


FINAL DEGREE PROJECT

# Contributions to ITACA: A tool to architect space communication networks

*Author:*
Iñigo del Portillo

*Supervisor:*
Dr. Bruce G. Cameron

December 11, 2014

# Contributions to ITACA: A tool to architect space communication networks

by

Iñigo del Portillo

Submitted to the Escola Técnica Superior d'Enginyeria de Telecomunicació de Barcelona
in partial fulfillment of the requirements for the degree of

Telecommunications Engineer and Industrial Engineer

at the

ESCOLA TÉCNICA SUPERIOR D'ENGINYERIA DE TELECOMUNICACIÓ DE BARCELONA

December 2014



Author .....
Escola Técnica Superior d'Enginyeria de Telecomunicació de Barcelona
December 12, 2014

Certified by .....
Dr. Bruce G. Cameron
Director, System Architecture Lab
Thesis Supervisor

Accepted by .....
Eduard Alarcón
Associate Professor, Department of Electrical Engineering

# Contributions to ITACA: A tool to architect space communication networks

by
Iñigo del Portillo

Submitted to the Escola Técnica Superior d'Enginyeria de Telecomunicació de Barcelona
on December 12, 2014, in partial fulfillment of the
requirements for the degree of
Telecommunications Engineer and Industrial Engineer

## Abstract

Communication and Navigation services are critical for any space mission. Nowadays, NASA provides services to its customers by using a triple network composed of the Near Earth Network (NEN), the Deep Space Network (DSN) and the Space Network (SN). The increasing mission requirements are forcing NASA to upgrade its assets. Moreover, in order to reduce the cost of the system an integrated network that comprises the NEN, DSN and SN has been proposed. With the objective of assisting NASA's UComm Study team during the process of architecting NASA's next-generation integrated-network for space communications, MIT's System Architecture Lab has developed a computational tool called the Integrated Tradespace Analysis of Communications Architectures (ITACA).

The objective of this thesis is to present the work conducted during this past year to expand the capabilities of ITACA. This thesis focuses on 1) the implementation of crossover algorithms for the architectural decisions and 2) the improvements performed on the scheduler algorithm that assesses the performance of the network. These improvements are intended to include new features and improve ITACA's computational performance. Finally, the utility of the tool is demonstrated through a case study on the evolution of the Space Network. In particular, four trade-offs involving the constellation pattern selection, the frequency band selection, the disaggregation of payloads in multiple spacecraft and use of inter-satellite links are studied.

Thesis Supervisor: Dr. Bruce G. Cameron
Title: Director, System Architecture Lab

**Contributions to ITACA: A tool to architect space communication networks**

by

Iñigo del Portillo

Submitted to the Escola Técnica Superior d'Enginyeria de Telecomunicació de Barcelona
on December 12, 2014, in partial fulfillment of the
requirements for the degree of
Telecommunications Engineer and Industrial Engineer

## Abstract

Los servicios de comunicaciones y navegación son críticos para cualquier misión espacial. En la actualidad, NASA proporciona servicio a sus usuarios por medio de una red triple compuesta por la Near Earth Network (NEN), la Deep Space Network (DSN) y la Space Network (SN). Los crecientes requerimientos de las misiones están obligando a NASA a actualizar sus sistemas. Además, con el objetivo de reducir el coste del sistema, se ha propuesto implementar una red integrada que comprenda la NEN, DSN y SN. Para asistir al equipo UComm Study de NASA durante el proceso de diseño de la próxima generación de la red integrada de comunicaciones espaciales, el Laboratorio de Arquitectura de Sistemas de MIT ha desarrollado una herramienta computacional llamada Integrated Tradespace Analysis of Communications Architectures (ITACA).

El objetivo de esta tesis es presentar el trabajo desarrollado durante el último año para expandir las capacidades de ITACA. Esta disertación se centra en 1) la implementación de algoritmos de crossover para las decisiones arquitectónicas y 2) las mejoras realizadas en el algoritmo de planificación que evalúa el desempeño de la red. Estas mejoras tienen como finalidad añadir nuevas características y mejorar su rendimiento computacional. Finalmente, la utilidad de la herramienta se demuestra a través del estudio de un caso sobre la evolución de la Space Network. En particular, cuatro compromisos relacionados con la selección de bandas frecuenciales, desagregación de antenas en múltiples naves y el uso de enlaces entre satélites son estudiados.

Thesis Supervisor: Dr. Bruce G. Cameron
Title: Director, System Architecture Lab

**Contributions to ITACA: A tool to architect space communication networks**

by

Iñigo del Portillo

Submitted to the Escola Técnica Superior d'Enginyeria de Telecomunicació de Barcelona
on December 12, 2014, in partial fulfillment of the
requirements for the degree of
Telecommunications Engineer and Industrial Engineer

## Abstract

Els serveis de comunicacions i navegació són crítics per a qualsevol missió espacial. Actualment, NASA proporciona servei als seus usuaris per mitjà d'una xarxa triple composta per la Near Earth Network (NEN), la Deep Space Network (DSN) i la Space Network (SN). Els creixents requeriments de les missions estan obligant a la NASA a actualitzar els seus sistemes. A més a més, amb l'objectiu de reduir el cost del sistema, s'ha proposat implementar una xarxa integrada que comprengui la NEN, la DSN i la SN. Per tal d'assistir a l'equip UComm Study de NASA durant el procés de disseny de la propera generació de la xarxa integrada de comunicacions espacials, el Laboratori d'Arquitectura de Sistemes de MIT ha desenvolupat una eina computacional anomenada Integrated Tradespace Analysis of Communications Architectures (ITACA).

L'objectiu d'aquesta tesi és presentar el treball desenvolupat durant l'últim any per expandir les capacitats d'ITACA. Aquesta dissertació es centra en 1) la implementació d'algorismes de crossover per a les decisions arquitectòniques i 2) les millores realitzades en l'algoritme de planificació que avalua el comportament de la xarxa. Aquestas milloras tenen com a finalitat afegir noves característiques i millorar el seu rendiment computacional. Finalment, la utilitat de l'eina es demostra a través de l'estudi d'un cas sobre l'evolució de la Space Network. En particular, quatre compromisos relacionats amb la selecció de bandes freqüencials, desagregació d'antenes en múltiples naus i l'ús d'enllaços entre satèl·lits són estudiats.

Thesis Supervisor: Dr. Bruce G. Cameron
Title: Director, System Architecture Lab

# Acknowledgments

This thesis is the result of 11 months of research in the System Architecture Lab at MIT. I never imagined I could get the chance to study here and I would like to devote some lines to thank all of those that helped me to fulfill this dream.

I want to first thank Dr. Bruce G. Cameron, my thesis supervisor, for giving me the opportunity of carrying out this project at the System Architecture Lab. Your valuable insights on my work and all your help and advice during this last months have been indispensable for the success of this endeavor. Also, thanks for making me feel part of the team from the very first moment I arrived. I am also very grateful to Eduard Alarcón, my thesis advisor in Spain, for trusting in me and giving me the chance of coming here to do this project.

Thanks to the rest of the guys in the lab: Morgan, Peter, Jakub, Chris, Demetrios, Marc and Dani. It was a ton of fun staying the office with you guys. Special mention to Marc and Dani; we shared a lot of hours of discussion (and laughs) during the last year. I am really grateful for how you embraced me in the project and your guidance during my whole stay at MIT.

I would also like to acknowledge all my friends in Boston, that made the best of this American experience. To my very close Spanish family in Boston: Adrián, Adriá, Jordina, Jordi, Laura, Joan and Lluís, thanks for all these good moments having some drinks at the Muddy Charles; they were always a good way to relax from work. Also I have wonderful memories of the moments shared with my international friends Patricia, Marco, Pandora and also with the people from Spain@MIT, specially to all of those in the soccer team. I enjoyed a lot playing with you (and winning the intramurals championship!) these months.

Finally, my deepest gratitude goes to my family. Mamá, papá, gracias por vuestro apoyo incondicional durante todos estos años. Gracias por animarme a alcanzar mis sueños y darme todas las herramientas y libertades para que lo logrará.

Thank you so much to all of you! I will never forget this experience.

# Contents


**1 Introduction** **13**
- 1.1 Motivation . . . 13
- 1.2 Background . . . 14
  - 1.2.1 System Architecture . . . 14
  - 1.2.2 Overview of ITACA . . . 17
  - 1.2.3 Rule-Based Expert Systems . . . 20
- 1.3 Thesis Overview . . . 23

**2 Crossover Algorithms for ITACA's SAP problems** **24**
- 2.1 Introduction to SAP Problems . . . 24
  - 2.1.1 The Assigning Problem . . . 25
  - 2.1.2 The Down-selecting Problem . . . 25
  - 2.1.3 The Partitioning Problem . . . 26
- 2.2 Introduction to Genetic Algorithms . . . 27
- 2.3 Crossover algorithms for GAs. A literature review . . . 28
  - 2.3.1 Binary string crossover operators . . . 29
  - 2.3.2 Real values crossover operators . . . 29
  - 2.3.3 Matrix crossover operators . . . 29
- 2.4 Implementation of ITACA's crossover algorithms . . . 30
  - 2.4.1 Crossover for the Assigning Problem . . . 30
  - 2.4.2 Crossover for the Down-selecting Problem . . . 31
  - 2.4.3 Crossover for the Partitioning Problem . . . 32

**3 An Enhanced Version of the Scheduling Algorithm** **36**
- 3.1 Scheduling techniques . . . 36
- 3.2 Initial scheduler . . . 38
- 3.3 Enhanced version of the scheduler . . . 38
  - 3.3.1 Data Structures . . . 39
  - 3.3.2 Visibility Windows Computation . . . 40
  - 3.3.3 Variable Initialization . . . 40
  - 3.3.4 Viable Window selection . . . 41
  - 3.3.5 Allocating Resources to Services . . . 43
  - 3.3.6 Trimming Affected Visibility Windows . . . 45
- 3.4 Performance Improvements . . . 46

**4 Using ITACA. A tradespace analysis of the Space Network** **47**
4.1 Scenario Description . . . 47
4.1.1 Tradespace Characterization . . . 47
4.1.2 Customer User Base Characterization . . . 50
4.2 Trade-offs Analysis . . . 53
4.2.1 Constellation pattern analysis . . . 54
4.2.2 Antenna Allocation Analysis . . . 55
4.2.3 ISL + Ground Stations . . . 58
4.2.4 Disaggregation trade-off . . . 59

**5 Conclusions** **63**
5.1 Summary . . . 63
5.2 Future Work . . . 64
5.2.1 Tool Expansion Tasks . . . 64
5.2.2 Architectural Analysis Tasks . . . 65

**A Additional functions used in the Partitioning crossover operator** **66**

**B Architectures Evaluated in the NASA UComm Case Study** **70**

# List of Figures


1-1 SCaN network architectures . . . 14
1-2 Architecture formal decomposition in its elements . . . 18
1-3 ITACA's architecture . . . 19
1-4 Flowchart of the Architecture Evaluator . . . 20
1-5 Common Architecture of a rule engine. The inference engine performs the pattern matching between the working memory facts and the rules in the rule database. It also determines the order of execution (via the agenda) and modifies the facts in the working memory (using the execution engine) . . . 21

2-1 Flowchart of a Simple Genetic Algorithm . . . 27
2-2 Crossover operator for the Assignment Problem . . . 31
2-3 Crossover operator for the Down-selecting Problem . . . 32
2-4 Example of the crossover algorithm for the user ground station down-selecting problem . . . 32
2-5 Crossover operator for the Down-selecting Problem . . . 34
2-6 Example of the crossover algorithm for the antenna allocation partitioning problem . . . 35

3-1 Flow chart of the Network Evaluator Module . . . 41
3-2 Flowchart of the Allocation process of Visibility Windows to Services 44
3-3 Flowchart of the best windows selection algorithm . . . 45

4-1 Decision Tree . . . 48
4-2 User case relative weighting to compute benefit . . . 51
4-3 FOM relative weighting for each user case . . . 52
4-4 Architectures represented in the benefit-cost space. Architectures have been color coded attending the constellation pattern they use. . . . 54
4-5 Tradespace Architectures represented in the Benefit-Cost space. The black diamonds represent the score of current TDRSS architecture on each scenario . . . 55
4-6 Figure of Merit relative score for each architecture option and scenario 56
4-7 Architecture Options Robustness . . . 57
4-8 Difference in cost among ISL-neighbor architectures plotted against the number of satellites launched . . . 58

4-9 Delta of benefit among ISL-neighbor architectures vs. number of satellites launched. . . . . 59
4-10 Different levels of disaggregation for a certain architecture . . . . . 60
4-11 Extra cost incurred due to disaggregation (per constellation) for GEO and MEO architectures . . . . . 60
4-12 Spacecraft cost decomposition . . . . . 62

# List of Tables

1.1 Architectural decisions . . . . . 17

3.1 Example of the data structures involved in the scheduling algorithm. Columns correspond to the type of slot, name of slot, example of value and units respectively . . . . . 40
3.2 Performance results for different versions of the Scheduler and different scenarios . . . . . 46

4.1 Users base characteristics . . . . . 49
4.2 Users base characteristics . . . . . 49
4.3 Architectural parameters . . . . . 50
4.4 Users base characteristics . . . . . 51
4.5 Service characteristics . . . . . 52

# Chapter 1

# Introduction

## 1.1 Motivation

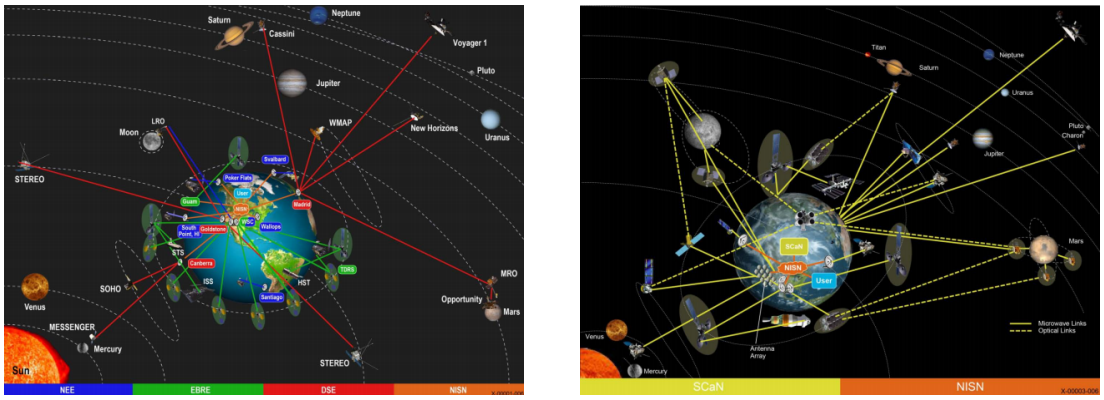
Providing communication and navigation services is fundamental for the success of space missions. In that sense, since its origins NASA has developed different networks to support their science missions. The differences among mission requirements have rendered a set of three independent networks: the Near Earth Network (NEN), that offers its services to the satellites flying in LEO orbits; the Space Network (SN), which is a set of geosynchronous satellites that provides full coverage communication relay capabilities to some satellites flying in LEO and the ISS; and the Deep Space Network (DSN), that supports those missions flying in deep space.

In 2006, the Space Communications and Navigation (SCaN) program was started. Its main goal was to lay out an architecture that integrated these three different networks, while being capable of meeting present and future needs. Reference [9] includes a high level description of the requirements and the challenges implied when evolving the current system.

The initial architecture as well as the desired architecture in the 2040 horizon are depicted in Fig.1-1a.

In order to support the SCaN system architecture effort, the System Architecture Lab at MIT has conducted several studies since 2012. Previous work includes a comprehensive stakeholder analysis [47] to estimate and characterize the needs of future customers, the definition of a set of metrics to evaluate the goodness of an architecture and the development of a tool capable of enumerating and evaluating thousands of architectures [38], [40] and point out the trades that exist among some architectural decisions. [39]

Besides, the System Architecture Lab has developed a System Architecting Tool to aid NASA identifying the network architectures that better address the needs of future near-Earth space-missions. At the beginning of this year, this tool, the Integrated




(a) Current SCaN network (b) SCaN network in 2025

Figure 1-1: SCaN network architectures

Tradespace Analysis for Communications Architectures (ITACA), had capabilities to model the performance and cost of a set of near-Earth communication satellites that, together with their ground stations, operate as a space network that relays information to and from customer spacecraft. However, in order to have more accurate results several improvements were proposed to be developed during the course of this year. The main points included: a) implement a $3^{rd}$ generation scheduling algorithm to drive down the computational time per architecture evaluated and to add new features (contact overlapping, MA support, ground station support to customer missions, etc.) and b) implement effective crossover algorithms for each architectural decision in order to have a functional GA to explore large spaces of architectures.

This thesis describes the work carried out in order to implement these new features and presents the results of an analysis conducted for NASA's UComm study team with ITACA.

## 1.2 Background

This section intention is to provide the context necessary by the reader to understand the rest of the document. In particular, the system architecture paradigm is described and some terminology is defined. Then, rule-based systems are introduced and finally an overview of ITACA's architecture is provided.

### 1.2.1 System Architecture

System Architecture is a discipline that emerged from Civil engineering in the late 80's. It develops the tools, methodologies and frameworks to systematically approach the design of architectures of large and complex systems.

The architecture of a system consists, basically, in its higher level system design [10]. An architecture, though, is defined from a holistic approach; it takes into account technical and non-technical factors, its main goal is to deliver value to its stakeholders

and it takes into account all the phases of the life-cycle, from design to operations and disposal [42].

A more formal definition of an **architecture** can be found in [11]: "An abstract description of the entities of a system and the relationships between those entities". This leads to precisely define a **system**: "A system is a set of entities and their relationships, whose functionality is greater than the sum of the individual entities". This extra functionality **emerges** from the operation of the system. The most common type of emergence in a system is functional emergence. For example, the time keeping function emerges from sand and a funnel, as in a hourglass.

Based on these definitions, Crawley defines **System Architecture** as "The embodiment of concept, the allocation of physical/informational *function* to the elements of *form*, and definition of *relationships* among the elements and with the surrounding context" [10]. This definition clearly differentiates three concepts:

- Form: the form is "*what the system is*". Form is the physical embodiment of the system.
- Function: the function is "*what the system does*" to provide value to the stakeholders.
- Relationships: the relationships include both structural and functional interrelations among objects of form of the system and its environment.

Hence, System Architecting is the process of creating a system architecture. Simmons [44] describes this process as a decision making process. Selva [42] further expands this concept by defining a system architecture problem as an optimization problem over the combinatorial space of decisions, where the utility function captures the value delivered to the stakeholders.

Therefore, we can model an architecture as a set of parameters and decisions that largely determine the system performance and cost [11]. Parameters $p$ are constant across architectures, whereas decisions $d$ represent the set of alternative choices that the system architect has to take when defining the system.

From this standpoint, the architecting problem can be seen as an optimization problem formulated as [37]:

$$\begin{aligned} \{arch^*\} = \{(\boldsymbol{d^*}, \boldsymbol{p})\} = \arg\min_{\boldsymbol{d}} \boldsymbol{J}\left(\boldsymbol{d}, \boldsymbol{p}\right) \\ s.t. \\ \boldsymbol{g}\left(\boldsymbol{d}, \boldsymbol{p}\right) \leq \boldsymbol{0} \\ \boldsymbol{h}\left(\boldsymbol{d}, \boldsymbol{p}\right) = \boldsymbol{0} \\ \boldsymbol{d} \in \boldsymbol{D} \end{aligned} \tag{1.1}$$

where $J(d,p) = [J_1(d,p), ..., J_m(d,p)]$ represents the evaluation function that com-

putes the metrics that capture the cost and performance of a certain architecture, $(d, p)$ is a tuple that represents an architecture, $g(d, p)$ and $h(d, p)$ are respectively inequality and equality constrains, and $D$ is the space of options for each of the decisions (D can be understood as the morphological matrix of the architecture).

In his thesis, Selva defines Eq. 1.1 as a **System Architecting Problem** (SAP). He further expands this concept stating that a SAP is *either a decision making problem, a combinatorial optimization problem or a search problem.*

### System Architecting Tools

Several issues raise within the SAP formulation: 1) the optimization domain is non-convex and with multiple optima, 2) variables can be discrete, continuous or combinatorial, 3) they are large scale problems (most time due to the combinatorial explosion [cite something] and 4) the objective function and the constraint functions are non-lineal. In order to ease the task of the system architect, several System Architecting Tools have been developed.

A **System Architecting Tool** is a computational tool that aids the system architect to solve a SAP. System Architecting Tools include among others simple morphological matrices, decision trees, tools to create system representations, model simulators, and tools to aid the decision-making process.

As described by Simmons in his thesis [44], any System Architecting Tool must have methods and functions in one or various of the four following aspects:

- Representational Aspect: Methods and tools to encode architectural decisions, constraints between decision and a set of metrics to calculate system properties.
- Structural Reasoning Aspect: Methods to analyze the structure of interconnected decision variables.
- Simulation Aspect: Methods to identify the feasible combinations of architectural variables, and the computation of system properties using metrics.
- Viewing Aspect: Tools to present the decision support data in a human understandable format, such as plots or tables.

On the other hand, Selva [42] classifies the System Architecting Tools as decision support tools (DST), combinatorial optimization algorithms (COA) and constraint satisfaction algorithms (CSA).

This thesis presents two of the contributions to ITACA, a System Architecting Tool to solve the SAP of defining te architecture of NASA's next generation of space networks, that were produced during my stay within the System Architecture Lab at MIT.

### 1.2.2 Overview of ITACA

ITACA was developed as a System Architecting Tool to identify the network architectures that better address the needs of future near Earth space missions [36]. This is done by exploring the tradespace of network architectures defined by combinations of decisions that characterize the network, while taking into account the needs of future space missions as characterized in [47].

A thorough description of the decisions that might define the tradespace as well as the reasons to choose them can be found in [36]. Table 1.1 contains a summary of these decisions, as well as their model as mathematical problems using the SAP taxonomy developed in [42]. Section 2.1 contains further explanation of the SAP classes.

Table 1.1: Architectural decisions

| *Decision* | *Parameters* | *SAP* | *Num. options* |
|---|---|---|---|
| Antenna assignment | $\{a_1, ...a_{N_a}\}$ | Down-selecting problem | $2^{N_a N_c}$ |
| Antenna allocation | $\{a_1, ...a_{N_a}\}$ | Partitioning problem | $Bell(N_a)$ |
| ISL antenna assignment | {yes, no} | Assigning problem | $2^{N_c}$ |
| Contract modality | {Procurement, hosted payloads} | Assigning problem | $2^{N_c}$ |
| Relay ground stations | $\{gs_1, ...gs_{N_{gs}}\}$ | Down-selecting problem | $2^{N_{gs}}$ |
| User ground stations | $\{gu_1, ...gu_{N_{gu}}\}$ | Down-selecting problem | $2^{N_{gu}}$ |
| Ground antennas | $\{1, ..., N_{max}\}$ | Assigning problem | $2^{N_{gu}}$ |

In particular, each architectural decision consists on:

1. **Antenna assignment**: Given the set of *available antennas* $\{a_1, ...a_{N_a}\}$ and the set of $N_c$ *available constellation patterns*, both defined as architectural parameters, assign subsets of antennas to each constellation pattern.
2. **Antenna allocation**: Given the set of antennas assigned to each constellation, choose to place them together on a single spacecraft or in a cluster of formation flying satellites.
3. **ISL antenna allocation**: Decide if each of the $N_c$ constellations will use inter-satellite links.
4. **Contract modality**: Decide if each of the $N_c$ constellations will be entirely procured or its antennas will be flown as hosted payloads.
5. **Relay ground stations**: Given the set of *available ground stations* $\{gs_1, ...gs_{N_{gs}}\}$, decide which ones will be operated to support the space segment.
6. **User ground stations**: Given the set of *available user ground stations* $\{gu_1, ...gu_{N_{gu}}\}$, decide which ones will be operated to support the network customers.
7. **Ground antennas**: Select how many antennas per ground station will be available to support network customers.

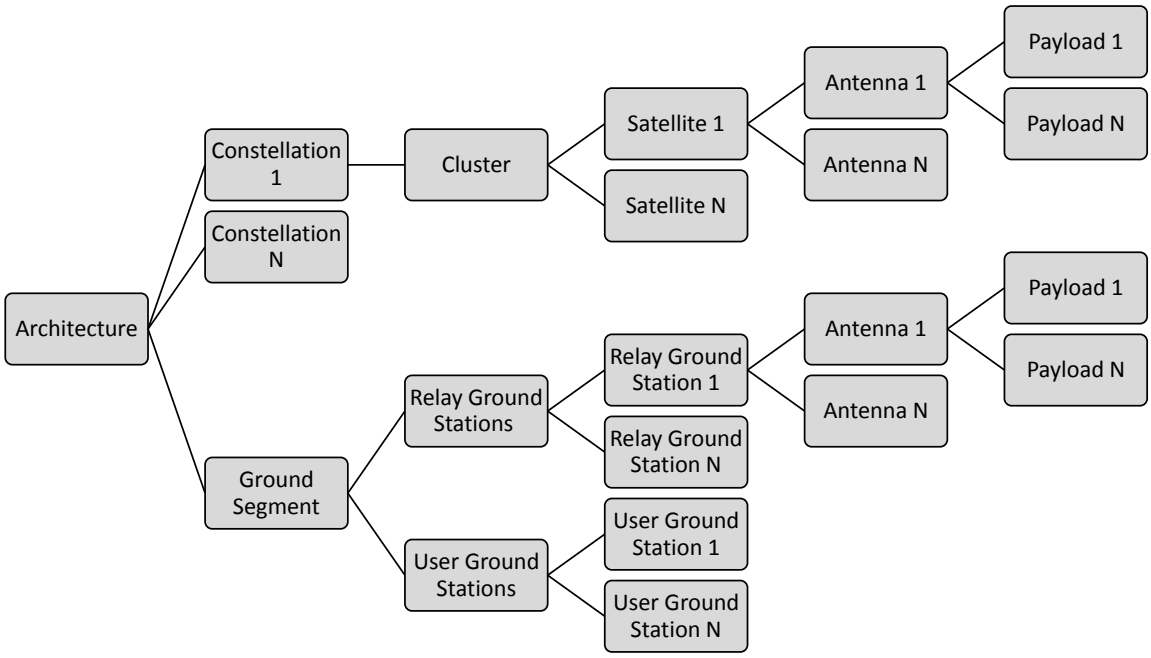



Figure 1-2: Architecture formal decomposition in its elements

On the other hand, the elements of an architecture are decomposed hierarchically as shown in Fig.1-2. An architecture is composed of the space segment and the ground segment.

The space segment is composed of several constellations, each of them with a certain number of orbital positions. On each orbital position there can be one or multiple satellites forming a cluster. Each satellite carries one or several antennas that at the same time can be used by one or more payloads.

The ground segment is composed of relay ground stations (those used to communicate with the relay satellites in the space segment) and user ground stations (those used to communicate directly with the customer missions). Each ground station contains one or multiple antennas with one or multiple payloads.

### ITACA's Architecture

ITACA is mainly composed of four different modules. First, a GUI allows the user to define the scenario that will be analyzed, that is, the parameters and decisions that will compose the architectures. Second, visibility windows (contact opportunities between customers and potential assets of the network) are computed using STK. Third, the Tradespace Evaluator enumerates and computes the performance and cost of thousands of architectures. Finally, the Tradespace Visualizer can be used to import and explore the results generated. Details on the implementation of ITACA can be found in [37].

Figure 1-3 shows a schema of the architecture of ITACA. The programming language used on each module is explicitly indicated.

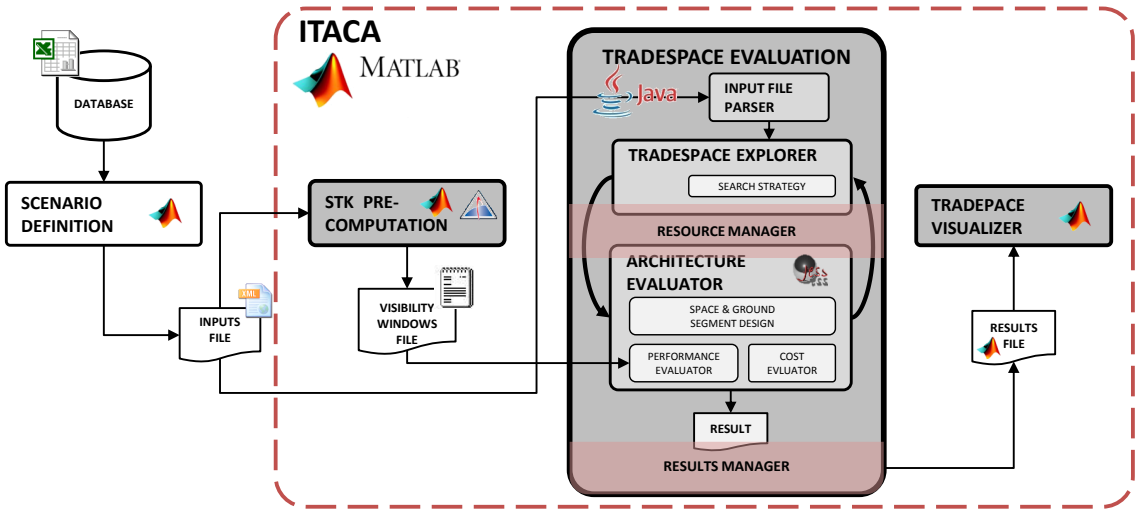



Figure 1-3: ITACA's architecture

Even the author of this thesis has contributed to the improvements performed in the four modules previously defined, due to extension limitations this document focuses only in the third module, the Tradespace Evaluator.

The Tradespace Evaluator is composed of the Tradespace Exporer and the Architecture Evaluator:

On one hand, the Tradespace Explorer generates architectures based on the parameters and decisions specified by the user. It includes different search strategies to explore the space of alternative architectures. In particular, it allows to perform a full enumeration of the whole tradespace or to use a genetic algorithm (GA). A genetic algorithm (GA) is a population-based meta-heuristic optimization algorithm that uses heuristics inspired by natural evolution, such as crossover, mutation and selection [27]. The GA implemented in ITACA follows the prescriptions of the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [14], a Multi-Objective Genetic Algorithm (MOGA)[13].

On the other hand, the Architecture Evaluator implements all the functionality to evaluate the behavior of a single architecture. In other words, it implements the evaluation function $J$ as described in Eq. (1.1). This means that given the tuple $(d, p)$ that represents an architecture, the architecture evaluator returns the benefit and cost metrics. In addition, the Architecture Evaluator returns the facts containing the information related to the design of the architectural elements (satellites, antennas, payloads), thus making the design traceable.

ITACA has two metrics defined (cost and benefit) and three figures of merit (returned data-volume, maximum latency, and user-burden). The cost of an architecture is estimated based on the number of launches, mass of the satellites, commonality,

number of ground assets, etc. On the other hand, the *benefit* figures of merit (data-volume, latency and user burden) are computed by simulating a day of operations of the system. These figures of merit are aggregated in a single value (benefit metric) using the VASSAR methodology [41].

The Architecture Evaluator is implemented in Jess [21], a Java-based rule engine used to build RBES (see Sec.1.2.3). The code is divided into several modules that encode human knowledge in form of rules. Each module is designed to perform a specific function. (i.e: there is a module to size the antennas of the assets, there is another module to compute the cost of a spacecraft, there is a module to compute the satisfaction of the users, etc.) Figure 1-4 depicts the flowchart of the Architecture Evaluator. Red modules correspond to benefit related modules, while green modules correspond to cost related modules.

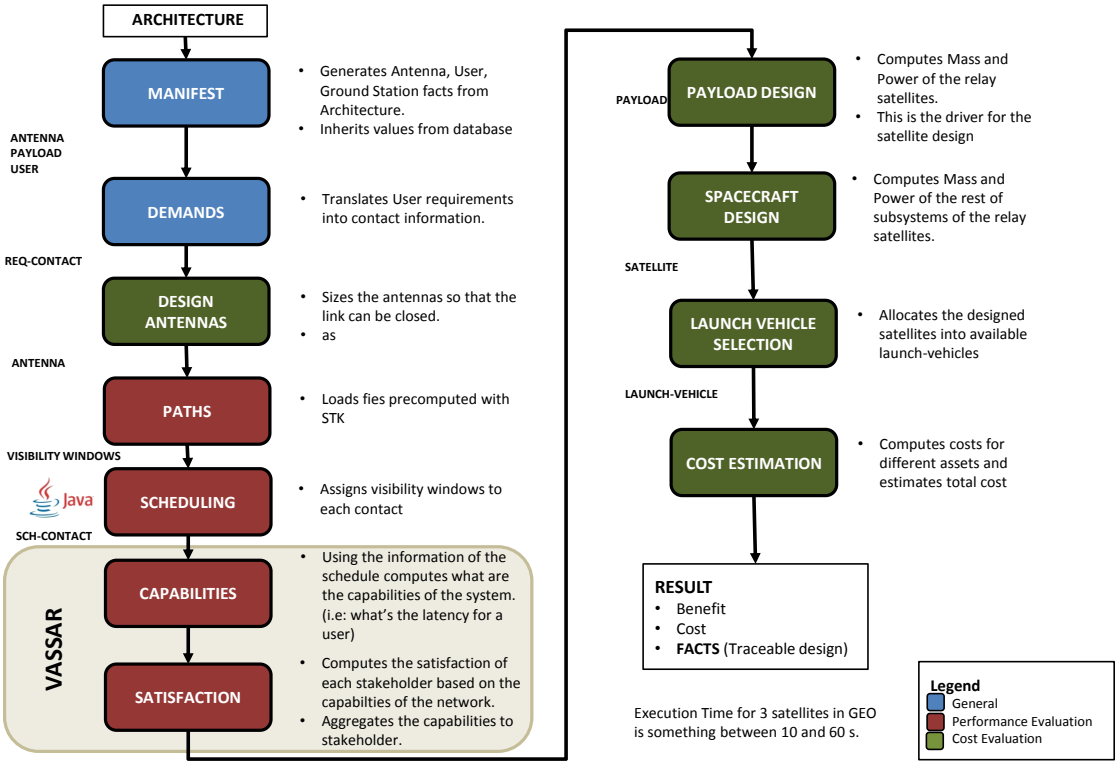



Figure 1-4: Flowchart of the Architecture Evaluator

### 1.2.3 Rule-Based Expert Systems

This section presents an introduction to Rule-Based Expert Systems (RBES). First, RBES are presented from a theoretical perspective, analyzing its components and how they interact. Second, Jess, a Java-based programing environment to develop RBES and its use in ITACA is described.

### Introduction to RBES

Expert systems were first introduced by Edward Feigenbaum in 1977 [19] and proliferated in the 1980's as forms of AI software [6],[43], [15]. Their aim is to mimic the decision making ability of humans by modeling human knowledge as logical rules. [26].

A typical rule engine contains three main items: The working memory (or fact database), the rule database and the inference engine. The inference engine, in turn, is composed of a pattern matching algorithm, the agenda and the execution engine. How these elements interact is shown in Fig.1-5

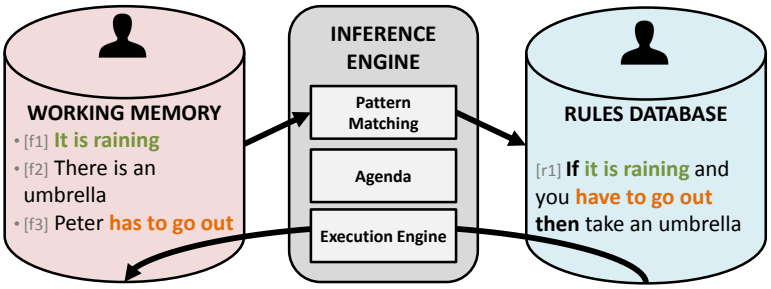



Figure 1-5: Common Architecture of a rule engine. The inference engine performs the pattern matching between the working memory facts and the rules in the rule database. It also determines the order of execution (via the agenda) and modifies the facts in the working memory (using the execution engine)

The working memory contains information about the world in the form of facts. Some of this facts are known by the programmer (initial conditions) while others are generated as a consequence of the execution of the RBES. The rule database contains human knowledge encoded in form of rules. Rules usually take the form of if-then statements, although some languages such as Prolog allow more complex structures. Rules are conditional statements that link given conditions to actions or outcomes. [22] These actions modify the fact database by asserting, retracting or modifying facts in the working memory. Finally, the inference engine is an automated reasoning system that evaluates the current state of the working memory and applies matching rules systematically.

Two inference methods are common within expert systems: *forward chaining* and *backward chaining*. In a *forward chaining* system, facts are processed first and rules draw new conclusions given those facts. In a *backward chaining* system, the conclusion is processed first and the inference engine looks for rules that would render that conclusion. This thesis only uses *forward chaining* techniques. Expert systems use a pattern (rules) to object (facts) matching algorithm to perform this forward chaining inference.

Advantages of expert system versus traditional programming languages include ease to expand and maintain the code, ease to write rules for non-trained programmers

and traceability of results by inspecting the facts generated during execution. Main disadvantages include the difficulties to debug the code due to the lack of control over its execution and from a computation efficiency standpoint, performance issues.

### Jess and its use in ITACA

Jess is a rule engine and a scripting environment developed in Java writen by Ernest Friedman-Hill at Sandia National Laboratories [1]. Jess implements an enhanced version of the Rete algorithm [20], an efficient many-to-many object-pattern matching algorithm.

As many rule engines, Jess follows a declarative paradigm instead of a procedural paradigm. In traditional programming languages such as C/C++, Java or Python, the order of execution of the instructions is determined by the programmer when he writes the code. On the other hand, in declarative languages the execution flow cannot be controlled by the programmer. The coder specifies *what* the program should accomplish and not *how* to accomplish it. Other examples of programming languages that use declarative programming are SQL, Haskell, Prolog or HTML.

Besides, Jess is a superset of the CLISP programming language. As in LISP, all code in Jess (assignment, procedures, control calls) takes the form of a function call. A function call is a list where the first element is the name of an existing function and the rest of the elements correspond to the arguments of the function. Variables in Jess are atoms, as Jess is not a strongly typed language. This means that variables don't need to be explicitly declared and that during execution time the type of a variable can be casted.

The typical structure of a Jess program is as follows: First, fact-templates and modules are defined. Then, rules that belong to each module are loaded and initial facts are asserted. Finally, execution is controlled by setting the focus in a module and letting the inference engine to match rules and facts by querying the working memory. This process continues indefinitely until there are no more possible matchings. Some of these concepts are further clarified:

- Fact-template: A fact-template defines the attributes that a certain fact contains.
- Module: A module is a section of code that groups a set of rules and facts that have common characteristics. Modules can be addressed from other modules.
- Rules: Rules in Jess have an if-then structure. The *if* part is commonly referred as the Left Hand Side (LHS) of the rule whereas the *then* statement is the Right Hand Side (RHS). The LHS contains the pattern that when met by the facts in the working memory, triggers the activation of the RHS. These patterns have the form of logical clauses.

[1] http://www.jessrules.com/

- Conditional element: Conditional elements are the logical clauses that define the patterns on the LHS of a rule. Examples of conditional elements include the `not` clause, the `exists` clause, or *regular-expressions* match patterns. More information on conditional elements can be found on chapter 7 of Ref. [21].
- Queries: A query searches the working memory (facts database) in order to find relationships between facts and rules.

In ITACA, the evaluation function that computes the values of the cost and benefit metrics of a particular architecture is implemented using Jess. In this implementation, facts represent network assets. Therefore, there are templates defined for elements such as satellites, antennas, ground-stations and so forth.

Rules represent the human knowledge necessary to size and design al these elements, and are contained in modules that perform a specific function (such as size antennas, schedule contacts or estimate the cost of the ground segment), as depicted in Fig.1-4. The inference engine focus is set successively on each module until the architecture has been completely designed and simulated.

## 1.3 Thesis Overview

The rest of this thesis is structured as follows:

Chapter 2 explains the implementation of the crossover operators for several of the decisions that are included in ITACA. This crossover operators are a key element in the new architectures enumerating process when the GA search strategy is selected.

Chapter 3 shows the work done to expand the functionality of the network scheduler. Work has been conducted to 1) improve the execution time of the scheduler and 2) provide the scheduler with the capability of scheduling multiple access contacts, overlapping contacts and use ground assets so that customer mission are capable of establishing a direct link to Earth.

Chapter 4 exercises the tool to analyze a case study on how to evolve the Space Network, using as inputs the set of architecture options currently under consideration by NASA's UComm Study Group. Several trade-offs are considered, including frequency band selection, spacecraft disaggregation, constellation pattern selection and ISL + Ground segment design.

Finally, chapter 5 present the conclusions and proposes several areas of improvement for future work.

# Chapter 2

# Crossover Algorithms for ITACA's SAP problems

This chapter describes the implementation of the crossover algorithms for ITACA's System Architecting Problems (SAPs). It is structured as follows: First, Sec.2.1 introduces the SAP problems that appear in ITACA architectural decisions. Second, Sec.2.3 presents a literature review of existent crossover algorithms for the described SAP. Finally, the implementation details for the proposed crossover algorithms are depicted in Sec. 2.4.

## 2.1 Introduction to SAP Problems

Selva showed in his dissertation [42] that the system architecting process can essentially be seen as the process of solving one or several related System Architecting Problems (SAPs). He explicitly defines a System Architecting Problem as: 1) a decision making problem; 2) a combinatorial optimization problem; or 3) a search problem. [42] In particular he argues that a SAP can be formulated as a constrained, multi-objective combinatorial optimization problem.

He presents a library with the classes of SAPs that appear more often in System Architecting. It contains efficient grammars, evaluation rules, search heuristics and selection criteria for each of the classes. For complex systems, it is common that more than a SAP class is necessary to fully represent the decision space. This is the case within the architecting problem that we are facing in this thesis. Moreover, precedence relations among decisions impose an order on the decision making process.

This section's objective is twofold: first, to present these grammars and introduce the reader in the rationale they are founded on. Second, to relate these grammars to ITACA's decisions.

### 2.1.1 The Assigning Problem

In short, an assignment problem consists on given a set of options for a decision, assign the optimal option to that decision. Formally, Selva defines this problem as:

Given a set of n generic decisions $D = \{d_1, d_2, \ldots, d_n\}$, where each decision $d_i$ has a discrete set of options $O_i = \{d_{ij}\}$, the goal of the Assigning Problem is to find the assignment of options to decisions $A = \{d_i \leftarrow d_{ij}\}_i$ that maximizes value delivery to stakeholders:

$$A^* = argmax_{\{d_i \leftarrow d_{ij}\}_i} V(\{d_i \leftarrow d_{ij}\}_i) \tag{2.1}$$

where $V(\cdot)$ is a generic value property function that may have more than one dimension (e.g. benefit and cost). Under this conditions, the number of possible combinations of the assignment problem equals the number of options, $n$. Selva proves in his thesis that any other class of SAP can be reduced to an Assigning Problem simply by enumerating all the possible options.

This SAP class is useful to model decisions where the architect has to make a choice among a set of exclusive options. In ITACA, the *contract modality* and the *ISL antenna assignment* decisions are modeled as assigning problems. A good encoding scheme to represent this kind of problems is assigning each option a number form 1 to $n$ (or a representative text string), and then associating to $A^*$ that number.

### 2.1.2 The Down-selecting Problem

The down-selecting problem consists on given a set of elements, choose the subset that maximizes the value delivered to the stakeholders. Selva presents this SAP formally as:

Given a set of m generic elements of function or form $U = \{e_1, e_2, \ldots, e_m\}$, the goal of the down-selecting problem is to find the subset $S_i$ or subsets of elements $\{S_i\}$ that maximize value delivery to stakeholders, or more precisely, that optimizes the trade-off between benefit and cost.

$$S_i^* = argmax_{S_i}[B(S_i)\ C(S_i)] \tag{2.2}$$

where $B(\cdot)$ and $C(\cdot)$ are benefit and cost property functions respectively. A total of $2^m$ possible combinations exist for the unconstrained down-selecting problem. However, normally some restrictions apply when selecting elements, due to incompatibilities among them. This SAP class is useful to model those architectural decisions where the architect has to choose one or more elements from a set. A real-life situation that exemplifies this class occurs when one has to choose what assets to buy with a limited amount of money.

In ITACA, the *antenna selection* and the *relay* and *user ground stations selection* decisions are modeled as down-selecting problems. A good encoding scheme to represent this kind of problems is using a binary string of length $n$, where bit in position $i$ represents if element $i$ in the set has been selected. For example, in the user ground station decision, the set of elements to choose might be Dongara, Hawaii, Poker-Flat, Svalbard, Kiruna. Vector [01011] means that for a particular architecture, stations in Hawaii, Svalbard and Kiruna have been selected.

### 2.1.3 The Partitioning Problem

In order to describe the partitioning problem, a set partition must be defined first: Given a set of m generic elements of function or form U = $\{e_1, e_2, \ldots, e_m\}$, a partition $P_i = \{S_i\}$ of a set $U$ is a division of $U$ into a number of non-overlapping subsets $S_i$ that are mutually exclusive and exhaustive.

A Partitioning Problem consists on given a set of m generic elements of function or form U = $\{e_1, e_2, \ldots, e_m\}$, the goal of the Set Partitioning Problem is to find the set partition $P_i = \{S_i\}$ that maximizes value delivery to stakeholders:

$$P_i^* = argmax_{P_i} V(P_i) \tag{2.3}$$

where $V(\cdot)$ is a generic value property function that may have more than one dimension (e.g. benefit and cost). The total number of possible set partitions in an unconstrained problem equals the Bell number of m, which can be computed as shown in Eq. (2.4). Note that if unconstrained, the number of options render the enumeration process intractable, as Bell numbers grow faster than exponentially. A recursive enumeration rule is used to compute all the possible combinations for the set partitioning problem.

$$B(m) = \sum_{k=0}^{m} (\frac{1}{k!} \sum_{i=0}^{k} ((-1)^i \binom{k}{i} (k-i)^m)) \tag{2.4}$$

A common instance of the set partitioning problem is the scheduling and distribution of flight crew into spacecraft that face many airlines all over the world.

In ITACA, the antenna allocation into spacecraft decision is modeled as a Partitioning Problem. The encoding used to represent the antenna-partitioning information is a numerical code. For each element, a number is assigned representing the subset it belongs to. For example, if three antennas are present in a constellation, an allocation of 122 means that the first antenna is in satellite 1 whereas the second and the third antennas are in satellite 2. As a convention, we will always choose the lowest number possible to represent the set-partitioning (Hence, even though a codification like 211 represents the same set-partitioning than 122, we will always chose the later).

This encoding has the limitation that no more than 9 subsets can be formed. Its main advantage is that it is very easy to interpret from a human standpoint (as compared to other encodings such as relation matrices or data structures) while at the same time easy to deal with computationally.

## 2.2 Introduction to Genetic Algorithms

A Genetic Algorithm (GA) is a meta-heuristic algorithm that mimics the problem of natural selection in order to solve an optimization or a search problem. They were first described by Holland in 1962 [27], and further developed by Goldberg in the forthcoming years [23]. Figure 2-1 shows the typical steps followed in a GA.

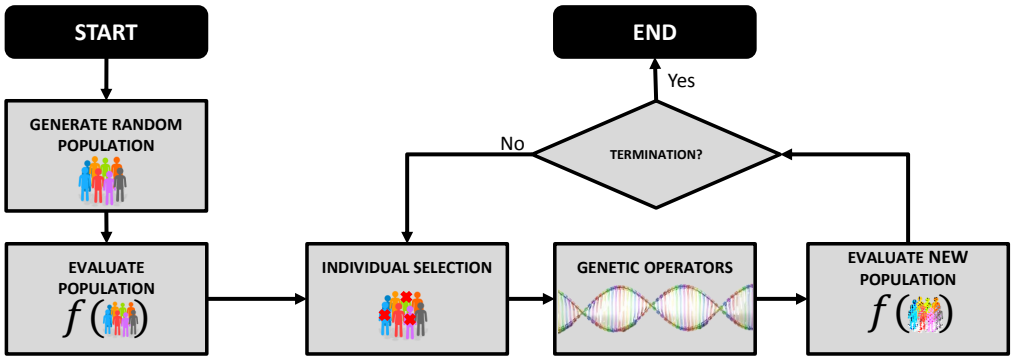



Figure 2-1: Flowchart of a Simple Genetic Algorithm

The GA starts by creating a random initial population. The size of the initial population depends on the nature of the problem, but normally a generation is compose of hundreds or thousands of individuals. After evaluating each individual using a fitness function (evaluation function, objective function), a subset of them is chosen to breed a new generation. Those architectures with a higher score (better fitted) are more likely to be chosen.

In order to generate a new population, several genetic operators are applied over the set of parent architectures. Two operators are always present in a GA, crossover and mutation. The former generates a children by combining the genes of its parents whereas the later creates a new individual by altering the value of one or more genes of a parent.

Crossover is a procedure analogous to reproduction, where the chromosomes of two parents are combined in order to produce the chromosome of a child. In that sense, a GA is based in the premise that the result of merging two good individuals is a better individual. Different techniques exist to combine parent architectures: although the more common ones use two parents to produce a new child (or two children), there are methods where multiple parents can be used. Section 2.3 provides a literature

review of crossover algorithms. Mutation occurs during evolution according to a user-defined mutation probability, which should be low as otherwise the algorithm turns into a random search algorithm. The purpose of mutation in GAs is preserving and introducing new diversity. In that sense, mutation allows the algorithm to avoid getting stuck at local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. Although crossover and mutation are known as the main genetic operators, it is possible to use other operators such as regrouping, colonization-extinction, or migration [4].

Finally, a termination criteria must be set in order for the algorithm to finalize. Common termination conditions include "a maximum number of generations is reached" or "a solution that satisfies the minimum criteria is found".

In a GA, the variables are called chromosomes. The simplest chromosomes are encoded in the form of bit strings, although more complex structures such as integers, floating point values or complex data structures can be used. In those cases the genetic operators have to be defined appropriately in order to deal particularities of those structures.

As stated by Holland in [5] in his "Fundamental theorem of genetic algorithms", GAs are good at finding good schema, that is, subsets of a string with similarities in certain string positions. In particular, the theorem says that short, low-order schemata with above-average fitness increase exponentially in successive generations.

The GA implemented in ITACA is based in the Non-Dominated Sorting Algorithm NSGA-II [14]. This algorithm performs the parent selection by computing two parameters: the Pareto ranking and the crowding distance among individuals with the same Pareto ranking (how far is the score obtained by that individual from the score obtained by any other individual in the Pareto Front). Criteria to choose the $N$ parents of the next generation of architectures are:

1. Choose the lowest Pareto ranking individuals first.
2. Among individuals with the same Pareto ranking, choose those with a higher crowding distance.

## 2.3 Crossover algorithms for GAs. A literature review

Crossover algorithms for GAs highly depend on how chromosomes are encoded. In that sense, different techniques are applied depending on the nature of the variables used to represent the data that intervenes in the optimization problem to solve. We can classify them in binary string, real values and matrix crossovers.

### 2.3.1 Binary string crossover operators

For binary strings multiple algorithms have been developed. Most famous are single-point crossover, two-point crossover, uniform crossover and shuffle crossover.

- Single-point crossover produces two offsprings from a pair of parents by randomly selecting a point on the chromosomes and replacing the former and latter half of each parent from the point at a probability r.
- Two-point crossover produces two offsprings from a pair of parents by randomly replacing an interval on the chromosome of each parent with the same interval on the chromosome of the other parent [46]. A generalization of a two point crossover is the multi-point crossover where multiple segments are created and they are alternatively swapped among parent individuals.
- Uniform crossover produces two offsprings from a pair of parents by uniformly replacing the elements on each locus of the parents at probability $r$ [48]. Uniform crossover is an extreme case of multi-point crossover where the number of segments equals the number of bits in the sting.
- Shuffle crossover is related to uniform crossover. A single crossover position (as in single-point crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offsprings are unshuffled in reverse. This way, the positional bias is removed [7].

### 2.3.2 Real values crossover operators

For floating point variables, several methods have been proposed in te literature. The most famous is simulated binary crossover proposed by Agrawal in [2], which generates a probability distribution based on the values of the parent architectures and then extracts a random sample of that probability distribution. Previous work conducted by Eshelman and Schaffer in [17] proposed an approach based on interval schemata where using a blend crossover operator (BLX-$\alpha$) generates a new point as a lineal combination of the values of both parents ($\alpha$ and $(1-\alpha)$ are the coefficients that multiply the values of the parents).

### 2.3.3 Matrix crossover operators

In [30] Levine described a GA used to solve the set partitioning problem. He creates a crossover mechanism among matrices. Crossover is performed at the column level, that is, for each column in the matrix the same column of the father or the mother is copied. In this case the size of the matrix is constant across individuals.

In [49] a matrix crossover algorithm is proposed to solve the Travel Salesman Problem. This crossover algorithm is intended to solve the permutation problem. The crossover algorithm tries to preserve the order of precedence among different cities. This information is encoded in a binary matrix.

## 2.4 Implementation of ITACA's crossover algorithms

ITACA models decisions as an abstract class that contains the following GA methods: `initializeDecision`, `crossover`, `mutateDecision`, `checkConsistency` and `repairDecision`. A decision also includes information about its name, type, and the decisions that precede it in the decisions making process. As described in [37], this precedence information is used to build a Decision Tree that manages the order of the decision taking process. Each of the SAP classes extends this abstract class and implements the particularities of the aforementioned methods. If precedence rules exist, the methods have to be tweaked manually to deal with the constraints this precedence imposes. A generalization of the decision class that solves this issue is part of the future work in the field.

The `initializeDecision` and `checkConsistency` methods are self explanatory. The first one initializes a random value for the decisions whereas the second one verifies that the assigned option satisfies the constraints (intrinsic and imposed by precedent decisions) the decision is subject to. Their implementation is trivial and thus, it is not described in this thesis. The `repairDecision` method is specific to each decision and acts in case `checkConsistency` returns constraints have been violated.

`mutateDecision` and `crossover` methods are specific to each decision. The first one alters randomly the value of the decision and provides a method for the GA to escape when it gets stuck in a local optima. The second one, takes the information of two parent architectures to generate one (or two) children that have a combination of the options assigned to the parents. This section is devoted to describe the implementation of the crossover algorithms for each of the SAPs.

### 2.4.1 Crossover for the Assigning Problem

The crossover algorithm starts with the information of the options assigned to each of the parents. As described in Sec. 2.1.1, only a value of the possible set of options can be chosen in an Assigning Problem, thus `crossover` assigns to one of the children the father's option and to the other children the mother's option. However if a decision is dependent on other decisions, it might happen that an option is not assigned for one (or both) of the parents. If only a parent has an option assigned, both children get that value. If none of the parents has an option assigned, `crossover` assigns randomly an option to the decision.

Listing 2-2 shows the Java code that implements this method in ITACA. As both the ISL assignment and the contract modality decisions must be preceded by the Antenna Assigning decision, the crossover operator shows explicitly the behavior described in the previous paragraph.

```
public void crossover(Architecture father, Architecture mother, List<Architecture> children) throws
Exception {
    // Get father and mother payload assignment
    ArrayList<Integer> father_pay_assign = (ArrayList) father.getVariable("payload-assignment");
    ArrayList<Integer> mother_pay_assign = (ArrayList) mother.getVariable("payload-assignment");

    // For each children, generate the crossover
    for (Architecture c : children) {
        ArrayList<Integer> child_pay_assign = (ArrayList) c.getVariable("payload-assignment");
        //Create the children ;
        List child_pay = new ArrayList<Integer>();
        if (child pay assign.get(i) != 0) {
            // If the payload is not null, decide which to copy for each orbital slot (father / mother)
            if (father_pay_assign.get(i) != 0 && mother_pay_assign.get(i) != 0) {
                // If both are present, choose one randomly
                if (rnd.nextBoolean()) {
                    child pay.add((Integer) ((ArrayList) father.getVariable(name)).get(i));
                } else {
                    child_pay.add((Integer) ((ArrayList) mother.getVariable(name)).get(i));
                }
            } // If only one was present before, choose it for the children
            else if (father_pay_assign.get(i) == 0 && mother_pay_assign.get(i) != 0) {
                child_pay.add((Integer) ((ArrayList) mother.getVariable(name)).get(i));
            } else if (father_pay_assign.get(i) != 0 && mother_pay_assign.get(i) == 0) {
                child_ pay.add((Integer) ((ArrayList) father.getVariable(name)).get(i));
            } // If none of them were present before, choose a random value for the children
            else {
                child_ pay.add(getIndividualRandomValue((Integer) child_pay_assign.get(i)));
            }
        } else {
            // If no payloads were assigned to that orbital slot, add '0'
            child_pay.add(0);
        }
        c.setVariable(name, child_pay);
        this.repairDecisionValue(c);
    }
}
```

Figure 2-2: Crossover operator for the Assignment Problem

### 2.4.2 Crossover for the Down-selecting Problem

As described in Sec.2.1.2 a down-selecting problem can be represented using a binary string that contains 1's on the elements that are present in the assigned option. Sec.2.3 presents several algorithms to perform the crossover operation when the genes can be represented using a binary string. As ITACA's down-selecting decisions do not have precedence conditions, there is no need to tweak any of those algorithms.

In particular, a uniform crossover algorithm has been implemented. Figure 2-3 shows the Java code that implements this method. Note that this code is the same for all of the down-selecting decisions in ITACA.

```
protected void crossover(Architecture father, Architecture mother, List<Architecture> children) throws
Exception
{
    List<String> f = (ArrayList<String>)father.getVariable(name);
    List<String> m = (ArrayList<String>)mother.getVariable(name);

    BitSet father = initializeBitSet(f);
    BitSet mother = initializeBitSet(m);

    BitSet child = new BitSet(father.size());
    BitSet invChild = new BitSet(father.size());

    for(int i = 0; i < child.size(); i++){
        if(rnd.nextBoolean()){
            child.set(i, father.get(i));
            invChild.set(i, mother.get(i));
        }
        else{
            child.set(i, mother.get(i));
            invChild.set(i, father.get(i));
        }
    }

    children.get(0).setVariable(name, updateGroundList(child));
    children.get(1).setVariable(name, updateGroundList(invChild));

    this.repairDecisionValue(children.get(0));
    this.repairDecisionValue(children.get(1));
}
```

Figure 2-3: Crossover operator for the Down-selecting Problem

Figure 2-4 shows a graphic representation of what the crossover algorithm does for the user ground station down-selecting problem.

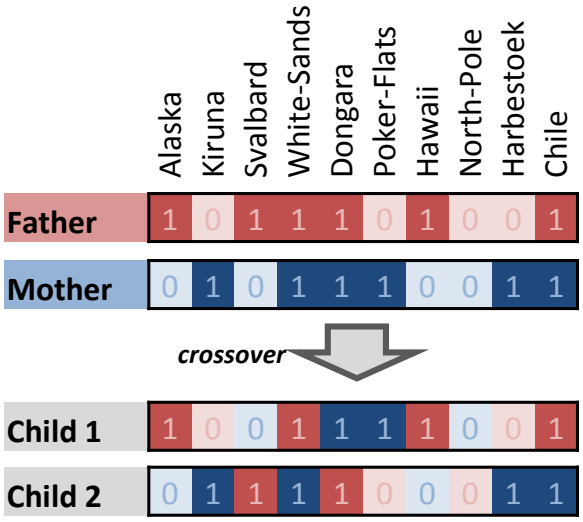



Figure 2-4: Example of the crossover algorithm for the user ground station down-selecting problem

### 2.4.3 Crossover for the Partitioning Problem

The representation chosen in ITACA to encode the antenna allocation is number-based, as described in 2.4.3. However, the real information of the antenna allocation decision is contained in the relations imposed by the subsets, that is, whether two antennas are together in a spacecraft or if they are in separated spacecraft. In that sense, a better representation (but less comprehensible from a human perspective) is a

relation matrix that shows which elements belong to the same subset. Thus, the first step of the partitioning problem is to create the matrix representation of the antenna allocation decisions of the parent architectures. Function `buildContraintMatrix` performs this transformation. As it occurred in the assignment problem, it might happen that some of the antennas were not present in some or both of the parent architectures. In that case, as an antenna is not assigned to a parent architecture, the antenna allocation problem has no information about it and a -1 will appear in the corresponding position.

The second step is to crossover both matrixes. At each time, a position in the matrix is randomly selected. If both parent architectures have a value different to -1, one of the parent's value is selected radomly. If the antenna is only present in one parent, the value of that parent is selected. Otherwise, a value of 0 or 1 is randomly selected.

Finally, due to the characteristics of the problem, fixing a 0 or a 1 in any position of the matrix might fix the value in other positions in the same row or the same column. Therefore, a `fixDecisions` method must be executed recursively in order to propagate the effects of fixing a number on the rest of the values of its row /column . For example, if elements 1 and 2 are in the same set, and elements 3 and 4 are together in a different set, fixing that 2 and 3 are in different sets also fixes the following conditions: 1 and 3 are in different sets; 1 and 4 are in different sets; 2 and 4 are in different sets.

Listing 2-5 shows the Java code that implements the crossover algorithm for the Antenna allocation decision. The rest of the functions involved in the algorithm can be found in Annex A.

```
protected void crossover(Architecture father, Architecture mother, List<Architecture> children) throws
Exception {
    // Get father and mother payload assignment
    List<Integer> father_pay_assign = (ArrayList) father.getVariable("payload-assignment");
    List<Integer> mother_pay_assign = (ArrayList) mother.getVariable("payload-assignment");

    List<Long> father_pay_alloc = (ArrayList<Long>) father.getVariable(name);
    List<Long> mother_pay_alloc = (ArrayList<Long>) mother.getVariable(name);
    //System.out.println("Crossover dad = " + father_pay_alloc + " mom = " + mother_pay_alloc);
    int Npayloads = ((List) father.getVariable("payloads")).size();

    for (Architecture c : children) {
        // For each children, generate the crossover
        List<Integer> child_pay_assign = (ArrayList) c.getVariable("payload-assignment");
        List<Long> child_alloc = new ArrayList<Long>();
        for (int i = 0; i < child_pay_assign.size(); i++) {
            int p = child_pay_assign.get(i);

            if (p != 0) {

                int[][] father_M = buildConstraintMatrix(father_pay_assign.get(i),
father_pay_alloc.get(i), p, Npayloads);
                int[][] mother_M = buildConstraintMatrix(mother_pay_assign.get(i),
mother_pay_alloc.get(i), p, Npayloads);

                int[][] child_M = crossoverMatrixes(p, father_M, mother_M, Npayloads);
                child_alloc.add(buildAllocationFromMatrix(child_M, p));

            } else {
                // If no payloads were assigned to that orbitl slot, add '0'
                child_alloc.add(0L);
            }
        }
        // set the value to the architecture.
        c.setVariable(name, child_alloc);
    }
}
```

Figure 2-5: Crossover operator for the Down-selecting Problem

Figure 2-6 shows an example of the application of the crossover algorithm. In this case, the antenna assignment decision contains 5 possible options. A SMA antenna, two RF SA antennas and two optical telescopes. The father architecture is composed of a constellation with a SMA and a RF SA antenna in a satellite and another satellite with two optical telescopes, whereas the mother architecture is composed of three satellites; one carrying a SMA antenna, another carrying a RF SA antenna and the last one with an optical telescope. Note that the father architecture has the SMA and the RF antennas in the same spacecraft whereas the mother architecture has them in different satellites.

In addition the child architecture has an antenna not present in any of its parents (The second RF SA antenna), probably due to the effects of mutation over it. After converting the payload allocation value to the relation matrices the crossover function generates the matrix of the offspring. In this case this process is done in 5 iterations. First position (4,5) is randomly selected (pic. 2 in Fig.2-6), and as the father architecture is the only one with a valid value (1) in that position, the children inherits it. `fixDecision` method does not propagate any value at this step. In next step (pic. 3 in Fig.2-6), position (3,4) is selected. As none of the parents has a value assigned in that position, a random one is assigned (0). `fixDecision` propagates this value to position (3,5); as the two optical telescopes are in the same spacecraft and the second RF SA antenna is not with telescope 1, it can not be together with telescope 2. In pic. 4 position (1,2) is selected and the value from the father inherited. No propagation is necessary. Next, (pic. 5) a similar procedure than in pic. 3. occurs. Finally, in pic. 6 position (1,4) is selected and the mother value is inherited. First

propagations fixes values in positions (1,5) and (2,4). Second round of propagation (using this last two values as initial points) fixes the value in position (2,5). In natural language that last step can be understood as: "The SMA and the optical telescope 1 are not together. As the SMA and RF SA 1 antennas are together in a satellite and the two optical telescopes are together in another spacecraft, neither the SMA and the optical telescopes or the RF SA antenna and the optical telescopes can be in the same satellite" .

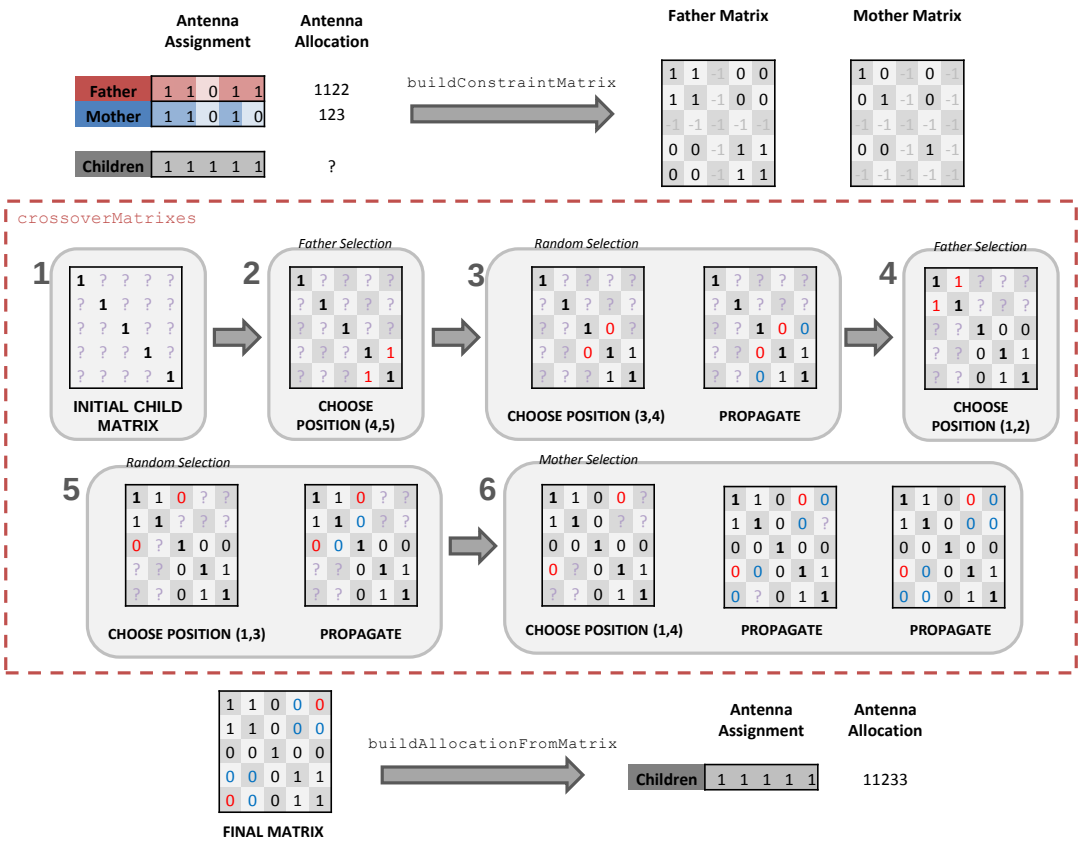



Figure 2-6: Example of the crossover algorithm for the antenna allocation partitioning problem

Last step consists in creating the payload allocation numerical value from the final matrix using the `buildAllocationFromMatrix` function.

The propagation process is carried by the `fixDecisions` method. `fixDecisions` loops through the column where the last value was fixed and whenever it finds two 1's in different positions, it substitutes the "?" in one of the rows where the 1's are by the values in the other row. Then the loop is performed over the rows (instead of the columns) and the substitution is done using the values in the columns. The code of these functions can be found in Annex A.

# Chapter 3

# An Enhanced Version of the Scheduling Algorithm

The performance of the network in ITACA is assessed by simulating a day of operations of the system. During this simulation, a schedule that assigns contact time (resources) between each customer mission and a network asset is created. Based on the number of contacts scheduled, the maximum time between them, and the data-volume returned, the latency and data-volume figures of merit (FOM) are computed.

## 3.1 Scheduling techniques

Scheduling problems constitute a fundamental family in optimization problems. Best known instances of this family include the job-shop scheduling problem [8], where a set of jobs have to be assigned to a set of machines minimizing the timespan, the open-shop scheduling problem [24], where a set of jobs have to be processed for given amounts of time in certain machines at an arbitrary order and the flow shop scheduling problem [32], where the jobs have to be processed in certain machines in a certain order. These problems are also popular in other domains such as operations research [29], where the aim is to find a valid schedule that satisfies all the constraints imposed.

As a generalization, scheduling problems can be formulated as: determining the times that certain resources will be serving certain jobs while optimizing a metric that captures the goodness of the solution [50]. Our particular problem consists on assigning satellite-antenna contact time (the resources) to the customer-missions services the network has to support (the jobs). Several hard constraints apply. For example, a link can only be established if both the customer mission and the relay satellite are in line of sight during the whole duration of the contact. Besides, the antenna in the relay satellite has to have a free payload in the same band of the user

antenna and the data rate it provides must be high enough to support the service to schedule. Also, at any time a SA antenna can only be serving an unique customer while a MA antenna can support simultaneously a maximum number of users (5 in TDRSS).

Several approaches have been proposed to solve the job scheduling problem in the satellite communications context [45] [34] [35]. Classical ones involve a discretization of the time domain, creating a static snapshot that captures the state of the network at certain intervals of time, and the use of Mixed Integer Linear Programming techniques to optimize the solution. However, this kind of algorithms are computationally intensive, as the algorithm explores most of the possible solutions which increase exponentially with the number of users and resources. One of the main advantages of this approach is that, given a sufficient amount of time, the optimality of the solution is guaranteed. This approach has been used in [31] to schedule a daily imaging scheduling system for a low-orbit, earth observation satellite and in [3], where the problem of scheduling users access to satellite communication channels is dissected.

Other techniques use heuristic based general purpose algorithms such as Genetic Algorithms, Colony Optimization Techniques or Simulated Annealing. This techniques use a series of metaheuristics to determine good solutions for the optimization problem. However, even though it has been proved that this mechanism converges to an optima, the speed of convergence is difficult to quantify. In any case, for the problem we are facing this time would be too high, due to the high amount of constraints and users. Reference [25] presents a generic hybrid genetic algorithm to solve the job shop scheduling problem. A valid schedule is constructed using a priority rule in which the priorities are defined by the genetic algorithm.

In addition, scheduling where constraints impose hard conditions can be tackled as Constraint Satisfaction Problems, thus the objective is not anymore to optimize the performance metric but just to find a valid solution. This technique is used in [18] combined with the use of a GA in order to find optimal scheduling solutions for current TDRSS.

Finally, the last method consists on the use of heuristic techniques that use human knowledge gathered from experts to build a schedule solution. This techniques do not guarantee the optimality of the solution as most of the times render in a suboptimal solution. However, given that the scheduling process follows a series of simple rules its execution time is very small. This method is used in [1].

As our tool must be able to evaluate thousands of architectures systematically, having a low execution time to analyze a single architecture is crucial. Because of this fact, MILP methods and metaheuristics methods that need multiple iterations to converge were discarded. Instead, a greedy heuristic approach that uses a set of rules to schedule contacts based on the priority of the users has been followed. This allows us to evaluate the performance of complex architectures in less than 30 seconds, thus allowing us to assess large tradespaces in a reasonable time.

## 3.2 Initial scheduler

The evolution of ITACA has followed different stages. Initially, the tool infrastructure was completely MATLAB-based, whereas the expert knowledge was contained in Jess scripts. At that time, the scheduling process was carried out using a heuristic MATLAB algorithm, that allocated link usage to users depending on its priority. A shortest path algorithm together with the system state matrices were used to determine the best route from the customer mission to the ground terminal. More information on this scheduler the can be found in [40]

Due to performance issues, and most importantly, as by that time it was not possible to use parallel computing in MATLAB, the tool infrastructure was migrated to Java in early 2013. After this migration the scheduler code was ported to Jess, creating a heuristic rule-based scheduler. The scheduler had capabilities of scheduling users through a relay satellite, although only single accesses could be satisfied. Besides, due to the poor performance of the rule-engine the execution time for complex architectures (with lots of relays or lots of customer missions) was becoming a problem. A detailed description of the heuristic and how this algorithm worked, as well as its validation can be found in reference [39].

## 3.3 Enhanced version of the scheduler

Improving the computational performance and expanding the capabilities of the scheduler was an important task of the project. On one hand, it was necessary to decrease the execution time at least one order of magnitude in order to be capable of evaluating large tradespaces in a reasonable amount of time. On the other hand, several features needed to be added to the scheduler, in order to be able to assess more realistically the behavior of the network. These new capabilities include:

- Ground assets support: One of the objectives of the SCaN program was to define the architecture of an integrated network that comprised the Near Earth Network (NEN), the Space Network (SN) and the Deep Space Network (DSN). In order to accomplish that objective, it was necessary to provide the tool with the capability of scheduling users directly to ground stations, similar to the way the NEN works nowadays.
- Multiple access support: One of the current payloads in the Space Network is a phase-array antenna with 32 receive elements that can support up to 5 simultaneous contacts. One of the studies of relevant interest for NASA consists on how would the performance and cost change if users were encouraged to use the MA service instead of the SA. However, the initial scheduler didn't have the capability of scheduling multiple access contact and this capability was added in the enhanced version.

- Contact overlapping support: In real life, most of the customer missions overlap SD and TT&C services when using TDRSS. That is, as these two services use different frequency bands, the data can be multiplexed and transmitted simultaneously towards the same relay-antenna. In order to model the heterogeneity of the services the network has to serve, this capability had to be implemented in the scheduler.
- Maximum data volume: Some network assets (ground stations or relay-satellites) have limitations on the amount of data they can transmit. For example, McMurdo ground station, in Antarctica, has a limited backhaul connection between the station and the mission operation centers (MOCs).
- Force the use of some assets: In some cases, some users are forced to use certain assets to relay its data. This was another capability that the scheduler lacked and was incorporated in the enhanced version of the scheduler.

In order to implement all these modifications, a new version of the scheduler was created. Major changes related to a) viable window selection algorithm and b) resource allocation algorithm. In addition, new data structures were added, in order to maintain the information related to the novel capabilities of the scheduler.

The general structure of the scheduler is presented in Fig.3-1. Next subsections discuss in detail each of the steps of the scheduling process.

### 3.3.1 Data Structures

In order to implement the aforementioned new capabilities several modifications had to be made to the data structures used in the scheduling algorithm. On one hand, several slots were added to the visibility window fact. This fact, traditionally contained information about the origin node, the relay node, start and end time of the contact opportunity and the characteristics (frequency band, supported data-rate, pointing) of the antenna. In the enhanced version it also contains information about the type of antenna (SA or MA), if it's already occupied by a contact or if is an user-relay or a direct-to-earth window.

On the other hand, the viable windows fact was also modified to include information about the characteristics of the visibility window they refer; is the window occupied by another service (and thus we will be overlapping), is the relay antenna a MA antenna, are there multiple payloads in the relay antenna.

Besides in order to implement the new asset traffic-limit feature, a new data structure was created. These facts contains information on 1) the maximum traffic that can be routed through each asset, 2) the accumulated traffic until the moment, 3) the accumulated number of contacts routed and 4) the accumulated time the asset has been used to relay data.

Table 3.1 shows the slot-structure of each of the facts involved in the scheduler.

| SCHEDULER::TRAFFIC | | | |
|---|---|---|---|
| *slot* | max-traffic | 100e9 | bits |
| *slot* | traffic | 5.32e9 | bits |
| *slot* | asset-id | McMurdo | - |
| *slot* | num-contacts | 5 | - |
| *slot* | contact-time | 1650 | s |
| *slot* | copied | yes/no | - |

| SCHEDULER::VIABLE-WINDOW | | | |
|---|---|---|---|
| *slot* | id | 68 | - |
| *slot* | tStart | 1274 | s |
| *slot* | tEnd | 3546 | s |
| *slot* | overlapping | yes/no | - |
| *slot* | node2-pos | GEO-3-1 | - |
| *slot* | node2-cons | GEO | - |
| *slot* | multiple-bands | yes/no | - |
| *slot* | MA-scheme | SA / MA | - |

| STK::VISIBILITY-WINDOW | | | |
|---|---|---|---|
| *slot* | id | 68 | - |
| *slot* | node1 | ISS | - |
| *slot* | node1-type | user | - |
| *slot* | node1-pay | USR-Ku | - |
| *slot* | node1-ant | A-USR-Ku | - |
| *slot* | node1-cons | - | - |
| *slot* | node1-pos | - | - |
| *slot* | node2 | TDRSS-sat-2 | - |
| *slot* | node2-type | relay | - |
| *slot* | node2-pay | SAKu | - |
| *slot* | node2-beam | 1 | - |
| *slot* | node2-ant | ASAKu | - |
| *slot* | node2-cons | GEO | - |
| *slot* | node2-pos | GEO-3-1 | - |
| *slot* | tStart | 1274 | s |
| *slot* | tEnd | 3546 | s |
| *slot* | band | Ku | - |
| *slot* | contact-data-rate | 100e6 | bps |
| *slot* | multiple-bands | yes/no | - |
| *slot* | to-delete | yes/no | - |
| *slot* | occupied | yes/no | - |

| SCHEDULER::REQ-CONTACT | | | |
|---|---|---|---|
| *slot* | id | 1 | - |
| *slot* | user | ISS | - |
| *slot* | service | SD | - |
| *slot* | nominal-data-rate | 100e6 | bps |
| *slot* | contact-data-rate | 80e6 | bps |
| *slot* | band | Ku | - |
| *slot* | duration | 1200 | s |
| *slot* | priority | 5 | - |
| *slot* | tStart | | s |
| *slot* | tEnd | | s |
| *slot* | data-volume | 96e9 | bits |
| *slot* | attempted | yes/no | - |
| *slot* | satisfied | yes/no | - |
| *multislot* | mandatory-assets | [TDRSS, NEN] | - |

Table 3.1: Example of the data structures involved in the scheduling algorithm. Columns correspond to the type of slot, name of slot, example of value and units respectively

### 3.3.2 Visibility Windows Computation

A *visibility window* defines a time-tagged contact opportunity between a user and a network asset. Visibility windows are computed with STK taking into account the assets orbital movement and constraints on the antennas' FOV, orientation or elevation angle (for ground stations). Each visibility window is associated to a communication payload and therefore has a specified frequency band and maximum data rate.

Additionally, STK is also used to compute multiple hop paths between ground stations and relay satellites with and without the presence of ISLs. They are used to trim visibility windows that define contacts between a relay satellite and a network customer when the relay is not connected to a ground station, as we assume a bent-pipe architecture for the network.

### 3.3.3 Variable Initialization

The `Scheduler Initialization` task performs two functions. First, it uses the pre-computed paths to discard certain visibility windows depending on the architecture. For example, paths that contain elements that are not present in the architecture, such as relays, ground stations, or ISL, are eliminated. Second, the priority of each service is calculated using the heuristic defined in reference [39]. Then, the scheduling process will proceed by progressively schedule services in descending order of priority.

### 3.3.4 Viable Window selection

Given a service to schedule, only a small subset of all the visibility windows can actually be used to grant contacts. We refer to them as the *viable windows* subset. The criteria that define the viability of a visibility window are as follows:

- **Same band criterion:** The service and the visibility window have the same frequency band.
- **Link capacity criterion:** The visibility window maximum data rate is higher than the service data rate.
- **Overlapping access criterion:** If the antenna is a Single Access (SA) antenna, this criterion is always satisfied if there is no other contact scheduled in that visibility window in the same frequency band. If there is a scheduled contact in another band, the user of the scheduled contact and the selected contact must be the same and the antenna must support both bands.
- **Multiple access criterion:** If the antenna is a Multiple Access (MA) antenna, this criterion is satisfied if the number of scheduled contacts in that window is lower than the number of beams of the MA antenna.

  Similar criteria apply to *user - user ground station* visibility windows.

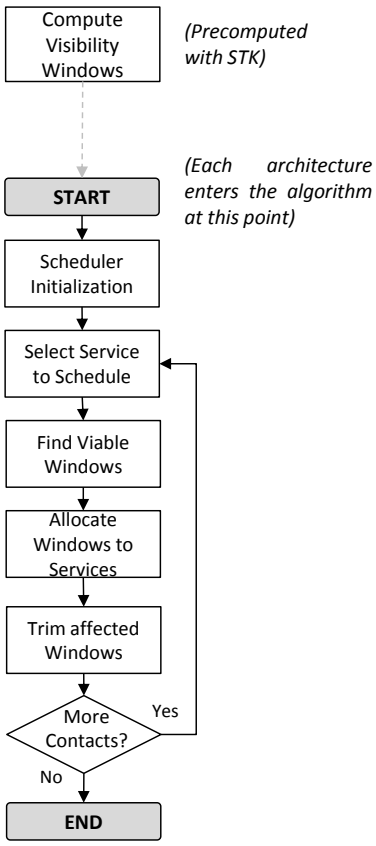



Figure 3-1: Flow chart of the Network Evaluator Module

The code that performs this comparison is written in Jess and can be found in the next page. Lines 1 to 31 conform the LHS of the rule that select the viable windows, and they implement the link capacity and the same band criterion. Line 33 verifies that the visibility window connects the customer mission with an asset of those specified in the mandatory-assets slot of the selected contact (if no assets were specified, it will be assumed that the user can contact any of the assets in the system).

Lines 40 to 43 verify, in case the selected visibility window uses a MA antenna, that the maximum number of contacts with that MA antenna has not been exceeded.

```
(defrule SCHEDULER2::create-visibility-windows-for-a-user-satellite
        "This functions creates all the viable windows user-relay-GS
         for the services attached to the selected user"

        (declare (no-loop TRUE))

        ;; Get General Information from Services / Users / Satellites
        (SCHEDULER::SEL-CONTACT (id 0) (user ?usr) (service ?s) (band ?b)
                (contact-data-rate ?dr) (mandatory-assets $?mand-assets))
        (MANIFEST::USER (id ?usr-id) (copy-from ?usr))

        ;; Check that the user payloads work in the band of the service
        (MANIFEST::PAYLOAD (satellite ?usr-id) (band ?b) (data-rate ?usr-dr) (id ?usr-pay))

        (MANIFEST::SATELLITE (id ?sat-id) (constel-name ?cons-name))

        ;; Get viable payloads from the relay side (same satellite, same band than the service)
        (MANIFEST::PAYLOAD (antenna ?sat-ant-id&~nil) (copy-from ?sat-pay) (band ?b)
                (data-rate ?ldr) (MA-scheme ?MA-scheme) (num-beams ?num-beams))
        (MANIFEST::ANTENNA (copy-from ?sat-ant) (id ?sat-ant-id&~nil) (satellite ?sat-id))

        ;; Get all the visibility windows between users and constellations
        ?f <- (STK::VISIBILITY-WINDOW (id ?w-id) (occupied ?oc) (multiple-bands ?mult-band)
                (node1 ?usr&~nil) (band ?b2) (node2-cons ?cons-name) (node2-ant ?sat-ant&~nil)
                (node2-pos ?node2-pos)(tStart ?tStart) (tEnd ?tEnd))

        (ASSETS::ELEMENT (name ?cons-name) (element-type ?as-type) (groups $?el-g) (sub-assets $?sa))

        (test (>= (float ?ldr) (float ?dr)))

        =>

        (if (check-for-mandatory-assets ?as-type $?mand-assets $?el-g $?sa) then
            (if (or (eq ?oc no) (and (neq ?b2 ?b) (eq ?mult-band yes)) (eq ?MA-scheme MA)) then


                (if (eq ?MA-scheme MA) then

                    ; Get the number of beams that are already used at that time in an MA-Payload
                    (bind ?c (count-query-results SCHEDULER::already-used-ma-windows ?cons-name
         ?node2-pos ?sat-ant ?sat-pay ?tStart ?tEnd))

                    (if (> ?num-beams ?c) then
                         (duplicate ?f (occupied no) (id ?*wnd-counter*) (node2 ?sat-id)
                        (node2-beam (+ 1 ?c))  (node1-pay ?usr-pay) (node2-pay ?sat-pay) (band ?b)
                        (contact-data-rate ?dr) )

                        (assert (SCHEDULER::VIABLE-WINDOW   (node2-cons ?cons-name) (overlapping no)
                        (id ?*wnd-counter*) (tStart ?tStart) (tEnd ?tEnd) (multiple-bands ?mult-band)
                        (node2-pos ?node2-pos) (MA-scheme ?MA-scheme)))

          (bind ?*wnd-counter* (+ ?*wnd-counter* 1))
                    )
                else
                    (if (eq ?oc yes) then
                            (duplicate ?f (occupied yes) (id ?*wnd-counter*) (node2 ?sat-id)
                          (node1-pay ?usr-pay) (node2-pay ?sat-pay) (band ?b) (contact-data-rate ?dr))

                            (assert (SCHEDULER::VIABLE-WINDOW   (node2-cons ?cons-name) (overlapping yes)
                                    (id ?*wnd-counter*) (tStart ?tStart) (tEnd ?tEnd) (multiple-bands
                                    ?mult-band) (node2-pos ?node2-pos) (MA-scheme ?MA-scheme)))

                            (bind ?*wnd-counter* (+ ?*wnd-counter* 1))
                    else
                        (if (eq ?w-id nil) then

                                (duplicate ?f (occupied no) (id ?*wnd-counter*) (node2 ?sat-id)
                                    (node1-pay ?usr-pay) (node2-pay ?sat-pay) (band ?b)
                                    (contact-data-rate ?dr))

                                (assert (SCHEDULER::VIABLE-WINDOW   (node2-cons ?cons-name)(overlapping
                                     no) (id ?*wnd-counter*) (tStart ?tStart) (tEnd ?tEnd) (multiple-
                                     bands ?mult-band) (node2-pos ?node2-pos) (MA-scheme ?MA-scheme)))

                                (bind ?*wnd-counter* (+ ?*wnd-counter* 1))

                            else
                                (assert (SCHEDULER::VIABLE-WINDOW   (node2-cons ?cons-name)(overlapping no)
                                      (id ?w-id) (tStart ?tStart) (tEnd ?tEnd) (multiple-bands ?mult-band)
                                      (node2-pos ?node2-pos) (MA-scheme ?MA-scheme)))
                            )
                    )
                )
            )
        )
    )
```

### 3.3.5 Allocating Resources to Services

The allocation of resources to services consists in assigning viable windows to individual contacts. For a contact to be eligible for allocation, the following conditions must be met: a) it must be possible to place the contact at least *mtbc* seconds after the last contact scheduled for that service; b) there must exist a window (or a train of concatenated windows) such that its total duration is higher than the duration of the contact.

The criteria to decide which viable visibility window best suits each contact is based on heuristics that try to mimic the current daily schedules generated by the Network Control Center Data System (NCCDS) for the TDRSS constellation. These criteria, in order of importance, are:

- Windows where a contact already exists are always preferred to empty windows. In other words, overlapping contacts of different services for the same user is always desirable.
- If a window offers the possibility of overlapping a contact in the future, it is preferred to windows that do not offer this possibility. In other words, multiband windows have preference over non-multiband windows.
- It is always preferable to schedule a contact through one single visibility window rather than concatenating several of them. This reduces the number of handovers in the network schedule.
- If concatenation is required, the window that creates the longest train of windows is preferred over the rest. In other words, the window that ends the latest is selected.

This part of the code was implemented in Java since the sequence of criteria to apply is constant and rule-based systems are known to be slower than traditional procedural languages in these circumstances. In particular, Fig.3-2 shows the flowchart of the algorithm used to allocate the visibility widows for a particular service. After importing and casting all the Jess facts to Java objects, we sort the viable windows in start time ascending order. This will enable us to more efficiently search through the list of windows, eliminating the need to loop through the whole list on each iteration.

Then, we search for a train of concatenated windows with a total duration higher than the contact duration. In case a valid train is found, the contact is successfully scheduled, time advances until the end of the train plus the minimum time between contacts and if there are more contacts to be scheduled, the process iterates. Should a valid train not be found, the scheduler advances the time until the start of the viable window that starts immediately after the ending time of the unsuccessful train and tries again. In case there are no more viable windows, the algorithm stops and the contacts that were successfully scheduled are asserted back in Jess' working memory.

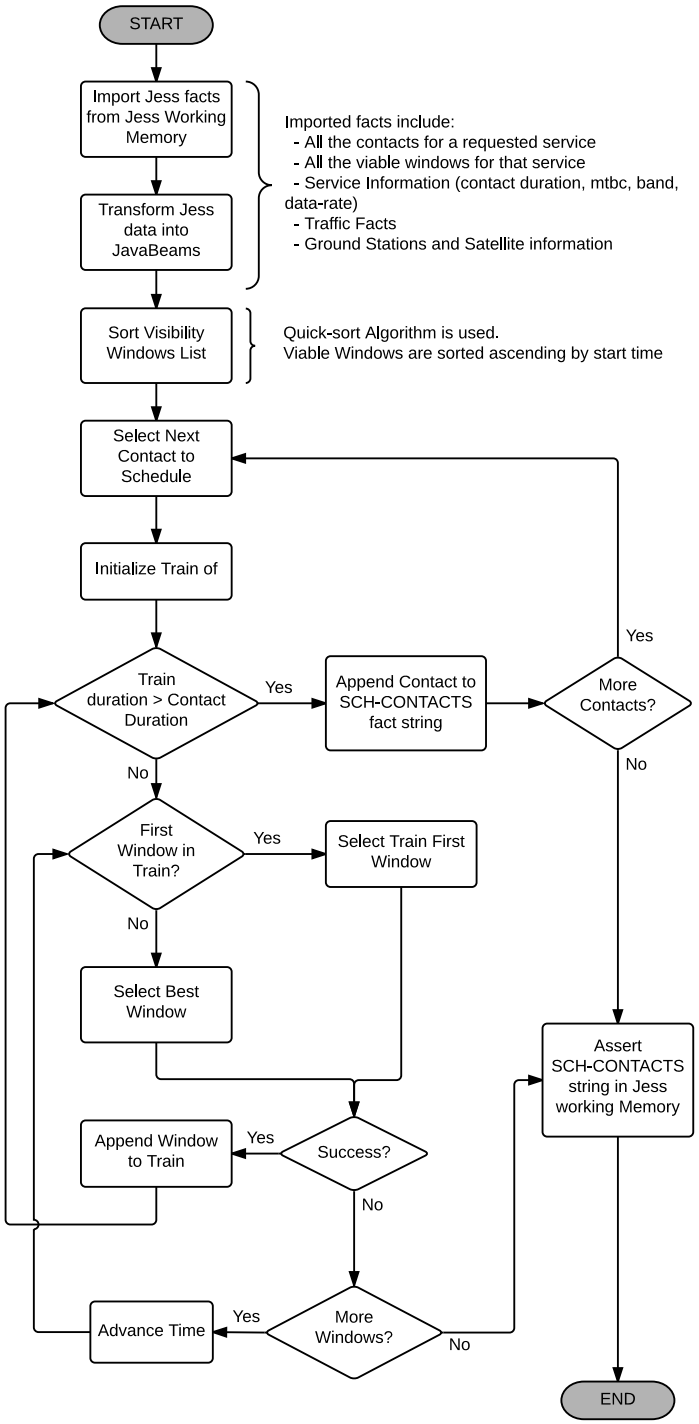



Figure 3-2: Flowchart of the Allocation process of Visibility Windows to Services

The algorithm that implements the aforementioned criteria to choose the best window to use is described next. The algorithm loops through all the candidate viable windows that can be appended to the train, and selects the best of them according

to the criteria presented above. Based on these criteria, a total of 11 conditions can render that the current window is better than the previously selected window. This conditions are checked using a set of nested if-then structures as depicted in Fig.3-3.

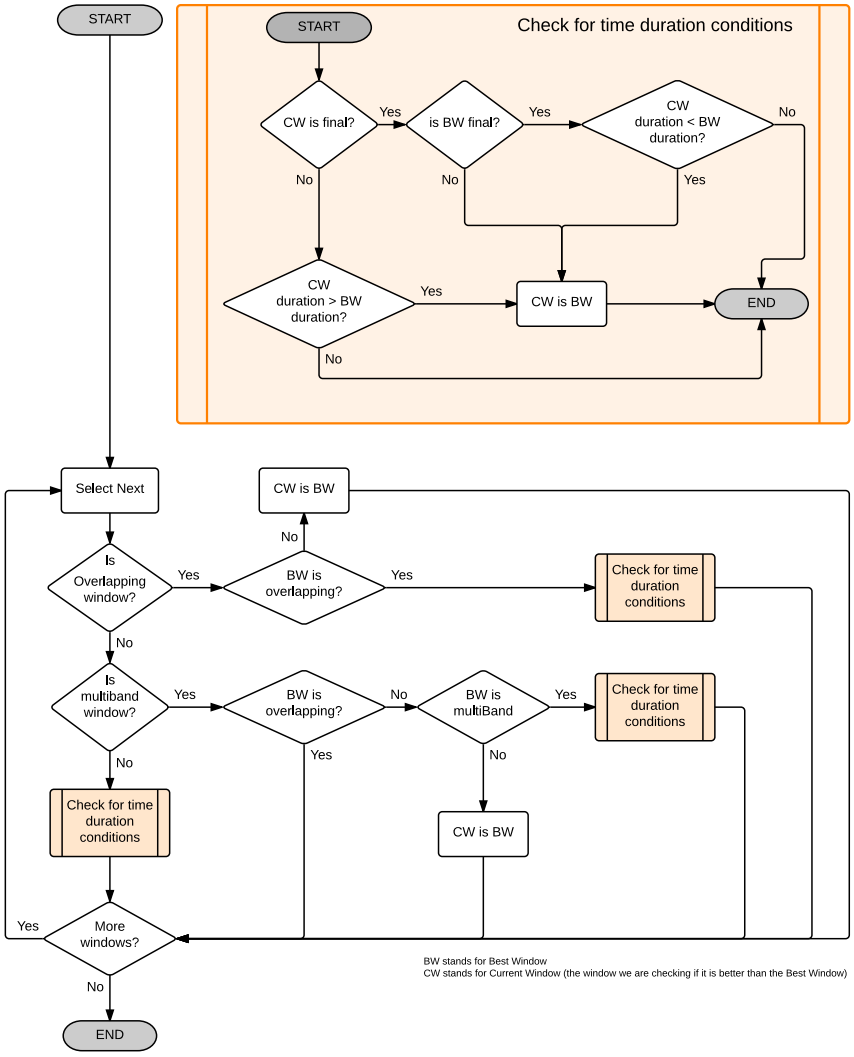



Figure 3-3: Flowchart of the best windows selection algorithm

### 3.3.6 Trimming Affected Visibility Windows

The assignment of contacts to certain visibility windows impacts the rest of visibility windows. For instance, if a visibility window is used by a SA antenna to provide a service to a certain user, no other user would be able to use that window during the duration of that contact. Hence, some visibility windows will need to be deleted, and

others might need to be trimmed to update their starting and ending times. Similarly, in the case of MA antennas, the number of beams available during the contacts' periods must be decreased according to the number of contacts scheduled.

## 3.4 Performance Improvements

Several tests were conducted in order to assess the improvements in terms of execution time among the different versions of the scheduler. Four scenarios were defined in order to assess the improvement under a variety of conditions. The baseline scenario corresponds to a case similar to a normal day of operations of the current version of TDRSS. The MA scenario is the same case but users are encouraged to user MA links. MEO + GEO and LEO + GEO scenarios are a scenario where in addition to TDRSS constellation, there is also a constellation of 6 and 66 satellites respectively in MEO and LEO. Table 3.2 shows the results for the different scenarios.

| **Scheduler version** | **Baseline Scenario** | **MA Scenario** | **MEO + GEO Scenario** | **LEO + GEO Scenario** |
|---|---|---|---|---|
| Initial Scheduler | 43.6 (s) | - | 221.0 (s) | 1734.3 (s) |
| Enhanced Version (No MA) | 7.4 (s) | - | 54.21 (s) | 168.9 (s) |
| Enhanced Version | 7.7 (s) | 12.3 (s) | 61.21 (s) | 180.2 (s) |

Table 3.2: Performance results for different versions of the Scheduler and different scenarios

As it can be observed, the enhanced version of the scheduler reduces the computation time in a factor comprised between 4 and 10 times as compared to the initial scheduler. This improvement is specially noticeable due to the reduced time of execution of the resource allocation step, now written in Java. On the other hand, the MA version of the scheduler incurs in a slight larger execution time than the no-MA version. This is more noticeable when a lot of satellites and visibility windows are present. Also, serving a lot of customers using the MA band causes the scheduler to decline in its performance, due to the visibility window explosion that MA generates.

Finally, during the experiments conducted in order to profile the execution time of the enhanced scheduler, it was determined that approximately 20 % of execution time was devoted to allocate resources, 60 % to trim the affected visibility windows and the rest of time was used in the rest of the steps. Knowing that, the author believes that the next step in order to achieve a higher performance consists on translating all the sections of the scheduler into Java.

# Chapter 4

# Using ITACA. A tradespace analysis of the Space Network

This chapter demonstrates the use of ITACA to analyze a tradespace of architectures for NASA's Space Network. This chapter is structured in two parts. First, the tradespace scenario is described, second, then results involving four different trade-offs are presented. These trades include the constellation design pattern, the frequency band selection, the disaggregation trade and the valuation of inter-satellite links (ISL) and ground stations.

## 4.1 Scenario Description

The analysis follows a series of guidelines (referring to the customer user base considered, the typology and topology of the space network as well as the available frequency bands) provided by NASA's UComm team. The following subsections describe

### 4.1.1 Tradespace Characterization

The notional architecture space, in terms of the topology of the network is:

- Constellation patterns: Two possible constellations are considered. A geostationary constellation with three evenly spaced orbital positions and a constellation in MEO with six orbital positions in an equatorial plane.
- Satellite disaggregation: In each orbital position, one or several satellites can be placed. In the former, we say that the architectures is monolithic whereas in the later we say the architecture is disaggregated with a degree of disaggregation equal to the number of satellites on each orbital position.

- Frequency band selection: Different antennas with payloads operating at different frequencies can be placed on each spacecraft. In particular, three different types of antennas are considered.
  - MA Antenna: A multiple access phased-array antenna with a payload that operates in S-band at 4 Mbps, based on current TDRS $2^{nd}$ capabilities.
  - SA Antenna: A single access 5 meter dish parabolic antenna with payloads that operates in S-band (10 Mbps), Ku-band (100 Mbps) and Ka-band (300 Mbps), similar to the ones on board TDRS $2^{nd}$ generation spacecraft.
  - Optical telescope: An optical telescope similar to the one under development for LCRD [16] that supports downlink speeds up to 1 Gbps.
- ISL presence: Inter-satellite links (ISL) are allowed among the satellites of the GEO-3-1 and the MEO-1-6 constellations. No inter-constellation links are allowed. This decision is coupled with the ground station decision.
- Ground stations: Two ground stations to relay the data from the Space Network relay satellites are available: Withe Sands Complex (WSC) and Guam Remote Ground Terminal (GRGT). If ISL is used only WSC will be used in the ground segment, whereas if spacecraft have no ISL capabilities both stations will be used.

The decision tree that represents the order in which these decisions must be taken, is displayed in Fig.4-1.

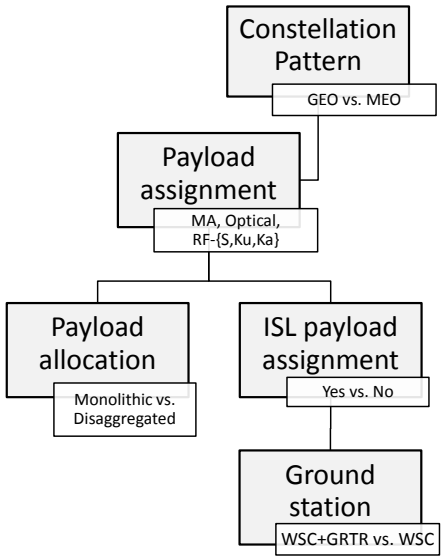



Figure 4-1: Decision Tree

In this study, only some predefined architecture options will be studied. We understand an architectural option as a set of configurations that have some decision options in common. These architecture options were specified by NASA's UComm team as the preferred set of architectures to analyze in their wave 2.1 analysis. Table 4.1

shows them. It can be observed that all the analyzed have at least a spacecraft with an optical payload, indicating their interest on moving forward from traditional RF technologies. Besides, some architecture options are paired being the only difference the fractionation strategy (monolithic architecture vs. disaggregated architectures), therefore this is considered to be a trade-off that deserves special analyses.

| | MA | | Optical | | RF | | S/C Approach | |
|---|---|---|---|---|---|---|---|---|
| | GEO | MEO | GEO | MEO | GEO | MEO | Monolithic | Disaggregated |
| AO1 | ✓ | - | ✓ | - | - | - | - | ✓ |
| AO2 | ✓ | - | ✓ | - | - | - | ✓ | - |
| AO3 | ✓ | - | - | ✓ | - | - | - | ✓ |
| AO4 | - | ✓ | - | ✓ | - | - | - | ✓ |
| AO5 | ✓ | - | ✓ | - | ✓ | - | - | ✓ |
| AO6 | ✓ | - | ✓ | - | ✓ | - | ✓ | - |
| AO7 | ✓ | - | - | ✓ | ✓ | - | - | ✓ |
| AO8 | ✓ | - | - | ✓ | ✓ | - | ✓ | - |
| AO9 | ✓ | - | ✓ | - | - | ✓ | - | ✓ |
| AO10 | ✓ | - | ✓ | - | - | ✓ | ✓ | - |
| AO11 | ✓ | - | - | ✓ | - | ✓ | - | ✓ |
| AO12 | ✓ | - | - | ✓ | - | ✓ | ✓ | - |
| AO13 | - | ✓ | - | ✓ | - | ✓ | ✓ | - |

Table 4.1: Users base characteristics

Finally, NASA specified a set of 19 possible canonical spacecraft to use as relay satellites as shown in 4.2. We generate all the possible combinations of canonical spacecraft in order to capture the disaggregation options. Two constraints are imposed; First, configurations can only have less than 4 spacecraft on each orbital position. Second, configurations cannot have different types of canonical spacecraft carrying similar payloads (i.e.: we cannot have a configuration with a SC1 and a SC3 on each orbital position).

| Spacecraft Name | Number of Links in Spacecraft | | | ITACA payload assignment | ITACA payload Allocation | Number Satellites |
|---|---|---|---|---|---|---|
| | MA | RF | Opt | | | |
| 'SC-1' | 1 | - | - | 256 | 1 | 1 |
| 'SC-2' | 2 | - | - | 384 | 11 | 1 |
| 'SC-3' | 3 | - | - | 448 | 111 | 1 |
| 'SC-4' | - | 1 | - | 32 | 1 | 1 |
| 'SC-5' | - | 2 | - | 48 | 11 | 1 |
| 'SC-6' | - | - | 1 | 8 | 1 | 1 |
| 'SC-7' | - | - | 2 | 12 | 11 | 1 |
| 'SC-8' | - | - | 3 | 14 | 111 | 1 |
| 'SC-9' | - | - | 4 | 15 | 1111 | 1 |
| 'SC-10' | 1 | 1 | - | 288 | 11 | 1 |
| 'SC-11' | 1 | 2 | - | 304 | 111 | 1 |
| 'SC-12' | 1 | - | 1 | 264 | 11 | 1 |
| 'SC-13' | 1 | - | 2 | 268 | 111 | 1 |
| 'SC-14' | 1 | - | 3 | 270 | 1111 | 1 |
| 'SC-15' | 1 | - | 4 | 271 | 11111 | 1 |
| 'SC-16' | 1 | 1 | 1 | 296 | 111 | 1 |
| 'SC-17' | 1 | 1 | 2 | 300 | 1111 | 1 |
| 'SC-18' | 1 | 2 | 1 | 312 | 1111 | 1 |
| 'SC-19' | 1 | 2 | 2 | 316 | 11111 | 1 |

Table 4.2: Users base characteristics

Given these constraints, the defined tradespace is composed of 332 configurations. A complete list of these architectures can be found in Appendix B.

Finally, some parameters are considered constant across all architectures. These include architectural decisions set to a fixed value such as the contract modality in which the payloads will be launched (procurement as opposed to hosted payloads) or

the type of network (bent-pipe as opposed to circuit switched or store-and-forward). Table 4.3 shows these values.

| Parameter | Value | Units |
|---|---|---|
| **Time horizon** | 30 | years |
| **Lifetime** | 10 | years |
| **Bus learning factor** | 1.00 | - |
| **Payload learning factor** | 1.00 | - |
| **Antennas learning factor** | 1.00 | - |
| **Epoch** | 1 April 2013 | - |
| **Number of steps** | 1440 | - |
| **Available Antennas** | SMA1, SMA2, SMA3, SA1, SA2, opt1, opt2, opt3, opt4 | - |
| **Constellation Patterns** | GEO-3-1, MEO-1-6 | - |
| **Ground Stations** | White Sands, Guam | - |
| **Contract Modality** | procurement | - |
| **Network type** | bent-pipe | - |
| **Number antennas NEN** | N/A | - |
| **NEN ground stations** | none | - |

Table 4.3: Architectural parameters

### 4.1.2 Customer User Base Characterization

This section describes the user base considered across the different scenarios analyzed. Input is based in a study performed by SCaN customer characterization team at NASA GSFC, that describes the different user cases that the integrated network os Space Communications and Navigation (SCaN) will have to support. The network load, that is, the number of customer missions of each user case is similar to the values in 2014. This configures the baseline scenario the different architectures will be evaluated against.

In particular, two big groups of users can be identified. First, LEO users include weather, Earth Observation Satellites and Human Space Flight missions such as the International Space Station (ISS). Second, Heliophysics users are located in high elliptical orbits (HEO) with an eccentricity close to 0.2 and an argument of the apogee of 20,000 km. This study deliberately omitted users in the Cis-Lunar environment, although its influence when architecting the network which will be analyzed in the future.

Table 4.4 shows the customer mission classes that the network provides service to. In total, 41 customers are served. 29 of them fly in LEO orbits and their missions include human space flight, weather services and some science missions, whereas the other 12 satellites fly in high-elliptical orbits in MEO and perform science missions. Also, for some of the user classes an explicit distinction between sun-synchronous (SSO) and equatorial orbits has been performed.

| User class | Orbit type | Num. Users | Schedule Priority | Inclination [deg] | Eccentricity [deg] | Altitude [km] |
|---|---|---|---|---|---|---|
| **LEO-Sci-Low** | SSO | 14 | 1 | 97.5 | 0.0 | 600 |
| **LEO-Sci-Low** | Equatorial | 7 | 1 | 40.0 | 0.0 | 600 |
| **LEO-Sci-Mod** | SSO | 1 | 2 | 97.5 | 0.0 | 600 |
| **LEO-Sci-High** | SSO | 3 | 4 | 97.5 | 0.0 | 700 |
| **HEO-Sci-Mod** | Equatorial | 12 | 2 | 0.0 | 0.2 | 20000 |
| **LEO-Weather** | Equatorial | 1 | 3 | 0.0 | 0.0 | 700 |
| **Sci-Alert** | SSO | 2 | 3 | 97.5 | 0.0 | 600 |
| **HSF-LEO** | Equatorial | 1 | 5 | 51.0 | 0.0 | 400 |
| **TOTAL** | - | 41 | - | - | - | - |

Table 4.4: Users base characteristics

When computing the benefit of the network, the satisfaction of each of the users is weighted according to the relative importance for the different stakeholders (namely NASA, UGSS and NOAA) that take benefit of using the SCaN network. In that sense, we assume that the ISS (represented by the HSF tag) will continue to be the main driver of the value delivered by the Space Network. Figure 4-2 shows the relative weight of each user class. The weight of all the customer missions on each user class is the same.

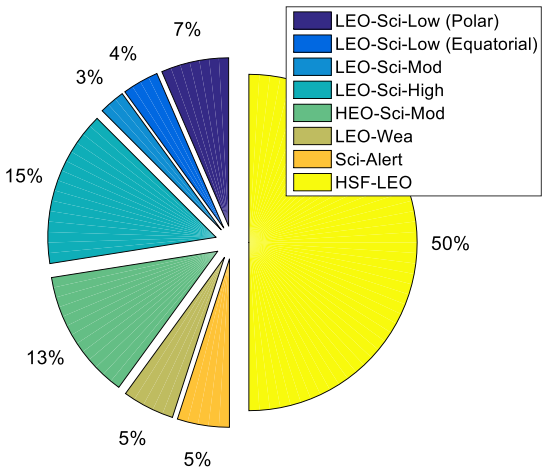



Figure 4-2: User case relative weighting to compute benefit

Each user case has defined a set of services that capture the heterogeneity of the different services the SCaN network has to serve. Most of the customer missions user the network to relay both Science Data (SD) and Telemetry, Tracking and Control (TT&C) services. While most of SD needs a high data-rate connection to download large volumes of information, TT&C can be served using connections with a data rate

of a few Mbps. Table 4.5 shows the characteristics of the different services associated to each user case.

| User class | description | contacts-per-day | duration | data-rate (Mbps) | data-volume (Gb) | min-time-btw-contacts (s) | latency (s) | weight | Scn. 1 band | Scn. 2 band | Scn. 3 band | Scn. 4 band |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LEO-Sci-Low** | TTC | 15 | 600 | 1 | 9 | 3600 | 7200 | 0.5 | * | * | * | S |
| **LEO-Sci-Low** | SD | 15 | 600 | 30 | 270 | 3600 | 7200 | 0.5 | * | * | opt | Ku |
| **LEO-Sci-Mod** | TTC | 15 | 600 | 1 | 9 | 3600 | 7200 | 0.5 | * | * | * | S |
| **LEO-Sci-Mod** | SD | 15 | 600 | 60 | 540 | 3600 | 7200 | 0.5 | * | * | opt | Ku |
| **LEO-Sci-High** | TTC | 15 | 900 | 1 | 18 | 3600 | 7200 | 0.5 | * | * | * | S |
| **LEO-Sci-High** | SD | 15 | 900 | 400 | 5400 | 3600 | 7200 | 0.5 | * | opt | opt | Ka |
| **HEO-Sci-Mod** | TTC | 15 | 600 | 1 | 9 | 3600 | 7200 | 0.5 | * | * | * | S |
| **HEO-Sci-Mod** | SD | 15 | 600 | 60 | 540 | 3600 | 7200 | 0.5 | * | * | opt | Ku |
| **LEO-Wea** | TTC | 15 | 900 | 1 | 18 | 3600 | 2100 | 0.5 | * | * | * | S |
| **LEO-Wea** | SD | 15 | 900 | 160 | 2160 | 3600 | 2100 | 0.5 | * | * | opt | Ku |
| **HSF-LEO** | TTC | 15 | 5760 | 0.072 | 6.2208 | 0 | 1800 | 0.4 | * | * | * | S |
| **HSF-LEO** | SD | 15 | 5760 | 150 | 12960 | 0 | 1800 | 0.4 | * | opt | opt | Ku |
| **HSF-LEO** | HDTV | 15 | 5760 | 150 | 12960 | 0 | 1800 | 0.2 | * | * | opt | Ku |

* designed bands

Table 4.5: Service characteristics

Besides, each service has associated a different weight for each figure of merit (data-volume, latency and user burden) in order to capture the importance and design constraints that affect them. For example, the ISS does not suffer from power budget constraints, therefore the relative weight of the user-burden FOM is set to zero. On the other hand, Science Alert (Sci-Alert) satellites such as SWIFT have very stringent requirements in terms of latency, thus the weight of this FOM was set to 0.6. Figure 4-3 shows the relative weight for each of the FOM and for each user case.

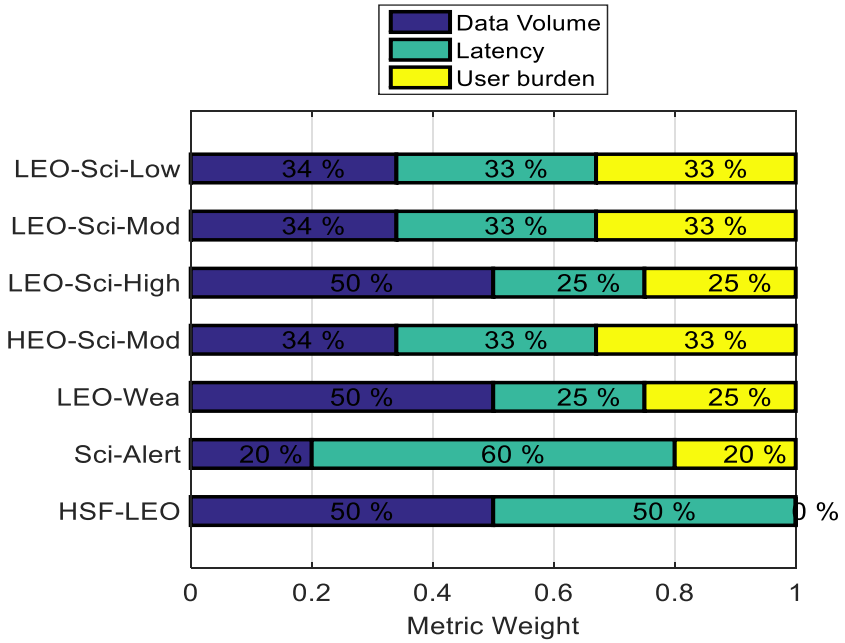



Figure 4-3: FOM relative weighting for each user case

The stakeholder satisfaction is then computed as:

$$\text{Stakeholder satisfaction} = \sum_{ob=1}^{OB} w_{ob} \sum_{usr=1}^{USR} w_{usr} \sum_{sr=1}^{SR} w_{sr} \sum_{fom=1}^{FOM} w_{fom} sat_{fom} \tag{4.1}$$

where $w_{ob}$ is the weight of each objective (each user class belongs to an objective), $w_{usr}$ are the weights of each user class as shown in Fig.4-2 , $w_{sr}$ are the weights of each services as depicted in Table 4.5 and $w_{fom}$ is the FOM weight as shows in Fig.4-3.

Finally, the user base characterization is subject to a high degree of uncertainty, due to the difficulties associated with having reliable predictions of how many customer missions will use the SCaN network in a budget constrained and delay-prone context. In particular, three different types of uncertainty have been identified: network load, users requirement and users communications system choice.

The first of them, the network load was not addressed in this study due to the lack of data about the different scenarios (in term of number of customer missions for each user case). The user requirements uncertainty is modeled by employing the user cases provided by NASA for the year 2014 [28]. Finally, the users' communications system choice was modeled by defining 4 different scenarios that address different policies and situations the agency might face in the future:

- Scenario 1: In the first scenario all the payloads used by each customer mission are designed taking into account the available bands in the network. Users will adopt a conservative approach to band selection, choosing the frequency band that allows them to transmit the desired data-rate at the lowest frequency.
- Scenario 2: This scenario represents a situation where high data rate customers are forced to transmit their science data using optical band.
- Scenario 3: In this case we analyze how the architectures behave when an agency level policy that forces all the customers to use optical band to transmit science data is imposed. This policy might be applied in order to reduce costs, as commonality across missions will be enforced.
- Scenario 4: In this scenario all the users transmit their data using the same bands that they are currently using. This scenario represent an extreme situation of customer risk aversion.

Last four columns in table 4.5 depict this information.

## 4.2 Trade-offs Analysis

This section presents and analyzes the aforementioned tradespace. Four main trades are identified, concerning to each of the decisions that generate the tradespace. During

the analysis, the effect of each of these decisions on the dependent metrics, benefit and cost, is isolated from the effect of the rest of the decisions. This process is conducted using the concept of neighbor architectures.

We define two architectures as neighbors with respect to a certain architectural decision if they only differ in the value assigned to that decision while having assigned similar options for the rest of the decisions. By doing pairwise comparisons among neighbor architectures we manage to isolate the effect that a particular decision takes over the metrics. Note that we are assuming that the effects of each decision on the network performance and cost are independent, that is, there are no interactions or synergies among different decisions.

### 4.2.1 Constellation pattern analysis

The first decision in he decision tree (as described in Fig.4-1) was the constellation pattern selection. Within the study, three options are possible. An only-GEO network with three evenly spaced orbital positions at a geostationary orbit, an only-MEO architecture with six evenly spaced orbital positions at a equatorial MEO orbit, or an architecture that combines both constellation patterns.

Figure 4-4 shows the different architectures' score in the benefit-cost space. Architectures are color coded based on the constellation pattern they present. It can be observed that only-GEO architectures always dominate the rest of the options; they achieve a higher benefit at a lower cost. In particular, it can be observed that only-GEO constellations provide full span of benefit. This trend was identified over te four analyzed scenarios, although due to space limitations only the plot for Scenario 1 was presented.

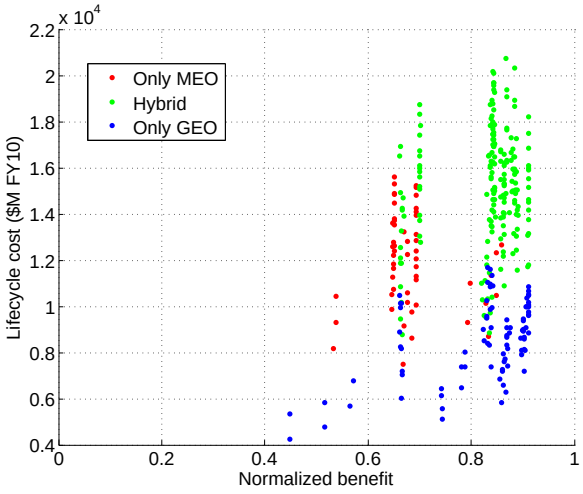
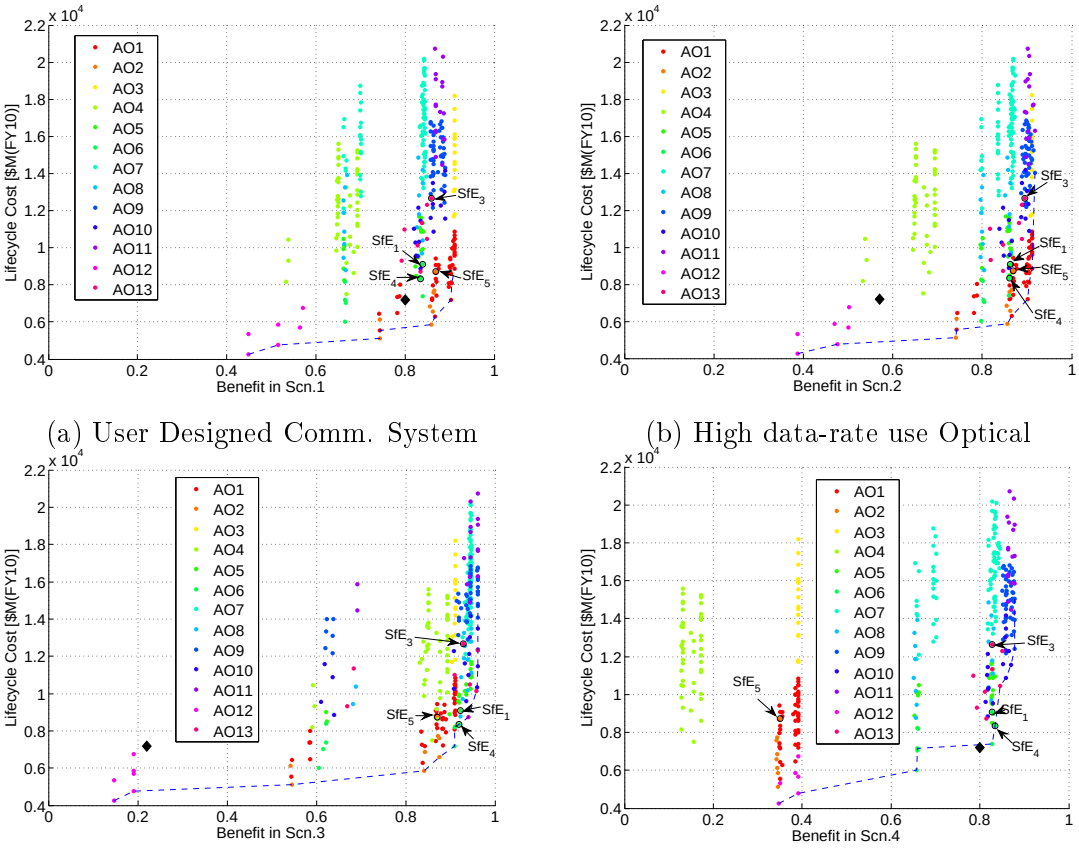



Figure 4-4: Architectures represented in the benefit-cost space. Architectures have been color coded attending the constellation pattern they use.

### 4.2.2 Antenna Allocation Analysis

The benefit achieved by each of the architectures is mainly driven by the frequency band selection for each of the relay spacecraft. As uncertainty in the communication technology utilization is not easy to quantify, four different scenarios were prepared. Figure **??** shows the benefit vs. cost plot results for each of the defined scenarios.




(a) User Designed Comm. System

(b) High data-rate use Optical

(c) All Science Data use Optical

(d) Current Design

Figure 4-5: Tradespace Architectures represented in the Benefit-Cost space. The black diamonds represent the score of current TDRSS architecture on each scenario

We will consider Scenario 1(Fig.4-5a) as the baseline scenario, as the users choose to design its communications system based on the available payloads in the network. It can be observed that Architecture Options 1,2,3 dominate the rest of the options. These architectures have SMA (mainly used to relay TT&C data) and optical (used to relay Science Data) payloads. In particular, monolithic architectures (AO2, AO5) outperform in terms on cost to the rest of the architectures. Besides, it can be observed that TDRSS is not in the Pareto front under this characteristics. It is also interesting to note that, for architectures carrying optical and the multi-RF bands SA antenna, none of the customer missions uses optical links to transmit their data. This is due to the fact that, given the service characteristics described in

Table 4.5 none of the users needs to use optical frequencies to transmit the requested amount of data. Finally, it is interesting noting that none the Systems for Evaluation, handpicked architectures chosen for deeper study was in the Pareto Front. This shows how traditional methods for architectural design might lack the broader view that characterizes a tradespace analysis.

Scenario 2 (Fig.4-5b) shows the outcome of the tool when high data-rate users are forced to move to optical band. In this scenario, the scores of architectural families AO1, AO2, AO3 and AO12 remains constant as compared to the baseline scenario. These architectures still dominate the rest of the options. In contrast, the TDRSS score decreases, due to the lack of optical capabilities that allow the network to serve the high science contacts in optical band. On the other hand, architectures where satellites carry both the RF & optical payloads achieve a slight increase in performance. This is due to the fact that using the optical band frees time in the RF SA antenna, allowing lower priority users to access the network in a more timely manner and increasing the score of the latency FOM.

Scenario 3 (as shown in Fig.4-5c) shows a situation where all the science data is transmitted using payloads in the optical band. It can be observed that TDRSS achieves a very low benefit, as the architecture is not capable of serving any of the science data services because of its lack of optical capabilities. On the other hand, enforcing an agency policy concerning the use of optical payloads results in architectural families AO5 - AO11 achieving the highest benefit score across the four scenarios. This is due to the increase in the user-burden metric, caused by the following reasons: 1) the optical links offer a better user burden than RF technology and 2) as the RF SA antennas are not used anymore to satisfy science data services, TT&C contacts can be scheduled using the 5 meter parabolic dish instead of the MA antenna, rendering in a better score for the user burden FOM.

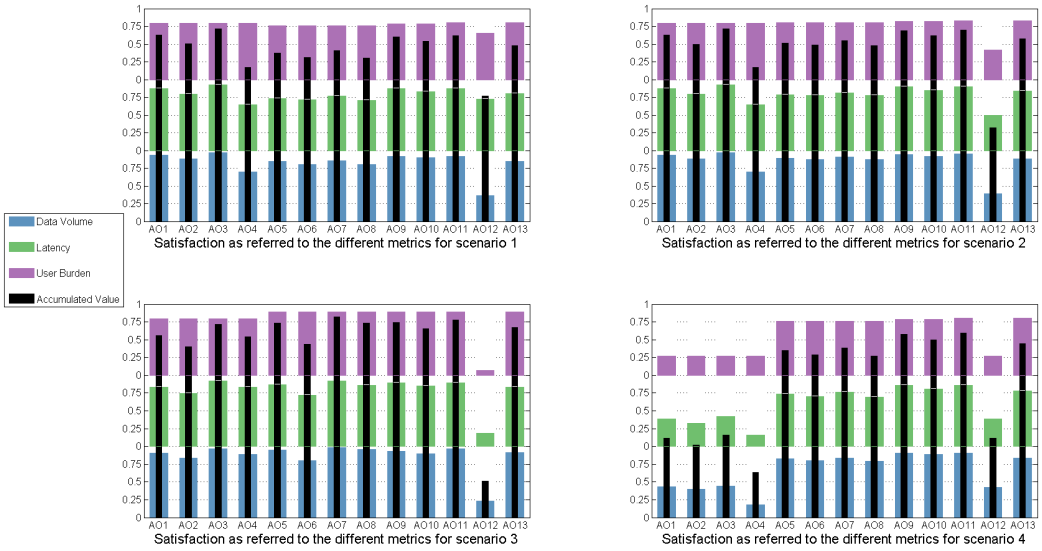



Figure 4-6: Figure of Merit relative score for each architecture option and scenario

Scenario 4 (Fig.4-5d) displays the results of the simulation when all the customers use a similar band to the one they use nowadays. In this context, optical architectures (AO 1, AO2 & AO3) achieve a very low benefit score, as they can only serve TT&C contacts. On the other hand, it can be shown that TDRSS is now a dominant architecture. SfE4, an upgraded version of TDRSS with two optical telescopes, achieves a very similar benefit score but at a higher cost (due to the addition of the telescopes). The metric scores achieved by AO-5 to AO-11 architectural families is similar to those obtained in the baseline scenario.

Finally, Figure 4-6 shows the average FOM value achieved by all the configurations in an architecture option for each of the four scenarios. It can be seen that for all the architecture options except AO12, the Scenario 3 offers the highest values for all the metrics. As mentioned above, architectures AO1, AO2 and AO3 show a decrease in the score of all the FOMs in Scenario 4.

Another interesting property to study was the robustness of an architecture across the four scenarios. In that sense, we compared the variability in the benefit score of an architecture (computed as the range of the benefit scores in the four identified scenarios) with its average benefit across scenarios. Figure 4-7 shows the robustness of different architectures. The x-axis determines the average benefit whereas the y-axes shows the variability in benefit. The utopia point (the global optima in the benefit - benefit variability space) is located in the bottom-right corner of the graph.

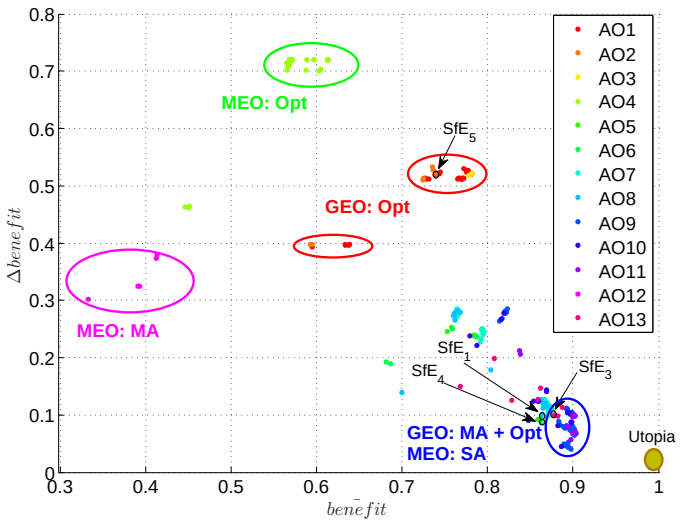



Figure 4-7: Architecture Options Robustness

It can be seen that architectures containing payloads in all the frequency bands are robust across different scenarios and at the same time obtain a high mean benefit. On the other hand, those architectures that only have the MA antenna and the optical telescope, present a much higher variability and a lower mean benefit, even though they were found to be optimal in the cost-benefit space for several of the scenarios presented before. This lower robustness comes from two facts: First only-optical architectures have a lower robustness towards customer demand uncertainty.

Second, the risk aversion from customers might limit the value they deliver, as no other alternatives apart from S-band are provided.

However, there is a dimension of the analysis missing in this picture: cost. Even though it is true that architectures with the full variety of frequency payloads are more robust, its cost doubles the cost of any only-optical architecture. To put it another way, robustness comes at a certain cost.

### 4.2.3 ISL + Ground Stations

Another interesting question that arose during the definition of the tradespace was the value of using ISL links to create a inter-satellite network and opening the possibility to close Guam Remote Ground Terminal. GRGT is a ground station built in 1990 to close the Exclusion Zone over the Indian Ocean, that is operated remotely from White-Sands Complex. Due to its remote location, it operation costs are among the highest in TDRSS.

In order to answer this question, we compare the cost difference between neighboring architectures with respect to the ISL and the ground station decisions.Figure 4-8 shows the difference in cost between neighbor architecture with ISL and without ISL but using WSC and GRGT. The x-axis shows the total number of satellites launched per generation on each architecture. It can be observed that only networks with high levels of disaggregation and both satellites in GEO and MEO prefer two stations as compared to using ISL and WSC. Moreover, the trade-off between ISL and ground segment seems to be lineal with the number of satellites launched. In particular, using ISL results beneficial if the number of satellites launched is below 20. This is due to the extra mass to be launched on each satellite to provide ISL connectivity, that renders in an increased costs of manufacturing and launch for the spacecraft.

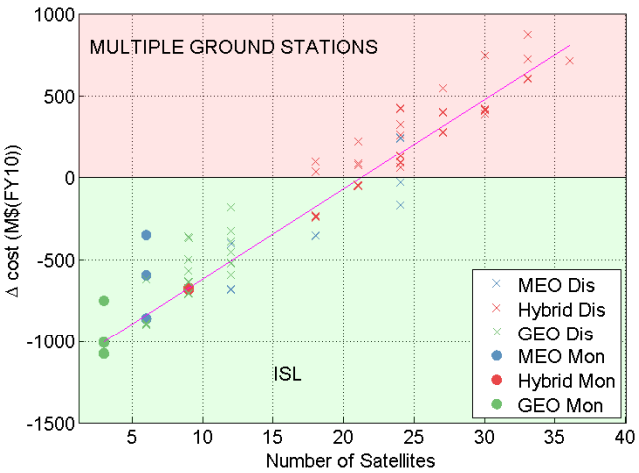



Figure 4-8: Difference in cost among ISL-neighbor architectures plotted against the number of satellites launched

On the other hand, Fig.4-9 shows how benefit is insensitive to the ISL and ground segment trade. In particular, the difference in benefit between neighbor architectures is always inferior to 5%.

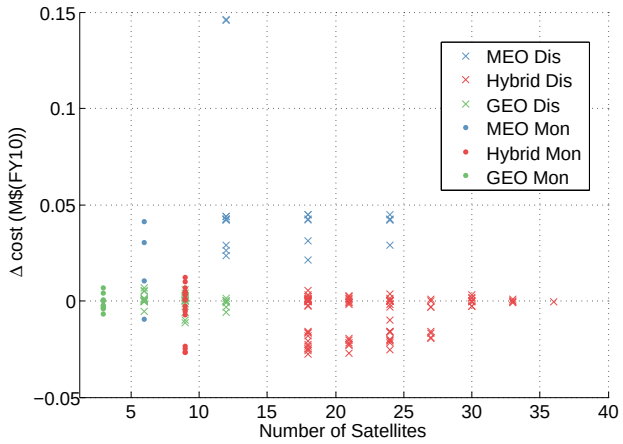



Figure 4-9: Delta of benefit among ISL-neighbor architectures vs. number of satellites launched.

### 4.2.4 Disaggregation trade-off

The last trade-off studied is the disaggregation trade. In that sense, it has been argued that flying the different payloads separately, using several spacecraft in the same orbital position might result in cost savings. In order to assess whether this statement is true or false, we compare the cost of neighbor architectures whit different disaggregation levels. We define the disaggregation level as the number of satellites on each orbital position. As such, neighboring architectures are defined as those that only differ in how payloads are allocated into spacecrafts. Note that there might be several configurations with the same disaggregation level for a certain architecture. Figure 4-10 illustrates this fact: for a disaggregation level of 2, two different options are presented. In option a), the monolithic satellite is decomposed in two identical satellites carrying an optical telescope and an RF single access antenna each, whereas in option b), the disaggregation is performed using a satellite carrying both of the optical telescopes and another one carrying the SA antennas.

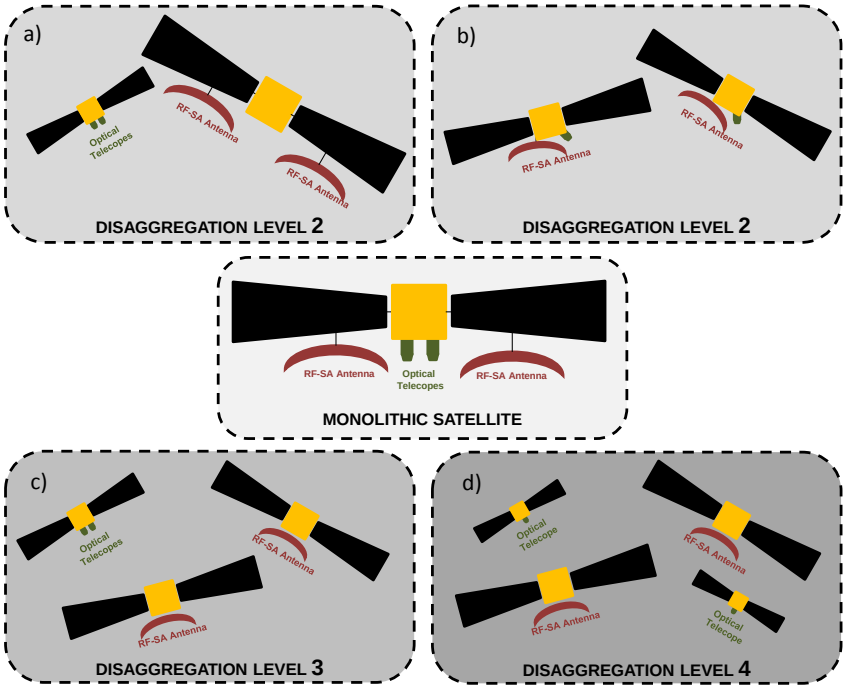



Figure 4-10: Different levels of disaggregation for a certain architecture

Figure 4-11 shows the difference in cost per year for neighbor architectures with different disaggregation levels. Value for only-GEO and only-MEO architectures are presented. We observe that the relationship between increment of the cost and the level of disaggregation is linear. Besides, it seems to be also linear with the number of satellites, as the slope of the MEO line is practically double than the one of the GEO line. In particular, we can allocate an extra cost of 3.5 \$M/year to each extra satellite in the constellation.

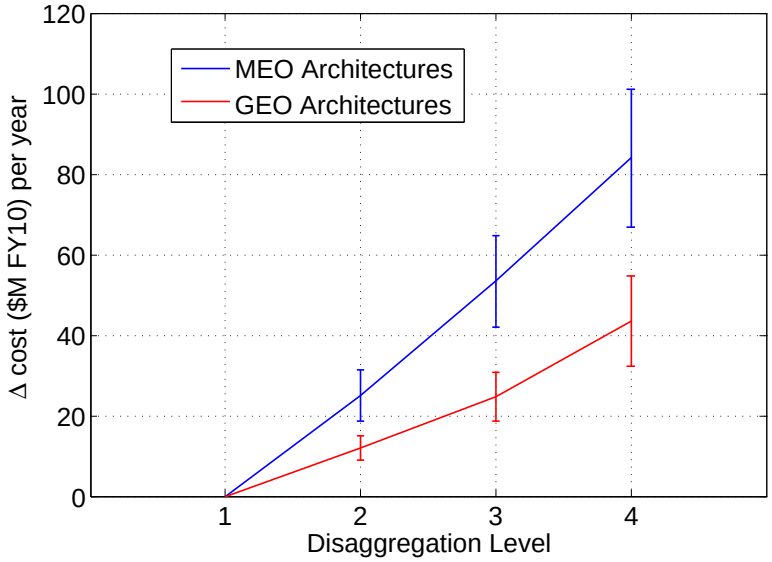



Figure 4-11: Extra cost incurred due to disaggregation (per constellation) for GEO and MEO architectures

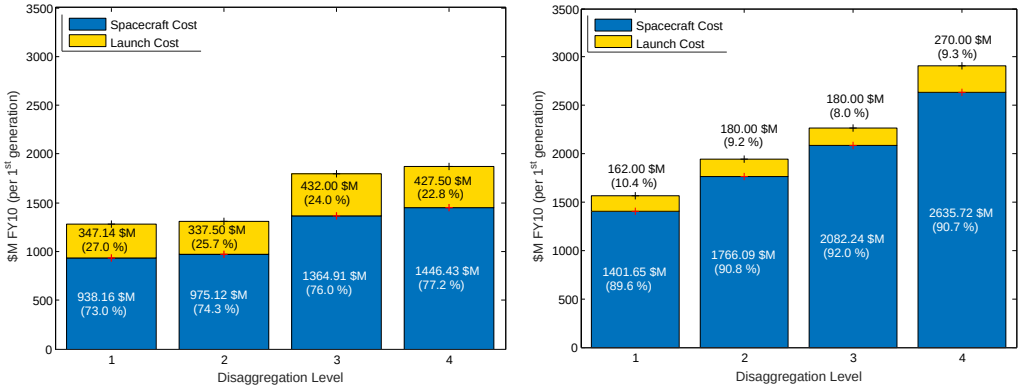
This extra cost can also be seen in another way. One of the clear benefits of disaggregation is robustness against unexpected events such as the loss of a spacecraft. In that sense, we can understand that the over-costs associated to disaggregation corresponds to the prize that has to be paid for the risk premium.

It is interesting to analyze the reasons behind this behavior. As for this trade-off analysis the only difference among neighbor architectures is the disaggregation level, we can impute this difference in cost to the difference of cost of the space segment (our model does not capture any extra operating cost in the ground segment depending on the number of satellites). The space segment cost is divided in the spacecraft cost (including recurring and non-recurring costs for the bus, payloads and antennas) and the launch cost.

Two arguments have been proposed to justify the use of fractionalization in new space architectures: First, someone hold that disaggregation induces lower launch costs, because launch-vehicle packaging efficiency increases as compared to monolithic architectures. Thus a lower number of launch vehicles is necessary. Second, some argue that the cost of manufacturing a satellite grows exponentially with the size and power of the payloads it carries, thus it is cheaper to manufacture several smaller spacecraft rather than a bigger monolithic one.

Figure 4-12a shows a cost decomposition (into launch costs and spacecraft costs) of the space segment costs for both GEO and MEO constellations. First, it can be inferred that the launch cost is not the driver of the space segment cost. In GEO constellations it always represents less than 25% whereas in MEO constellations it barely represents 10% of the total space segment cost. On the other hand, the absolute value of this cost increases with the disaggregation level, which indicates that a higher number of launches is necessary and thus the first reason for fractionalization does not hold. On the other hand, we observe that the spacecraft cost steadily increases with the disaggregation level. This leads to the conclusion that manufacturing multiple spacecraft, even smaller than a monolithic one has higher costs.

An important fact has to be noted at this point. In our analysis we assumed a learning factor of 1 during the manufacturing process. That is, all the satellites have a similar cost. However, we could compute the accumulative learning factor necessary to achieve a similar spacecraft cost independently of the disaggregation level. This learning factor results in average of 90 % for GEO satellites and 83 %. Both of these values are pretty high for the aerospace industry standards, more close to 95 %.




(a) GEO constellations (b) MEO constellations

Figure 4-12: Spacecraft cost decomposition

# Chapter 5

# Conclusions

## 5.1 Summary

This thesis has presented part of the work conducted this year in order to expand the capabilities of ITACA and to perform various analysis using this tool. Chapter 2 and Chapter 3 provided a detailed explanation of two of the improvements performed over the tool, while Chapter 4 includes the analysis and results of a tradespace analyzed for NASA's UComm Study group at Goddard Space Flight Center.

In particular, Chapter 2 described the crossover algorithms used when operating in GA mode to generate new architectures using the "genes" of previously evaluated ones. It first performed a literature review to present past research conducted in order to create crossover algorithms for similar problems to the ones faced in SAPs. Then, it described the algorithms for the down-selecting and assigning SAPs and it proposed a new crossover algorithm for the set partitioning SAP. It compared the performance of the latter in terms of bias to the performance achieved by other crossover algorithms for the same problem.

Chapter 3 presents the enhanced version of the scheduler. First, a detailed description of the overall operation of the scheduling algorithm was provided. Then, the implemented changes to provide the tool with multiple access, overlapping contacts, traffic limit and direct-contact to ground stations capabilities are described. Finally, how part of the code was ported from Jess to Java is depicted and then the performance of the new scheduling algorithm with the performance of the previous version is compared.

Finally, Chapter 4 presented a case study conducted for NASA's UComm Study group from Goddard Spaceflight Center. The analysis focused on the trade-offs involved in hosted payloads, use of inter-satellite links and constellation patterns for the spacecraft that will compose the next generation of NASA's Space Network. Four different scenarios were identified to capture the uncertainty associated to the communication system chosen by the users. Results showed that GEO constellations were

always preferred to those containing satellites in MEO. In addition, disaggregation was not recommended due to the extra cost incurred. Only-optical architectures had the best performance (in terms of benefit-cost ratio) compared to those carrying the RF SA antenna. To conclude, the use of ISL, a measurement that would allow to close Guam, was advantageous when the number of satellites in orbit was inferior to 20.

## 5.2 Future Work

The work on ITACA will continue during the next year. Two lines of work have been identified. On one hand, some work will be done to expand the capabilities of the tool. On the other hand, several analysis and studies of special interest have been identified.

### 5.2.1 Tool Expansion Tasks

One possible improvement is to implement a new rule-based module to translate mission communication requirements into concept of operations requirements. The theoretical work that founds this module [33]. This improvement will allow us to uncouple the network performance from the pre-assumed contact operations for each mission. Instead, each customer mission will design their conops taking into account the capabilities and characteristics of the architecture in evaluation.

Another improvement includes adding frequency band allocation as an architectural decision for the ISL and SGL antennas. The lack of this capability has been pointed out several times by the users of the tool at NASA GSFC and thus, is a special request from their side. From a implementation perspective this implies to create the new decisions and implement the crossover and mutation algorithms in Java. Also, the antenna template must be modified in order to capture the purpose of the antenna (SGL, ISL, user-relay). Finally, the links and RF spectrum management rules must be modified, in order to deal with infeasible architectures. A way of dealing with this is to upgrade the scheduler algorithm so that it performs basic network-simulation tasks that take into account the end-to-end routing process.

Finally, we want to expand the use of the tool so that time-dependent analyses can be conducted. In order to do that, a necessary improvement consists on integrating the graph-based tradespace exploration methodology developed in [12] in order to enable time-evolving architectural studies that take into account the phase-out of existing constellations, the evolution of the user base across time and the phase-in of new technologies into the system.

### 5.2.2 Architectural Analysis Tasks

Several new analyses to be performed in the course of the next year have been identified. In this sense, MIT will exercise the tool to examine key questions of interest to the NASA team. These analyses include:

- Perform sensitivity analysis of UComm networks against uncertainty in network load, user communication requirements and available communication technology.
- Use the model to include new frequency band allocations in the system. This includes the impact on cost and performance of using X, V and W-band transponders.
- Study possible architectures for a combined space and ground architecture that consolidates current SN and NEN networks, taking into account a time evolution of the demand forecast and current TDRSS reliability and failure rates.
- Analyze the coupling between network disaggregation and using hosted payloads to deploy network assets, as well as study the capabilities and cost of commercial service providers against future demands for the space segment.
- Determine the optimal designs for the ISL and SGL links given network performance and cost, communication technology and spectrum allocations.
- Include users in the Cis-Lunar Space and study how different network architectures satisfy their requirements.

# Appendix A

# Additional functions used in the Partitioning crossover operator

```
// Code below this line is used to calculate the crossover between two payloads allocations
private int[][] buildConstraintMatrix(int ant_pay_assign, long ant_pay_alloc, int child_pay_assign, int
Npayloads) {

    int[][] ret = new int[Npayloads][Npayloads];

    BitString child_s = new BitString(child_pay_assign, Npayloads);
    BitString ant_s = new BitString(ant_pay_assign, Npayloads);

    String ant_pay_alloc_s = String.valueOf(ant_pay_alloc);

    char[] tmp_alloc_s = new char[Npayloads];
    int i_c = 0;
    for (int i = 0; i < ant_s.length(); i++) {
        if (ant_s.getBit(i) == '1') {
            tmp_alloc_s[i] = ant_pay_alloc_s.charAt(i_c);
            i_c++;
        } else {
            tmp_alloc_s[i] = '0';
        }

    }
    String alloc_s = new String(tmp_alloc_s);

    for (int i = 0; i < Npayloads; i++) {
        for (int j = i; j < Npayloads; j++) {
            if (ant_s.getBit(i) == '1' && child_s.getBit(i) == '1') {
                if (ant_s.getBit(j) == '1' && child_s.getBit(j) == '1') {
                    if (alloc_s.charAt(i) == alloc_s.charAt(j)) {
                        ret[i][j] = 1;
                    } else {
                        ret[i][j] = 0;
                    }
                } else {
                    ret[i][j] = -1;
                }
            } else {
                ret[i][j] = -1;
            }
        }
    }
    return ret;
}
```

```
private long buildAllocationFromMatrix(int[][] m, int child_pay_assign) {

    int Npayloads = m.length;
    BitString child_s = new BitString(child_pay_assign, Npayloads);

    char[] pay_alloc = new char[Long.bitCount(child_pay_assign)];
    for (int i = 0; i < pay_alloc.length; i++) {
        pay_alloc[i] = '0';
    }

    char num = '1';
    for (int i = 0; i < Npayloads; i++) {
        if (pay_alloc[i] == '0') {
            pay_alloc[i] = num;
            for (int j = i + 1; j < Npayloads; j++) {
                if (m[i][j] == 1) {
                    pay_alloc[j] = num;
                }
            }
            num = String.valueOf(Math.min(Integer.parseInt(Character.toString(num)) + 1,
Nmax_sat)).charAt(0);
        }
    }

    return Long.parseLong((new String(pay_alloc)).replace("0", ""));
}
```

```
private int[][] crossoverMatrixes(int chil_pay_assign, int[][] father_m, int[][] mother_m, int Npayloads)
{
    Random rnd = new Random(System.currentTimeMillis());
    BitString child_s = new BitString(chil_pay_assign, Npayloads);

    int[][] ret = new int[Npayloads][Npayloads];
    int[][] res = new int[StringUtils.countMatches(child_s.getBits(),
"1")][StringUtils.countMatches(child_s.getBits(), "1")];

    ArrayList<String> both_present = new ArrayList<String>();
    ArrayList<String> one_present = new ArrayList<String>();
    ArrayList<String> none_present = new ArrayList<String>();

```

```
    // Build temporary matrix ret and populate array_lists with constraints that are defined in
    // both the parents, only on one of them or on none of them.
    for (int i = 0; i < Npayloads; i++) {
        for (int j = 0; j < Npayloads; j++) {
            if (j > i) {
                String str = new String("" + i + j);
                if (father_m[i][j] != -1 && mother_m[i][j] != -1) {
                    both_present.add(str);
                } else if (father_m[i][j] != -1 || mother_m[i][j] != -1) {
                    one_present.add(str);
                } else {
                    none_present.add(str);
                }
            }

            if (i != j) {
                ret[i][j] = -1;
            } else {
                ret[i][j] = 1;
            }
        }
    }

    int rnd_i;    int rnd_j;

    // Start the loop to take the decisions
    decision_fix_loop:
    while (both_present.size() + one_present.size() + none_present.size() > 0) {
        // Choose random decision (priority: both present > one present > none present
        do {
            if (both_present.size() > 0) {
                String str = both_present.remove(rnd.nextInt(both_present.size()));
                rnd_i = Integer.parseInt(Character.toString(str.charAt(0)));
                rnd_j = Integer.parseInt(Character.toString(str.charAt(1)));
            } else if (one_present.size() > 0) {
                String str = one_present.remove(rnd.nextInt(one_present.size()));
                rnd_i = Integer.parseInt(Character.toString(str.charAt(0)));
                rnd_j = Integer.parseInt(Character.toString(str.charAt(1)));
            } else {
                if (none_present.size() == 0) {
                    break decision_fix_loop;
                }
                String str = none_present.remove(rnd.nextInt(none_present.size()));
                rnd_i = Integer.parseInt(Character.toString(str.charAt(0)));
                rnd_j = Integer.parseInt(Character.toString(str.charAt(1)));
            }
        } while (ret[rnd_i][rnd_j] != -1);

        // Take the decision
        if (mother_m[rnd_i][rnd_j] != -1 && father_m[rnd_i][rnd_j] != -1) {
            if (rnd.nextBoolean()) {
                ret[rnd_i][rnd_j] = mother_m[rnd_i][rnd_j];
                ret[rnd_j][rnd_i] = mother_m[rnd_i][rnd_j];
            } else {
                ret[rnd_i][rnd_j] = father_m[rnd_i][rnd_j];
                ret[rnd_j][rnd_i] = father_m[rnd_i][rnd_j];
            }
        } else if (mother_m[rnd_i][rnd_j] != -1 && father_m[rnd_i][rnd_j] == -1) {
            ret[rnd_i][rnd_j] = mother_m[rnd_i][rnd_j];
            ret[rnd_j][rnd_i] = mother_m[rnd_i][rnd_j];
        } else if (mother_m[rnd_i][rnd_j] == -1 && father_m[rnd_i][rnd_j] != -1) {
            ret[rnd_i][rnd_j] = father_m[rnd_i][rnd_j];
            ret[rnd_j][rnd_i] = father_m[rnd_i][rnd_j];
        } else {
            ret[rnd_i][rnd_j] = rnd.nextInt(2);
            ret[rnd_j][rnd_i] = rnd.nextInt(2);
        }
        // Fix the matrix (as a decision might imply propagating earlier decisions
        fixDecisions(ret, rnd_i, rnd_j, Npayloads);
    }

    // Output formatting (create res matrix from ret matrix)
    int i_r = 0;    int j_r = 0;

    for (int i = 0; i < Npayloads; i++) {
        if (child_s.getBit(i) == '1') {
            for (int j = 0; j < Npayloads; j++) {
                if (child_s.getBit(j) == '1') {
                    res[i_r][j_r] = ret[i][j];
                    j_r++;
                }
            }
            j_r = 0;
            i_r++;
        }
    }
```

```
    return res;
}
```

```
private void fixDecisions(int[][] m, int rnd_i, int rnd_j, int Npayloads) {
    for (int i = 0; i < Npayloads; i++) {
        for (int i2 = i + 1; i2 < Npayloads; i2++) {
            if (m[i][rnd_j] == 1 && m[i2][rnd_j] == 1 && i != i2) {
                copy_row(m, i2, i, m[i2][rnd_j] == m[i][rnd_j], Npayloads);
                copy_row(m, i, i2, m[i2][rnd_j] == m[i][rnd_j], Npayloads);
            }
        }
    }

    for (int j = 0; j < Npayloads; j++) {
        for (int j2 = j + 1; j2 < Npayloads; j2++) {
            if (m[rnd_i][j] == 1 && m[rnd_i][j2] == 1 && j != j2) {
                copy_col(m, j2, j, m[rnd_i][j2] == m[rnd_i][j], Npayloads);
                copy_col(m, j, j2, m[rnd_i][j2] == m[rnd_i][j], Npayloads);
            }
        }
    }
}
```

```
private void copy_col(int[][] m, int rnd_j, int j, boolean b, int npayloads) {
    for (int i = 0; i < m.length; i++) {
        if (m[i][j] == -1 && m[i][rnd_j] != -1 && i != j) {
            if (b) {
                m[i][j] = m[i][rnd_j];
            } else if (m[i][rnd_j] == 1) {
                m[i][j] = 0;
            }

            fixDecisions(m, i, j, npayloads);
        }
    }
}
```

```
private void copy_row(int[][] m, int rnd_i, int i, boolean b, int npayloads) {
    for (int j = 0; j < m.length; j++) {
        if (m[i][j] == -1 && m[rnd_i][j] != -1 && i != j) {
            if (b) {
                m[i][j] = m[rnd_i][j];
            } else if (m[rnd_i][j] == 1) {
                m[i][j] = 0;
            }
            fixDecisions(m, i, j, npayloads);
        }
    }
}
```

# Appendix B

# Architectures Evaluated in the NASA UComm Case Study

| Name | AO | GEO Spacecraft | MEO Spacecraft | Antenna Assignment | | Antenna Allocation | | ISL | | Ground Stations | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | GEO | MEO | GEO | MEO | GEO | MEO | White-Sands | Guam |
| Arch-1 | AO1 | SC-1,SC-6 | | 12 | 0 | 264 | 0 | × | × | ✓ | ✓ |
| Arch-2 | AO1 | SC-1,SC-6 | | 12 | 0 | 264 | 0 | ✓ | ✓ | ✓ | - |
| Arch-3 | AO1 | SC-1,SC-7 | | 122 | 0 | 268 | 0 | × | × | ✓ | ✓ |
| Arch-4 | AO1 | SC-1,SC-7 | | 122 | 0 | 268 | 0 | ✓ | ✓ | ✓ | - |
| Arch-5 | AO1 | SC-1,SC-8 | | 1222 | 0 | 270 | 0 | × | × | ✓ | ✓ |
| Arch-6 | AO1 | SC-1,SC-8 | | 1222 | 0 | 270 | 0 | ✓ | ✓ | ✓ | - |
| Arch-7 | AO1 | SC-1,SC-9 | | 12222 | 0 | 271 | 0 | × | × | ✓ | ✓ |
| Arch-8 | AO1 | SC-1,SC-9 | | 12222 | 0 | 271 | 0 | ✓ | ✓ | ✓ | - |
| Arch-9 | AO1 | SC-1,2xSC-6 | | 123 | 0 | 268 | 0 | × | × | ✓ | ✓ |
| Arch-10 | AO1 | SC-1,2xSC-6 | | 123 | 0 | 268 | 0 | ✓ | ✓ | ✓ | - |
| Arch-11 | AO1 | SC-1,3xSC-6 | | 1234 | 0 | 270 | 0 | × | × | ✓ | ✓ |
| Arch-12 | AO1 | SC-1,3xSC-6 | | 1234 | 0 | 270 | 0 | ✓ | ✓ | ✓ | - |
| Arch-13 | AO1 | SC-1,2xSC-7 | | 12233 | 0 | 271 | 0 | × | × | ✓ | ✓ |
| Arch-14 | AO1 | SC-1,2xSC-7 | | 12233 | 0 | 271 | 0 | ✓ | ✓ | ✓ | - |
| Arch-15 | AO1 | SC-2,SC-6 | | 112 | 0 | 392 | 0 | × | × | ✓ | ✓ |
| Arch-16 | AO1 | SC-2,SC-6 | | 112 | 0 | 392 | 0 | ✓ | ✓ | ✓ | - |
| Arch-17 | AO1 | SC-2,SC-7 | | 1122 | 0 | 396 | 0 | × | × | ✓ | ✓ |
| Arch-18 | AO1 | SC-2,SC-7 | | 1122 | 0 | 396 | 0 | ✓ | ✓ | ✓ | - |
| Arch-19 | AO1 | SC-2,SC-8 | | 11222 | 0 | 398 | 0 | × | × | ✓ | ✓ |
| Arch-20 | AO1 | SC-2,SC-8 | | 11222 | 0 | 398 | 0 | ✓ | ✓ | ✓ | - |
| Arch-21 | AO1 | SC-2,SC-9 | | 112222 | 0 | 399 | 0 | × | × | ✓ | ✓ |
| Arch-22 | AO1 | SC-2,SC-9 | | 112222 | 0 | 399 | 0 | ✓ | ✓ | ✓ | - |
| Arch-23 | AO1 | SC-2,2xSC-6 | | 1123 | 0 | 396 | 0 | × | × | ✓ | ✓ |
| Arch-24 | AO1 | SC-2,2xSC-6 | | 1123 | 0 | 396 | 0 | ✓ | ✓ | ✓ | - |
| Arch-25 | AO1 | SC-2,3xSC-6 | | 11234 | 0 | 398 | 0 | × | × | ✓ | ✓ |
| Arch-26 | AO1 | SC-2,3xSC-6 | | 11234 | 0 | 398 | 0 | ✓ | ✓ | ✓ | - |
| Arch-27 | AO1 | SC-2,2xSC-7 | | 112233 | 0 | 399 | 0 | × | × | ✓ | ✓ |
| Arch-28 | AO1 | SC-2,2xSC-7 | | 112233 | 0 | 399 | 0 | ✓ | ✓ | ✓ | - |
| Arch-29 | AO1 | SC-3,SC-6 | | 1112 | 0 | 456 | 0 | × | × | ✓ | ✓ |
| Arch-30 | AO1 | SC-3,SC-6 | | 1112 | 0 | 456 | 0 | ✓ | ✓ | ✓ | - |
| Arch-31 | AO1 | SC-3,SC-7 | | 11122 | 0 | 460 | 0 | × | × | ✓ | ✓ |
| Arch-32 | AO1 | SC-3,SC-7 | | 11122 | 0 | 460 | 0 | ✓ | ✓ | ✓ | - |
| Arch-33 | AO1 | SC-3,SC-8 | | 111222 | 0 | 462 | 0 | × | × | ✓ | ✓ |
| Arch-34 | AO1 | SC-3,SC-8 | | 111222 | 0 | 462 | 0 | ✓ | ✓ | ✓ | - |
| Arch-35 | AO1 | SC-3,SC-9 | | 1112222 | 0 | 463 | 0 | × | × | ✓ | ✓ |
| Arch-36 | AO1 | SC-3,SC-9 | | 1112222 | 0 | 463 | 0 | ✓ | ✓ | ✓ | - |
| Arch-37 | AO1 | SC-3,2xSC-6 | | 11123 | 0 | 460 | 0 | × | × | ✓ | ✓ |
| Arch-38 | AO1 | SC-3,2xSC-6 | | 11123 | 0 | 460 | 0 | ✓ | ✓ | ✓ | - |
| Arch-39 | AO1 | SC-3,3xSC-6 | | 111234 | 0 | 462 | 0 | × | × | ✓ | ✓ |
| Arch-40 | AO1 | SC-3,3xSC-6 | | 111234 | 0 | 462 | 0 | ✓ | ✓ | ✓ | - |
| Arch-41 | AO1 | SC-3,2xSC-7 | | 1112233 | 0 | 463 | 0 | × | × | ✓ | ✓ |
| Arch-42 | AO1 | SC-3,2xSC-7 | | 1112233 | 0 | 463 | 0 | ✓ | ✓ | ✓ | - |
| Arch-43 | AO1 | 2xSC-1,2xSC-6 | | 1234 | 0 | 396 | 0 | × | × | ✓ | ✓ |
| Arch-44 | AO1 | 2xSC-1,2xSC-6 | | 1234 | 0 | 396 | 0 | ✓ | ✓ | ✓ | - |
| Arch-45 | AO1 | 2xSC-1,2xSC-7 | | 123344 | 0 | 399 | 0 | × | × | ✓ | ✓ |
| Arch-46 | AO1 | 2xSC-1,2xSC-7 | | 123344 | 0 | 399 | 0 | ✓ | ✓ | ✓ | - |
| Arch-47 | AO2 | SC-12 | | 11 | 0 | 264 | 0 | × | × | ✓ | ✓ |
| Arch-48 | AO2 | SC-12 | | 11 | 0 | 264 | 0 | ✓ | ✓ | ✓ | - |
| Arch-49 | AO2 | SC-13 | | 111 | 0 | 268 | 0 | × | × | ✓ | ✓ |
| Arch-50 | AO2 | SC-13 | | 111 | 0 | 268 | 0 | ✓ | ✓ | ✓ | - |
| Arch-51 | AO2 | SC-14 | | 1111 | 0 | 270 | 0 | × | × | ✓ | ✓ |
| Arch-52 | AO2 | SC-14 | | 1111 | 0 | 270 | 0 | ✓ | ✓ | ✓ | - |
| Arch-53 | AO2 | SC-15 | | 11111 | 0 | 271 | 0 | × | × | ✓ | ✓ |
| Arch-54 | AO2 | SC-15 | | 11111 | 0 | 271 | 0 | ✓ | ✓ | ✓ | - |
| Arch-55 | AO3 | 2xSC-1 | 2xSC-6 | 12 | 12 | 384 | 12 | × | × | ✓ | ✓ |
| Arch-56 | AO3 | 2xSC-1 | 2xSC-6 | 12 | 12 | 384 | 12 | ✓ | ✓ | ✓ | - |
| Arch-57 | AO3 | 2xSC-1 | 3xSC-6 | 12 | 123 | 384 | 14 | × | × | ✓ | ✓ |
| Arch-58 | AO3 | 2xSC-1 | 3xSC-6 | 12 | 123 | 384 | 14 | ✓ | ✓ | ✓ | - |
| Arch-59 | AO3 | 2xSC-1 | 2xSC-7 | 12 | 1122 | 384 | 15 | × | × | ✓ | ✓ |
| Arch-60 | AO3 | 2xSC-1 | 2xSC-7 | 12 | 1122 | 384 | 15 | ✓ | ✓ | ✓ | - |
| Arch-61 | AO3 | 2xSC-1 | 4xSC-6 | 12 | 1234 | 384 | 15 | × | × | ✓ | ✓ |
| Arch-62 | AO3 | 2xSC-1 | 4xSC-6 | 12 | 1234 | 384 | 15 | ✓ | ✓ | ✓ | - |
| Arch-63 | AO3 | 3xSC-1 | 2xSC-6 | 123 | 12 | 448 | 12 | × | × | ✓ | ✓ |
| Arch-64 | AO3 | 3xSC-1 | 2xSC-6 | 123 | 12 | 448 | 12 | ✓ | ✓ | ✓ | - |
| Arch-65 | AO3 | 3xSC-1 | 3xSC-6 | 123 | 123 | 448 | 14 | × | × | ✓ | ✓ |
| Arch-66 | AO3 | 3xSC-1 | 3xSC-6 | 123 | 123 | 448 | 14 | ✓ | ✓ | ✓ | - |
| Arch-67 | AO3 | 3xSC-1 | 2xSC-7 | 123 | 1122 | 448 | 15 | × | × | ✓ | ✓ |
| Arch-68 | AO3 | 3xSC-1 | 2xSC-7 | 123 | 1122 | 448 | 15 | ✓ | ✓ | ✓ | - |
| Arch-69 | AO3 | 3xSC-1 | 4xSC-6 | 123 | 1234 | 448 | 15 | × | × | ✓ | ✓ |
| Arch-70 | AO3 | 3xSC-1 | 4xSC-6 | 123 | 1234 | 448 | 15 | ✓ | ✓ | ✓ | - |
| Arch-71 | AO4 | MEO-SC-1,SC-6 | | 0 | 12 | 0 | 264 | × | × | ✓ | ✓ |
| Arch-72 | AO4 | MEO-SC-1,SC-6 | | 0 | 12 | 0 | 264 | ✓ | ✓ | ✓ | - |
| Arch-73 | AO4 | MEO-SC-1,SC-7 | | 0 | 122 | 0 | 268 | × | × | ✓ | ✓ |
| Arch-74 | AO4 | MEO-SC-1,SC-7 | | 0 | 122 | 0 | 268 | ✓ | ✓ | ✓ | - |
| Arch-75 | AO4 | MEO-SC-1,SC-8 | | 0 | 1222 | 0 | 270 | × | × | ✓ | ✓ |
| Arch-76 | AO4 | MEO-SC-1,SC-8 | | 0 | 1222 | 0 | 270 | ✓ | ✓ | ✓ | - |
| Arch-77 | AO4 | MEO-SC-1,SC-9 | | 0 | 12222 | 0 | 271 | × | × | ✓ | ✓ |
| Arch-78 | AO4 | MEO-SC-1,SC-9 | | 0 | 12222 | 0 | 271 | ✓ | ✓ | ✓ | - |
| Arch-79 | AO4 | MEO-SC-1,2xSC-6 | | 0 | 123 | 0 | 268 | × | × | ✓ | ✓ |
| Arch-80 | AO4 | MEO-SC-1,2xSC-6 | | 0 | 123 | 0 | 268 | ✓ | ✓ | ✓ | - |

| Name | AO | GEO Spacecraft | MEO Spacecraft | Antenna Assignment | | Antenna Allocation | | ISL | | Ground Stations | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | GEO | MEO | GEO | MEO | GEO | MEO | White-Sands | Guam |
| Arch-81 | AO4 | MEO-SC-1,3xSC-6 | | 0 | 1234 | 0 | 270 | × | × | ✓ | ✓ |
| Arch-82 | AO4 | MEO-SC-1,3xSC-6 | | 0 | 1234 | 0 | 270 | ✓ | ✓ | ✓ | - |
| Arch-83 | AO4 | MEO-SC-1,2xSC-7 | | 0 | 12233 | 0 | 271 | × | × | ✓ | ✓ |
| Arch-84 | AO4 | MEO-SC-1,2xSC-7 | | 0 | 12233 | 0 | 271 | ✓ | ✓ | ✓ | - |
| Arch-85 | AO4 | MEO-SC-2,SC-6 | | 0 | 112 | 0 | 392 | × | × | ✓ | ✓ |
| Arch-86 | AO4 | MEO-SC-2,SC-6 | | 0 | 112 | 0 | 392 | ✓ | ✓ | ✓ | - |
| Arch-87 | AO4 | MEO-SC-2,SC-7 | | 0 | 1122 | 0 | 396 | × | × | ✓ | ✓ |
| Arch-88 | AO4 | MEO-SC-2,SC-7 | | 0 | 1122 | 0 | 396 | ✓ | ✓ | ✓ | - |
| Arch-89 | AO4 | MEO-SC-2,SC-8 | | 0 | 11222 | 0 | 398 | × | × | ✓ | ✓ |
| Arch-90 | AO4 | MEO-SC-2,SC-8 | | 0 | 11222 | 0 | 398 | ✓ | ✓ | ✓ | - |
| Arch-91 | AO4 | MEO-SC-2,SC-9 | | 0 | 112222 | 0 | 399 | × | × | ✓ | ✓ |
| Arch-92 | AO4 | MEO-SC-2,SC-9 | | 0 | 112222 | 0 | 399 | ✓ | ✓ | ✓ | - |
| Arch-93 | AO4 | MEO-SC-2,2xSC-6 | | 0 | 1123 | 0 | 396 | × | × | ✓ | ✓ |
| Arch-94 | AO4 | MEO-SC-2,2xSC-6 | | 0 | 1123 | 0 | 396 | ✓ | ✓ | ✓ | - |
| Arch-95 | AO4 | MEO-SC-2,3xSC-6 | | 0 | 11234 | 0 | 398 | × | × | ✓ | ✓ |
| Arch-96 | AO4 | MEO-SC-2,3xSC-6 | | 0 | 11234 | 0 | 398 | ✓ | ✓ | ✓ | - |
| Arch-97 | AO4 | MEO-SC-2,2xSC-7 | | 0 | 112233 | 0 | 399 | × | × | ✓ | ✓ |
| Arch-98 | AO4 | MEO-SC-2,2xSC-7 | | 0 | 112233 | 0 | 399 | ✓ | ✓ | ✓ | - |
| Arch-99 | AO4 | MEO-SC-3,SC-6 | | 0 | 1112 | 0 | 456 | × | × | ✓ | ✓ |
| Arch-100 | AO4 | MEO-SC-3,SC-6 | | 0 | 1112 | 0 | 456 | ✓ | ✓ | ✓ | - |
| Arch-101 | AO4 | MEO-SC-3,SC-7 | | 0 | 11122 | 0 | 460 | × | × | ✓ | ✓ |
| Arch-102 | AO4 | MEO-SC-3,SC-7 | | 0 | 11122 | 0 | 460 | ✓ | ✓ | ✓ | - |
| Arch-103 | AO4 | MEO-SC-3,SC-8 | | 0 | 111222 | 0 | 462 | × | × | ✓ | ✓ |
| Arch-104 | AO4 | MEO-SC-3,SC-8 | | 0 | 111222 | 0 | 462 | ✓ | ✓ | ✓ | - |
| Arch-105 | AO4 | MEO-SC-3,SC-9 | | 0 | 1112222 | 0 | 463 | × | × | ✓ | ✓ |
| Arch-106 | AO4 | MEO-SC-3,SC-9 | | 0 | 1112222 | 0 | 463 | ✓ | ✓ | ✓ | - |
| Arch-107 | AO4 | MEO-SC-3,2xSC-6 | | 0 | 11123 | 0 | 460 | × | × | ✓ | ✓ |
| Arch-108 | AO4 | MEO-SC-3,2xSC-6 | | 0 | 11123 | 0 | 460 | ✓ | ✓ | ✓ | - |
| Arch-109 | AO4 | MEO-SC-3,3xSC-6 | | 0 | 111234 | 0 | 462 | × | × | ✓ | ✓ |
| Arch-110 | AO4 | MEO-SC-3,3xSC-6 | | 0 | 111234 | 0 | 462 | ✓ | ✓ | ✓ | - |
| Arch-111 | AO4 | MEO-SC-3,2xSC-7 | | 0 | 1112233 | 0 | 463 | × | × | ✓ | ✓ |
| Arch-112 | AO4 | MEO-SC-3,2xSC-7 | | 0 | 1112233 | 0 | 463 | ✓ | ✓ | ✓ | - |
| Arch-113 | AO4 | MEO-2xSC-1,2xSC-6 | | 0 | 1234 | 0 | 396 | × | × | ✓ | ✓ |
| Arch-114 | AO4 | MEO-2xSC-1,2xSC-6 | | 0 | 1234 | 0 | 396 | ✓ | ✓ | ✓ | - |
| Arch-115 | AO4 | MEO-2xSC-1,2xSC-7 | | 0 | 123344 | 0 | 399 | × | × | ✓ | ✓ |
| Arch-116 | AO4 | MEO-2xSC-1,2xSC-7 | | 0 | 123344 | 0 | 399 | ✓ | ✓ | ✓ | - |
| Arch-117 | AO5 | SC-10,2xSC-6 | | 1123 | 0 | 300 | 0 | × | × | ✓ | ✓ |
| Arch-118 | AO5 | SC-10,2xSC-6 | | 1123 | 0 | 300 | 0 | ✓ | ✓ | ✓ | - |
| Arch-119 | AO5 | SC-10,3xSC-6 | | 11234 | 0 | 302 | 0 | × | × | ✓ | ✓ |
| Arch-120 | AO5 | SC-10,3xSC-6 | | 11234 | 0 | 302 | 0 | ✓ | ✓ | ✓ | - |
| Arch-121 | AO5 | SC-10,2xSC-7 | | 112233 | 0 | 303 | 0 | × | × | ✓ | ✓ |
| Arch-122 | AO5 | SC-10,2xSC-7 | | 112233 | 0 | 303 | 0 | ✓ | ✓ | ✓ | - |
| Arch-123 | AO5 | SC-11,2xSC-6 | | 11123 | 0 | 316 | 0 | × | × | ✓ | ✓ |
| Arch-124 | AO5 | SC-11,2xSC-6 | | 11123 | 0 | 316 | 0 | ✓ | ✓ | ✓ | - |
| Arch-125 | AO5 | SC-11,3xSC-6 | | 111234 | 0 | 318 | 0 | × | × | ✓ | ✓ |
| Arch-126 | AO5 | SC-11,3xSC-6 | | 111234 | 0 | 318 | 0 | ✓ | ✓ | ✓ | - |
| Arch-127 | AO5 | SC-11,2xSC-7 | | 1112233 | 0 | 319 | 0 | × | × | ✓ | ✓ |
| Arch-128 | AO5 | SC-11,2xSC-7 | | 1112233 | 0 | 319 | 0 | ✓ | ✓ | ✓ | - |
| Arch-129 | AO5 | SC-12,2xSC-4 | | 1231 | 0 | 312 | 0 | × | × | ✓ | ✓ |
| Arch-130 | AO5 | SC-12,2xSC-4 | | 1231 | 0 | 312 | 0 | ✓ | ✓ | ✓ | - |
| Arch-131 | AO5 | SC-13,2xSC-4 | | 12311 | 0 | 316 | 0 | × | × | ✓ | ✓ |
| Arch-132 | AO5 | SC-13,2xSC-4 | | 12311 | 0 | 316 | 0 | ✓ | ✓ | ✓ | - |
| Arch-133 | AO5 | SC-14,2xSC-4 | | 123111 | 0 | 318 | 0 | × | × | ✓ | ✓ |
| Arch-134 | AO5 | SC-14,2xSC-4 | | 123111 | 0 | 318 | 0 | ✓ | ✓ | ✓ | - |
| Arch-135 | AO5 | SC-15,2xSC-4 | | 1231111 | 0 | 319 | 0 | × | × | ✓ | ✓ |
| Arch-136 | AO5 | SC-15,2xSC-4 | | 1231111 | 0 | 319 | 0 | ✓ | ✓ | ✓ | - |
| Arch-137 | AO6 | SC-16 | | 111 | 0 | 296 | 0 | × | × | ✓ | ✓ |
| Arch-138 | AO6 | SC-16 | | 111 | 0 | 296 | 0 | ✓ | ✓ | ✓ | - |
| Arch-139 | AO6 | SC-17 | | 1111 | 0 | 300 | 0 | × | × | ✓ | ✓ |
| Arch-140 | AO6 | SC-17 | | 1111 | 0 | 300 | 0 | ✓ | ✓ | ✓ | - |
| Arch-141 | AO6 | SC-18 | | 1111 | 0 | 312 | 0 | × | × | ✓ | ✓ |
| Arch-142 | AO6 | SC-18 | | 1111 | 0 | 312 | 0 | ✓ | ✓ | ✓ | - |
| Arch-143 | AO6 | SC-19 | | 11111 | 0 | 316 | 0 | × | × | ✓ | ✓ |
| Arch-144 | AO6 | SC-19 | | 11111 | 0 | 316 | 0 | ✓ | ✓ | ✓ | - |
| Arch-145 | AO7 | SC-1,SC-4 | 2xSC-6 | 12 | 12 | 288 | 12 | × | × | ✓ | ✓ |
| Arch-146 | AO7 | SC-1,SC-4 | 2xSC-6 | 12 | 12 | 288 | 12 | ✓ | ✓ | ✓ | - |
| Arch-147 | AO7 | SC-1,SC-4 | 3xSC-6 | 12 | 123 | 288 | 14 | × | × | ✓ | ✓ |
| Arch-148 | AO7 | SC-1,SC-4 | 3xSC-6 | 12 | 123 | 288 | 14 | ✓ | ✓ | ✓ | - |
| Arch-149 | AO7 | SC-1,SC-4 | 2xSC-7 | 12 | 1122 | 288 | 15 | × | × | ✓ | ✓ |
| Arch-150 | AO7 | SC-1,SC-4 | 2xSC-7 | 12 | 1122 | 288 | 15 | ✓ | ✓ | ✓ | - |
| Arch-151 | AO7 | SC-1,SC-4 | 4xSC-6 | 12 | 1234 | 288 | 15 | × | × | ✓ | ✓ |
| Arch-152 | AO7 | SC-1,SC-4 | 4xSC-6 | 12 | 1234 | 288 | 15 | ✓ | ✓ | ✓ | - |
| Arch-153 | AO7 | SC-1,SC-5 | 2xSC-6 | 122 | 12 | 304 | 12 | × | × | ✓ | ✓ |
| Arch-154 | AO7 | SC-1,SC-5 | 2xSC-6 | 122 | 12 | 304 | 12 | ✓ | ✓ | ✓ | - |
| Arch-155 | AO7 | SC-1,SC-5 | 3xSC-6 | 122 | 123 | 304 | 14 | × | × | ✓ | ✓ |
| Arch-156 | AO7 | SC-1,SC-5 | 3xSC-6 | 122 | 123 | 304 | 14 | ✓ | ✓ | ✓ | - |
| Arch-157 | AO7 | SC-1,SC-5 | 2xSC-7 | 122 | 1122 | 304 | 15 | × | × | ✓ | ✓ |
| Arch-158 | AO7 | SC-1,SC-5 | 2xSC-7 | 122 | 1122 | 304 | 15 | ✓ | ✓ | ✓ | - |
| Arch-159 | AO7 | SC-1,SC-5 | 4xSC-6 | 122 | 1234 | 304 | 15 | × | × | ✓ | ✓ |
| Arch-160 | AO7 | SC-1,SC-5 | 4xSC-6 | 122 | 1234 | 304 | 15 | ✓ | ✓ | ✓ | - |

| Name | AO | GEO Spacecraft | MEO Spacecraft | Antenna Assignment | | Antenna Allocation | | ISL | | Ground Stations | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | GEO | MEO | GEO | MEO | GEO | MEO | White-Sands | Guam |
| Arch-161 | AO7 | SC-1,2xSC-4 | 2xSC-6 | 123 | 12 | 304 | 12 | × | × | ✓ | ✓ |
| Arch-162 | AO7 | SC-1,2xSC-4 | 2xSC-6 | 123 | 12 | 304 | 12 | ✓ | ✓ | ✓ | - |
| Arch-163 | AO7 | SC-1,2xSC-4 | 3xSC-6 | 123 | 123 | 304 | 14 | × | × | ✓ | ✓ |
| Arch-164 | AO7 | SC-1,2xSC-4 | 3xSC-6 | 123 | 123 | 304 | 14 | ✓ | ✓ | ✓ | - |
| Arch-165 | AO7 | SC-1,2xSC-4 | 2xSC-7 | 123 | 1122 | 304 | 15 | × | × | ✓ | ✓ |
| Arch-166 | AO7 | SC-1,2xSC-4 | 2xSC-7 | 123 | 1122 | 304 | 15 | ✓ | ✓ | ✓ | - |
| Arch-167 | AO7 | SC-1,2xSC-4 | 4xSC-6 | 123 | 1234 | 304 | 15 | × | × | ✓ | ✓ |
| Arch-168 | AO7 | SC-1,2xSC-4 | 4xSC-6 | 123 | 1234 | 304 | 15 | ✓ | ✓ | ✓ | - |
| Arch-169 | AO7 | SC-2,SC-4 | 2xSC-6 | 112 | 12 | 416 | 12 | × | × | ✓ | ✓ |
| Arch-170 | AO7 | SC-2,SC-4 | 2xSC-6 | 112 | 12 | 416 | 12 | ✓ | ✓ | ✓ | - |
| Arch-171 | AO7 | SC-2,SC-4 | 3xSC-6 | 112 | 123 | 416 | 14 | × | × | ✓ | ✓ |
| Arch-172 | AO7 | SC-2,SC-4 | 3xSC-6 | 112 | 123 | 416 | 14 | ✓ | ✓ | ✓ | - |
| Arch-173 | AO7 | SC-2,SC-4 | 2xSC-7 | 112 | 1122 | 416 | 15 | × | × | ✓ | ✓ |
| Arch-174 | AO7 | SC-2,SC-4 | 2xSC-7 | 112 | 1122 | 416 | 15 | ✓ | ✓ | ✓ | - |
| Arch-175 | AO7 | SC-2,SC-4 | 4xSC-6 | 112 | 1234 | 416 | 15 | × | × | ✓ | ✓ |
| Arch-176 | AO7 | SC-2,SC-4 | 4xSC-6 | 112 | 1234 | 416 | 15 | ✓ | ✓ | ✓ | - |
| Arch-177 | AO7 | SC-2,SC-5 | 2xSC-6 | 1122 | 12 | 432 | 12 | × | × | ✓ | ✓ |
| Arch-178 | AO7 | SC-2,SC-5 | 2xSC-6 | 1122 | 12 | 432 | 12 | ✓ | ✓ | ✓ | - |
| Arch-179 | AO7 | SC-2,SC-5 | 3xSC-6 | 1122 | 123 | 432 | 14 | × | × | ✓ | ✓ |
| Arch-180 | AO7 | SC-2,SC-5 | 3xSC-6 | 1122 | 123 | 432 | 14 | ✓ | ✓ | ✓ | - |
| Arch-181 | AO7 | SC-2,SC-5 | 2xSC-7 | 1122 | 1122 | 432 | 15 | × | × | ✓ | ✓ |
| Arch-182 | AO7 | SC-2,SC-5 | 2xSC-7 | 1122 | 1122 | 432 | 15 | ✓ | ✓ | ✓ | - |
| Arch-183 | AO7 | SC-2,SC-5 | 4xSC-6 | 1122 | 1234 | 432 | 15 | × | × | ✓ | ✓ |
| Arch-184 | AO7 | SC-2,SC-5 | 4xSC-6 | 1122 | 1234 | 432 | 15 | ✓ | ✓ | ✓ | - |
| Arch-185 | AO7 | SC-2,2xSC-4 | 2xSC-6 | 1123 | 12 | 432 | 12 | × | × | ✓ | ✓ |
| Arch-186 | AO7 | SC-2,2xSC-4 | 2xSC-6 | 1123 | 12 | 432 | 12 | ✓ | ✓ | ✓ | - |
| Arch-187 | AO7 | SC-2,2xSC-4 | 3xSC-6 | 1123 | 123 | 432 | 14 | × | × | ✓ | ✓ |
| Arch-188 | AO7 | SC-2,2xSC-4 | 3xSC-6 | 1123 | 123 | 432 | 14 | ✓ | ✓ | ✓ | - |
| Arch-189 | AO7 | SC-2,2xSC-4 | 2xSC-7 | 1123 | 1122 | 432 | 15 | × | × | ✓ | ✓ |
| Arch-190 | AO7 | SC-2,2xSC-4 | 2xSC-7 | 1123 | 1122 | 432 | 15 | ✓ | ✓ | ✓ | - |
| Arch-191 | AO7 | SC-2,2xSC-4 | 4xSC-6 | 1123 | 1234 | 432 | 15 | × | × | ✓ | ✓ |
| Arch-192 | AO7 | SC-2,2xSC-4 | 4xSC-6 | 1123 | 1234 | 432 | 15 | ✓ | ✓ | ✓ | - |
| Arch-193 | AO7 | SC-3,SC-4 | 2xSC-6 | 1112 | 12 | 480 | 12 | × | × | ✓ | ✓ |
| Arch-194 | AO7 | SC-3,SC-4 | 2xSC-6 | 1112 | 12 | 480 | 12 | ✓ | ✓ | ✓ | - |
| Arch-195 | AO7 | SC-3,SC-4 | 3xSC-6 | 1112 | 123 | 480 | 14 | × | × | ✓ | ✓ |
| Arch-196 | AO7 | SC-3,SC-4 | 3xSC-6 | 1112 | 123 | 480 | 14 | ✓ | ✓ | ✓ | - |
| Arch-197 | AO7 | SC-3,SC-4 | 2xSC-7 | 1112 | 1122 | 480 | 15 | × | × | ✓ | ✓ |
| Arch-198 | AO7 | SC-3,SC-4 | 2xSC-7 | 1112 | 1122 | 480 | 15 | ✓ | ✓ | ✓ | - |
| Arch-199 | AO7 | SC-3,SC-4 | 4xSC-6 | 1112 | 1234 | 480 | 15 | × | × | ✓ | ✓ |
| Arch-200 | AO7 | SC-3,SC-4 | 4xSC-6 | 1112 | 1234 | 480 | 15 | ✓ | ✓ | ✓ | - |
| Arch-201 | AO7 | SC-3,SC-5 | 2xSC-6 | 11122 | 12 | 496 | 12 | × | × | ✓ | ✓ |
| Arch-202 | AO7 | SC-3,SC-5 | 2xSC-6 | 11122 | 12 | 496 | 12 | ✓ | ✓ | ✓ | - |
| Arch-203 | AO7 | SC-3,SC-5 | 3xSC-6 | 11122 | 123 | 496 | 14 | × | × | ✓ | ✓ |
| Arch-204 | AO7 | SC-3,SC-5 | 3xSC-6 | 11122 | 123 | 496 | 14 | ✓ | ✓ | ✓ | - |
| Arch-205 | AO7 | SC-3,SC-5 | 2xSC-7 | 11122 | 1122 | 496 | 15 | × | × | ✓ | ✓ |
| Arch-206 | AO7 | SC-3,SC-5 | 2xSC-7 | 11122 | 1122 | 496 | 15 | ✓ | ✓ | ✓ | - |
| Arch-207 | AO7 | SC-3,SC-5 | 4xSC-6 | 11122 | 1234 | 496 | 15 | × | × | ✓ | ✓ |
| Arch-208 | AO7 | SC-3,SC-5 | 4xSC-6 | 11122 | 1234 | 496 | 15 | ✓ | ✓ | ✓ | - |
| Arch-209 | AO7 | SC-3,2xSC-4 | 2xSC-6 | 11123 | 12 | 496 | 12 | × | × | ✓ | ✓ |
| Arch-210 | AO7 | SC-3,2xSC-4 | 2xSC-6 | 11123 | 12 | 496 | 12 | ✓ | ✓ | ✓ | - |
| Arch-211 | AO7 | SC-3,2xSC-4 | 3xSC-6 | 11123 | 123 | 496 | 14 | × | × | ✓ | ✓ |
| Arch-212 | AO7 | SC-3,2xSC-4 | 3xSC-6 | 11123 | 123 | 496 | 14 | ✓ | ✓ | ✓ | - |
| Arch-213 | AO7 | SC-3,2xSC-4 | 2xSC-7 | 11123 | 1122 | 496 | 15 | × | × | ✓ | ✓ |
| Arch-214 | AO7 | SC-3,2xSC-4 | 2xSC-7 | 11123 | 1122 | 496 | 15 | ✓ | ✓ | ✓ | - |
| Arch-215 | AO7 | SC-3,2xSC-4 | 4xSC-6 | 11123 | 1234 | 496 | 15 | × | × | ✓ | ✓ |
| Arch-216 | AO7 | SC-3,2xSC-4 | 4xSC-6 | 11123 | 1234 | 496 | 15 | ✓ | ✓ | ✓ | - |
| Arch-217 | AO7 | 2xSC-1,2xSC-4 | 2xSC-6 | 1234 | 12 | 432 | 12 | × | × | ✓ | ✓ |
| Arch-218 | AO7 | 2xSC-1,2xSC-4 | 2xSC-6 | 1234 | 12 | 432 | 12 | ✓ | ✓ | ✓ | - |
| Arch-219 | AO7 | 2xSC-1,2xSC-4 | 3xSC-6 | 1234 | 123 | 432 | 14 | × | × | ✓ | ✓ |
| Arch-220 | AO7 | 2xSC-1,2xSC-4 | 3xSC-6 | 1234 | 123 | 432 | 14 | ✓ | ✓ | ✓ | - |
| Arch-221 | AO7 | 2xSC-1,2xSC-4 | 2xSC-7 | 1234 | 1122 | 432 | 15 | × | × | ✓ | ✓ |
| Arch-222 | AO7 | 2xSC-1,2xSC-4 | 2xSC-7 | 1234 | 1122 | 432 | 15 | ✓ | ✓ | ✓ | - |
| Arch-223 | AO7 | 2xSC-1,2xSC-4 | 4xSC-6 | 1234 | 1234 | 432 | 15 | × | × | ✓ | ✓ |
| Arch-224 | AO7 | 2xSC-1,2xSC-4 | 4xSC-6 | 1234 | 1234 | 432 | 15 | ✓ | ✓ | ✓ | - |
| Arch-225 | AO8 | SC-10 | SC-6 | 11 | 1 | 288 | 8 | × | × | ✓ | ✓ |
| Arch-226 | AO8 | SC-10 | SC-6 | 11 | 1 | 288 | 8 | ✓ | ✓ | ✓ | - |
| Arch-227 | AO8 | SC-10 | SC-7 | 11 | 11 | 288 | 12 | × | × | ✓ | ✓ |
| Arch-228 | AO8 | SC-10 | SC-7 | 11 | 11 | 288 | 12 | ✓ | ✓ | ✓ | - |
| Arch-229 | AO8 | SC-10 | SC-8 | 11 | 111 | 288 | 14 | × | × | ✓ | ✓ |
| Arch-230 | AO8 | SC-10 | SC-8 | 11 | 111 | 288 | 14 | ✓ | ✓ | ✓ | - |
| Arch-231 | AO8 | SC-10 | SC-9 | 11 | 1111 | 288 | 15 | × | × | ✓ | ✓ |
| Arch-232 | AO8 | SC-10 | SC-9 | 11 | 1111 | 288 | 15 | ✓ | ✓ | ✓ | - |
| Arch-233 | AO8 | SC-11 | SC-6 | 111 | 1 | 304 | 8 | × | × | ✓ | ✓ |
| Arch-234 | AO8 | SC-11 | SC-6 | 111 | 1 | 304 | 8 | ✓ | ✓ | ✓ | - |
| Arch-235 | AO8 | SC-11 | SC-7 | 111 | 11 | 304 | 12 | × | × | ✓ | ✓ |
| Arch-236 | AO8 | SC-11 | SC-7 | 111 | 11 | 304 | 12 | ✓ | ✓ | ✓ | - |
| Arch-237 | AO8 | SC-11 | SC-8 | 111 | 111 | 304 | 14 | × | × | ✓ | ✓ |
| Arch-238 | AO8 | SC-11 | SC-8 | 111 | 111 | 304 | 14 | ✓ | ✓ | ✓ | - |
| Arch-239 | AO8 | SC-11 | SC-9 | 111 | 1111 | 304 | 15 | × | × | ✓ | ✓ |
| Arch-240 | AO8 | SC-11 | SC-9 | 111 | 1111 | 304 | 15 | ✓ | ✓ | ✓ | - |

| Name | AO | GEO Spacecraft | MEO Spacecraft | Antenna Assignment | | Antenna Allocation | | ISL | | Ground Stations | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | GEO | MEO | GEO | MEO | GEO | MEO | White-Sands | Guam |
| Arch-241 | AO9 | SC-1,SC-6 | 2xSC-4 | 12 | 12 | 264 | 48 | × | × | ✓ | ✓ |
| Arch-242 | AO9 | SC-1,SC-6 | 2xSC-4 | 12 | 12 | 264 | 48 | ✓ | ✓ | ✓ | - |
| Arch-243 | AO9 | SC-1,SC-7 | 2xSC-4 | 122 | 12 | 268 | 48 | × | × | ✓ | ✓ |
| Arch-244 | AO9 | SC-1,SC-7 | 2xSC-4 | 122 | 12 | 268 | 48 | ✓ | ✓ | ✓ | - |
| Arch-245 | AO9 | SC-1,SC-8 | 2xSC-4 | 1222 | 12 | 270 | 48 | × | × | ✓ | ✓ |
| Arch-246 | AO9 | SC-1,SC-8 | 2xSC-4 | 1222 | 12 | 270 | 48 | ✓ | ✓ | ✓ | - |
| Arch-247 | AO9 | SC-1,SC-9 | 2xSC-4 | 12222 | 12 | 271 | 48 | × | × | ✓ | ✓ |
| Arch-248 | AO9 | SC-1,SC-9 | 2xSC-4 | 12222 | 12 | 271 | 48 | ✓ | ✓ | ✓ | - |
| Arch-249 | AO9 | SC-1,2xSC-6 | 2xSC-4 | 123 | 12 | 268 | 48 | × | × | ✓ | ✓ |
| Arch-250 | AO9 | SC-1,2xSC-6 | 2xSC-4 | 123 | 12 | 268 | 48 | ✓ | ✓ | ✓ | - |
| Arch-251 | AO9 | SC-1,3xSC-6 | 2xSC-4 | 1234 | 12 | 270 | 48 | × | × | ✓ | ✓ |
| Arch-252 | AO9 | SC-1,3xSC-6 | 2xSC-4 | 1234 | 12 | 270 | 48 | ✓ | ✓ | ✓ | - |
| Arch-253 | AO9 | SC-1,2xSC-7 | 2xSC-4 | 12233 | 12 | 271 | 48 | × | × | ✓ | ✓ |
| Arch-254 | AO9 | SC-1,2xSC-7 | 2xSC-4 | 12233 | 12 | 271 | 48 | ✓ | ✓ | ✓ | - |
| Arch-255 | AO9 | SC-2,SC-6 | 2xSC-4 | 112 | 12 | 392 | 48 | × | × | ✓ | ✓ |
| Arch-256 | AO9 | SC-2,SC-6 | 2xSC-4 | 112 | 12 | 392 | 48 | ✓ | ✓ | ✓ | - |
| Arch-257 | AO9 | SC-2,SC-7 | 2xSC-4 | 1122 | 12 | 396 | 48 | × | × | ✓ | ✓ |
| Arch-258 | AO9 | SC-2,SC-7 | 2xSC-4 | 1122 | 12 | 396 | 48 | ✓ | ✓ | ✓ | - |
| Arch-259 | AO9 | SC-2,SC-8 | 2xSC-4 | 11222 | 12 | 398 | 48 | × | × | ✓ | ✓ |
| Arch-260 | AO9 | SC-2,SC-8 | 2xSC-4 | 11222 | 12 | 398 | 48 | ✓ | ✓ | ✓ | - |
| Arch-261 | AO9 | SC-2,SC-9 | 2xSC-4 | 112222 | 12 | 399 | 48 | × | × | ✓ | ✓ |
| Arch-262 | AO9 | SC-2,SC-9 | 2xSC-4 | 112222 | 12 | 399 | 48 | ✓ | ✓ | ✓ | - |
| Arch-263 | AO9 | SC-2,2xSC-6 | 2xSC-4 | 1123 | 12 | 396 | 48 | × | × | ✓ | ✓ |
| Arch-264 | AO9 | SC-2,2xSC-6 | 2xSC-4 | 1123 | 12 | 396 | 48 | ✓ | ✓ | ✓ | - |
| Arch-265 | AO9 | SC-2,3xSC-6 | 2xSC-4 | 11234 | 12 | 398 | 48 | × | × | ✓ | ✓ |
| Arch-266 | AO9 | SC-2,3xSC-6 | 2xSC-4 | 11234 | 12 | 398 | 48 | ✓ | ✓ | ✓ | - |
| Arch-267 | AO9 | SC-2,2xSC-7 | 2xSC-4 | 112233 | 12 | 399 | 48 | × | × | ✓ | ✓ |
| Arch-268 | AO9 | SC-2,2xSC-7 | 2xSC-4 | 112233 | 12 | 399 | 48 | ✓ | ✓ | ✓ | - |
| Arch-269 | AO9 | SC-3,SC-6 | 2xSC-4 | 1112 | 12 | 456 | 48 | × | × | ✓ | ✓ |
| Arch-270 | AO9 | SC-3,SC-6 | 2xSC-4 | 1112 | 12 | 456 | 48 | ✓ | ✓ | ✓ | - |
| Arch-271 | AO9 | SC-3,SC-7 | 2xSC-4 | 11122 | 12 | 460 | 48 | × | × | ✓ | ✓ |
| Arch-272 | AO9 | SC-3,SC-7 | 2xSC-4 | 11122 | 12 | 460 | 48 | ✓ | ✓ | ✓ | - |
| Arch-273 | AO9 | SC-3,SC-8 | 2xSC-4 | 111222 | 12 | 462 | 48 | × | × | ✓ | ✓ |
| Arch-274 | AO9 | SC-3,SC-8 | 2xSC-4 | 111222 | 12 | 462 | 48 | ✓ | ✓ | ✓ | - |
| Arch-275 | AO9 | SC-3,SC-9 | 2xSC-4 | 1112222 | 12 | 463 | 48 | × | × | ✓ | ✓ |
| Arch-276 | AO9 | SC-3,SC-9 | 2xSC-4 | 1112222 | 12 | 463 | 48 | ✓ | ✓ | ✓ | - |
| Arch-277 | AO9 | SC-3,2xSC-6 | 2xSC-4 | 11123 | 12 | 460 | 48 | × | × | ✓ | ✓ |
| Arch-278 | AO9 | SC-3,2xSC-6 | 2xSC-4 | 11123 | 12 | 460 | 48 | ✓ | ✓ | ✓ | - |
| Arch-279 | AO9 | SC-3,3xSC-6 | 2xSC-4 | 111234 | 12 | 462 | 48 | × | × | ✓ | ✓ |
| Arch-280 | AO9 | SC-3,3xSC-6 | 2xSC-4 | 111234 | 12 | 462 | 48 | ✓ | ✓ | ✓ | - |
| Arch-281 | AO9 | SC-3,2xSC-7 | 2xSC-4 | 1112233 | 12 | 463 | 48 | × | × | ✓ | ✓ |
| Arch-282 | AO9 | SC-3,2xSC-7 | 2xSC-4 | 1112233 | 12 | 463 | 48 | ✓ | ✓ | ✓ | - |
| Arch-283 | AO9 | 2xSC-1,2xSC-6 | 2xSC-4 | 1234 | 12 | 396 | 48 | × | × | ✓ | ✓ |
| Arch-284 | AO9 | 2xSC-1,2xSC-6 | 2xSC-4 | 1234 | 12 | 396 | 48 | ✓ | ✓ | ✓ | - |
| Arch-285 | AO9 | 2xSC-1,2xSC-7 | 2xSC-4 | 123344 | 12 | 399 | 48 | × | × | ✓ | ✓ |
| Arch-286 | AO9 | 2xSC-1,2xSC-7 | 2xSC-4 | 123344 | 12 | 399 | 48 | ✓ | ✓ | ✓ | - |
| Arch-287 | AO10 | SC-12 | SC-4 | 11 | 1 | 264 | 32 | × | × | ✓ | ✓ |
| Arch-288 | AO10 | SC-12 | SC-4 | 11 | 1 | 264 | 32 | ✓ | ✓ | ✓ | - |
| Arch-289 | AO10 | SC-12 | SC-5 | 11 | 11 | 264 | 48 | × | × | ✓ | ✓ |
| Arch-290 | AO10 | SC-12 | SC-5 | 11 | 11 | 264 | 48 | ✓ | ✓ | ✓ | - |
| Arch-291 | AO10 | SC-13 | SC-4 | 111 | 1 | 268 | 32 | × | × | ✓ | ✓ |
| Arch-292 | AO10 | SC-13 | SC-4 | 111 | 1 | 268 | 32 | ✓ | ✓ | ✓ | - |
| Arch-293 | AO10 | SC-13 | SC-5 | 111 | 11 | 268 | 48 | × | × | ✓ | ✓ |
| Arch-294 | AO10 | SC-13 | SC-5 | 111 | 11 | 268 | 48 | ✓ | ✓ | ✓ | - |
| Arch-295 | AO10 | SC-14 | SC-4 | 1111 | 1 | 270 | 32 | × | × | ✓ | ✓ |
| Arch-296 | AO10 | SC-14 | SC-4 | 1111 | 1 | 270 | 32 | ✓ | ✓ | ✓ | - |
| Arch-297 | AO10 | SC-14 | SC-5 | 1111 | 11 | 270 | 48 | × | × | ✓ | ✓ |
| Arch-298 | AO10 | SC-14 | SC-5 | 1111 | 11 | 270 | 48 | ✓ | ✓ | ✓ | - |
| Arch-299 | AO10 | SC-15 | SC-4 | 11111 | 1 | 271 | 32 | × | × | ✓ | ✓ |
| Arch-300 | AO10 | SC-15 | SC-4 | 11111 | 1 | 271 | 32 | ✓ | ✓ | ✓ | - |
| Arch-301 | AO10 | SC-15 | SC-5 | 11111 | 11 | 271 | 48 | × | × | ✓ | ✓ |
| Arch-302 | AO10 | SC-15 | SC-5 | 11111 | 11 | 271 | 48 | ✓ | ✓ | ✓ | - |
| Arch-303 | AO11 | 2xSC-1 | SC-6,2xSC-4 | 12 | 123 | 384 | 56 | × | × | ✓ | ✓ |
| Arch-304 | AO11 | 2xSC-1 | SC-6,2xSC-4 | 12 | 123 | 384 | 56 | ✓ | ✓ | ✓ | - |
| Arch-305 | AO11 | 2xSC-1 | SC-7,2xSC-4 | 12 | 1233 | 384 | 60 | × | × | ✓ | ✓ |
| Arch-306 | AO11 | 2xSC-1 | SC-7,2xSC-4 | 12 | 1233 | 384 | 60 | ✓ | ✓ | ✓ | - |
| Arch-307 | AO11 | 2xSC-1 | SC-8,2xSC-4 | 12 | 12333 | 384 | 62 | × | × | ✓ | ✓ |
| Arch-308 | AO11 | 2xSC-1 | SC-8,2xSC-4 | 12 | 12333 | 384 | 62 | ✓ | ✓ | ✓ | - |
| Arch-309 | AO11 | 2xSC-1 | SC-9,2xSC-4 | 12 | 123333 | 384 | 63 | × | × | ✓ | ✓ |
| Arch-310 | AO11 | 2xSC-1 | SC-9,2xSC-4 | 12 | 123333 | 384 | 63 | ✓ | ✓ | ✓ | - |
| Arch-311 | AO11 | 3xSC-1 | SC-6,2xSC-4 | 123 | 123 | 448 | 56 | × | × | ✓ | ✓ |
| Arch-312 | AO11 | 3xSC-1 | SC-6,2xSC-4 | 123 | 123 | 448 | 56 | ✓ | ✓ | ✓ | - |
| Arch-313 | AO11 | 3xSC-1 | SC-7,2xSC-4 | 123 | 1233 | 448 | 60 | × | × | ✓ | ✓ |
| Arch-314 | AO11 | 3xSC-1 | SC-7,2xSC-4 | 123 | 1233 | 448 | 60 | ✓ | ✓ | ✓ | - |
| Arch-315 | AO11 | 3xSC-1 | SC-8,2xSC-4 | 123 | 12333 | 448 | 62 | × | × | ✓ | ✓ |
| Arch-316 | AO11 | 3xSC-1 | SC-8,2xSC-4 | 123 | 12333 | 448 | 62 | ✓ | ✓ | ✓ | - |
| Arch-317 | AO11 | 3xSC-1 | SC-9,2xSC-4 | 123 | 123333 | 448 | 63 | × | × | ✓ | ✓ |
| Arch-318 | AO11 | 3xSC-1 | SC-9,2xSC-4 | 123 | 123333 | 448 | 63 | ✓ | ✓ | ✓ | - |
| Arch-319 | AO12 | SC-1 | | 1 | 0 | 256 | 0 | × | × | ✓ | ✓ |
| Arch-320 | AO12 | SC-1 | | 1 | 0 | 256 | 0 | ✓ | ✓ | ✓ | - |

| Name | AO | GEO Spacecraft | MEO Spacecraft | Antenna Assignment | | Antenna Allocation | | ISL | | Ground Stations | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | GEO | MEO | GEO | MEO | GEO | MEO | White-Sands | Guam |
| Arch-321 | AO12 | SC-2 | | 11 | 0 | 384 | 0 | × | × | ✓ | ✓ |
| Arch-322 | AO12 | SC-2 | | 11 | 0 | 384 | 0 | ✓ | ✓ | ✓ | - |
| Arch-323 | AO12 | SC-3 | | 111 | 0 | 448 | 0 | × | × | ✓ | ✓ |
| Arch-324 | AO12 | SC-3 | | 111 | 0 | 448 | 0 | ✓ | ✓ | ✓ | - |
| Arch-325 | AO13 | MEO-SC-16 | | 0 | 111 | 0 | 296 | × | × | ✓ | ✓ |
| Arch-326 | AO13 | MEO-SC-16 | | 0 | 111 | 0 | 296 | ✓ | ✓ | ✓ | - |
| Arch-327 | AO13 | MEO-SC-17 | | 0 | 1111 | 0 | 300 | × | × | ✓ | ✓ |
| Arch-328 | AO13 | MEO-SC-17 | | 0 | 1111 | 0 | 300 | ✓ | ✓ | ✓ | - |
| Arch-329 | AO13 | MEO-SC-18 | | 0 | 1111 | 0 | 312 | × | × | ✓ | ✓ |
| Arch-330 | AO13 | MEO-SC-18 | | 0 | 1111 | 0 | 312 | ✓ | ✓ | ✓ | - |
| Arch-331 | AO13 | MEO-SC-19 | | 0 | 11111 | 0 | 316 | × | × | ✓ | ✓ |
| Arch-332 | AO13 | MEO-SC-19 | | 0 | 11111 | 0 | 316 | ✓ | ✓ | ✓ | - |

# Bibliography


[1] M Adinolfi and A Cesta. Contributed paper heuristic scheduling of the drs communication system, 1995.

[2] Ram Bhusan Agrawal, Kalyanmoy Deb, and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. 1994.

[3] C Ahara and J Rossbach. The scheduling problem in satellite communications systems. *Communication Technology, IEEE Transactions on*, 15(3):364–371, 1967.

[4] Reza Akbari and Koorush Ziarati. A multilevel evolutionary algorithm for optimizing numerical functions. *International Journal of Industrial Engineering Computations*, 2(2):419–430, 2011.

[5] Clayton L Bridges and David E Goldberg. An analysis of reproduction and crossover in a binary-coded genetic algorithm. *Grefenstette*, 878:9–13, 1987.

[6] Bruce G Buchanan, Edward Hance Shortliffe, et al. *Rule-based expert systems*, volume 3. Addison-Wesley Reading, MA, 1984.

[7] Richard A Caruana, Larry J Eshelman, and J David Schaffer. Representation and hidden bias ii: Eliminating defining length bias in genetic search via shuffle crossover. In *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 1*, pages 750–755. Morgan Kaufmann Publishers Inc., 1989.

[8] Edward Grady Coffman and John L Bruno. *Computer and job-shop scheduling theory*. John Wiley & Sons, 1976.

[9] Space Communications and Navigation Program. Space communications and navigation (scan) network architecture definition document (add) volume 1: Executive summary. Technical report, NASA, Washington, DC, October 2011.

[10] E. F. Crawley. System arhitecture. Class Notes, 2004.

[11] E. F. Crawley, B. Cameron, and D. Selva. *System Architecture*. Pearson, 2015.

[12] Peter Davison. Tradespace exploration for space system architecture: A weighted graph framework. Master's thesis, Massachusetts Institute of Technology Department of Aeronautics and Astronautics, Cambridge, Massachusetts, June 2014.

[13] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[14] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Lecture notes in computer science*, 1917:849–858, 2000.

[15] Richard Duda, John Gaschnig, and Peter Hart. Model design in the prospector consultant system for mineral exploration. *Expert systems in the microelectronic age*, 1234:153–167, 1979.

[16] Bernard Edwards. Overview of nasa's laser communications relay demonstration. *NASA Goddard Space Flight Center*, 2012.

[17] LJ Eshelman and JD Schaffer. Real-coded genetic algorithms and intervalschemata, foundations of ga 2, 1993.

[18] Yan-Shen Fang and Ying-Wu Chen. Constraint programming model of tdrss single access link scheduling problem. In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 948–951. IEEE, 2006.

[19] Edward A Feigenbaum. The art of artificial intelligence. 1. themes and case studies of knowledge engineering. Technical report, DTIC Document, 1977.

[20] Charles L Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*, 19(1):17–37, 1982.

[21] E. Friedman-Hill. *JESS in Action*, volume 1930110893. Manning Greenwich, CT, 2003.

[22] Joseph C Giarratano and Gary Riley. *Expert systems: principles and programming*. Brooks/Cole Publishing Co., 1989.

[23] David E Goldberg. *Genetic algorithms*. Pearson Education India, 2006.

[24] Teofilo Gonzalez and Sartaj Sahni. Open shop scheduling to minimize finish time. *Journal of the ACM (JACM)*, 23(4):665–679, 1976.

[25] Jose Fernando Gonzalves, Jorge Jose de Magalhaes Mendes, and Mauricio GC Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.

[26] Crina Grosan and Ajith Abraham. *Intelligent systems: A modern approach*, volume 17. Springer, 2011.

[27] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–72, 1992.

[28] William Horne. Architecture use cases for space communications. Technical report, NASA, 2015.

[29] Lynwood A Johnson and Douglas C Montgomery. *Operations research in production planning, scheduling, and inventory control*, volume 7. Wiley New York, 1974.

[30] David Levine. *A parallel genetic algorithm for the set partitioning problem.* Springer, 1996.

[31] Wei-Cheng Lin, Da-Yin Liao, Chung-Yang Liu, and Yong-Yao Lee. Daily imaging scheduling of an earth observation satellite. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(2):213–223, 2005.

[32] Muhammad Nawaz, E Emory Enscore Jr, and Inyong Ham. A heuristic algorithm for the< i> m</i>-machine,< i> n</i>-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

[33] Marc Sanchez Net, Inigo del Portillo, Bruce Cameron, and Edward Crawley. Architecting space communication networks under mission demand uncertainty. pages 1–10, March 2015.

[34] Joseph C Pemberton and Flavius Galiber. A constraint-based approach to satellite scheduling. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 57:101–114, 2001.

[35] JD Rao, P Soma, and GS Padmashree. Multi-satellite scheduling system for leo satellite operations. *SpaceOps, Thokyo, Japan, 2b002*, 1998.

[36] M. Sanchez. Architecting space communications networks. Master's thesis, Massachusetts Institute of Technology Department of Aeronautics and Astronautics, Cambridge, MA, June 2014.

[37] M. Sanchez, I. del Portillo, D. Selva, B. Cameron, and E. Crawley. Itaca: Modeling and optimization of space network architectures. *AIAA Journal of Aerospace Information Systems*, pages 66–72, 2015 (under review).

[38] M. Sanchez, D. Selva, B. Cameron, E. Crawley, A. Seas, and B. Seery. Exploring the architectural trade space of nasas space communication and navigation program. In *Aerospace Conference, 2013 IEEE*, pages 1–16, Big Sky, MT, March 2013. IEEE.

[39] M. Sanchez, D. Selva, B. Cameron, E. Crawley, A. Seas, and B. Seery. Results of the mit space communication and navigation architecture study. In *2014 IEEE Aerospace Conference*, pages 1–14, Big Sky, MT, March 2014. IEEE.

[40] Marc Sanchez Net et al. Architectural model for evaluating space communication networks.

[41] D. Selva and E. F Crawley. Vassar: Value assessment of system architectures using rules. In *Aerospace Conference, 2013 IEEE*, pages 1–21, Big Sky, MT, 2013. IEEE.

[42] D. Selva Valero. *Rule-based system architecting of Earth observation satellite systems.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2012.

[43] Edward H Shortliffe, Randall Davis, Stanton G Axline, Bruce G Buchanan, C Cordell Green, and Stanley N Cohen. Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the mycin system. *Computers and biomedical research*, 8(4):303–320, 1975.

[44] W. L. Simmons. *A framework for decision support in systems architecting*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, February 2008.

[45] P Soma, S Venkateswarlu, S Santhalakshmi, Tapan Bagchi, and Sanjay Kumar. Multi-satellite scheduling using genetic algorithms. *ISTRAC/ISRO, SpaceOps*, 2004.

[46] William M Spears and Kenneth A De Jong. An analysis of multi-point crossover. Technical report, DTIC Document, 1990.

[47] T. A. Sutherland, B. G. Cameron, and E. F. Crawley. Program goals for the nasa/noaa earth observation program derived from a stakeholder value network analysis. *Space Policy*, 28(4):259 – 269, November 2012.

[48] Gilbert Syswerda. Uniform crossover in genetic algorithms. 1989.

[49] N Thang and Byung R Moon. A new genetic approach for the traveling salesman problem. 1994.

[50] Matthew Bartschi Wall. *A genetic algorithm for resource-constrained scheduling*. PhD thesis, Massachusetts Institute of Technology, 1996.