic tools that focus on problems other than finding the shortest development path through the graph. With architecture relationships quantitatively encoded in the tradespace graph, a variety of other graph statistics and domain-independent algorithms can be implemented to extract a greater variety of information from the graph. Clustering algorithms could help to identify families of architectures that make up the tradespace; different concepts of centrality could be used to highlight architectures that are on the critical path of different evolutions and identify edges that if modified or added could have a large impact on the development options; and graph conversion algorithms could be developed to generate decision trees for the concise presentation of multiple paths through the tradespace.

A second avenue of pursuit is to extend the framework from its current form in which vertices and edges are static (i.e. the graph itself does not change), to one in which relationships between vertices, and even vertices themselves, are dynamic. Such an extension would require a new set of analysis methods, but would allow the modeling of much more complex relationships between the architecture, asset portfolio, time, and the external environment (e.g. system demand or cost fluctuations).

6.4.2 HEXANE Tradespace Application

The application of the tradespace graph to the HEXANE tradespace presents several possibilities for extension and refinement as the work in this thesis has used just a portion of the tradespace enumeration capabilities of the HEXANE model. In this thesis, all architectures in the HEXANE tradespace had identical mission parameters (i.e. every architecture did the exact same thing). A much richer set of analysis would be possible if mission duration, number of crew and science payload were allowed to

vary from architecture to architecture. Such analysis would require a model of overall system performance based on mission parameters, along with a way in which to determine system cost that combines IMLEO and TDC metrics.

Another area of future work to improve the HEXANE results is to develop a more accurate (and likely more complex) metric for determining the switching cost between architectures. This metric would likely take into account both the functional architecture as we have defined it, and the need to develop new elements of form versus simply modifying existing ones.

The final area of future work relating to the HEXANE tradespace, which is also the most promising in terms of the long term value of the model, is the ability to analyze 'multi-destination' paths through the tradespace. This type of work would examine how an architecture can be developed over the long term to fulfill all of NASA's in-space transportation needs as both mission requirements and destinations change. This problem scenario is in line with NASA's current 'flexible path' policy of sending humans to Mars over the coming decades, and as such the type of analysis produced within the tradespace graph framework would be highly valuable to NASA decision makers currently exploring future evolution options.

6.4.3 SCaN Tradespace Application

Finally, the application of the graph framework to the SCaN tradespace also offers a number of promising options for future work as the SCaN model itself continues to evolve to generate more complex and richer tradespaces. In the work presented in this thesis only the evolution of the in-space segment of the Space Network was analyzed, however the ground segment also offers a rich decision space that NASA is currently exploring. Incorporating the ground station decisions into the tradespace graph model would help to make the decision support model richer and more valuable to SCaN stakeholders.

Currently the SCaN tradespace model is moving in the direction of exploring architectures at a higher degree of fidelity by modeling and defining the architecture of each satellite in each constellation as a unique entity. Mirroring this process in

the rules that define our tradespace graph model would increase the resolution of our decision model. Another relatively simple possibility for future work in this area is to incorporate constellation decommissioning costs into the definition of edge weight. Although this would likely not have a major impact on the structure of the graph, it would help to make the decision model more palatable to potential users.

Finally, as we have the ability to analyze paths through the tradespace that last several decades, a crucial addition to the tradespace graph will be the ability to model a demand profile that evolves over time. This will mean that architecture benefit will no longer be a constant between vertices with the same architecture, which presents a variety of challenges in terms of the current implementation of the methods to find development paths through the tradespace.

Bibliography

[1] Dale Arney. *Rule-Based Graph Theory to Enable Exploration of the Space System Architecture Design Space*. PhD thesis, Georgia Institute of Technology, 2012.

[2] Dale Arney and Alan Wilhite. A Flexible Modeling Environment for Evaluating Space System Architectures. In *AIAA Modeling and Simulation Technologies Conference*, Toronto, 2010.

[3] Dale Arney and Alan W. Wilhite. Modeling Space System Architectures with Graph Theory. *Journal of Spacecraft and Rockets*, Advance on, 2014.

[4] Arman Avadikyan and Patrick Llerena. A Real Options Reasoning Approach to Hybrid Vehicle Investments. *Technological Forecasting and Social Change*, 77(4):649–661, 2010.

[5] Jonathan A. Battat, Bruce Cameron, Alexander Rudat, and Edward F. Crawley. Technology Decisions Under Architectural Uncertainty. *Journal of Spacecraft and Rockets*, 5(5), 2013.

[6] David A. Bearden. A Complexity-Based Risk Assessment of Low-Cost Planetary Missions: When is a mission too fast and too cheap? *Acta Astronautica*, 52(2):371–379, 2003.

[7] Michel Benaroch and Robert J. Kauffman. A Case for Using Real Options Pricing Analysis to Evaluate Information Technology Project Investments. *Information Systems Research*, 10(1):70–86, 1999.

[8] Geoffrey Boothroyd and Peter Dewhurst. *Product Design for Assembly*. Boothroyd Dewhurst Incorporated, 1987.

[9] Alan Brown and Mark Thomas. Reengineering the Naval Ship Concept Design Process. In *Ship Systems Engineering Symposium, ASNE*, 1998.

[10] Tyson R. Browning. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management*, 48(3), 2001.

[11] Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest Path Algorithms: Theory and Experimental Evaluation. *Mathematical Programming*, 73(2):129–174, 1996.

[12] Committee on Assessment of Impediments to Interagency Cooperation on Space and Earth Science Missions. Assessment of Impediments to Interagency Collaboration on Space and Earth Science Missions. Technical report, National Research Council, Washington D.C., 2011.

[13] Zvi Covaliu and Robert M. Oliver. Representation and Solution of Decision Problems Using Sequential Decision Diagrams. *Management Science*, 41(12):1860–1881, 1995.

[14] Edward F. Crawley, Olivier L. de Weck, Steven Eppinger, Christopher Magee, Joel Moses, Warren Seering, Joel Schindall, David Wallace, and Daniel Whitney. Engineering Systems Monograph: The Influence of Architecture in Engineering Systems. In *Engineering Systems Symposium*, Cambridge, MA, 2004.

[15] William H.E. Day. The Complexity of Computing Metric Distances Between Partitions. *Mathematical Social Sciences*, 1(3):269–287, 1981.

[16] Olivier de Weck, Richard de Neufville, and Mathieu Chaize. Staged Deployment of Communications Satellite Constellations in Low Earth Orbit. *Journal of Aerospace Computing, Information and Communication*, 1(3):119–136, 2004.

[17] Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[18] Dov Dori. *Object Process Methodology: A Holistic Systems Paradigm*. Springer, 2002.

[19] Claudia Eckert, P. John Clarkson, and Winfried Zanker. Change and Customisation in Complex Engineering Domains. *Research in Engineering Design*, 15(1):1–21, 2004.

[20] Andreas Eder and Matthias Linde. Efficient and Dynamic - The BMW Group Roadmap for the Applications of Thermoelectric Generators, 2011.

[21] Steven Eppinger. Innovation at the Speed of Information. *Harvard Business Review*, 79(1):149–158, 2001.

[22] Michael L. Fredman and R.E. Tarjan. Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms. In *25th Annual Symposium on Foundations of Computer Science*, San Diego, CA, 1984. IEEE.

[23] Ernest Friedman-Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications, Greenwich, CT, 2003.

[24] Joseph C. Giarratano and Gary D. Riley. *Expert Systems: Principles and Programming*. Course Technology, fourth edition, 2004.

[25] R. Douglas Hamelin, David D. Walden, and Michael E. Krueger. *INCOSE Systems Engineering Handbook*. International Council on Systems Engineering, 2011.

[26] Khaled Haris and Cihan H. Dagli. Adaptive Reconfiguration of Complex System Architecture. *Procedia Computer Science*, 6:147–152, 2011.

[27] W. Michael Hawes and Michael R. Duffey. Formulation of Financial Valuation Methodologies for NASA's Human Spaceflight Projects. *Project Management Journal*, 39(1):85–94, 2008.

[28] Kurt Heidenberger and Christian Stummer. Research and Development Project Selection and Resource Allocation: A Review of Quantitative Modelling Approaches. *International Journal of Management Reviews*, 1(2):197–224, 1999.

[29] Rebecca M. Henderson and Kim B. Clark. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Administrative Science Quarterly*, 35(1):9–30, 1990.

[30] Wilfried Hofstetter. *A Framework for the Architecting of Aerospace Systems Portfolios for Commonality*. PhD thesis, Massachusetts Institute of Technology, 2009.

[31] Benjamin H. Y. Koo, Willard L. Simmons, and Edward F. Crawley. Algebra of Systems: A Metalanguage for Model Synthesis and Evaluation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(3):501–513, May 2009.

[32] Ram Kumar, Haya Ajjan, and Yuan Niu. Information Technology Portfolio Management: Literature Review, Framework, and Research Issues. *Information Resources Management Journal*, 21(3):64–87, 2008.

[33] Matthew J. Liberatore and Anthony C. Stylianou. The Development Manager's Advisory System: A KnowledgeBased DSS Tool for Project Assessment. *Decision Sciences*, 24(5):953–976, 1993.

[34] Mary L. Maher and Josiah Poon. Modeling Design Exploration as Co-Evolution. *Computer-Aided Civil and Infrastructure Engineering*, 11(3):195–209, 1996.

[35] Mark W. Maier and Eberhardt Rechtin. *The Art of Systems Architecting*. CRC Press, New York, NY, third edit edition, 2009.

[36] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary Design Optimization: A Survey of Architectures. *AIAA Journal (Submitted for Publication)*, 2012.

[37] Hugh L. McManus and Joyce M. Warmkessel. Creating Advanced Architectures for Space Systems: Emergent Lessons from New Processes. *Journal of Spacecraft and Rockets*, 41(1):69–74, 2004.

[38] Ali Farhang Mehr and Irem Y. Tumer. Risk-Based Decision-Making for Managing Resources During the Design of Complex Aerospace Systems. *Journal of Mechanical Design*, 128(4):1014–1022, 2006.

[39] Abraham Mehrez. Selecting R&D Projects: A Case Study of the Expected Utility Approach. *Technovation*, 8(4):299–311, 1988.

[40] Taiga Nakamura and Victor R. Basili. Metrics of Software Architecture Changes Based on Structural Distance. In *Proc. of the 11th IEEE International Software Metrics Symposium*, page 8, 2005.

[41] NASA. Human Exploration of Mars Design Reference Architecture 5.0. Technical report, NASA, 2009.

[42] NASA. Preliminary Report Regarding NASAs Space Launch System and Multi-Purpose Crew Vehicle. Technical report, 2011.

[43] NASA. Space Communications and Navigation (SCaN) Network Architecture Definition Document (ADD) Volume 1: Executive Summary. Technical report, 2011.

[44] NASA. Space Communications and Navigation: Program Overview, 2013.

[45] NASA Goddard Space Flight Center. Tracking and Data Relay Satellite, 2014.

[46] National Research Council. The Global Positioning System: A Shared National Asset. Technical report, The National Academies Press, Washington D.C., 1995.

[47] Office of the USAF Chief Scientist. United States Air Force Technology Horizons 2010-2030. Technical report, 2010.

[48] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, 2010.

[49] Iris Reinhartz-Berger, Dov Dori, and Shmuel Katz. OPM/Web Object-Process Methodology for DevelopingWeb Applications. *Annals of Software Engineering*, 13(1):141–161, 2002.

[50] Adam M. Ross and Daniel E. Hastings. The Tradespace Exploration Paradigm. In *INCOSE International Symposium 2005*, pages 13–25, 2005.

[51] Alexander Rudat. *HEXANE: Architecting Manned Space Exploration Missions Beyond Low Earth Orbit*. Master's thesis, Massachusetts Institute of Technology, 2013.

[52] Alexander Rudat, Jonathan A. Battat, Bruce Cameron, Morgan Dwyer, and Edward F. Crawley. Tradespace Exploration Approach For Architectural Definition of In-Space Transportation Infrastructure Systems For Future Human Space Exploration. In *International Astronautical Conference*, 2012.

[53] Joseph H. Saleh, Elisabeth Lamassoure, and Daniel E. Hastings. Space Systems Flexibility Provided by On-Orbit Servicing: Part 1. *Journal of Spacecraft and Rockets*, 39(4):551–560, 2002.

[54] Marc Sanchez, Daniel Selva, Bruce Cameron, Edward F. Crawley, Antonios Seas, and Bernie Seery. Exploring the Architectural Trade Space of NASAs Space Communication and Navigation Program. In *IEEE Aerospace Conference*, 2013.

[55] Marc Sanchez, Daniel Selva, Bruce Cameron, Edward F. Crawley, Antonios Seas, and Bernie Seery. Results of the MIT Space Communication and Navigation Architecture Study. In *IEEE Aerospace Conference*, 2014.

[56] Daniel Selva. *Rule-Based System Architecting of Earth Observation Satellite Systems*. PhD thesis, Massachusetts Institute of Technology, 2012.

[57] Daniel Selva and Edward F. Crawley. VASSAR: Value Assessment of System Architectures using Rules. In *IEEE Aerospace Conference*, 2013.

[58] Robert Shishko, Donald H. Ebbeler, and George Fox. NASA Technology Assessment Using Real Options Valuation. *Systems Engineering*, 7(1):1–13, 2004.

[59] Matthew R. Silver and Olivier L. de Weck. Time-Expanded Decision Networks: A Framework for Designing Evolvable Complex Systems. *Systems Engineering*, 10(2):167–186, 2007.

[60] Willard L. Simmons. *A Framework for Decision Support in Systems Architecting*. PhD thesis, Massachusetts Institute of Technology, 2008.

[61] Dan A. Simovici and Szymon Jaroszewicz. An Axiomatization of Partition Entropy. *Theory, IEEE Transactions on Information*, 48(7):2138–2142, 2002.

[62] Atmika Singh and Cihan H. Dagli. Multi-objective Stochastic Heuristic Methodology for Tradespace Exploration of a Network Centric System of Systems. In *IEEE Systems Conference*, pages 218–223, 2009.

[63] Kaushik Sinha and Olivier de Weck. Structural Complexity Quantification for Engineered Complex Systems. In *Proceedings of the International Design Engineering Conference*, Portland, OR, 2013.

[64] Rudolf Smaling and Olivier de Weck. Assessing Risks and Opportunities of Technology Infusion in System Design. *Systems Engineering*, 10(1):1–25, 2007.

[65] Steering Committee for NASA Technology Roadmaps and National Research Council. NASA Space Technology Roadmaps and Priorities: Restoring NASA's Technological Edge and Paving the Way for a New Era in Space. Technical report, 2012.

[66] Zoe Szajnfarber, Matthew G. Richards, and Annalisa Weigel. Challenges to Innovation in the Government Space Sector. *Defense Acquisition Review Journal*, 59:257–276, 2011.

[67] Sunny Tsiao. *Read You Loud and Clear: The Story of NASA's Spaceflight Tracking and Data Network*. NASA History Division, Washington D.C., 2008.

[68] Vassilios Tzerpos and Richard C. Holt. MoJo: A Distance Metric for Software Clusterings. In *Proc. 6th Working Conference on Reverse Engineering*, pages 187–193. IEEE, 1999.

[69] U.S. Senate. National Aeronautics and Space Administration Authorization Act of 2010, 2010.

[70] Myles A. Walton and Daniel E. Hastings. Applications of Uncertainty Analysis to Architecture Selection of Satellite Systems. *Journal of Spacecraft and Rockets*, 41(1):75–84, 2004.

[71] Jin Y. Yen. Finding the k Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, 1971.