Anomaly Resilience, Detection, and Reaction in Astrodynamics Problems

by

Manuel Indaco

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 3, 2024

Keywords: Trajectory Design, Cislunar Space, Surrogate Modeling, P-LEO Satellite Constellation, Anomaly Detection, Transformer Neural Network, Adversarial Games, Competitive Coevolution, Genetic Programming

Approved by

Davide Guzzetti, Chair, Assistant Professor of Aerospace Engineering
Daniel R. Tauritz, Associate Professor of Computer Science and Software Engineering
Ehsan Taheri, Assistant Professor of Aerospace Engineering
Nan Li, Assistant Professor of Aerospace Engineering
Samuel Mulder, Associate Research Professor of Computer Science and Software Engineering

Abstract

This manuscript is divided into three parts, each associated with three of the main projects I undertook throughout my doctoral journey. In particular, this dissertation focuses on the topics of anomaly resilience, detection, and reaction in astrodynamics problems, which are crucial aspects to consider for ensuring mission success, especially when dealing with the harsh space environment. In this manuscript, resilience, detection and reaction are investigated in two selected domains: trajectory design in cislunar regime and satellite constellations, both susceptible to the possibility of unexpected events. Therefore, effective identification capabilities and swift reaction to unforeseen phenomena become vital to support mission integrity.

Resilience to off-nominal behaviors is investigated for the trajectory design problem in the first part of this manuscript. In particular, we explore the convergence and dynamical structure of the trajectory design space associated with lunar landing and ascent abort scenarios for a crewed module departing from and returning to the Deep Space Gateway. Numerical methods for the identification of abort trajectory solutions are employed within a two-step optimization pipeline, through which we discover the existence of regions of stiff convergence where traditional pipelines may fail. Hence, we present an extensive analysis of problem parameters, demonstrating how the presence of such regions can be traced to the formulation of employed correction algorithms, problem dynamics sensitivity, and transfers geometry. To reduce the computational cost, we introduce a three-step optimization pipeline relying on surrogate models trained via adaptive sampling for the fast generation of initial guesses, which are then corrected for the recovery of abort trajectories within the defined scenario. Results obtained from the application of the optimization pipeline on the two scenarios underscore the complexity of the solution space, while providing useful information to inform the trajectory design process.

Anomaly detection and reaction in the context of Proliferated Low Earth Orbit (P-LEO) satellite constellations are then explored in the remaining two parts of this manuscript, with a focus on anomalous behaviors originating from adversarial actions against a constellation. For the detection problem, we present a transformer neural network- based pipeline for the identification of anomalous connections between satellites, modeling a P-LEO as a dynamic graph, which is capable of capturing spatial-temporal correlations characterizing a temporal network. Our analyses demonstrate how temporal and spatial signals alone are insufficient for effectively discriminating anomalous connections, requiring additional features to enrich the information extracted from the network dynamics. Notably, we discover how the introduction of edge-frequency information positively impacts our algorithm performance, reaching up to 95% in AUC score, here used as quality metric. Additionally, extensive analyses on variations of problem parameters demonstrate the robustness of the method over a wide range of scenarios, and highlight the existence of interesting couplings between satellite dynamics, spatial ground node distribution, and algorithm performance.

For the reaction component, a combination of competitive coevolutionary algorithms and genetic programming is employed to evolve reactive strategies to respond to adversarial actions against a constellation system. In particular, genetic programming trees are employed for the representation of reactive policies to respond to presented and different, unseen scenarios. The analysis demonstrates how the utilized approach provides effective solutions, beating both minimally complex strategies and human-developed ones, and showing adaptability to the introduction of multiple constraints. In both the detection and reaction problems, the proposed methodologies display the potential to reduce the cognitive load on operators of large constellation systems, and to possibly enable resolution of critical situations in a timely manner.

Acknowledgments

If you asked me five years ago where I would have been after college, I would have probably replied with "working for some company somewhere around in Europe". Funny how the events unfolded. I came to Auburn in 2019 for my Master's thesis, as I had the desire to spend some time abroad, to prepare myself for living away from home and see if I could make it on my own. Never I would have thought that those six months would have turned into four more years. I remember the day I left Italy at the beginning of January 2021 (mostly due to the risk of not being able to get on the plane for having added "PHD" as a prefix to my Delta account, which resulted into a mismatch between my name on the passport and the one on the boarding pass...got a little carried away there), so excited to begin this new journey. "Three and a half years in the US, this is going to be amazing!". And I must say, it truly was. You know, I claimed this in my Master's thesis, and I still believe it...the acknowledgments are the toughest part to write. There are just so many memories, so many people I would like to thank between those from my home country, and those I met here, my second home. But I promise, I will keep this short (maybe).

First and foremost, I want to thank my advisor and mentor, Dr. Davide Guzzetti, for giving me this extraordinary opportunity. Since day one, I considered your guidance and encouragement as invaluable gifts, and I put all the effort I could to repay the trust you gave me, in every single day of work. And if today I can call myself a Researcher, I owe that to you. So, thank you, Professor.

Next, I would like to thank all the members of my committee: Dr. Ehsan Taheri, Dr. Nan Li, Dr. Daniel R. Tauritz, and Dr. Samuel Mulder, for all the precious suggestions and insightful conversations held throughout these years, working on different projects.

Mom, Dad, "Frer", the greatest of the acknowledgments goes to you, for all the love you have always demonstrated to me. And I am not referring just to these last years, but to my entire academic journey, since elementary school, through high school and college. Most of the merit for the man I am today is yours, and I promise I will continue to do my best to make you proud.

To my grandparents. Even though you are not here anymore, I know that you have never stopped watching over me, and for this I want to thank you. I wish you were still here, cheering with me this accomplishment of mine; but I guess you are, even though just in my mind.

To the rest of my family: my aunts and uncles, and my cousins. Thank you for all the love and support you have always demonstrated for me, and for always pushing me in pursuing my ambitions.

And now, the most difficult part: friends. Just like four years ago, I do not know where to start from, too worried of forgetting someone. The problem is that there is no easy way to start, as it may seem one is giving more importance to who comes first. But it is not like that. Every single friendship to me has its own meaning. So, I will cheat and play the chronological order card.

First, I would like to express my gratitude to Robert (my godson) and Ryo, the first people I met when I came to Auburn. Thank you for having been such amazing friends.

To Stefano, the first Italian I met here in Auburn (we are everywhere). I'm so glad you replied to this random guy who texted you "Ma sei italiano?" on a Zoom call for international students. Thank you for having been such a good friend, for all the conversations, for the support, and for all the runs (I'm sorry I can't make it to the New York marathon in November...maybe next year!).

To all my lab mates in Davis 153: Joe (I will let you continue the ranking of the best pastry at the grocery store, I trust your judgment), Kanak, Dhathri, Luke, Laith, Eirik, Nick, Praveen ("rook to C4, it will open something in the future"), Keziban, Jack and Francesco. Thanks for all the chats and laughs we had during these years. You are the best, and I really hope we'll meet again.

To Rehman (I know you thought "did he just leave me out?!"), my "homie". Thank you for the great friend you are, for all the conversations, for the Wednesday's "life&cannoli" (great podcast name, right?), and for the help you gave me throughout all these years, teaching me how to live in the US. This time we'll meet in Rome, Italy; perhaps next time it will be in Rome, New York!

Finally, I would like to thank a few other people I had a chance to spend many memorable moments (somehow most of them involving pool games): Michele, Mattia, Elzon, Guilherme, Stefano, Erik, Carly and Skye. It's been great meeting you all!

If there is one more thing I am grateful for, is the possibility that Auburn gave me to meet with so many amazing people from all around the world: Spain, Colombia, Mexico, Greece. The list goes on, and it would be hard to mention each individual person and country. But to all of you, even if it was just for some laughs or for a few nights out, thank you!

Being an international student, especially on the other side of the world, is not simple. Sure, moving to a foreign country is exciting, but we must deal with somewhat leaving behind our home. However, if like me you are lucky enough, the geographic distance is meaningless compared with true, long-lasting friendships. So, I would like to thank a few amazing people: Paul, "Man", Pampa, Turk, Sara, Luchino, Eli, Simo, Ire, Morgan, AB, "Picci", and my friends from college (Phil, Pier, and the whole Morpheus group). A special thank goes to the "bomber", Biagio, who has always found the time to chat with me when I needed to. A second and last special thank goes to Brando, with whom I started this journey in the US. Thank you for the great friend you have always been, for all the phone calls, and for all the laughs.
Lastly, I want to thank Giorgia. Thank you for having been my Sancho, and for having always supported me when I needed it the most.

As I conclude these acknowledgments, the last section I'm adding to this manuscript, I realize this also symbolically concludes this experience. But "part of the journey is the end", and I am eager to see what life is reserving for me as I embark on my next adventure. War Eagle!

Manuel

Conclusive Note. I would like to acknowledge the multiple sources of support for the research I conducted here at Auburn, including the NASA Grant 80NSSC20K1017, the Auburn Cyber Research Center (ACRC) (https://cyber.auburn.edu), Auburn University's Intramural Grant Program (IGP), and matching funds from the Samuel Ginn College of Engineering, the Department of Aerospace Engineering, and the Department of Computer Science and Software Engineering.

Table of Contents

List of Tables

List of Figures

“Many places I have been

Many sorrows I have seen

But I don’t regret

Nor will I forget

All who took that road with me”

William “Billy” Nathan Boyd, *The Last Goodbye*

Chapter 1

Introduction

1.1 Motivation

Anomaly resilience, detection, and reaction are essential components in the design and management of space missions due to the unpredictability of the harsh space environment. The ability to rapidly identify and react to abnormal behavior in space systems has several benefits, such as ensuring the correct functionality of a space asset, the reduction of overall risks and costs, and maximizing the probability of mission success. Unfortunately, there exist a variety of events that may endanger a space asset, including natural phenomena (solar flares, debris collision, magnetic storms, etc.) or sudden system failure. As such, identifying an anomaly and its causes, as well as quickly selecting countermeasures, become complex tasks. The ability to respond to unforeseen events is even more crucial in the context of manned missions, where guaranteeing the safety of astronauts is the primary objective.

Over the past years, multiple nations have renewed their interest in the Moon, announcing programs with the ambitious goal of returning mankind to the lunar surface within the next decade. One such program is the National Aeronautics and Space Administration's (NASA's) Artemis mission, which aims to establish a human outpost on the Moon. The lunar surface base will be a proving ground to test new technologies for future deep-space exploration and a field laboratory to conduct lunar science. However, different mission segments may be susceptible to multiple risks. One such risk is constituted by a potential anomaly during the landing or ascent phases, which necessitates the design of abort trajectories. Unfortunately, trajectory design in chaotic environments such as the cislunar one possesses many sources of uncertainty, requiring large, complex, time-demanding simulations that span over the entirety of the solution space. As in a time-constrained environment conducting such analyses may be unfeasible, novel approaches to reduce the computational complexity of the problem are required. In this work, surrogate modeling is explored as a potential solution to enable acceleration of large scale trajectory design simulations. Surrogate models are indeed fast and computationally-light to train, and may provide a good approximation (which can be refined in a second moment) in substitution to an exact, expensive-to-obtain solution. However, appropriate selection of the surrogate model requires a sense of the surface to be approximated. Hence, an extensive investigation of key problem modeling parameters is conducted, aiming to generate a cartography of the abort transfers landscape.

Detecting and reacting to anomalous events becomes even more challenging when the number of space assets to manage exponentially increases. One notable example is constituted by the most recent proliferated LEO constellations. These are systems composed of hundreds or thousands of satellites dedicated to providing ubiquitous services (telecommunication, Earth observation and navigation - to name a few), with military, commercial and scientific applications. However, the size of such infrastructures poses challenges for the constant monitoring of the space segment. The increasing number of units makes continuous monitoring of individual elements impractical for human operators. Moreover, intrinsic system dynamics introduce a temporal factor which may hinder effective detection of abnormal behaviors, especially when system elements are not monitored individually. Finally, the large variety of possible anomalies further complicates the decision-making process when evaluation of an event is needed. A particularly concerning aspect is then introduced by the possibility of adversarial actions against these large systems. In fact, such constellations being a critical component for modern society, they constitute valuable targets for cyber and physical attacks, which may disrupt, or even disable, the entire constellation.

Within the realm of anomaly detection on satellite systems, current approaches exploit a combination of human expertise and automatic anomaly identification through algorithms often designed for a specific mission. However, the variety of threats renders the task of identifying isolated events even more challenging; additionally, traditional methods do not consider the system as an aggregate, thus potentially missing anomalous behaviors extending to multiple units. To overcome these challenges, we propose the application of deep learning - specifically, employing a transformer neural network - for the detection of anomalies on P-LEO constellations. In particular, considering the problem of adversarial actions acting on satellite links, we tackle the challenge of anomalous edge detection. The large satellite system is modeled as a graph, which enables capturing network information both locally and globally. The hypothesis here is that the transformer network can extrapolate meaningful features from its input, thus becoming capable of understanding the underlying network dynamical behavior, hence recognizing anomalies more effectively.

Once the attack on the satellite constellation is identified, a prompt reaction is crucial to mitigate damage and restore nominal operations. In fact, P-LEO constellations provide services to support multiple industries, including finance, transportation and communication, and an attack that disrupts satellite services can result in a significant economic loss. To model this problem, an adversarial ground station transit time game is introduced and investigated as a proxy to study the ability to rapidly respond to satellite attacks. Competitive coevolution is employed to evolve two populations: one attacker, representative of a malevolent actor aiming to degrade a constellation service, and one defender, whose goal is to re-establish nominal operations. Genetic programming trees are exploited to encode reactive strategies for the defender, which is then capable of dynamically responding to present and newly faced situations.

1.2 Research Objectives

To tackle the anomaly resilience, detection, and reaction tasks, we employ different techniques depending on the specific case. Regarding the trajectory design domain, surrogate modeling is employed to accelerate the trajectory design process in the context of lunar descent and ascent scenarios, exploiting adaptive sampling for the rapid exploration of the design space. For what concerns the investigation related to the satellite constellation domain, we consider the problem divided into two segments, that are 1) detection and 2) reaction, with both segments sharing the assumption of an adversarial attack against a P-LEO system. For the detection segment, we propose the application of a neural network-based architecture as a more general methodology to identify anomalous connections on a satellite network represented as a dynamic graph. For the reaction segment, competitive coevolution and genetic programming are combined to evolve efficient strategies to counter an adversarial action against a satellite network.

Objective-1. Assessment of the convergence and dynamical structures of the Near Rectilinear Halo Orbit-to-Low Lunar Orbit abort trajectory design space. The purpose of this objective is to generate a cartography for the trajectory abort design space in the context of humans' lunar exploration. To this aim, two representative scenarios - descent and ascent - are introduced and investigated. For each case, the convergence structure is analyzed through the implementation of a two-step optimization pipeline. To reduce the computational cost, surrogate modeling in combination of adaptive sampling techniques is explored as a way to rapidly generate initial guesses to be corrected by traditional schemes for the generation of desired trajectory solutions. Further analyses are conducted to assess how the problem geometry influences the dynamics and the quality of the obtained trajectory solutions.

Objective-2. Establishment of a framework for detecting temporal-topological anomalies on satellite P-LEO constellation networks. The purpose of this objective is to define a framework for efficiently detecting anomalous connections over P-LEO networks, with the goal of creating an automatic tool that can aid ground operators to monitor large satellite systems. To this aim, the P-LEO constellation is first represented as a dynamic graph, which offers a flexible way to represent large dynamic networks, providing both spatial and temporal rich information. Next, graph features are extracted and processed by a transformer-encoder network. Graph and constellation parameters are investigated to determine their effect on network performance.

Objective-3. Establishment of a framework for the investigation of reactive strategies to counter threats against satellite P-LEO constellations. The goal of this objective is to characterize a framework for the exploration of effective reactive strategies to counter adversarial attacks against a satellite system. To achieve this objective, the threat-to-satellites problem is framed as an adversarial ground station transit time game, which is developed and characterized. The proposed formulation, while simplifying real-world scenarios, offers flexibility and constitutes a launch pad to investigate more complex dynamics. A solution is explored through employment of competitive coevolution to evolve two populations representing a hypothetical attacker aiming to disrupt communication services, and a defender whose goal is to re-establish nominal operations. In this regard, both competitive coevolution and genetic programming trees are employed to identify effective strategies for both the attacker and the defender as a solution to the problem.

1.3 Manuscript Organization

As mentioned in the previous section, three problems are investigated in this dissertation: 1) trajectory design in cislunar space, 2) anomaly detection in satellite P-LEO constellations, and 3) identification of reactive strategies within a P-LEO constellation adversarial setting. Due to the diverse nature of these challenges, each of these problems is discussed separately. Hence, the structure of this work is organized as follows:

- Chapter 2 is dedicated to the anomaly resilience challenge, and partially contains the findings of our work “Structure of the NRHO-to-LLO Abort Trajectory Design Space” [4]. At first, the problem and its relevance are discussed in the context of the Artemis program. Next, the dynamical modeling and algorithmic methodologies are presented. Finally, the descent and ascent scenarios are presented and key findings are thoroughly discussed.
- Chapter 3 discusses the anomaly detection problem and presents a modified version of our journal publication “Transformer-based anomaly detection in P-LEO constellations: A dynamic graph approach”[5]. At first, the problem is framed in the context of security of satellite constellations. Next, theoretical background on graphs and deep learning is provided, followed by a description of the employed methodology. Finally, multiple scenarios are considered, demonstrating and discussing the robustness of the method with the variation of problem parameters.

- Chapter 4 deals with the anomaly reaction task and presents a modified version of our journal publication “Coevolving Defender Strategies Within Adversarial Ground Station Transit Time Games via Competitive Coevolution” [6]. At first, the problem is again framed in the context of space assets security. Next, a background on the methodology is provided, followed by details on our particular implementation. Finally, a wide pool of scenarios is presented and the associated results are analyzed.

- Chapter 5 concludes the manuscript and presents future potential avenues of research for each topic.

Chapter 2

Investigating Trajectory Abort Scenarios for Mission Resilience Enhancement

2.1 Introduction

As highlighted in the Global Exploration Roadmap [7], over the past decade multiple nations have expressed a renewed interest in returning to the Moon, announcing programs targeting the return of mankind to the lunar surface by the end of this decade. One such program is the NASA's Artemis mission, which reached its first major milestone in December 2022 with the successful completion of Artemis I. One of the primary objectives of the program involves the establishment of a human outpost on the lunar surface, which will be exploited as a proving ground for the testing of new technologies and demonstration of reliability of human long-duration habitats for future deep-space exploration (such as future missions to Mars), and a field laboratory to conduct lunar science.

Toward these objectives, part of the mission includes the realization of the Deep Space Gateway (DSG), an orbiting space station with both operational and scientific purposes. The DSG will in fact provide essential support for lunar surface missions, an environment to conduct science, and it will constitute a staging point for further deep space missions. Due to its pivotal role, a convenient operational orbit must be carefully identified. Among several options, a 9:2 resonant ($\sim$ 6.5 days) Near Rectilinear Halo Orbit (NRHO) [8] has been selected. Such an orbit belongs to the NRHO family, a class of orbits identifiable within a three-body dynamical system and defined by orbits extending close to the smaller body, almost perpendicularly to the line connecting the primary bodies. Several operational advantages are offered by the selected NRHO, including the possibility of continuous communication with the Earth, simpler shadow avoidance, lower station-keeping costs, and facilitated accessibility to the lunar surface and deep space [9, 10, 11].

Starting from Artemis III, the lunar gateway will serve as a staging point for crewed missions to the lunar surface, enabling access to multiple regions of the Moon. According to the Human Landing System (HLS) mission conops [12], operations for crewed missions will exploit a circular Low Lunar Orbit (LLO) as intermediate step in the descent/ascent phases. Leveraging a LLO offers several advantages, including flexibility for Lunar Orbit Insertion (LOI), reduced-cost options (in terms of Δv) for lunar descents, and in general, a convenient intermediate break point for landing and ascent operations [13]. A schematic representation for a three-elements mission concept is provided in Fig. 2.1.

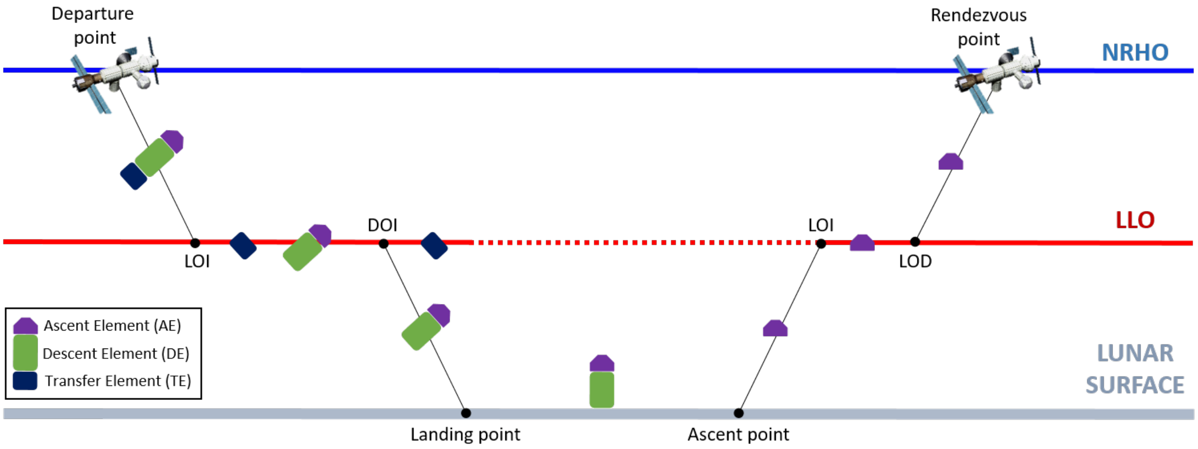


Figure 2.1: Schematic representation of the structure of a roundtrip NRHO-lunar surface for a three-elements mission concept. LOI: lunar orbit injection, DOI: descent orbit initiation, LOD: lunar orbit departure.

While nominal mission trade studies provide meaningful information to define system performance requirements, one must consider the possibility of failures throughout the mission operational lifetime. This aspect becomes even more relevant when considering crewed missions. As such, positive abort capabilities must be considered. Of note, these off-nominal scenarios do not necessarily impede successful completion of the mission, but they may lead to partial fulfilment of the missions objectives. Therefore, integration of abort modes into the design process may still lead to a successful mission, though with sub-optimal performance, ultimately enhancing overall design resilience. Unfortunately, the uncertainty on the mission schedule (initial plans promised a manned mission by the end of 2024, now pushed beyond 2026) requires the analysis of a large number of abort trajectories, potentially necessitating the scanning of all combinations of the problem search space parameters. However, conducting this operation is often unfeasible, especially in a time-constrained environment. Therefore, more effective ways to identify feasible solutions must be considered. Traditionally, two approaches are employed in the trajectory design process: 1) direct search, which relies on the utilization of numerical methods for the exploration of the search space and the identification of trajectories, and 2) flow-informed search, which leverages knowledge of the dynamical environment to reveal potential solutions and gain further understanding of the solution space.

Traditional direct search methods adopt brute-force grid search strategies, such as full-factorial Monte Carlo analyses, to explore the design space. While direct search methods can cover large regions of the solution space, their computational cost grows exponentially with the number of design variables, such as trajectory segment epochs, corrections, abort modes, etc. Additionally, existing optimization pipelines typically rely on a set of initial guesses whose quality is directly correlated to the convergence robustness of the corresponding pipeline. The identification of such initial guesses is a non-trivial task, especially when dealing with a complex dynamical environment. Furthermore, black-box searches do not directly provide an understanding of the underlying dynamical structures, which are generally important to gain confidence in operating within highly non-linear dynamical environments, such as the cislunar space. As direct search methods may have limitations when dealing with complex, large search spaces, alternative methodologies are necessary.

Flow-informed search techniques often complement direct search methods. Notably, such techniques are agnostic to model fidelity, and can thus become applicable where traditional approaches lose their effectiveness. Within the context of spacecraft trajectory design, previous studies have highlighted the advantages introduced by flow-informed methodologies [14, 15], as they can provide an additional geometrical understanding of the motion of a spacecraft under complex dynamics fields. Among these techniques, Finite-Time Lyapunov Exponents (FTLEs) have emerged as an effective tool to describe the dynamical structure of complex spaces [16]. In fact, FTLEs are informative of stretching directions in the dynamical flow. As such, trajectory designers are aided in the characterization of useful directions of motion, and in the parametrization of regions of the design space associated with a common trajectory pattern.

In this chapter, we explore the structures of the NRHO-to-LLO abort trajectory design space on two scenarios.

1. Descent. The first scenario considers the trajectory segment that goes from the NRHO to a selected LLO, and it assumes a failure occurs along the trajectory, such that the module is commanded to rendezvous with the Gateway.
2. Ascent. The second scenario considers the full ascent from the lunar surface to the NRHO, and it assumes the surface stay is aborted such that the crew has to return to the DSG before the nominal mission duration. In conformity with the mission conops, an intermediate LLO is selected as break point between the lunar surface and the NRHO.

We employ direct and flow-informed search approaches to characterize the solution space, presenting different levels of analysis for the specific scenario.
For the descent problem, we first capture a snapshot of the convergence structure for a selected case study. Through this initial investigation, we highlight the existence of regions of stiff convergence for different variations of the trajectory correction pipeline. In an attempt to gain additional insight into the obtained results, we consider the same abort scenario while exploring the utilization of FTLEs maps [17], which provide qualitative and quantitative information regarding the types of motion associated with the identified trajectories. By leveraging this tool, classes of trajectories associated with particular types of motion can be identified, thus providing a first characterization of the problem's dynamical structure. Finally, we demonstrate an initial guess generator based on surrogate models trained via adaptive sampling techniques, which enable smart sampling of the parameter space, reducing the overall number of points required to train an effective metamodel. While the training of the surrogate model adds extra operations, our preliminary results on the net acceleration of the trajectory sweep process indicate a positive improvement. By providing the downstream optimization with a better guess, we reduce the

number of high-fidelity model evaluations required for convergence, reducing the total computational cost. The enhancement is also reflected in the diminished areas of stiff convergence within the solution space's structure.

Similar studies are conducted for the ascent problem, which also displays regions of stiff convergence based on variations of the pipeline. In particular, our analysis reveals how the solution surface is much more complex than the one identified in the descent problem, hindering the efficacy of a surrogate model. Hence, an in-depth analysis is conducted to determine the key factors behind the underlying structure behavior. Specifically, we investigate the most sensitive cases from a numerical and dynamical perspective, and we conduct further analysis by considering additional problem parameters. The outcome of this study is then summarized in a series of key considerations which become instrumental for the trajectory design problem.

The remainder of this chapter is organized as follows. Sec. 2.2 presents a description of the dynamical model and introduces the reference frames utilized throughout the work. Sec. 2.3 provides an overview of numerical integration techniques. Sec. 2.4 presents the set up of the two abort scenarios. Sec. 2.5 contains an overview of the different methodologies utilized in the investigation. Sec. 2.6 presents the analyses and discussion associated with the descent problem. A similar structure then follows in Sec. 2.7 for the ascent scenario.

2.2 Dynamical Models

The Circular Restricted Three-Body Problem (CR3BP) is adopted to describe the dynamical environment. The CR3BP constitutes a simplification of the more general three body problem, which assume three gravitationally attractive bodies, namely two primaries and the spacecraft, each influencing the motion of the others. In the case of the CR3BP, the mass of the third body (the spacecraft) is negligible. For the case of the Earth-Moon system, the mass ratio between the primaries is larger with respect to the one in many other systems. This makes the CR3BP framework particularly suitable to gain sufficient insight into the dynamical behavior of a spacecraft in cislunar regime without the burden of the computational cost required by higher-fidelity models [18].
Let P_1, P_2 and P_3 be the three primary bodies, with associated masses m_1, m_2 and m_3. Then, the main assumptions of the CR3BP can be formalized as:

- The spacecraft, P_3 has a negligible mass, that is, $m_3 \ll m_1, m_2$, such that the motion of P_1 and P_2 is not influenced by P_3.
- P_1, P_2 and P_3 are all considered point masses solely capable of translational motion.
- P_1 and P_2, with $m_1 > m_2$, revolve on a circular Keplerian orbit with respect to a common barycenter. The circularity assumption constitutes a further simplification to the three body problem, which would otherwise consider elliptical orbits, resulting in the slightly more general Elliptical Restricted Three Body Problem (ER3BP).

Despite the simplification, the resulting framework remains sufficiently accurate as a first representation of the motion of a spacecraft in cislunar regime, and leads to an autonomous model when formulated in a rotating, or synodic, frame.

2.2.1 Equations of Motion

To derive the equations of motions (EOMs) in the synodic frame, let us start by defining an inertial reference frame $\hat{X}$-$\hat{Y}$-$\hat{Z}$, centered at the barycenter of the primaries P_1 and P_2. As mentioned, the primaries revolve on Keplerian circular orbits, with their motion lying on the $\hat{X}$-$\hat{Y}$ plane. In such a way, the $\hat{Z}$ axis results to be normal to the primaries orbital plane, and parallel to the angular momentum vector. Conversely, P_3 is free to move in all spatial dimensions. Newton's Second Law can then be applied to describe the motion of P_3 as influenced by the gravitational pull of the primaries [19]:

$$\ddot{\mathbf{r}}_3 = -\frac{Gm_1}{r_{13}^3}\mathbf{r}_{13} - \frac{Gm_2}{r_{23}^3}\mathbf{r}_{23} \tag{2.1}$$

with G being the universal gravity constant, and $\mathbf{r}_{ij}$ (such that $\mathbf{r}_{ij}$ = $\mathbf{r}_i$ - $\mathbf{r}_j$) expressing the position of the body P_i with respect to the body P_j. Next, to express Eq. (2.1) in the synodic frame, an appropriate rotating coordinate frame must be introduced. To this aim, the CR3BP assumes such frame defined as follows: the $\hat{x}$ axis is defined along the P_1-P_2 direction, with the $\hat{z}$ axis oriented parallel to the primary angular momentum vector. The $\hat{y}$ axis completes then the orthogonal triad. The rotation of the $\hat{x}$-$\hat{y}$-$\hat{z}$ frame is described by an angle, θ, such that θ = Nt, considering θ_0 = Nt_0 = 0. In the previous expression, N represents the mean motion associated with the primary bodies, while t is the dimensional time elapsed since the initial time t_0. Of note, the expression of constant angle rate holds due to the circularity assumption. A representation of the frames is reported in Fig. 2.2.

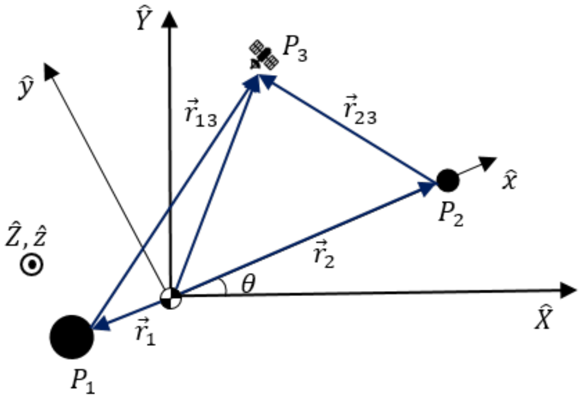


Figure 2.2: Definitions of barycentric synodic and inertial frames: rotating frame ($\hat{x}, \hat{y}$) oriented with angle θ relative to the inertial frame ($\hat{X}, \hat{Y}$).

As no closed-form analytical solution exists for the CR3BP, numerical integration must be performed. To this aim, a common step involves non-dimensionalization of the EOMs, which makes the results more general and improves numerical conditioning for integration. Hence, a few characteristic quantities are introduced for mass (Eq. (2.2)), distance (Eq. (2.3)) and time (Eq. (2.4)):

$$m^* = m_1 + m_2 \tag{2.2}$$

$$l^* = r_1 + r_2 \tag{2.3}$$

$$t^* = \sqrt{\frac{l^{*3}}{Gm^*}} \tag{2.4}$$

Notably, the characteristic time, which derives from Kepler's third law, is such that the non-dimensional mean motion, from now on denoted with "n", is equal to 1. A summary for the value of the characteristic quantities in the Earth-Moon system is reported in Tab. 2.1.

Parameter	Value
m^* [kg]	6.0458×10^{24}
l^* [km]	384400
t^* [s]	375200

Table 2.1: Transfer trajectory characteristics.

Introduction of these characteristic quantities allows to express Eq. (2.1) in its non-dimensional form. Defining:

$$\boldsymbol{\rho} = \frac{\mathbf{r}_3}{l^*} = x\hat{x} + y\hat{y} + z\hat{z} \tag{2.5}$$

$$\mathbf{r} = \frac{\mathbf{r}_{23}}{l^*} = (x - 1 + \mu)\hat{x} + y\hat{y} + z\hat{z} \tag{2.6}$$

$$\mathbf{d} = \frac{\mathbf{r}_{13}}{l^*} = (x + \mu)\hat{x} + y\hat{y} + z\hat{z} \tag{2.7}$$

where $\mu = \frac{m_2}{m_1+m_2}$ denotes the system mass ratio, Eq. (2.1) becomes:

$$\ddot{\boldsymbol{\rho}} = -\frac{(1-\mu)}{d^3}\mathbf{d} - \frac{\mu}{r^3}\mathbf{r} \tag{2.8}$$

Please note that Eq. (2.8) represents the dynamics of P_3 as seen from an inertial observer, expressed in the rotating frame. Representation of the same dynamics from the perspective of a rotating observer can then be obtained by applying the transport theorem. Given $\boldsymbol{\rho}$, and denoting the synodic frame through the letter "S ", the expression for the velocity and acceleration reads:

$$\dot{\boldsymbol{\rho}} = {}^{\mathrm{I}}\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{\rho}) = {}^{\mathrm{S}}\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{\rho}) + \boldsymbol{\omega}_{S/I} \times \boldsymbol{\rho} \tag{2.9}$$

$$\ddot{\boldsymbol{\rho}} = {}^{\mathrm{I}}\frac{\mathrm{d}^2}{\mathrm{d}t^2}(\boldsymbol{\rho}) = {}^{\mathrm{S}}\frac{\mathrm{d}}{\mathrm{d}t}(\dot{\boldsymbol{\rho}}) + \boldsymbol{\omega}_{S/I} \times \dot{\boldsymbol{\rho}} = {}^{\mathrm{S}}\frac{\mathrm{d}^2}{\mathrm{d}t^2}(\boldsymbol{\rho}) + 2\boldsymbol{\omega}_{S/I} \times {}^{\mathrm{S}}\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{\rho}) + \boldsymbol{\omega}_{S/I} \times \boldsymbol{\omega}_{S/I} \times \boldsymbol{\rho} \tag{2.10}$$

where $\boldsymbol{\omega}_{S/I}$ represents the angular velocity vector of the synodic frame **S** relative to the inertial frame **I**. Recalling the assumption of the model, this corresponds to a vector of constant magnitude and direction, and it is expressed as $\boldsymbol{\omega}_{S/I} = n\hat{z}$. Finally, substituting Eq. (2.5) into Eq. (2.10) and considering its individual components, the non-dimensional EOMs for the CR3BP expressed in the rotating frame become:

$$\begin{aligned} \ddot{x} - 2n\dot{y} - n^2x &= -\tfrac{(1-\mu)(x+\mu)}{d^3} - \tfrac{\mu(x-1+\mu)}{r^3}, \\ \ddot{y} + 2n\dot{x} - n^2y &= -\tfrac{(1-\mu)y}{d^3} - \tfrac{\mu y}{r^3}, \\ \ddot{z} &= -\tfrac{(1-\mu)z}{d^3} - \tfrac{\mu z}{r^3}, \end{aligned} \tag{2.11}$$

where n, here equal to 1, can be omitted. As concluding remark, this set of equations can be expressed in a more compact form with the introduction of the pseudo-potential function:

$$U^* = U + \frac{1}{2}n^2(x^2 + y^2) = \frac{1-\mu}{d} + \frac{\mu}{r} + \frac{1}{2}n^2(x^2 + y^2) \tag{2.12}$$

Following the introduction of this quantity, the re-written EOMs become:

$$\begin{aligned} \ddot{x} - 2n\dot{y} &= \tfrac{\partial U^*}{\partial x} \\ \ddot{y} + 2n\dot{x} &= \tfrac{\partial U^*}{\partial y} \\ \ddot{z} &= \tfrac{\partial U^*}{\partial z} \end{aligned} \tag{2.13}$$

2.2.2 Jacobi Constant

The Jacobi constant is the only integral of motion in the CR3BP problem, and it can be interpreted as an energy-like term, which remains constant within the restricted three body formulation. To derive the expression of such a constant, one can simply take the dot product of Eq. (2.13) with the velocity expressed in rotating frame, thus obtaining:

$$\dot{x}\ddot{x} + \dot{y}\ddot{y} + \dot{z}\ddot{z} = \frac{\partial U^*}{\partial x}\dot{x} + \frac{\partial U^*}{\partial y}\dot{y} + \frac{\partial U^*}{\partial z}\dot{z} \tag{2.14}$$

As the pseudo-potential U^* is a function of the position alone, the right hand side can be expressed as its time total derivative, $\frac{\mathrm{d}U^*}{\mathrm{d}t}$, thus allowing to re-write the previous equation as:

$$\dot{x}\ddot{x} + \dot{y}\ddot{y} + \dot{z}\ddot{z} = \frac{\mathrm{d}U^*}{\mathrm{d}t} \tag{2.15}$$

Time integration of Eq. (2.15) let then introduce an integration constant, C, as:

$$\frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) = U^* + \frac{C}{2} \tag{2.16}$$

Finally, rearranging the terms in this last equation and recognizing the left hand side to be the square of the velocity expressed in rotating frame, the Jacobi constant assumes the form:

$$C = 2U^* - V^2 \tag{2.17}$$

This quantity is directly related to the energy of the system. In particular, an increase of the energy corresponds to a decrease of the Jacobi constant (and the other way around.) The Jacobi constant serves multiple functions within the CR3BP problem, including monitoring the numerical stability of integration, identification of the necessary energy variation required by maneuvers, and identification of boundaries for the motion of a particle.

2.2.3 Frame conversion

Rotating Frame. Often times, as it will be later discussed in this chapter, it is necessary to convert quantities from the rotating frame to the inertial frame (and the other way around).

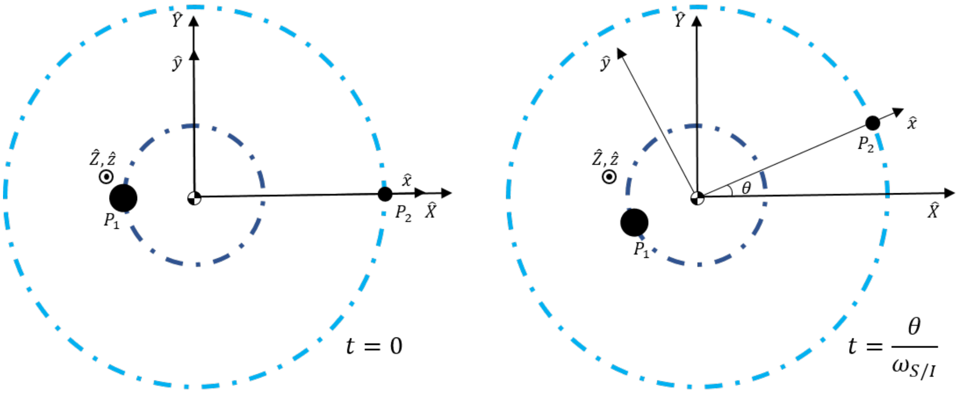


Figure 2.3: Relative orientation of inertial and rotating frames at two representative times.

With reference to Fig. 2.3, and considering the previously introduced notation with capital letters for the inertial frame and lowercase for the rotating frame, the transformation from rotating to inertial coordinates is given as [20]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathcal{R}_{S/I}\boldsymbol{\rho} \tag{2.18}$$

To compute the vector time derivative, one can consider each component individually, where the time information is contained within the rotation angle through the relation $\theta = \boldsymbol{\omega}_{S/I}t$, resulting into the following expressions:

$$\dot{X} = \dot{x}\cos\theta - \dot{y}\sin\theta - (x\sin\theta + y\cos\theta) \tag{2.19}$$

$$\dot{Y} = \dot{x}\sin\theta + \dot{y}\cos\theta + (x\cos\theta - y\sin\theta) \tag{2.20}$$

$$\dot{Z} = \dot{z} \tag{2.21}$$

Hence, the rotation of a full state, here intended as a combination of position and velocity, can be expressed through:

$$\begin{bmatrix} X \\ Y \\ Z \\ \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{S/I} & \mathbf{0}_{3\times 3} \\ \\ \dot{\mathcal{R}}_{S/I} & \mathcal{R}_{S/I} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{2.22}$$

where the derivative of the rotation matrix $\dot{\mathcal{R}}_{S/I}$ is given by:

$$\dot{\mathcal{R}}_{S/I} = \begin{bmatrix} -\sin\theta & -\cos\theta & 0 \\ \cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.23}$$

To transform from inertial to rotating frame, the inverse of the total rotation matrix can be taken. It is worth mentioning that the above procedure, though general, is here meant to be applied between the inertial frame centered at the system barycenter and the synodic frame. Depending on the task, or by the necessity to gain additional insight, one may desire to center the origin of the system at other basepoints, such as of one of the primaries. In such a case, the coordinates can be first translated to the desired basepoints, and the transformation matrix is then applied to the translated vector. This procedure will be used throughout the next sections to convert from rotating to arbitrarily centered inertial frames.

Other Frames. Sometimes it is convenient to express quantities with respect to a different reference frame, as it may provide a better understanding of the problem. One such frame is the $\hat{R}$-$\hat{S}$-$\hat{W}$ coordinate system, that is a local frame co-moving with the spacecraft [21]. The system is defined as follows: the $\hat{R}$ axis indicates the radial direction pointing out from the spacecraft along the planetocentric radius vector, the $\hat{S}$ axis is normal to the radial direction and points toward the direction of motion (along-track direction), and the $\hat{W}$ is normal to the orbital plane (cross-track direction). A representation of the coordinate system is depicted in Fig. 2.4.

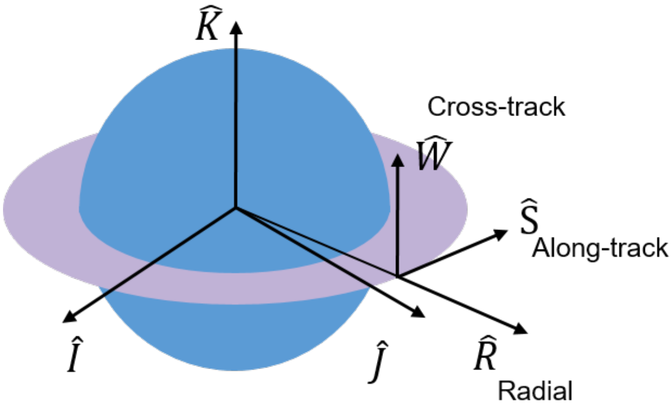


Figure 2.4: Representation of the $\hat{R}$-$\hat{S}$-$\hat{W}$ frame with respect to a generic inertial frame $\hat{I}$-$\hat{J}$-$\hat{K}$.

To convert from the inertial frame to the local frame (and the other way around) we first define the direction of the $\hat{R}$-$\hat{S}$-$\hat{W}$ axes as follows:

$$\hat{R} = \frac{\mathbf{r}}{|\mathbf{r}|} \tag{2.24}$$

$$\hat{W} = \frac{\mathbf{r} \times \mathbf{v}}{|\mathbf{r} \times \mathbf{v}|} \tag{2.25}$$

$$\hat{S} = \hat{W} \times \hat{R} \tag{2.26}$$

where $\mathbf{r}$ and $\mathbf{v}$ represent the position and velocity vector expressed in the inertial frame. Next, the conversion from inertial to local frame reads:

$$\mathbf{r}_{RSW} = [\hat{R} \;\; \hat{S} \;\; \hat{W}]^T \mathbf{r}_{IJK} \tag{2.27}$$

2.3 Numerical schemes

With solely one integral of motion, that is, the Jacobi constant (see Sec. 2.2.2), the CR3BP does not present a closed-form solution. As such, numerical integration is required to propagate trajectories. However, due to the chaotic nature of the system, it is not possible to identify a priori a set of initial conditions that yield a desired behavior. As such, differential correction schemes are typically introduced. In the following, the techniques adopted within this work are briefly described.

2.3.1 State Transition Matrix

Before describing the most prominent differential correction schemes, it is first relevant to introduce the concept of State Transition Matrix (STM). In fact, the definition of these schemes relies on the concept of sensitivity, thus requiring to measure how variations to an initial state relate to deviations in a downstream state. Consider at first the nonlinear system of differential equations expressed in its first-order form as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \tag{2.28}$$

with $\mathbf{x}^\top = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$. Next, consider some set of initial conditions generating a reference path $\mathbf{x}^*(t)$. Then, a trajectory originating from the variation of such a path can be expressed as:

$$\mathbf{x}(t) = \mathbf{x}^*(t) + \delta\mathbf{x}(t) \tag{2.29}$$

where $\delta\mathbf{x}(t)$ denotes the deviation with respect to the nominal path through time. Plugging the last equation in Eq. (2.28) and taking the Taylor series expansion about the reference trajectory provides the linear approximation (higher order terms are neglected) for the time evolution of the deviation:

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}^* + \delta\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}^* + \delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^*) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^*} \delta\mathbf{x} \tag{2.30}$$

Cancelling out the terms related to the reference trajectory, the above equation reduces to:

$$\delta\dot{\mathbf{x}} \approx \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^*} \delta\mathbf{x} = A\delta\mathbf{x} \tag{2.31}$$

where A ia a matrix denoting the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$ evaluated along the reference. Such matrix has an analytical expression given as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ U^*_{xx} & U^*_{xy} & U^*_{xz} & 0 & 2 & 0 \\ U^*_{yx} & U^*_{yy} & U^*_{yz} & -2 & 0 & 0 \\ U^*_{zx} & U^*_{zy} & U^*_{zz} & 0 & 0 & 0 \end{bmatrix} \tag{2.32}$$

with the notation U^*_{ij} expressing the second order partial derivatives of the pseudo-potential function (Eq. (2.12)).

The solution to Eq. (2.31) can be expressed in a form that maps the variations in the initial state to variations in a downstream state at some time t, according to:

$$\delta\mathbf{x}(t) = \left(\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t_0)}\right)\delta\mathbf{x}(t_0) \tag{2.33}$$

where the matrix $\left(\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t_0)}\right)$ provides the linear mapping, and it represents the STM, denoted as $\Phi(t, t_0)$. To determine the STM, one can take its time derivative:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t_0)}\right) = \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}(t_0)}\dot{\mathbf{x}}(t) = \frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}(t)}\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t_0)} = A\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t_0)} \tag{2.34}$$

In conclusion, the evolution of the STM can be obtained as the solution of the first-order differential equation:

$$\dot{\Phi}(t, t_0) = A\Phi(t, t_0) \tag{2.35}$$

To propagate Eq. (2.35), the initial conditions for the STM are given as $\Phi(t, t_0) = \mathcal{I}_{6\times6}$. Then, the elements in the STM can be determined from the A matrix, and integrated according to Eq. (2.35) along with the equations of motion.

2.3.2 Differential Corrections Algorithms

In the context of trajectory design, differential correction algorithms are a class of algorithms which aim to identify solutions such that a given trajectory satisfies a set of constraints. These methods are extremely popular in the CR3BP framework, where initial conditions can sometime be guessed from a linear approximation, which rarely leads to the desired behavior in non-linear regime. In particular, a constraint/free variable Newton method is traditionally employed to solve two-point boundary value problems, where the free variables correspond to the set of variables allowed to be adjusted during the optimization process. Common free variables include state-related quantities (position and/or velocity components), time of flight and epochs. To begin with, let $\mathbf{X}$ be the free variable vector with n free variables, that is:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \tag{2.36}$$

The components of $\mathbf{X}$ have then to be adjusted to meet some constraints, which are enclosed into a constraint vector $\mathbf{F}(\mathbf{X})$, defined as:

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} F_1(\mathbf{X}) \\ F_2(\mathbf{X}) \\ \vdots \\ F_m(\mathbf{X}) \end{bmatrix} = \mathbf{0} \tag{2.37}$$

with m being the number of constraints. Similarly to the free variable vector, constraints are generally applied to position, velocity, or time of flight, though many other can be imposed based on the specific problem. Given these two quantities, the goal is to identify a design vector

$\mathbf{X}^*$ such that $\mathbf{F}(\mathbf{X}^*) = 0$. To this aim, let $\mathbf{X}^0$ be the initial guess for such a vector. Then, the constraint vector can be approximated with its first-order Taylor expansion about the initial guess:

$$\mathbf{F}(\mathbf{X}) = \mathbf{F}(\mathbf{X}^0) + \frac{\partial \mathbf{F}(\mathbf{X}^0)}{\partial \mathbf{X}^0}(\mathbf{X} - \mathbf{X}^0) \tag{2.38}$$

where $\frac{\partial \mathbf{F}(\mathbf{X}^0)}{\partial \mathbf{X}^0}$ is the $m \times n$ Jacobian of the constraint vector, from here on denoted as $\mathbf{DF}(\mathbf{X}^0)$. Hence, recalling the objective is to identify $\mathbf{X}$ such that $\mathbf{F}(\mathbf{X}) = \mathbf{0}$, Eq. (2.38) can be written in an iterative update form, that is:

$$\mathbf{F}(\mathbf{X}^j) + \mathbf{DF}(\mathbf{X}^j)(\mathbf{X}^{j+1} - \mathbf{X}^j) = \mathbf{0} \tag{2.39}$$

where $\mathbf{X}^j$ and $\mathbf{X}^{j+1}$ represent the free variable vector at the current and next iteration respectively. As the variable vector and the constraint vector are both available at the j-th iteration, the Jacobian can be computed, thus allowing to rearrange Eq. (2.39) to define the update equation for the design variable vector:

$$\mathbf{X}^{j+1} = \mathbf{X}^j - \mathbf{DF}(\mathbf{X}^j)^{-1}\mathbf{F}(\mathbf{X}^j) \tag{2.40}$$

The iterative process is continued till the norm of the constraint vector decreases below an arbitrarily small tolerance ϵ, that is, until $||\mathbf{F}(\mathbf{X}^{j+1})|| < \epsilon$.

A few conclusive notes must be done in regard of the existence of a solution. In particular, if $n > m$, infinite many solutions exist. In such a case, a minimum norm approach can be taken, which involves considering the update equation as:

$$\mathbf{X}^{j+1} = \mathbf{X}^j - \mathbf{DF}(\mathbf{X}^j)^T[\mathbf{DF}(\mathbf{X}^j)\mathbf{DF}(\mathbf{X}^j)^T]\mathbf{F}(\mathbf{X}^j) \tag{2.41}$$

Instead, if $n < m$, the system is over-constrained and no solution exists. For a square system, exactly one solution can instead be obtained.

2.3.3 Single Shooting

The single shooting (SS) method is a popular differential correction algorithm in the context of trajectory design. With reference to Fig. 2.5, consider a trajectory whose initial conditions are given as $\mathbf{x}_0^\top = [x_0, y_0, z_0, \dot{x}_0, \dot{y}_0, \dot{z}_0]$ at some initial time t_0. The trajectory is then propagated for a time $t = t_0 + T$, such that $\mathbf{x}(t) = \mathbf{x}(t_0 + T)$. Furthermore, consider a target position is desired, that is, the trajectory is desired to terminate at some state $\mathbf{r}^* = [x^*, y^*, z^*]$ by modifying the initial velocity and the time of flight.

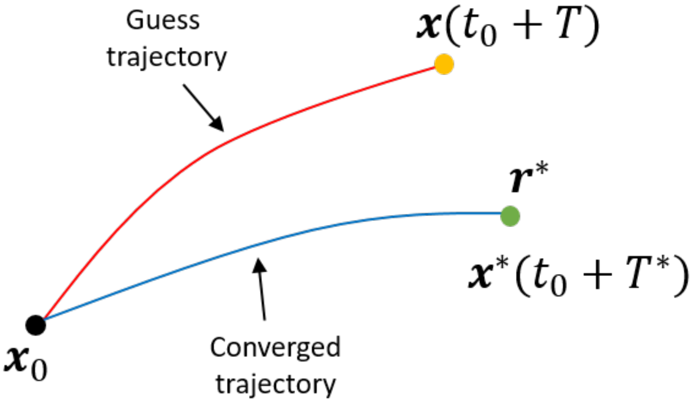


Figure 2.5: Single shooting targeting scheme representation.

In such a case, the design variable and the constraint vectors can be respectively defined as:

$$\mathbf{X} = \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ T \end{bmatrix} \tag{2.42}$$

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} x(t_0 + T) - x^* \\ y(t_0 + T) - y^* \\ z(t_0 + T) - z^* \end{bmatrix} \tag{2.43}$$

As $n > m$, the minimum norm approach (Eq. (2.41)) must be adopted. The Jacobian of the constraints can be written as:

$$\mathbf{DF}(\mathbf{X}) = \begin{bmatrix} \frac{\partial(x(t_0+T)-x^*)}{\partial \dot{x}_0} & \frac{\partial(x(t_0+T)-x^*)}{\partial \dot{y}_0} & \frac{\partial(x(t_0+T)-x^*)}{\partial \dot{z}_0} & \frac{\partial(x(t_0+T)-x^*)}{\partial T} \\ \frac{\partial(y(t_0+T)-y^*)}{\partial \dot{x}_0} & \frac{\partial(y(t_0+T)-y^*)}{\partial \dot{y}_0} & \frac{\partial(y(t_0+T)-y^*)}{\partial \dot{z}_0} & \frac{\partial(y(t_0+T)-y^*)}{\partial T} \\ \frac{\partial(z(t_0+T)-z^*)}{\partial \dot{x}_0} & \frac{\partial(z(t_0+T)-z^*)}{\partial \dot{y}_0} & \frac{\partial(z(t_0+T)-z^*)}{\partial \dot{z}_0} & \frac{\partial(z(t_0+T)-z^*)}{\partial T} \end{bmatrix} \tag{2.44}$$

Next, noticing that the desired state is independent of the free variables, and recognizing the remaining terms to be partial derivatives of the current state with respect to an initial state, the Jacobian of the constraint vector can be expressed in terms of elements of the STM:

$$\mathbf{DF}(\mathbf{X}) = \begin{bmatrix} \frac{\partial x(t)}{\partial \dot{x}_0} & \frac{\partial x(t)}{\partial \dot{y}_0} & \frac{\partial x(t)}{\partial \dot{z}_0} & \frac{\partial x(t)}{\partial T} \\ \frac{\partial y(t)}{\partial \dot{x}_0} & \frac{\partial y(t)}{\partial \dot{y}_0} & \frac{\partial y(t)}{\partial \dot{z}_0} & \frac{\partial y(t)}{\partial T} \\ \frac{\partial z(t)}{\partial \dot{x}_0} & \frac{\partial z(t)}{\partial \dot{y}_0} & \frac{\partial z(t)}{\partial \dot{z}_0} & \frac{\partial z(t)}{\partial T} \end{bmatrix} = \begin{bmatrix} \Phi_{14} & \Phi_{15} & \Phi_{16} & \dot{x}(t) \\ \Phi_{24} & \Phi_{25} & \Phi_{26} & \dot{y}(t) \\ \Phi_{34} & \Phi_{35} & \Phi_{36} & \dot{z}(t) \end{bmatrix} \tag{2.45}$$

2.3.4 Multiple Shooting

For cases involving more complex trajectories, a multi-segment approach is typically leveraged. Differently from the single shooting, the multiple shooting (MS) scheme solves simultaneously several two-point boundary value problems, aiming to satisfy some desired constraints. To formulate the multiple shooting routine, a trajectory is segmented into $n-1$ arcs connecting n patch points (Fig. 2.6).

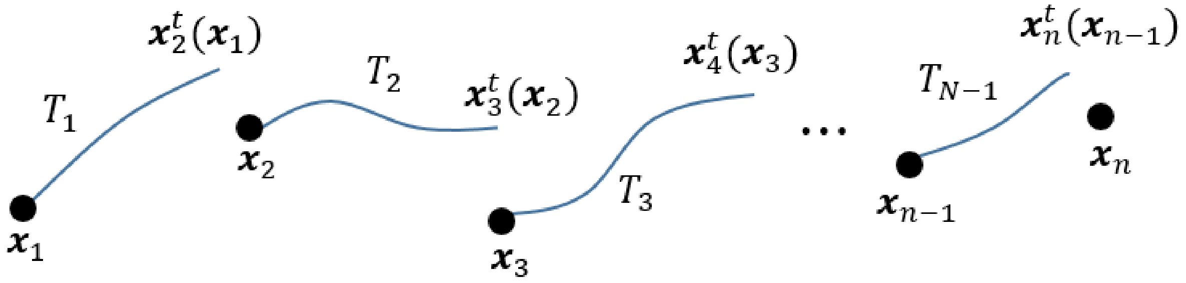


Figure 2.6: Multiple shooting targeting scheme representation.

For the sake of generality, let us assume each arc to have a different propagation time. Hence, each state of an i-th patch point, $\mathrm{x}_i = [x(t_i), y(t_i), z(t_i), \dot{x}(t_i), \dot{y}(t_i), \dot{z}(t_i)]^\top$, is propagated for a time T_i, reaching a terminal state x_{i+1}.

Given these preliminaries, the design variable vector can be defined as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \\ T_1 \\ T_2 \\ \vdots \\ T_{n-1} \end{bmatrix} \tag{2.46}$$

As the arcs have to be continuous in position and velocity to achieve a smooth trajectory, the constraint vector can be formulated as follows[1]:

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} \mathbf{x}_2^t(\mathbf{x}_1) - \mathbf{x}_2 \\ \mathbf{x}_3^t(\mathbf{x}_2) - \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n^t(\mathbf{x}_{n-1}) - \mathbf{x}_n \end{bmatrix} = \mathbf{0} \tag{2.47}$$

Therefore, the Jacobian of the constraint vector assumes the following form:

$$\begin{bmatrix} \Phi_1 & -\mathcal{I}_{6\times 6} & & & \dot{\mathbf{x}}_2^t & \\ & \ddots & \ddots & & & \ddots & \\ & & \Phi_{n-1} & -\mathcal{I}_{6\times 6} & & & \dot{\mathbf{x}}_n^t \end{bmatrix} \tag{2.48}$$

where the notation Φ_j refers to the STM along the j-th trajectory segment.

As a final note, the formulation here presented assumes each arc with different duration, which is referred to as "variable-time multiple-shooting" method. An analogous formulation exists with fixed arc time, which would result into the design variable vector not having the time-related terms.

[1] while this is is the traditional approach, one may also introduce slack variables in the design and constraint vector to enforce the time-related terms to remain positive.

Direct and flow-informed techniques are employed to characterize the convergence and dynamical structures of the cislunar space within two selected HLS abort scenarios.

Descent. In according with the mission conops, the descent problem formulation assumes a nominal NRHO-to-LLO transfer. Previous studies have considered multiple options for the departure point, finding NRHO-to-LLO transfers requiring about half a day [22] as the most convenient solutions in terms of total Δv budget and satisfaction of constraints of the mission. As our primary focus is narrowed solely to the NRHO-to-LLO transfer, we select the NRHO apolune as the departure point, which grants a larger time horizon for potential abort maneuvers, hence providing richer information regarding the design space. As a representative problem, we arbitrarily target a mean longitude L = 90° on a 100 km circular LLO. Under these settings, we assume an abort maneuver is performed along the transfer from the NRHO to the LLO. As a result, the HLS is commanded to rendezvous with the Gateway on the NRHO. As a proof of concept for our analysis, we consider a minimal scenario involving a two-dimensional search space, utilizing the time of flight (TOF) over the two trajectory segments (i.e., from the departure point to the abort point and from the abort point to the rendezvous point), from now on referred to as TOF1 and TOF2, as problem parameters. A schematic of the approach is shown in Fig. 2.7.

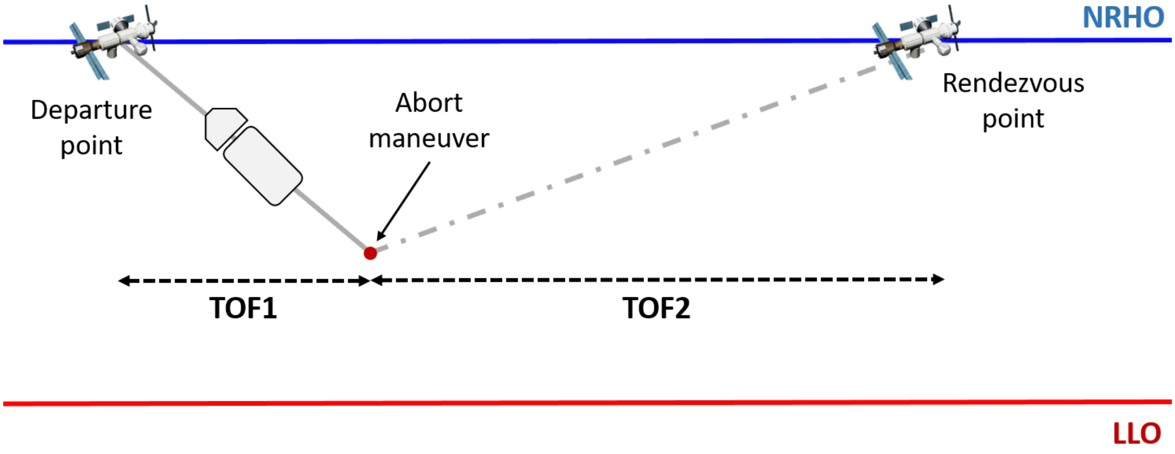


Figure 2.7: Schematic representation for the descent scenario.

Ascent. The ascent scenario presents a similar structure to the previous one. In this case, we assume the abort to be associated with the duration of the crew stay on the lunar surface, such that astronauts must return to the Gateway ahead of the nominal time. Also in this case, the mission conops consider an intermediate LLO as intermediate point. In this analysis, a representative polar LLO with an altitude of 150 km is selected. Analogously to the descent scenario, the time of flight associated with two trajectory phases is selected. The first segment combines the ascent from the lunar surface to the LLO and a loitering period on the LLO. The second segment is instead associated with the LLO-to-NRHO transfer. A representation of the scenario set up is depicted in Fig. 2.8.

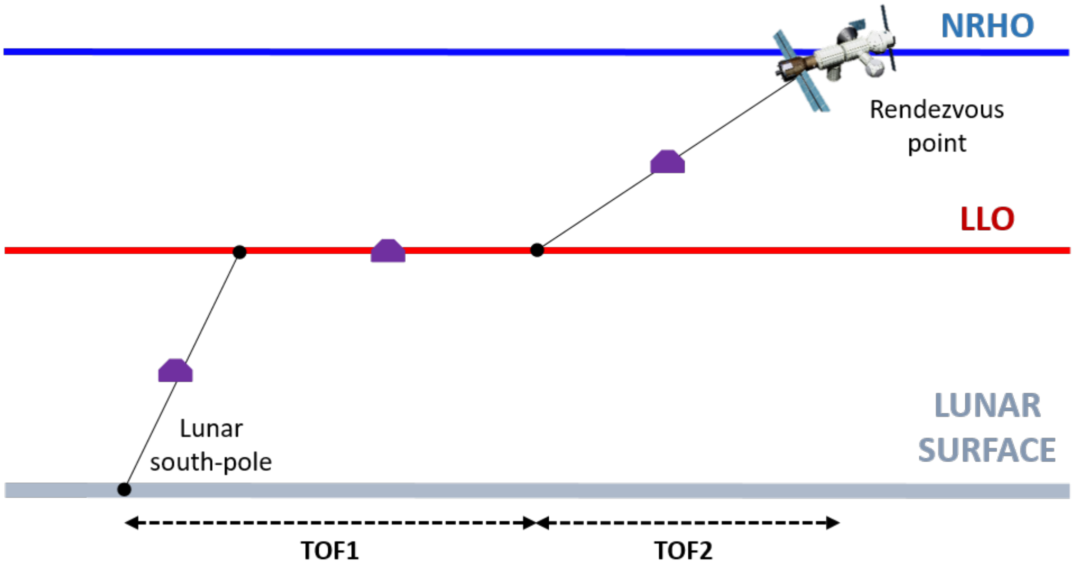


Figure 2.8: Schematic representation for the ascent scenario.

For what concern the ascent trajectory, the ascent module is assumed to depart from the lunar south-pole, with a motion governed by the following dynamics [23]:

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{T}{m} - g\sin\gamma \tag{2.49}$$

$$\frac{\mathrm{d}\gamma}{\mathrm{d}t} = -\frac{1}{v}\left(g - \frac{v^2}{R_M + h}\right)\cos\gamma \tag{2.50}$$

$$\frac{\mathrm{d}h}{\mathrm{d}t} = v\sin\gamma \tag{2.51}$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{R_M}{R_M + h} v\cos\gamma \tag{2.52}$$

$$\frac{\mathrm{d}m}{\mathrm{d}t} = -\frac{T}{I_{sp}g_0} \tag{2.53}$$

where v is the velocity, T is the thrust, g is the Moon gravitational acceleration, γ is the pitch angle, R_M is the Moon radius, h is the altitude, x is the downrange, m is the mass, I_{sp} is the specific impulse and g_0 is the gravitational constant at sea level. This set of equation is then propagated until the selected orbit altitude is reached, allowing to derive the true anomaly at the insertion point. Of note, for a selected right ascension of the LLO, it is assumed the launch occurs in a direction prograde with the orbital motion.

2.5 Methodology

This section provides the general background for the different methods utilized in our investigation. First, a direct-search technique is discussed in the context of the two selected scenarios. Next, surrogate modeling aided by adaptive sampling is introduced as a strategy aimed to improve the computational cost of the employed direct search method, followed by the description of a surrogate-aided optimization pipeline. Finally, a flow-informed method aimed at gaining additional insight into generated trajectory solutions from a dynamical standpoint is presented.

2.5.1 Direct Search

We employ a traditional grid search method for the scanning of the defined two-dimensional (TOF1-TOF2) design space. Abort trajectory solutions are then identified via a conventional two-step design approach. At first, an initial trajectory estimate is derived using a lower-fidelity orbit dynamics model, which is based on Keplerian dynamics for our preliminary proof-of-concept. Within the Keplerian model, the abort transfer is solved as a Lambert problem [23]. For the descent scenario, the Lambert problem is solved between the abort point and the rendezvous point. For the ascent scenario, the terminal state again coincides with the state of the Gateway on the NRHO, while the starting point corresponds to the state on the LLO after TOF1 has elapsed. Next, the trajectory identified as solution of the Lambert is refined within a higher-fidelity dynamics model (here, the CR3BP) by employing numerical correction techniques [24]. In order to gain more comprehensive observations, transfer trajectories are corrected using both single-shooting and multiple-shooting algorithms, with the latter formulated as a fixed-time shooting scheme. Both correction algorithms solely target a final position vector without constraints on the arrival Δv. Please note that while we acknowledge the transfer itself may be associated with an elevated total cost in terms of Δv, our primary interest is the definition of a reference trajectory. As such, an optimal (i.e., minimum-Δv) transfer is out of our scope. Notably, the fidelity of both models and the correction approach can vary in complexity. For instance, the lower-fidelity model could utilize CR3BP, while the higher-fidelity one can leverage NAIF ephemeris [2]. A schematic representation of the described approach is provided in Fig. 2.9.

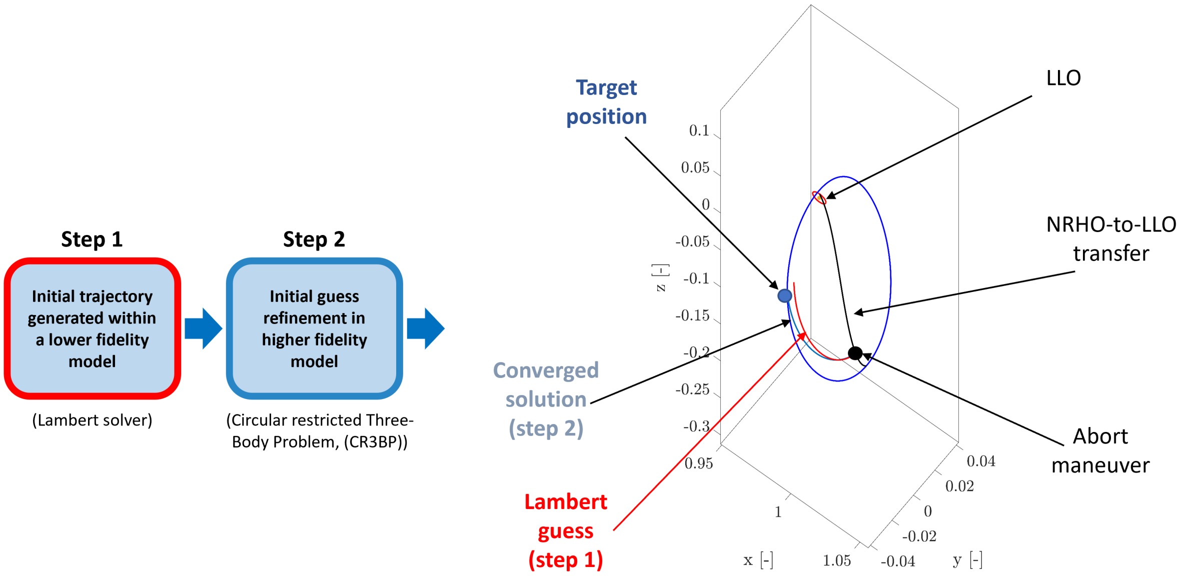

Step 1
Initial trajectory generated within a lower fidelity model
(Lambert solver)
Step 2
Initial guess refinement in higher fidelity model
(Circular restricted Three-Body Problem, (CR3BP))
Target position
Converged solution (step 2)
Lambert guess (step 1)
LLO
NRHO-to-LLO transfer
Abort maneuver
z [-]
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
0.95
1
1.05
x [-]
0.04
0.02
0
-0.02
-0.04
y [-]

Figure 2.9: Two-step, initial-guess-based optimization pipeline. Descent trajectory scenario utilized as an example on the right side of the figure.

2.5.2 Surrogate Modeling and Adaptive Sampling

The systematic exploration of a large search space rapidly becomes computationally expensive as the number of problem parameters increases. Additionally, any automatic exploration of the design space is further compromised by ill-conditioned initial guesses, which may affect the quality and the convergence robustness of an optimization algorithm. A promising strategy for reducing intensive computational costs is provided by surrogate models, also referred to as metamodels. Surrogate models are mathematical models that can be utilized to approximate various input/output relations describing the behavior of complex systems. In this sense, they can be interpreted as a data-driven abstraction of an original system (Fig. 2.10).

A plethora of methods exist for the construction of a metamodel, including support vector machines (SVMs) [25], radial basis function networks (RBFNs) [26], polynomial regression

[2] https://naif.jpl.nasa.gov/naif/data_generic.html

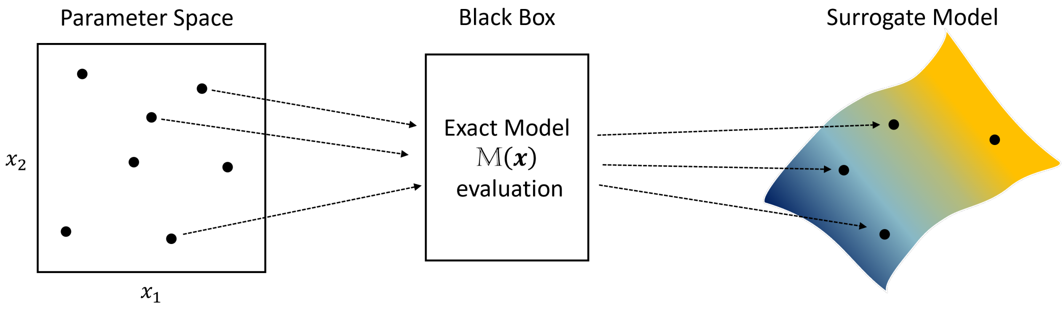


Figure 2.10: Schematic outlining the steps for generating surrogate models.

[27] and Kriging [28, 29]. In this investigation, we employ a class of the Kriging methodologies known as ordinary Kriging (OK) [30] as a metamodeling technique.

Ordinary Kriging. OK aims to approximate an exact mapping, $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$, between an input, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$, and a univariate output, $y \in \mathcal{Y} \subset \mathbb{R}$, as the mean of a stochastic process [31]:

$$Y(\mathbf{x}) = \mu + Z(\mathbf{x}) \tag{2.54}$$

where μ here represents a global mean contribution, while $Z(\mathbf{x})$ is a stationary Gaussian process. The latter, can be further expanded as:

$$Z(\mathbf{x}) = W(\mathbf{x}) + \eta(\mathbf{x}) + \epsilon(\mathbf{x}) \tag{2.55}$$

with $W(\mathbf{x})$ and $\eta(\mathbf{x})$ accounting for small-scale and micro-scale variations respectively, while $\epsilon(\mathbf{x})$ represents a noise term. The idea of OK is then to obtain a metamodel, $\hat{\mathcal{M}}$, that represents the most accurate approximation of an exact model, $\mathcal{M}$, for any point $\mathbf{x^0} \in \mathcal{X}$, as the mean of the realization of the stochastic process defined in Eq. (2.54) at $\mathbf{x^0}$, i.e.:

$$\mathcal{M}(\mathbf{x^0}) \approx \hat{\mathcal{M}}(\mathbf{x^0}) = \mu_{\hat{Y}(\mathbf{x^0})} = \hat{\mu} + \mathbf{r}_0^T \mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{2.56}$$

In Eq. (2.56), $\hat{\mu}$ represents the a priori estimate of the global mean (which can be obtained via a least-square estimate), $\mathbf{r}_0$ and $\mathbf{R}$ are the cross-correlation and autocorrelation vector and matrix respectively, $\mathbf{y}$ is a vector of true observations and $\mathbf{1}$ is a vector of ones. In particular, $\mathbf{r}_0$

and $\mathbf{R}$ are dependent on a n-dimensional parameter vector, $\boldsymbol{\theta}$, with n being the dimensionality of the input space, whose components are obtained via the solution of an optimization problem [32] based on the Maximum Likelihood Estimation (MLE):

$$\Psi = \frac{1}{2}\left\{m \ln\left[\frac{1}{m}\left(\mathbf{y} - \mathbf{1}\hat{\mu}\right)^T \mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{1}\hat{\mu}\right)\right] + \ln\left(|\mathbf{R}|\right)\right\} \tag{2.57}$$

with m being the current number of samples. In other words, the correlation parameters are optimized such they maximize the likelihood that the process described by the model produced the data that were actually observed. Notably, OK provides information about the variance of the model, $\sigma^2_{\hat{Y}(\mathbf{x^0})}$, which can be exploited to assess the quality of the model during the training process, here intended as the model's parameters adjustments as new points are sampled.

Adaptive Sampling. Prediction quality of surrogate models strongly depends on the size and distribution of the training points. In particular, an effective metamodel should be obtained from a reduced number of newly sampled points, which are added to an initially established set of evaluation points. In fact, as noted in the previous paragraph, the generation of a surrogate model necessitates a vector of true observations, meaning that the exact model must be evaluated. Hence, exploitation of surrogate models becomes appealing solely when the cost of generating the surrogate model itself positively impact the total simulation cost. Typically, exact knowledge of the input/output mapping of a given system may be unavailable; therefore, the identification of the distribution of the new sample points becomes a challenging task. A simple heuristic is provided by one-shot sampling techniques, which rely on sampling at once the points over the parameter space with an even distribution, aiming to capture the general behavior of the output surface. However, such a technique may miss relevant design surface information, making the filling technique inefficient. To mitigate this issue, a smart methodology is required to rapidly identify points of interest in the design space. One such methodology is provided by adaptive sampling techniques [33], belonging to the broader class of sampling strategies known as sequential sampling, which aim at identifying points of interest in an iterative manner based on previous predictions (i.e., on the current quality of the surrogate model). In such a way, the number of sampled points remains contained, while ensuring a sufficiently proficient metamodel.

Several adaptive sampling algorithms exist, each featuring either a local exploitation or a global exploration component, or a combination thereof. Examples of such techniques include [33] Monte Carlo-Intersite-proj-th (MIPT), Accumulative Error (ACE), Expected Improvement (EI) and Expected Improvement for Global Fit (EIGF).

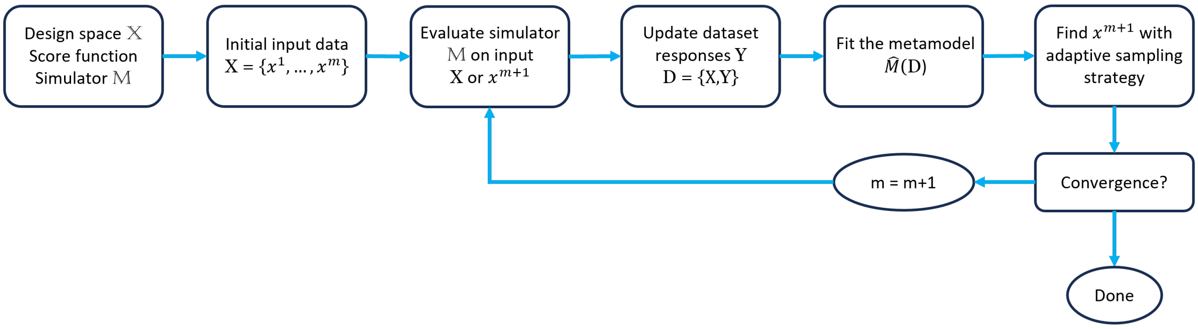


Figure 2.11: Schematic illustration of the training process of a metamodel leveraging an adaptive sampling technique.

Fig. 2.11 depicts the general procedure to train a surrogate model via adaptive sampling techniques. To begin with, the metamodel must be initialized. Toward this aim, for a given design space $\mathcal{X}$ and an exact model $\mathcal{M}$, an initial set of m-sample points $\mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^m\}$ is selected. As a general rule of thumb, as suggested in [34], the number of initial samples can be derived as m = 10n. Hence, the exact model is evaluated and its response is recorded, thus originating the initial training set $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$. The OK model is then initialized by fitting over such a training set. Next, a new point $\mathbf{x}^{m+1}$ is identified via an adaptive sampling technique, which selects a new point as the result of the optimization process:

$$\mathbf{x}^{m+1} = \arg\max_{\mathbf{x}^* \in \mathcal{X}} RC(\mathbf{x}^*) \tag{2.58}$$

where RC represents a “refinement criterion” reflecting the quality of the model. The training set $\mathcal{D}$ is then augmented with $\mathbf{x}^{m+1}$, and the process iterates until a user-defined stopping criterion is reached. Traditional choices include stopping when reaching a maximum number of adaptive points, criteria based on time constraints, or accuracy requirements.

2.5.3 Three-Step, Surrogate-Based Optimization Pipeline

While a surrogate model has a lower computational footprint, it cannot completely substitute an exact model due to accuracy limitations when used to predict technical measures of performance. This becomes especially true for systems characterized by a complex dynamic structure, where even small numerical variations can lead to large solution differences. Nonetheless, as the Kriging algorithm is trained over the higher-fidelity dynamics, it can be utilized as an educated and self-tuning guess generator for any downstream optimization. Initially, the downstream corrector or optimizer would rely on a hard-coded, lower-fidelity guess generator (like the Lambert solver employed in this work). Meanwhile, the surrogate model trains on the high-fidelity trajectory solutions generated by the pipeline. When a user-defined condition is met (either number of adaptive points or error on prediction), the surrogate model replaces the lower-fidelity guess generator, providing input for the subsequent correction or optimization process. When the surrogate model replaces the lower-fidelity guess generator, we expect an overall decrease in the number of iterations required to converge a solution within the high-fidelity model. Hence, the grid search is accelerated. Additionally, the surrogate model may be re-trained every M-steps to ensure acceleration performance remains consistent and the model auto-refines near catastrophic collapse conditions. Figure 2.12 provides a high-level flowchart of the proposed three-step, surrogate-based optimization pipeline. It is worth noting that the base version of the algorithm works for univariate predictions, whereas in our case multiple outputs are required. To achieve this goal, multiple surrogate models are trained in parallel.

2.5.4 Informed Search: Finite-Time Lyapunov Exponents

Additional insight into the underlying dynamics of a trajectory solution can be gained by leveraging flow-informed techniques. These techniques are traditionally popular in fluid dynamics, but an analogy can be constructed when considering an initial state under the influence of a complex gravitational field on a particle following the natural flow of a system. In particular, our flow-informed approach leverages a flow-based technique based on FTLEs, which offer a quantitative criterion to characterize the type of motion for a trajectory propagated over a finite time horizon

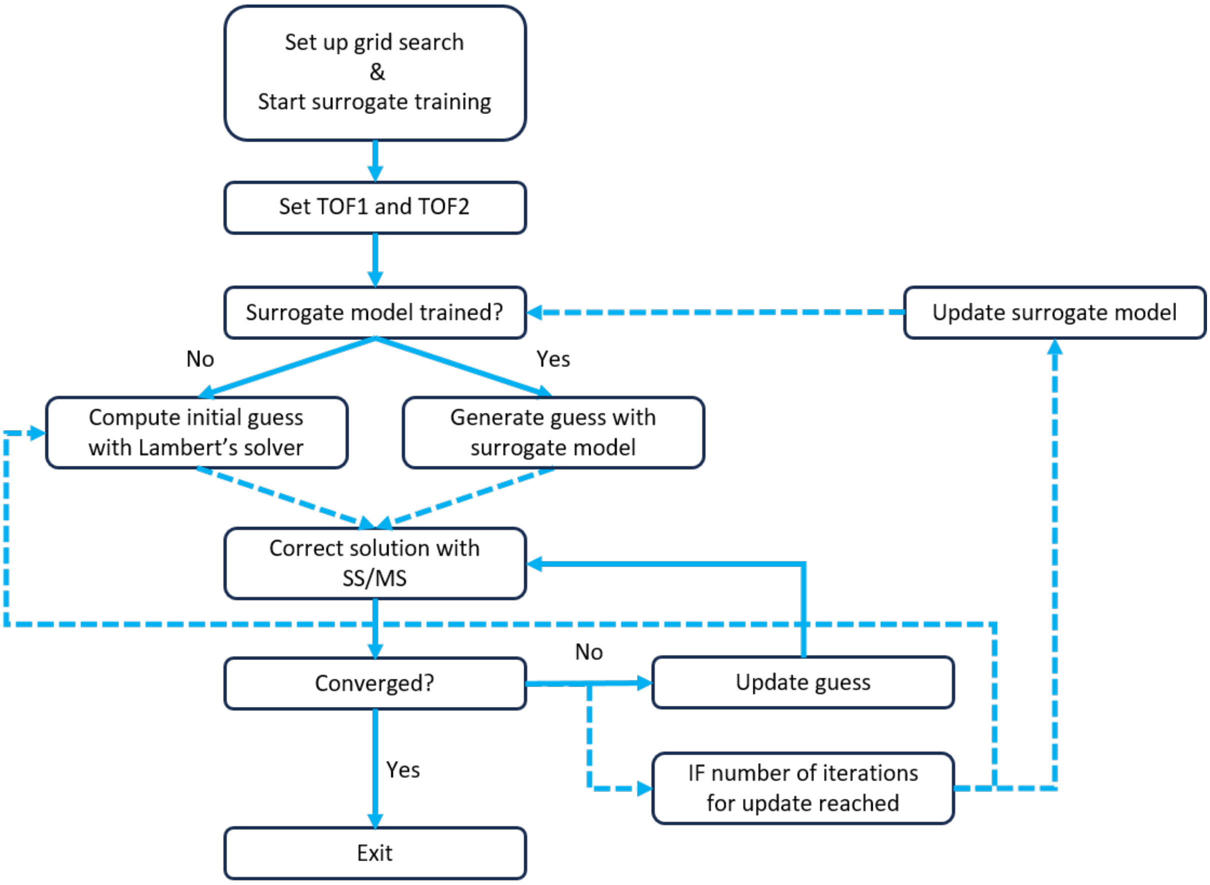

Set up grid search
&
Start surrogate training
Set TOF1 and TOF2
Surrogate model trained?
Update surrogate model
No
Yes
Compute initial guess with Lambert's solver
Generate guess with surrogate model
Correct solution with SS/MS
Converged?
No
Update guess
IF number of iterations for update reached
Yes
Exit

Figure 2.12: Grid search enhancement via surrogate modeling.

[35, 36]. In practice, their numerical value represents the sensitivity of trajectories originating from small perturbations applied to neighboring initial conditions. Projection of multiple FTLEs over a design section generates what are usually referred to as FTLE maps, which can be used to identify the existence of Lagrangian Coherent Structures (LCSs) [37]. LCSs define barriers in the dynamical flow, and can be exploited to qualitatively assess boundaries separating different types of motion. Knowledge of these dynamical structures can be instrumental for the trajectory design process, as they offer the opportunity to identify fuel- and/or time-efficient solutions. Specifically, FTLE maps typically display ridges separating regions characterized by different behaviors; such ridges represent larger deviations downstream a flow, thus possibly constituting favorable conditions for the placement of maneuvers. Computation of the FTLE values requires the derivation of the Cauchy-Green Strain Tensor (CGST), which is given by:

$$\text{CGST} = \Phi(t_f, t_0)^T \Phi(t_f, t_0), \tag{2.59}$$

with $\Phi(t_f, t_0)$ denoting the STM. Next, the FTLE can be directly computed as:

$$FTLE = \frac{1}{|t_f - t_0|} \ln \sqrt{\lambda_{\text{max}}(\text{CGST})}, \tag{2.60}$$

where λ_{max} denotes the maximum eigenvalue of the CGST, which is associated with the direction of maximum dynamical stretching, while the term $t_f - t_0$ indicates the time horizon and serves as a normalization factor. An example of FTLE map and associated classes of trajectories with a one day time horizon is provided in Fig. 2.13.

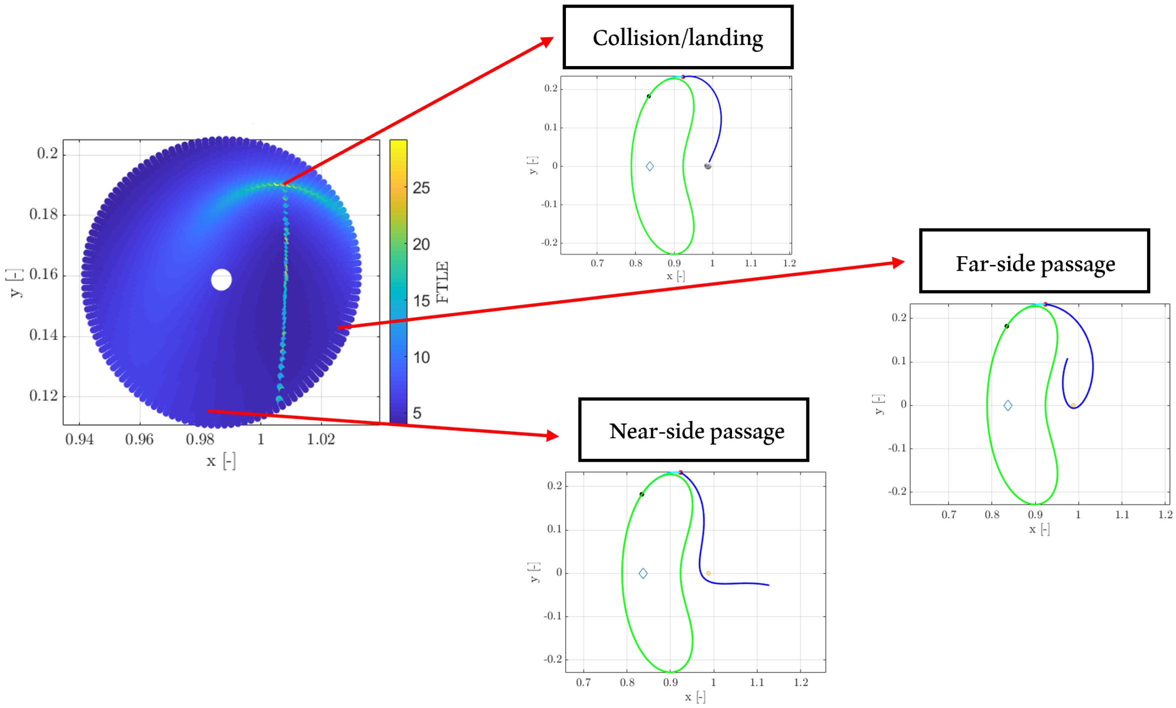


Figure 2.13: Examples of trajectory behaviors from different regions of the FTLE map.

Alternatively, the FTLE value can be computed as the ratio between the maximum singular value obtained through singular value decomposition (SVD) of the STM and the time horizon [35]. In fact, the STM can be decomposed through SVD as:

$$\Phi(t_f, t_0) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.61}$$

where $\mathbf{U}$ and $\mathbf{V}$ are two mutually orthogonal matrices, with the column of $\mathbf{U}$ expressing the direction of the stretching at t_f, while $\mathbf{\Sigma}$ is a diagonal matrix containing the eigenvalues representing the magnitude of the stretching in different direction in descending order:

$$\mathbf{\Sigma} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{2.62}$$

where $\lambda_1 > \lambda_2 > ... > \lambda_n$. A graphical representation is provided in Fig. 2.14.

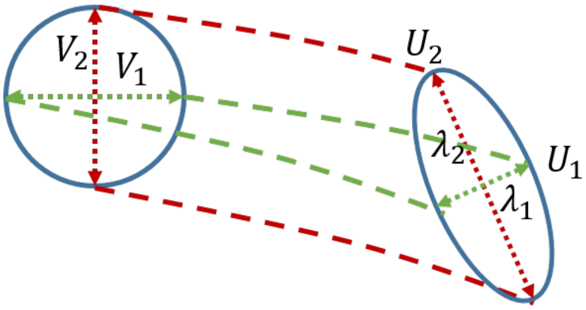


Figure 2.14: Visual representation of the stretching associated with principal eigenvalues.

Finally, as $\lambda_1 = \lambda_{max}$, the FTLE is obtained as:

$$FTLE = \frac{\lambda_1}{|t_f - t_0|} \tag{2.63}$$

2.6 Case Study I: Descent

In this section, the results associated with the descent abort scenario are presented and discussed. As described in Sec. 2.4, an abort maneuver is performed along a nominal trajectory departing from the apolune of the NRHO and targeting a 100 km altitude LLO, such that the HLS is commanded to rendezvous with the Gateway. The representative nominal NRHO-to-LLO transfer is depicted in Fig. 2.15, whereas the initial conditions for the transfer are reported in Tab. 2.2.

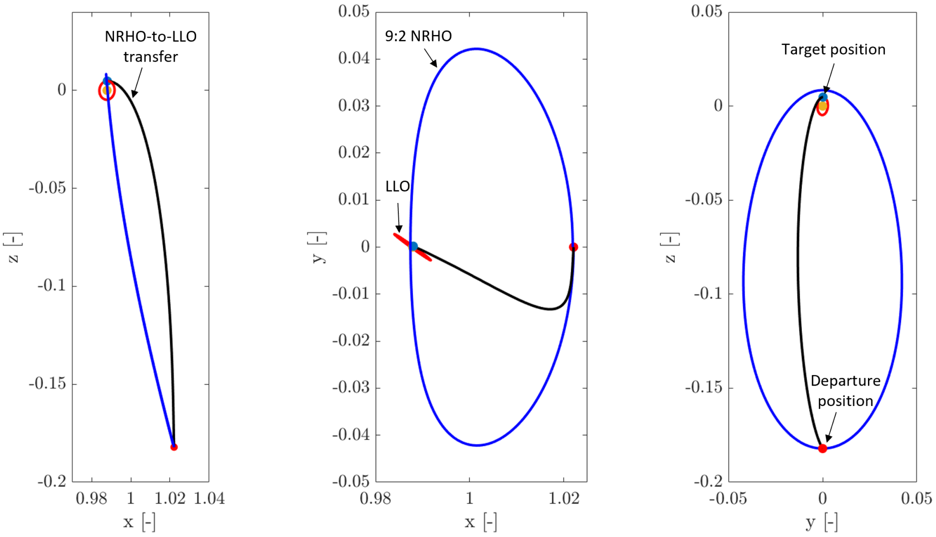


Figure 2.15: Representative NRHO-to-LLO transfer trajectory. Convergence tolerance set to 1e-10.

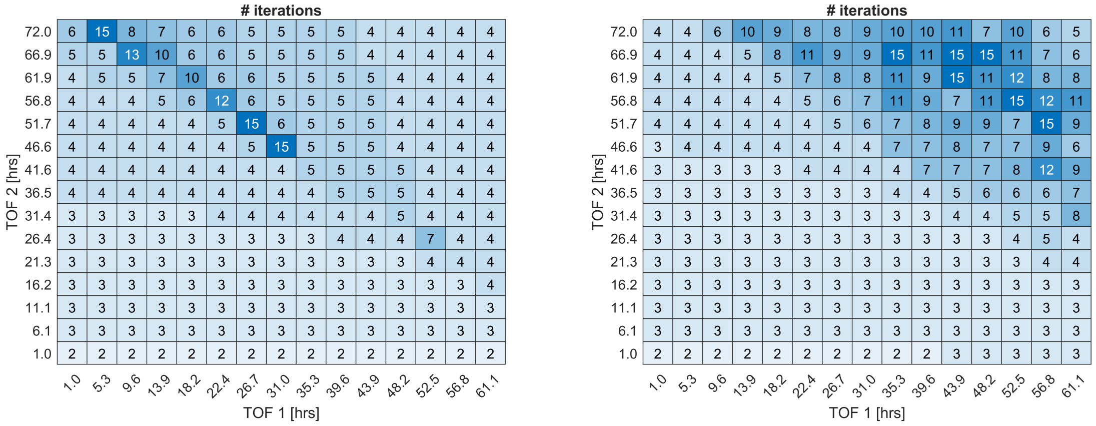
x [-]	y [-]	z [-]	$\dot{x}$ [-]	$\dot{y}$ [-]	$\dot{z}$ [-]	TOF [-]
1.0221	0	-0.1821	-0.0017	-0.0356	0.0455	0.6496

Table 2.2: Transfer trajectory initial conditions.

At first, the performance of a two-step initial-guess-based optimization pipeline is assessed, and the convergence structure of the abort scenario is analyzed. Next, we introduce a surrogate-based optimization pipeline to demonstrate computational cost reduction via a metamodel initial guess generator. Finally, the dynamical structure of the problem is investigated by leveraging FTLEs maps.

2.6.1 Convergence Structure

We employ a two-step optimization pipeline (Sec. 2.5) to identify a baseline computational cost structure within the direct search approach. Specifically, we define the computational structure as the number of iterations required by the implemented SS and MS targeting algorithms to correct the transfer guess provided by the lower-fidelity (Lambert) solver for different combinations of the search space parameters. Due to our grid search set up, the TOF2 design variable corresponds to the transfer time from to abort point to the rendezvous point. As such, both SS and MS are formulated through a fixed time formulation. Specifically, the single shooting algorithm targets the rendezvous state in position by operating on the initial velocity at the abort state. The multiple shooting is formulated in a similar fashion, though continuity on both position and velocity is enforced on the intermediate patch points.



TOF 2 [hrs]	1.0	5.3	9.6	13.9	18.2	22.4	26.7	31.0	35.3	39.6	43.9	48.2	52.5	56.8	61.1
72.0	6	15	8	7	6	6	5	5	5	5	4	4	4	4	4
66.9	5	5	13	10	6	6	5	5	5	5	4	4	4	4	4
61.9	4	5	5	7	10	6	6	5	5	5	5	4	4	4	4
56.8	4	4	4	5	6	12	6	5	5	5	5	4	4	4	4
51.7	4	4	4	4	4	5	15	6	5	5	5	4	4	4	4
46.6	4	4	4	4	4	4	5	15	5	5	5	4	4	4	4
41.6	4	4	4	4	4	4	4	4	5	5	5	5	4	4	4
36.5	4	4	4	4	4	4	4	4	4	5	5	5	4	4	4
31.4	3	3	3	3	3	4	4	4	4	4	4	5	4	4	4
26.4	3	3	3	3	3	3	3	3	3	4	4	4	7	4	4
21.3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4
16.2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4
11.1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
6.1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1.0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

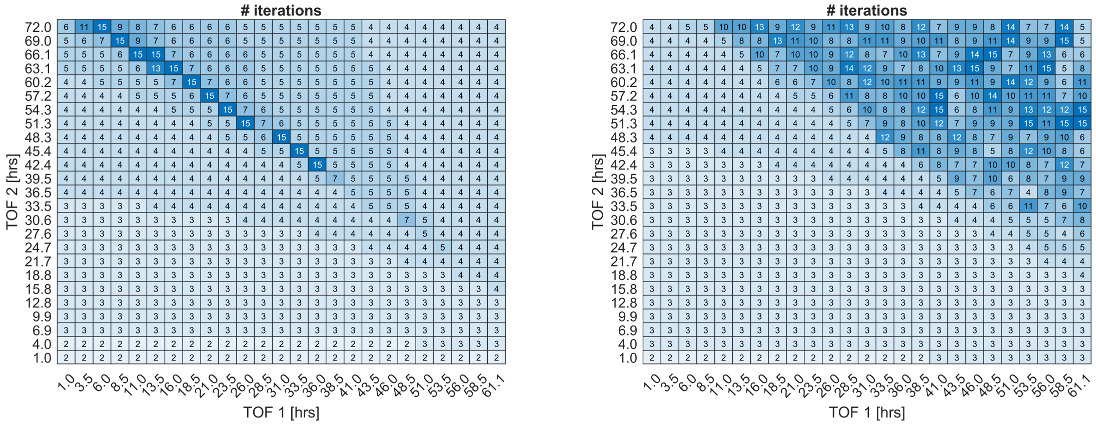
(a)

TOF 2 [hrs]	1.0	5.3	9.6	13.9	18.2	22.4	26.7	31.0	35.3	39.6	43.9	48.2	52.5	56.8	61.1
72.0	4	4	6	10	9	8	8	9	10	10	11	7	10	6	5
66.9	4	4	4	5	8	11	9	9	15	11	15	15	11	7	6
61.9	4	4	4	4	5	7	8	8	11	9	15	11	12	8	8
56.8	4	4	4	4	4	5	6	7	11	9	7	11	15	12	11
51.7	4	4	4	4	4	4	5	6	7	8	9	9	7	15	9
46.6	3	4	4	4	4	4	4	4	7	7	8	7	7	9	6
41.6	3	3	3	3	3	4	4	4	4	7	7	7	8	12	9
36.5	3	3	3	3	3	3	3	3	4	4	5	6	6	6	7
31.4	3	3	3	3	3	3	3	3	3	3	4	4	5	5	8
26.4	3	3	3	3	3	3	3	3	3	3	3	3	4	5	4
21.3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4
16.2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
11.1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
6.1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1.0	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3

(b)

Figure 2.16: (a) Single shooting and (b) multiple shooting comparison. Grid resolution 15×15.

Figs. 2.16 and 2.17 show the number of iterations required for the SS and MS to converge to a solution starting from an initial guess provided by the Lambert solver at two different resolutions of the grid search (15×15 and 25×25). For both cases, the maximum number of iterations is set to 15, which is deemed to be a sufficiently high threshold to allow the differential correction schemes to converge to a solution. For the cases where the maximum number of iterations is reached without a feasible solution, we tested that the SS algorithm fails to converge even with 500 iterations, while the MS provides a solution within 50 iterations. Note how the convergence structure, which is rendered by the number of iterations as a function



(a)

(b)

Figure 2.17: (a) Single shooting and (b) multiple shooting comparison. Grid resolution 25×25.

of TOF1-TOF2 combinations, includes highly nonlinear regions. In the proximity of certain combinations of search space parameters, there exists a sudden and catastrophic collapse of the trajectory design pipeline, where the pipeline fails to converge to a solution, or the computational cost exponentially increases with respect to neighbouring cases. These regions of collapse are likely correlated to an ill-conditioned initial guess, physical problem geometry, and numerical sensitivity of the optimization algorithm. An example is provided in Fig. 2.18, where the iterative steps associated with a non-converging solution are displayed. In particular, one can observe how the algorithm fails to reduce the error below the imposed threshold, oscillating around a constant value beyond 10 iterations.

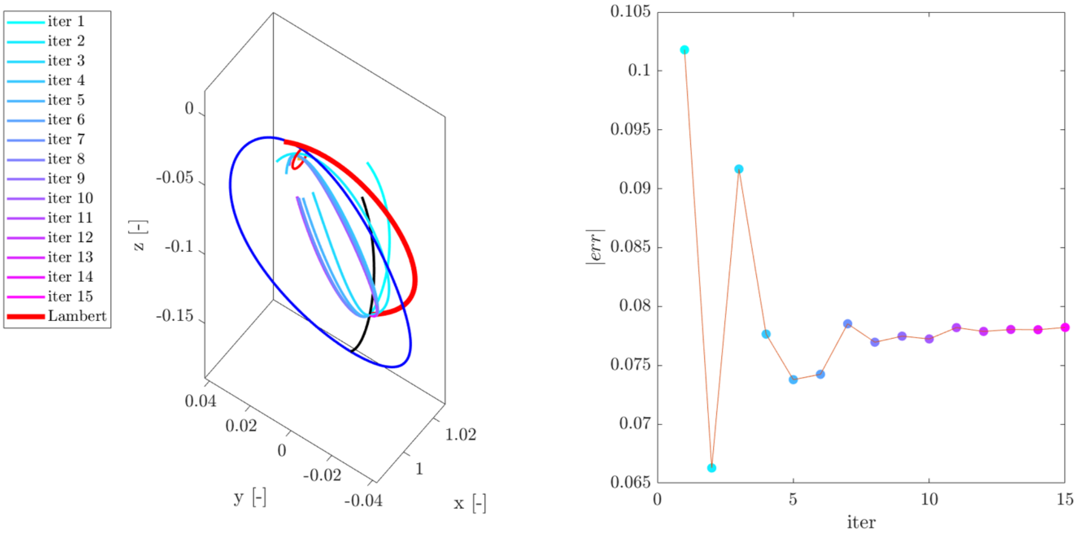


Figure 2.18: Non-converging solution insight.

The manual inspection of failure regions is challenging in a resource- and time-constrained design environment, and represents a barrier to the full automation of large-scale trajectory analysis pipelines. In fact, ad-hoc solution could be implemented to resolve some of the most sensitive cases, though their difference in nature would not allow for an adequate and unique comprehensive solution. Furthermore, as evident in Figs. 2.16 and 2.17, despite the global trend is generally maintained, catastrophic collapse regions shift and deform as a function of the downstream correction algorithms and grid resolution, rendering heuristic solutions brittle to small changes in the pipeline.

2.6.2 Surrogate-based Optimization Pipeline

We preliminarily test the three-step pipeline under simplified conditions as a proof of concept. In our descent abort scenario example, the initial guess to the correction algorithms is provided by the solution of a Lambert problem; however, as such a solution is available in two-body dynamics, it can be ill-conditioned, especially when the dynamics become more sensitive. In particular, we seek to utilize the surrogate model to generate an initial guess for the correction schemes in substitution to the Lambert's guess. Instead of directly building the framework depicted in Fig. 2.12, we first train a surrogate model offline. Next, we run a grid search analysis similar to those represented in Fig. 2.16, this time using the guess generated by the surrogate model in substitution to the Lambert solver.

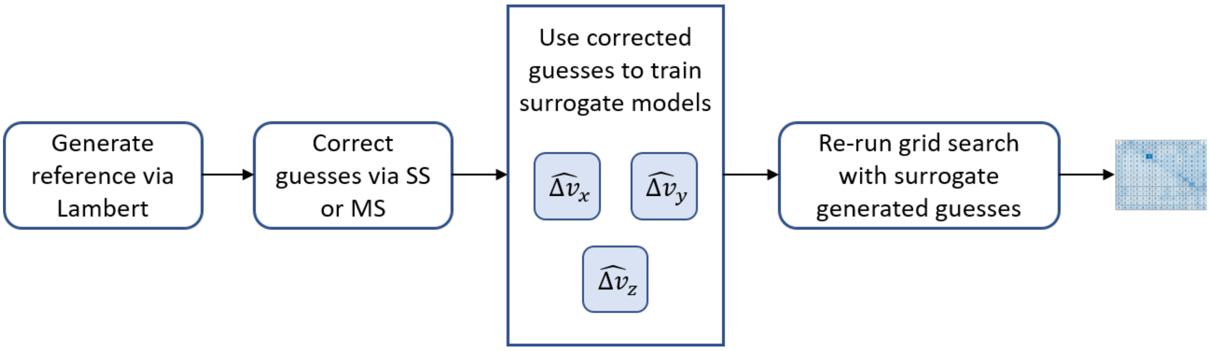

Generate reference via Lambert
Correct guesses via SS or MS
Use corrected guesses to train surrogate models
$\widehat{\Delta v_x}$
$\widehat{\Delta v_y}$
$\widehat{\Delta v_z}$
Re-run grid search with surrogate generated guesses

Figure 2.19: Offline pipeline testing scheme.

In this example, our goal is to predict the Δv vector components at the abort point. As discussed in Sec. 2.5, the OK methodology applies to cases with univariate output, that is, a single quantity is predicted. However, in this study multiple quantities corresponding to the components of the Δv vector are required. To bridge the difference with the standard implementation, we explore three options, each assuming a common initial set of sample points:

1. Independent training: each model predicts a new point of interest, and it is added to its own sample points set. This is equivalent to simply running multiple metamodel training sessions in parallel.

2. Semi-independent training: each model predicts a new point, but all the predicted points are added to each model initial sample set.

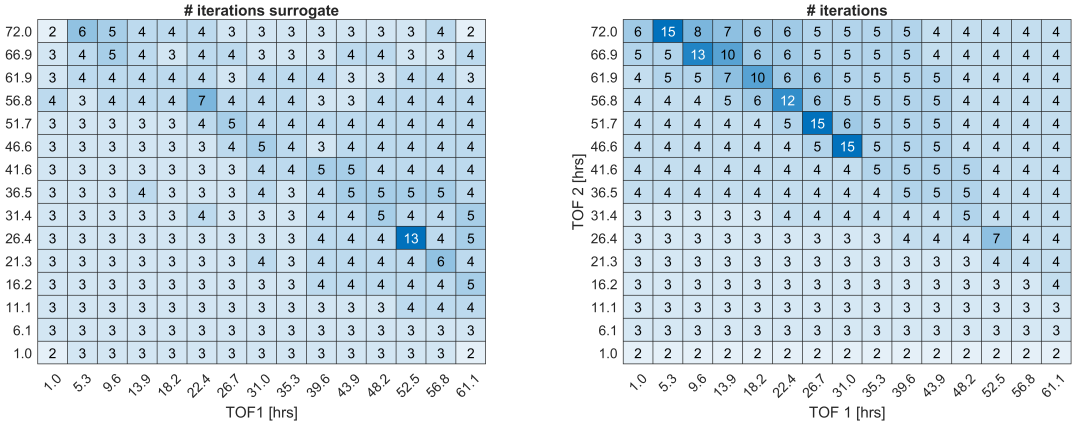
3. Shared training: this case follows a more traditional [38] implementation of a multiple-output Kriging metamodel. Each model is trained using k-adaptive sampling criterion, with each model generating k-points. Next, the "Technique for Order of Preference by Similarity to Ideal Solution" (TOPSIS) is utilized to select the set of points to be added to all the models. Following the implementation in [38], "Maximum and Minimum over the Expected Improvement" and "Mean Square Error" are adopted as adaptive sampling strategies.

Each model is initialized with 40 points for all the options, and a total of 60 points is added.

Option	**Grid Size**	**Lambert guess**	**Surrogate guess**	**% improvement**
1	15x15	932	899.6	**+3.48**
2			861.1	**+7.61**
3			892.3	**+4.26**
1	25x25	2577	2461.7	**+4.47**
2			2435.9	**+5.48**
3			2446.6	**+5.06**
1	35x35	5078	4851.6	**+4.46**
2			4763.2	**+6.20**
3			4789.2	**+5.69**
1	50x50	10362	9914.1	**+4.32**
2			9792.1	**+5.50**
3			9788.5	**+5.53**

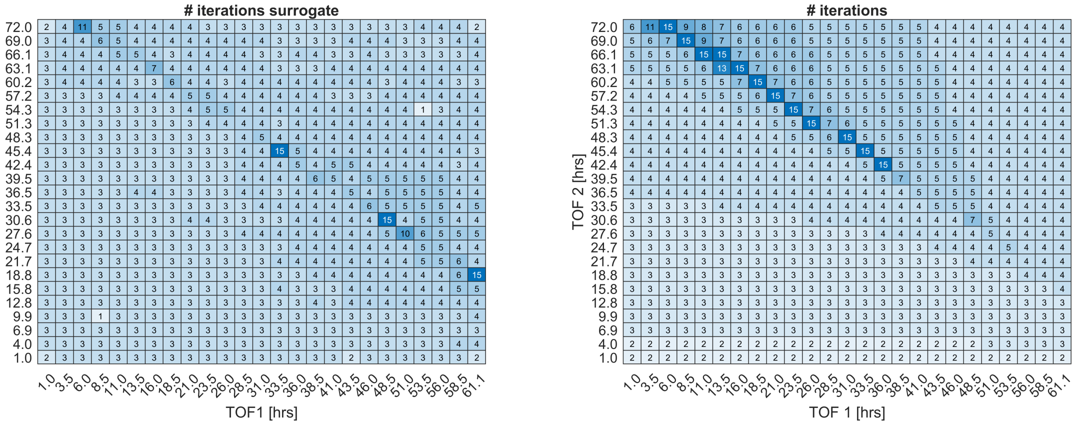
Table 2.3: Total number of iterations required for the SS to converge.

Tab. 2.3 summarizes the results for our preliminary test, using the total number of iterations in the entire grid search as global metric. As the training of the models includes a stochastic component, ten runs are conducted for each case, taking the average number of iterations achieved within each run to quantify the performance. Interestingly, for all the cases, the surrogate models perform better than the Lambert-based guess generator at different granularity of the search space. This could indicate that a model can be ideally trained with the addition of tens of sample points while remaining proficient on a much larger scale.



(a) **(b)**

Figure 2.20: Number of iterations required for the SS to converge, utilizing the surrogate model (a) and the Lambert solver (b) for generating the initial guess. Grid resolution 15×15.



(a) **(b)**

Figure 2.21: Number of iterations required for the SS to converge, utilizing the surrogate model (a) and the Lambert solver (b) for generating the initial guess. Grid resolution 25×25.

Figs. 2.20 and 2.21 report an example of the number of iterations required for the SS to converge to a solution starting from an initial guess provided by the surrogate models (left) and by the Lambert solver (right), for two different grid resolutions. The surrogate model performs substantially better than the Lambert solver as an initial guess generator, providing initial conditions almost always leading to a converged solution within the iterations boundary. In this particular case, the total iteration count associated with the surrogate model case amounts to 778 and 2220, respectively. Notably, an ideal case would have a total count corresponding to the number of cells in the grid (here being 225 and 625). As such, there exists further room for improvement, potentially achievable via an increment of adaptive points or a refinement of the method.

2.6.3 Dynamical Structure Analysis: FTLE Maps

Due to the complexity of the convergence structure of the problem (Sec. 2.6.1), a direct search approach is not always guaranteed to provide a feasible solution (within the problem and constraints formulation). Furthermore, we highlighted how this approach provides little to no insight regarding the nature of converged/non-converged trajectories. To mitigate these issues, an understanding of the environment's natural dynamics may offer multiple benefits. To this purpose, inspired by previous studies [39, 17], we employ a flow-informed search approach based on the exploitation of FTLE maps to explore the dynamical structure of the environment. In fact, FTLE maps can be informative of the type of motion characterizing trajectories in complex dynamics, thus aiding the trajectory design process.

FTLE maps are generated here for the HLS abort descent scenario defined in Sec. 2.4. Four separate segments are considered:

1. On the first segment, the HLS travels along the descent trajectory till an abort state.
2. On the second segment, a maneuver is performed to initiate the rendezvous with the DSG.
3. On the third segment, a second maneuver is performed to inject the module on a transfer trajectory toward the NRHO.
4. One last maneuver is performed for the injection on the terminal orbit (rendezvous point).

The time of flight after the abort (TOF2), the Δv magnitude for the abort maneuver, and the direction of the Δv are used as design variables to generate the FTLE maps. Regarding the second and third segments, we propagate the dynamics from the abort point forward in time and from the rendezvous point backward in time for a time horizon of TOF2/2. A schematic representation of this four-step process is shown in Fig. 2.22.

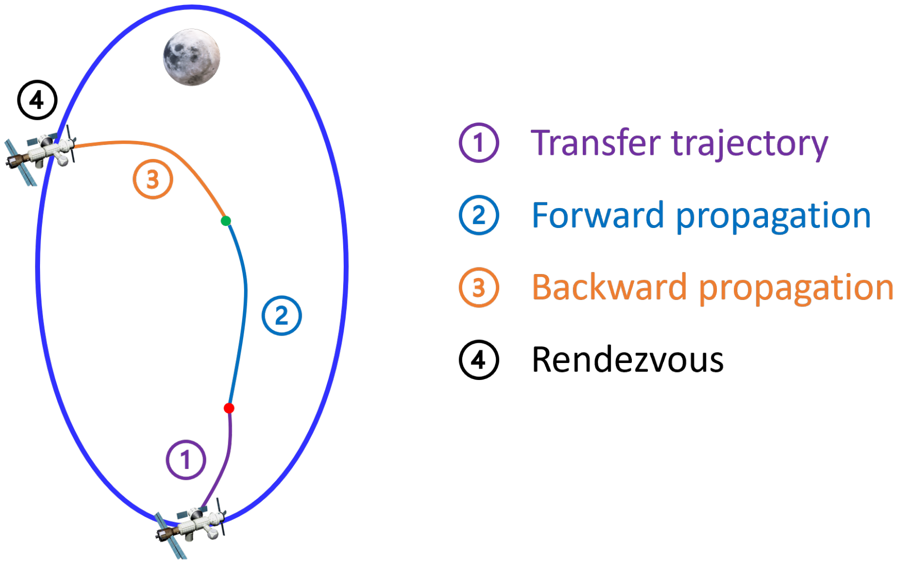


Figure 2.22: Schematic approach representation for the generation of FTLE maps.

As the goal is to characterize the terminal conditions obtained through the forward and backward propagation in terms of the FTLE values, such conditions are propagated for an arbitrarily long time horizon, here set to two weeks. In such a way, we can evaluate the sensitivity to small perturbations when applied to the terminal states obtained after the TOF2/2 forward and backward propagation. While this procedure slightly differs from the set-up of the direct search approach (which assumes a direct abort-point-to-NRHO transfer without intermediate maneuvers), it allows us to gather meaningful information on the general dynamics, potentially including (or not) the trajectories identified through our representative direct search approach.

Figs. 2.23 to 2.25 show the trajectory propagated from the same abort point for three different Δv values, with central and right plots showing the terminal points of such trajectories. For this case, a TOF1-TOF2 combination corresponding to a condition of non-convergence on the SS 15 $\times$ 15 grid is considered (see Fig. 2.16). One can observe how, as expected, higher Δv magnitudes generate a larger deviation downstream of the propagated trajectories, with the

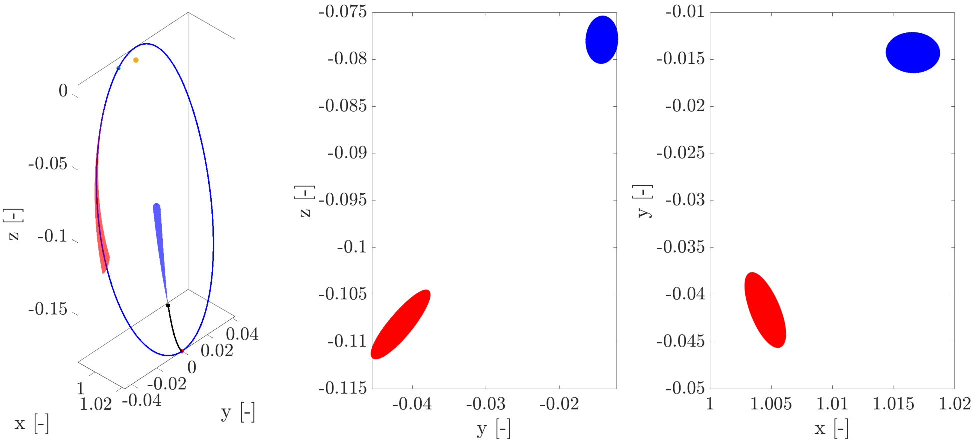


Figure 2.23: Dynamics propagation example. Δv = 10 m/s.

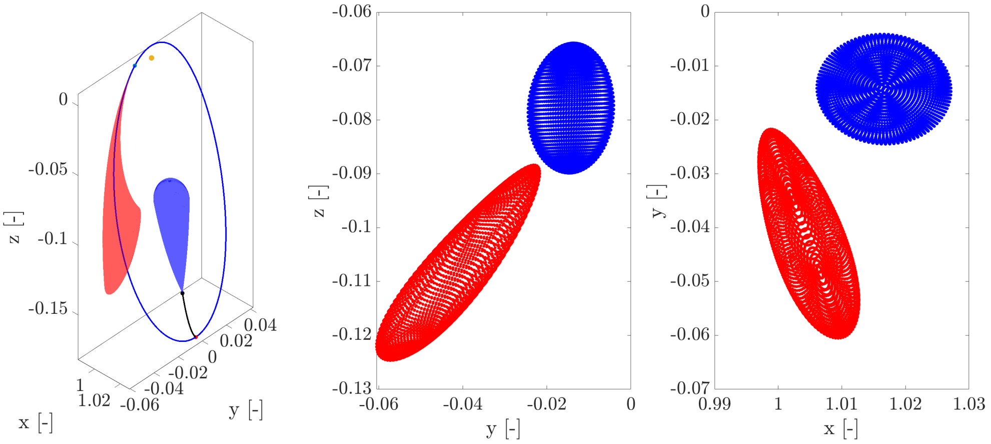


Figure 2.24: Dynamics propagation example. Δv = 50 m/s.

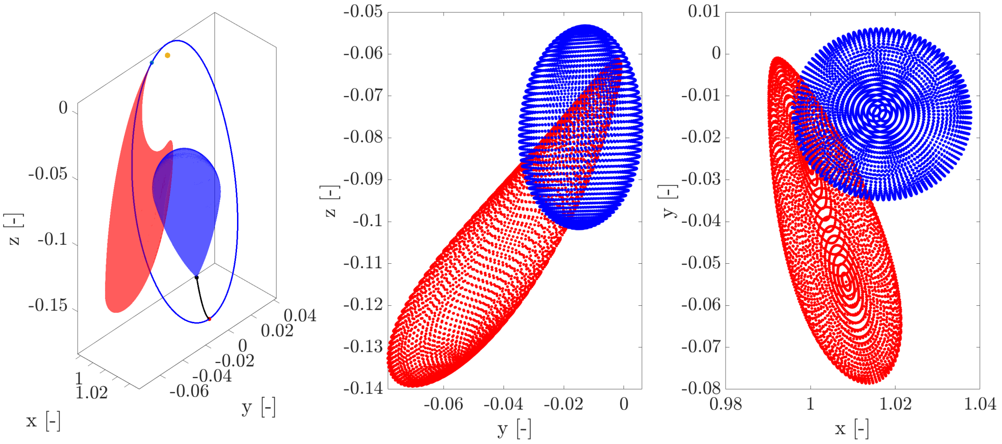


Figure 2.25: Dynamics propagation example. Δv = 100 m/s.

emergence of regions of recovery trajectories at the intersection of the forward and backward flows. These terminal regions can then be characterized through the associated FTLE values.

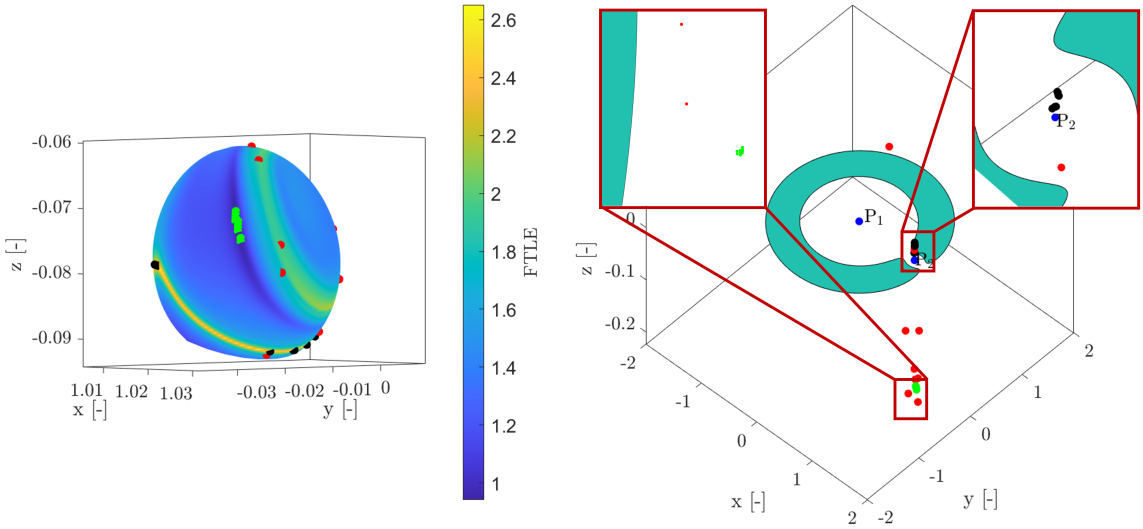


Figure 2.26: FTLE values to trajectory mapping. $\Delta v = 70$ m/s, 2 weeks propagation. Dots in the right picture represent the terminal condition of individual trajectory after propagation. Section of the zero velocity surface with Jacobi constant level $C = -3.16$.

Fig. 2.26 reports the FTLE forward map coloring of the terminal surface associated with a case assuming a $\Delta v = 70$ m/s is applied at the abort point while considering the same TOF1-TOF2 combination. Distinct dynamical behaviors can be identified considering different regions of the map. For this particular example, we considered trajectories originating from states characterized by the highest FTLE values (left picture, black dots), lowest FTLE values (left picture, green dots), and randomly picked intermediate values (left picture, red dots). Propagation of such conditions reveals how higher FTLE values are associated with trajectories persisting in the proximity of the Moon (within the considered two-week limit), while those associated with lower FTLE values (right picture, green cluster) are escaping.

A similar behavior is observed in multiple cases. For example, Fig. 2.27 reports a case assuming a $\Delta v = 50$ m/s is applied at the abort point while the forward propagation is extended to four weeks. Similarly to the previous case, trajectories associated with minimum and maximum FTLE values are associated with similar behaviors, with trajectories corresponding to lower values rapidly moving away from the system, while those corresponding to higher values persist (though with more dispersion, as expected due to the extended propagation time) in the proximity of the Moon.

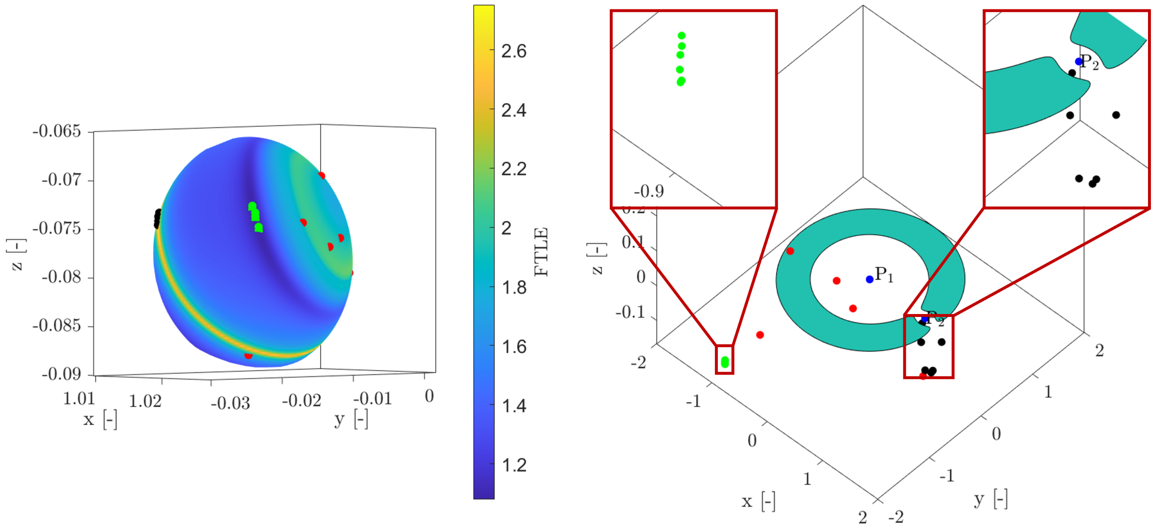


Figure 2.27: FTLE values to trajectory mapping. $\Delta v = 50$ m/s, 4 weeks propagation. Dots in the right picture represent the terminal condition of individual trajectory after propagation. Section of the zero velocity surface with Jacobi constant level $C = -3.16$.

Analogous considerations seem to apply when considering a different TOF1-TOF2 combination, this time selected in correspondence to a convergence condition. In particular, Figs. 2.28 and 2.29 report the FTLE forward map coloring when a Δv of 30 m/s and 60 m/s is applied, with a propagation time of four and two weeks, respectively. As for the previous examples, trajectories associated with the highest FTLE values remain close to the Moon, while those corresponding to intermediate and lowest values display much more dispersion, moving far from the Moon. Knowledge of this information may then become instrumental for trajectory planning. We speculate that performing a maneuver associated with states corresponding to higher FTLE values may demand a lower propellant consumption while ensuring the spacecraft remains in the proximity of the NRHO, while a maneuver placed in correspondence with lower values may necessitate a higher change of energy to keep the spacecraft closer to the DSG.

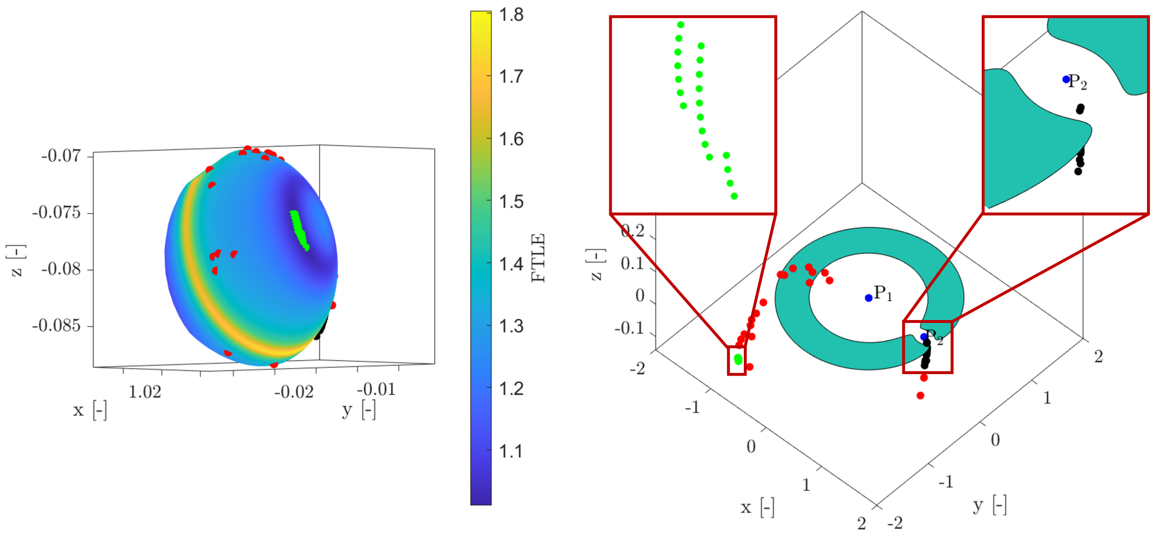


Figure 2.28: FTLE values to trajectory mapping. $\Delta v = 30$ m/s, 4 weeks propagation. Dots in the right picture represent the terminal condition of individual trajectory after propagation. Section of the zero velocity surface with Jacobi constant level $C = -3.16$.

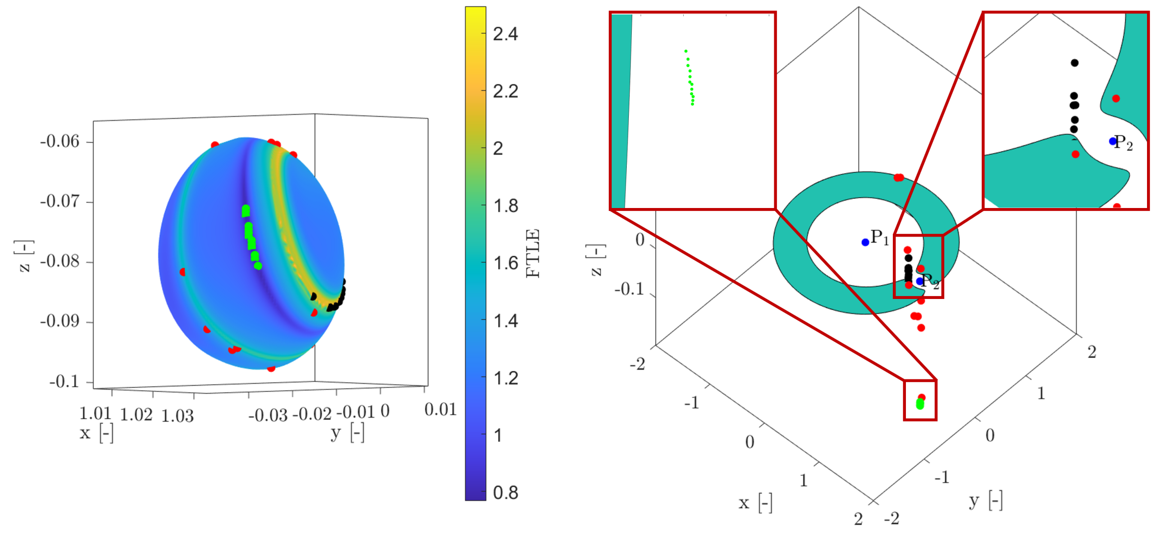


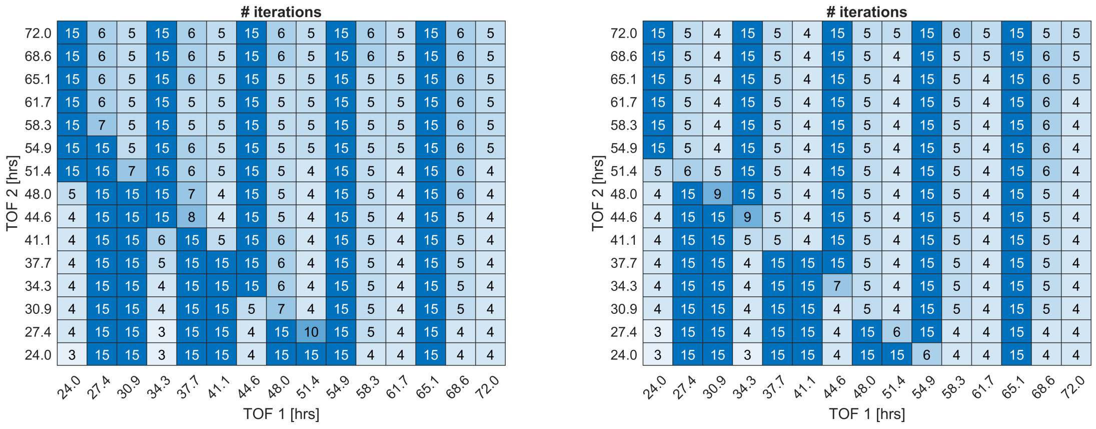
Figure 2.29: FTLE values to trajectory mapping. $\Delta v = 60$ m/s, 2 weeks propagation. Dots in the right picture represent the terminal condition of individual trajectory after propagation. Section of the zero velocity surface with Jacobi constant level $C = -3.16$.

2.7 Case Study II: Ascent

In this section, the results associated with the ascent abort scenario are thoroughly described and analyzed. As mentioned in Sec. 2.4, it is assumed the ascent module must return to the Gateway ahead of the nominal mission duration. A circular LLO of 150 km of altitude is considered in this scenario as intermediate orbit for the Moon-to-Gateway transfer. To begin with, the results concerning our two-step optimization pipeline are presented, followed by a discussion on the performance of the surrogate-based model. Next, an in-depth analysis of the design landscape is presented, providing additional insight to the dynamics of the problem and culminating into a series of considerations and recommendation to aid the trajectory design process.

2.7.1 Convergence Structure

Analogously to the descent case study, a two-step optimization pipeline is employed to gain an understanding of the convergence structure of the problem. Throughout the analysis, a lower and upper bound of 1 day and 3 days respectively is considered for both TOF1 and TOF2, thus providing a broader view of the resulting surface. Again, the maximum number of iterations is set to 15, with termination conditions on runtime and minimum altitude throughout the path.



iterations

	24.0	27.4	30.9	34.3	37.7	41.1	44.6	48.0	51.4	54.9	58.3	61.7	65.1	68.6	72.0
72.0	15	6	5	15	6	5	15	6	5	15	6	5	15	6	5
68.6	15	6	5	15	6	5	15	6	5	15	6	5	15	6	5
65.1	15	6	5	15	6	5	15	5	5	15	5	5	15	6	5
61.7	15	6	5	15	5	5	15	5	5	15	5	5	15	6	5
58.3	15	7	5	15	5	5	15	5	5	15	5	5	15	6	5
54.9	15	15	5	15	6	5	15	5	5	15	5	5	15	6	5
51.4	15	15	7	15	6	5	15	5	4	15	5	4	15	6	4
48.0	5	15	15	15	7	4	15	5	4	15	5	4	15	6	4
44.6	4	15	15	15	8	4	15	5	4	15	5	4	15	5	4
41.1	4	15	15	6	15	5	15	6	4	15	5	4	15	5	4
37.7	4	15	15	5	15	15	15	6	4	15	5	4	15	5	4
34.3	4	15	15	4	15	15	15	6	4	15	5	4	15	5	4
30.9	4	15	15	4	15	15	5	7	4	15	5	4	15	5	4
27.4	4	15	15	3	15	15	4	15	10	15	5	4	15	4	4
24.0	3	15	15	3	15	15	4	15	15	15	4	4	15	4	4

(a)

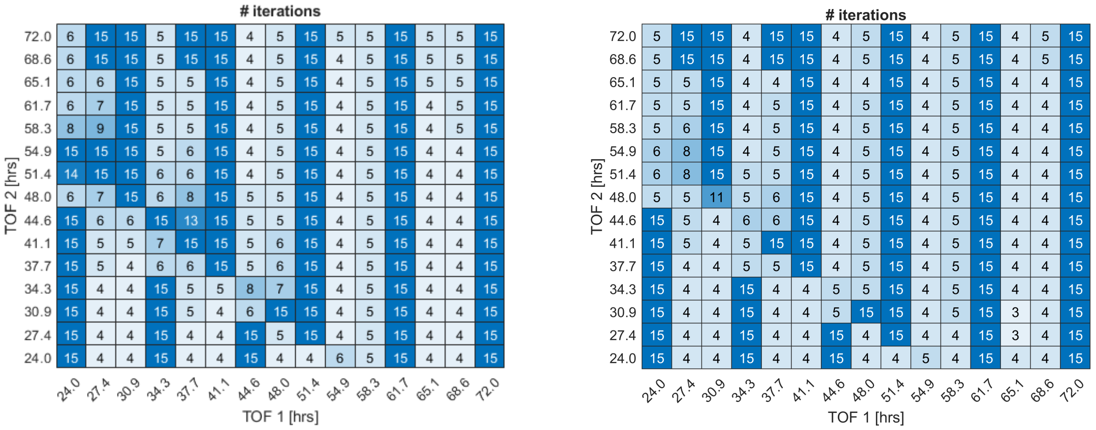
iterations

	24.0	27.4	30.9	34.3	37.7	41.1	44.6	48.0	51.4	54.9	58.3	61.7	65.1	68.6	72.0
72.0	15	5	4	15	5	4	15	5	5	15	6	5	15	5	5
68.6	15	5	4	15	5	4	15	5	4	15	5	5	15	6	5
65.1	15	5	4	15	5	4	15	5	4	15	5	4	15	6	5
61.7	15	5	4	15	5	4	15	5	4	15	5	4	15	6	4
58.3	15	5	4	15	5	4	15	5	4	15	5	4	15	6	4
54.9	15	5	4	15	5	4	15	5	4	15	5	4	15	6	4
51.4	5	6	5	15	5	4	15	5	4	15	5	4	15	6	4
48.0	4	15	9	15	5	4	15	5	4	15	5	4	15	5	4
44.6	4	15	15	9	5	4	15	5	4	15	5	4	15	5	4
41.1	4	15	15	5	5	4	15	5	4	15	5	4	15	5	4
37.7	4	15	15	4	15	15	15	5	4	15	5	4	15	5	4
34.3	4	15	15	4	15	15	7	5	4	15	5	4	15	5	4
30.9	4	15	15	4	15	15	4	5	4	15	5	4	15	5	4
27.4	3	15	15	4	15	15	4	15	6	15	4	4	15	4	4
24.0	3	15	15	3	15	15	4	15	15	6	4	4	15	4	4

(b)

Figure 2.30: (a) Single shooting and (b) multiple shooting convergence comparison on a 15×15 grid resolution, employing a *prograde* Lambert setting. LLO RAAN fixed to 0°.

Figs. 2.30 and 2.31 display the iterations required by the SS and the MS to correct the guess provided by the Lambert solver on a 15×15 grid resolution, assuming the LLO to have

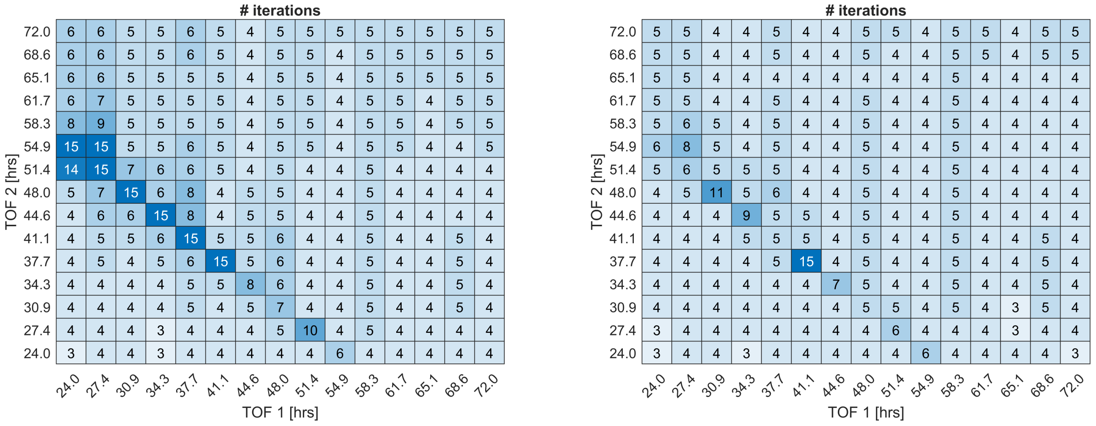


(a)

(b)

Figure 2.31: (a) Single shooting and (b) multiple shooting convergence comparison on a 15×15 grid resolution, employing a *retrograde* Lambert setting. LLO RAAN fixed to 0°.

a right ascension of 0°. Similarly to what observed in the descent case study, regions of stiff convergence appear on the grid, shifting and mutating based on the simulation settings and downstream correction scheme. Furthermore, acknowledging the multiple shooting to be able to resolve a few more cases with respect to the single shooting scheme, one can notice how prograde and retrograde convergence structure are (almost) complementary, indicating how particular combinations of time of flight can provide a solution solely for certain settings. This can be visualized in Fig. 2.32, which portrays the iterations map obtained via both prograde and retrograde transfer types. Of note, if both typologies provide a converged solution, the cheapest one in terms of departure Δv is retained.



(a)

(b)

Figure 2.32: (a) Single shooting and (b) multiple shooting convergence comparison on a 15×15 grid resolution, employing a *mixed* Lambert setting. LLO RAAN fixed to 0°.

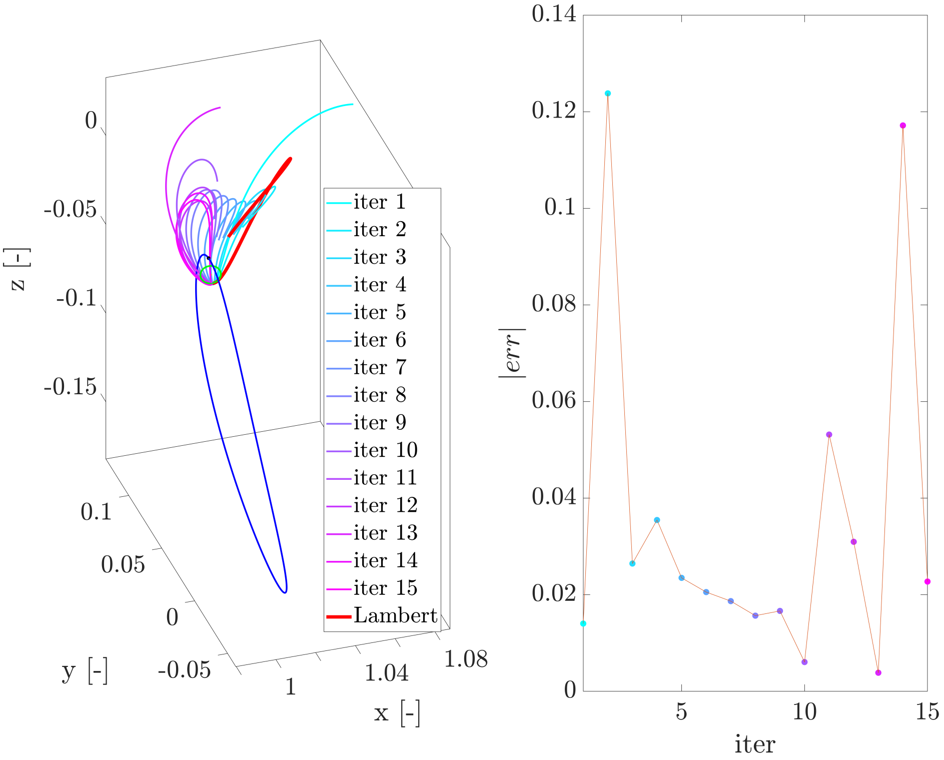
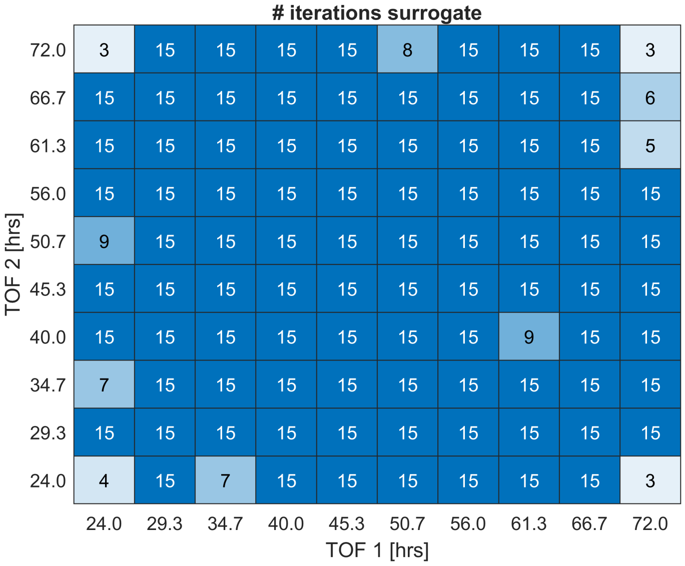


Figure 2.33: Non-converging solution insight.

Fig. 2.33 reports an example associated with a non-converging solution. In this case, numerical instability occurs during the correction process, determining an unbounded oscillation of the error. As previously discussed, ad-hoc mitigation may be introduced. For example, the correction algorithm assumes a full step is taken in the update equation, which may lead to instability if highly sensitive cases are considered. One possibility may be to introduce an adaptive step, or to reduce the step size. However, this may increase the number of iterations required to converge within the imposed tolerance, thus determining a reduction in computational speed (or, equivalently, an increment in total computational cost).

2.7.2 Surrogate-based Optimization Pipeline: Surface Complexity

Similarly to the previous case, we here test the offline three-step optimization pipeline, employing ordinary Kriging to generate an initial guess, here in terms of components of the Δv required to initialize the transfer from the LLO to the NRHO. To allow a comparison with the previous case, we again initialize each model with 40 points, sampled via Latin hypercube, adding a total of 60 adaptive points. Concurrently, as inclusion of bad guesses in the training process may hinder the surrogate performance, the simulation is conducted employing both Lambert settings, such that the chance of non-converging examples is minimized.



TOF 2 \ TOF 1	24.0	29.3	34.7	40.0	45.3	50.7	56.0	61.3	66.7	72.0
72.0	3	15	15	15	15	8	15	15	15	3
66.7	15	15	15	15	15	15	15	15	15	6
61.3	15	15	15	15	15	15	15	15	15	5
56.0	15	15	15	15	15	15	15	15	15	15
50.7	9	15	15	15	15	15	15	15	15	15
45.3	15	15	15	15	15	15	15	15	15	15
40.0	15	15	15	15	15	15	15	9	15	15
34.7	7	15	15	15	15	15	15	15	15	15
29.3	15	15	15	15	15	15	15	15	15	15
24.0	4	15	7	15	15	15	15	15	15	3

Figure 2.34: Iterations required to correct the guess provided by the surrogate models employing single shooting as correction scheme.

Fig. 2.34 reports the number of iterations required by a single shooting scheme to correct the guess generated by a surrogate model. As can be observed, the model performs extremely poorly if compared with the descent case study, leading to non-converging solutions for almost the entirety of the 10 × 10 grid here employed as test case. The causes of this behavior can be traced to the complexity of the surface to be approximated, which is directly related to the capabilities of the chosen metamodel.

Fig. 2.35 displays the surfaces that each surrogate model must approximate. As can be observed, each surface is characterized by steep variations, with sudden changes in either TOF1 and TOF2 directions. However, OK assumes that the spatial variability of the phenomenon being modeled is smooth and continuous, which does not hold for the case here considered. We believe the observed behavior to derive from the geometry and the dynamics of the problem. In fact, the two variables TOF1 and TOF2 correlates departure positions on the LLO to arrival conditions on the NRHO. Notably, the two orbits are characterized by substantially different dynamics, with a faster one associated with the LLO, and an overall slower one related to the NRHO.

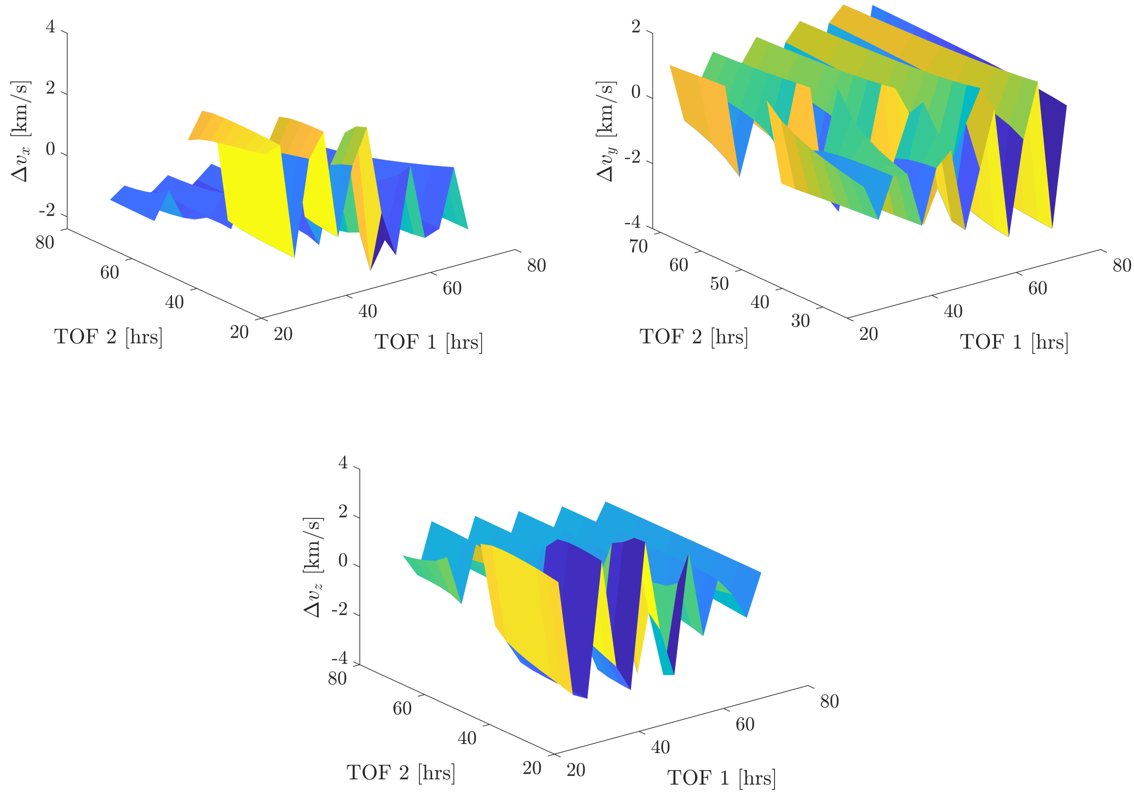


Figure 2.35: Δv components magnitude as a function of TOF1 and TOF2. LLO RAAN = 0°.

As such, conditions that are somewhat close in the TOF1-TOF2 space, may be relating highly diverse pairs of departure and arrival conditions from the geometric perspective. Concurrently, the surface is generated including guesses obtained correcting prograde and retrograde types of transfer, which is necessary to cover the solution space. Hence, the components of the Δv vector result to significantly variate in directions, thus determining the observed rapid changes. One natural solution may involve discretization of the [1,3] day-range for TOF1 and TOF2 into smaller grids, while training multiple surrogate models on each grid. Nonetheless, this requires identifying the spatial scale at which the surface approaches the characteristics required by OK. Unfortunately, a quick analysis reveals how this assumption generally holds for small time scale.

Fig. 2.36 depicts the departure Δv components when considering a small region of the parameter space. One can observe how the surface is significantly smoother with respect to what displayed in Fig. 2.35, hence making OK suitable for the task. The result is reported in Fig. 2.37,

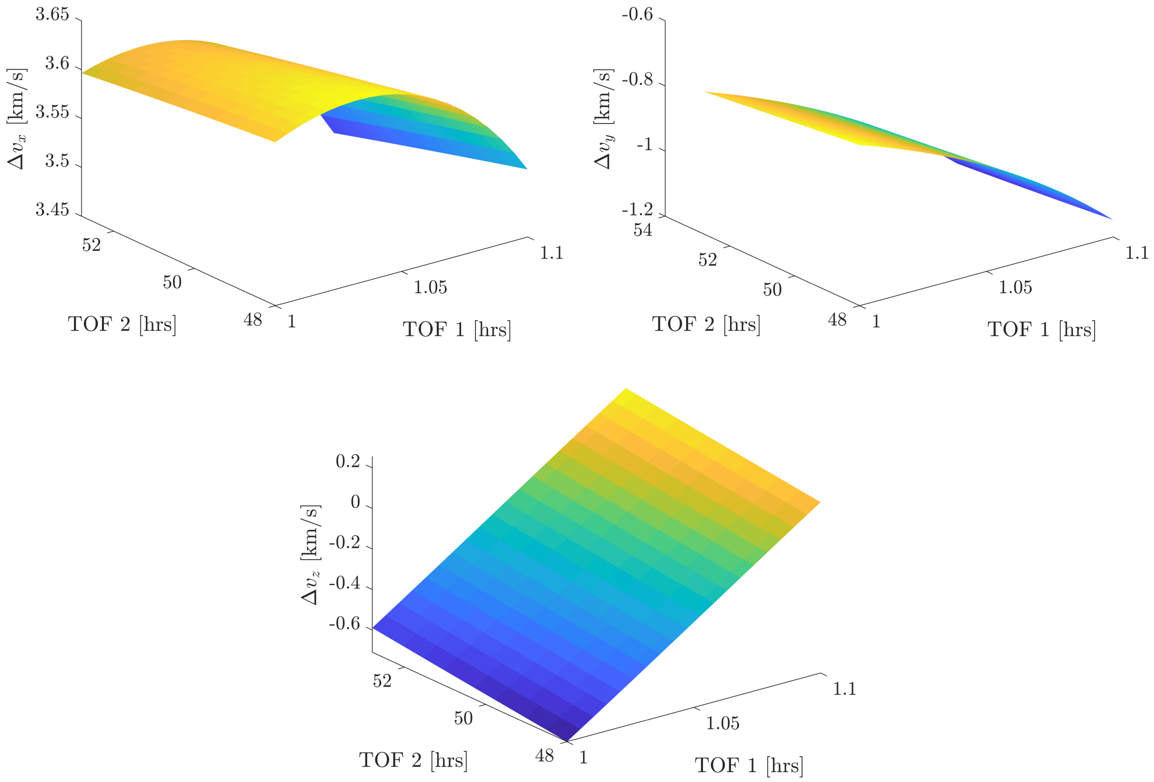
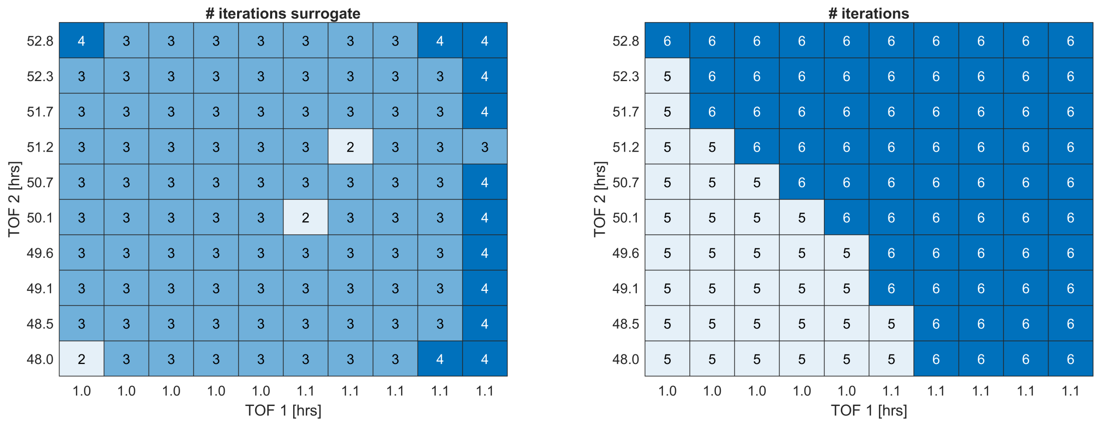


Figure 2.36: ∆v components magnitude as a function of TOF1 and TOF2. TOF1 ∈ [1,1.1] hrs and TOF2 ∈ [2,2.2] days.

where a surrogate model initialized with 20 points and improved through 15 total adaptive points is shown to outperform the Lambert guess generator on a 10 × 10 grid.



TOF 2 [hrs]										
52.8	4	3	3	3	3	3	3	3	4	4
52.3	3	3	3	3	3	3	3	3	3	4
51.7	3	3	3	3	3	3	3	3	3	4
51.2	3	3	3	3	3	3	2	3	3	3
50.7	3	3	3	3	3	3	3	3	3	4
50.1	3	3	3	3	3	2	3	3	3	4
49.6	3	3	3	3	3	3	3	3	3	4
49.1	3	3	3	3	3	3	3	3	3	4
48.5	3	3	3	3	3	3	3	3	3	4
48.0	2	3	3	3	3	3	3	3	4	4
TOF 1 [hrs]	1.0	1.0	1.0	1.0	1.0	1.1	1.1	1.1	1.1	1.1

(a)

TOF 2 [hrs]										
52.8	6	6	6	6	6	6	6	6	6	6
52.3	5	6	6	6	6	6	6	6	6	6
51.7	5	6	6	6	6	6	6	6	6	6
51.2	5	5	6	6	6	6	6	6	6	6
50.7	5	5	5	6	6	6	6	6	6	6
50.1	5	5	5	5	6	6	6	6	6	6
49.6	5	5	5	5	5	6	6	6	6	6
49.1	5	5	5	5	5	6	6	6	6	6
48.5	5	5	5	5	5	5	6	6	6	6
48.0	5	5	5	5	5	5	6	6	6	6
TOF 1 [hrs]	1.0	1.0	1.0	1.0	1.0	1.1	1.1	1.1	1.1	1.1

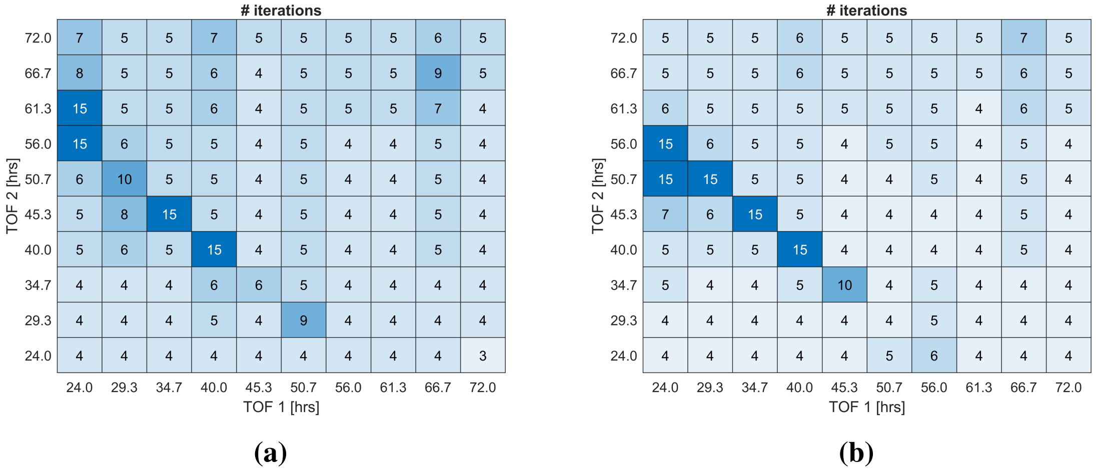
(b)

Figure 2.37: Iterations required to correct the guess provided by (a) the surrogate model and (b) the Lambert solver. TOF1 ∈ [1,1.1] hrs and TOF2 ∈ [2,2.2] days.

2.7.3 Dynamical Structure Investigation

In the previous section we discussed how the Δv surface is highly complex, allowing for applicability of an OK-based pipeline solely within special conditions. Another possible alternative would involve a change of surrogate model; nonetheless, this operation still necessitates a thorough understanding of the expected surface. Hence, we conducted an in-depth analysis aimed at identifying salient characteristics of the surface structure which resulted in a series of key considerations and the revelation of interesting dynamical behaviors. Of note, as no substantial difference has been observed, all analyses and discussions that follow will be based on results obtained employing solely the single shooting correction scheme.

Resolving Sensitive Cases. The analysis discussed in Sec. 2.7.1 highlighted how regions of stiff convergence appear throughout the parameter space. Interestingly, such results suggest these regions to be generally related to the passage of the Gateway across the perilune region, that is, to conditions where the Gateway has a higher speed on the NRHO, while being at its closest approach to the Moon (and hence, to the LLO). We substantiated this observation through further analyses, imposing different LLO orientations.



(a)

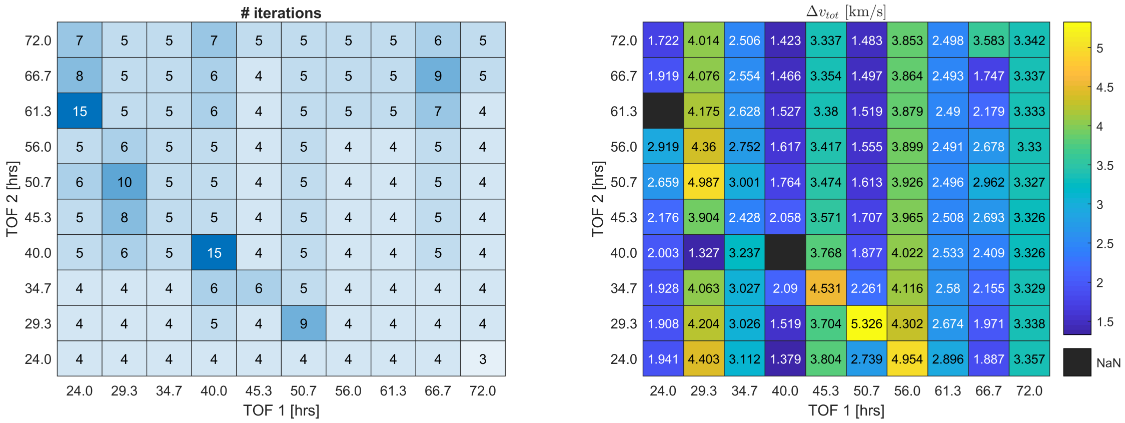
	24.0	29.3	34.7	40.0	45.3	50.7	56.0	61.3	66.7	72.0
72.0	7	5	5	7	5	5	5	5	6	5
66.7	8	5	5	6	4	5	5	5	9	5
61.3	15	5	5	6	4	5	5	5	7	4
56.0	15	6	5	5	4	5	4	4	5	4
50.7	6	10	5	5	4	5	4	4	5	4
45.3	5	8	15	5	4	5	4	4	5	4
40.0	5	6	5	15	4	5	4	4	5	4
34.7	4	4	4	6	6	5	4	4	4	4
29.3	4	4	4	5	4	9	4	4	4	4
24.0	4	4	4	4	4	4	4	4	4	3

(b)

	24.0	29.3	34.7	40.0	45.3	50.7	56.0	61.3	66.7	72.0
72.0	5	5	5	6	5	5	5	5	7	5
66.7	5	5	5	6	5	5	5	5	6	5
61.3	6	5	5	5	5	5	5	4	6	5
56.0	15	6	5	5	4	5	5	4	5	4
50.7	15	15	5	5	4	4	5	4	5	4
45.3	7	6	15	5	4	4	4	4	5	4
40.0	5	5	5	15	4	4	4	4	5	4
34.7	5	4	4	5	10	4	5	4	4	4
29.3	4	4	4	4	4	4	5	4	4	4
24.0	4	4	4	4	4	5	6	4	4	4

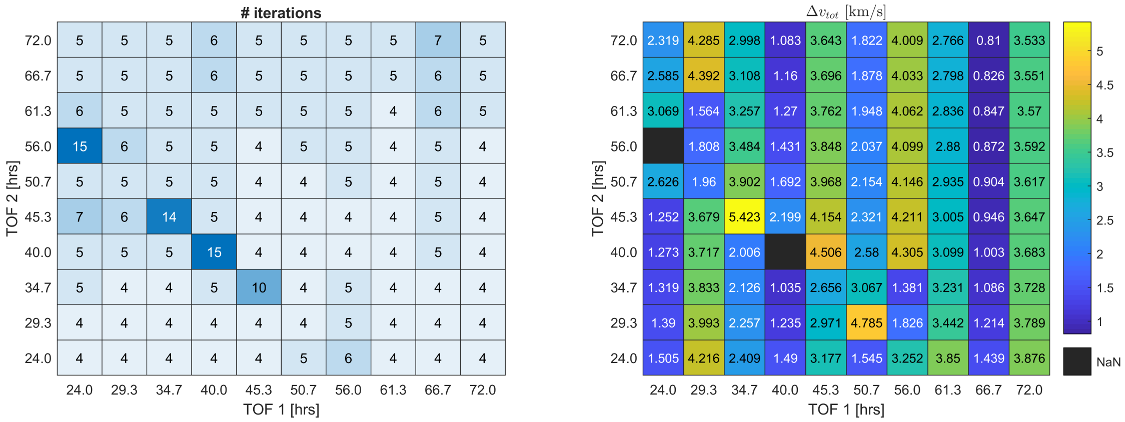
Figure 2.38: Stiff convergence region assuming a (a) RAAN = 150° and (b) RAAN = 300° .

An example is provided in Fig. 2.38, where results associated with two additional cases, RAAN = 150° and RAAN = 300°, are reported. Particularly, one can observe how non-converging cases are typically related to conditions for which the total time, given by the sum of TOF1 and TOF2, is about 80 hours. Considering all our simulations assume t_0 marking the passage of the Gateway at the aposelene, such a time is close to half of the NRHO period (∼78 hours), that is, to the time the DSG crosses the perilune. The observed trend is indicative that good initial guesses for relatively close departure and arrival points necessitate either smaller orbital paths or a shorter time of flight (observing Fig. 2.38 one can notice how solutions on the 80 hours diagonal converge when TOF2 goes below a certain threshold). As the imposed time of flight must be respected, the former implies utilization of multi-revolution transfers.



(a) **(b)**

Figure 2.39: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 150°.



(a) **(b)**

Figure 2.40: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 300°.

The solution is depicted in Figs. 2.39 and 2.40, where $Nrev_{max}$, that is, the maximum number of revolutions, is set 8. As can be observed , half of the non-converging cases are resolved via multi-revolution-like transfers. A visualization of the trajectories associated with the RAAN = 150° case is reported in Fig. 2.41.

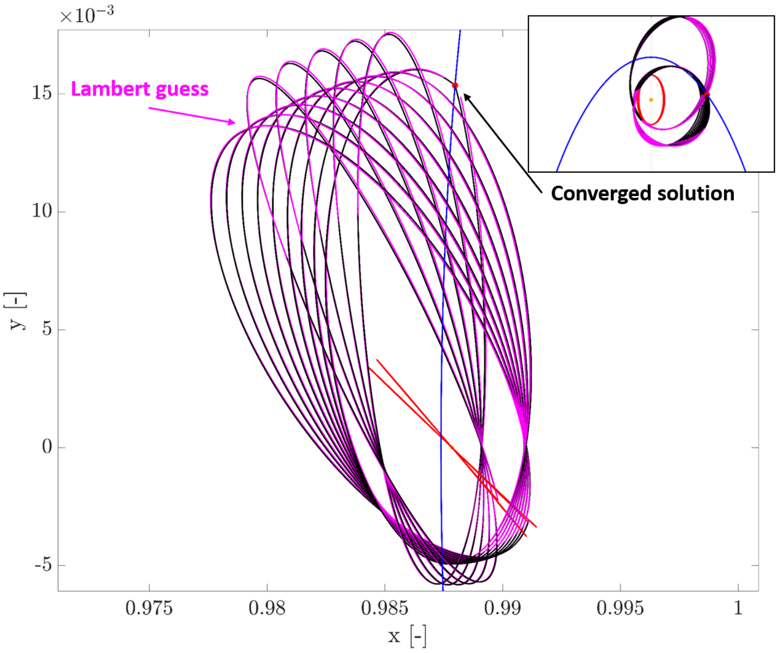
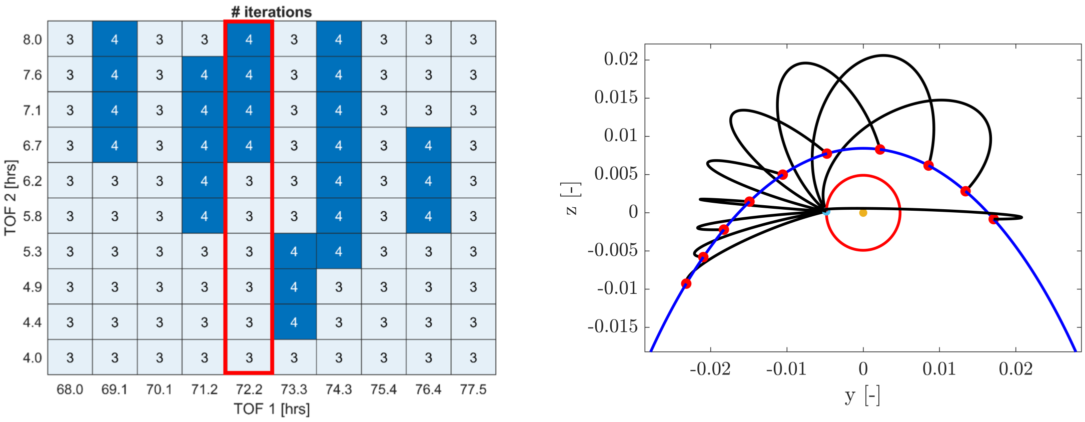


Figure 2.41: Corrected trajectories (top and side views), $Nrev_{max}$ = 8. LLO represented at the two different departure times.

As mentioned, an alternative consists of considering a shorter transfer duration. An example for the RAAN = 300° case is reported in Fig. 2.42.



TOF 2 [hrs] \ TOF 1 [hrs]	68.0	69.1	70.1	71.2	72.2	73.3	74.3	75.4	76.4	77.5
8.0	3	4	3	3	4	3	4	3	3	3
7.6	3	4	3	4	4	3	4	3	3	3
7.1	3	4	3	4	4	3	4	3	3	3
6.7	3	4	3	4	4	3	4	3	4	3
6.2	3	3	3	4	3	3	4	3	4	3
5.8	3	3	3	4	3	3	4	3	4	3
5.3	3	3	3	3	3	4	4	3	3	3
4.9	3	3	3	3	3	4	3	3	3	3
4.4	3	3	3	3	3	4	3	3	3	3
4.0	3	3	3	3	3	3	3	3	3	3

(a) **(b)**

Figure 2.42: Reducing the transfer time: (a) iteration map, and (b) sample of trajectories.

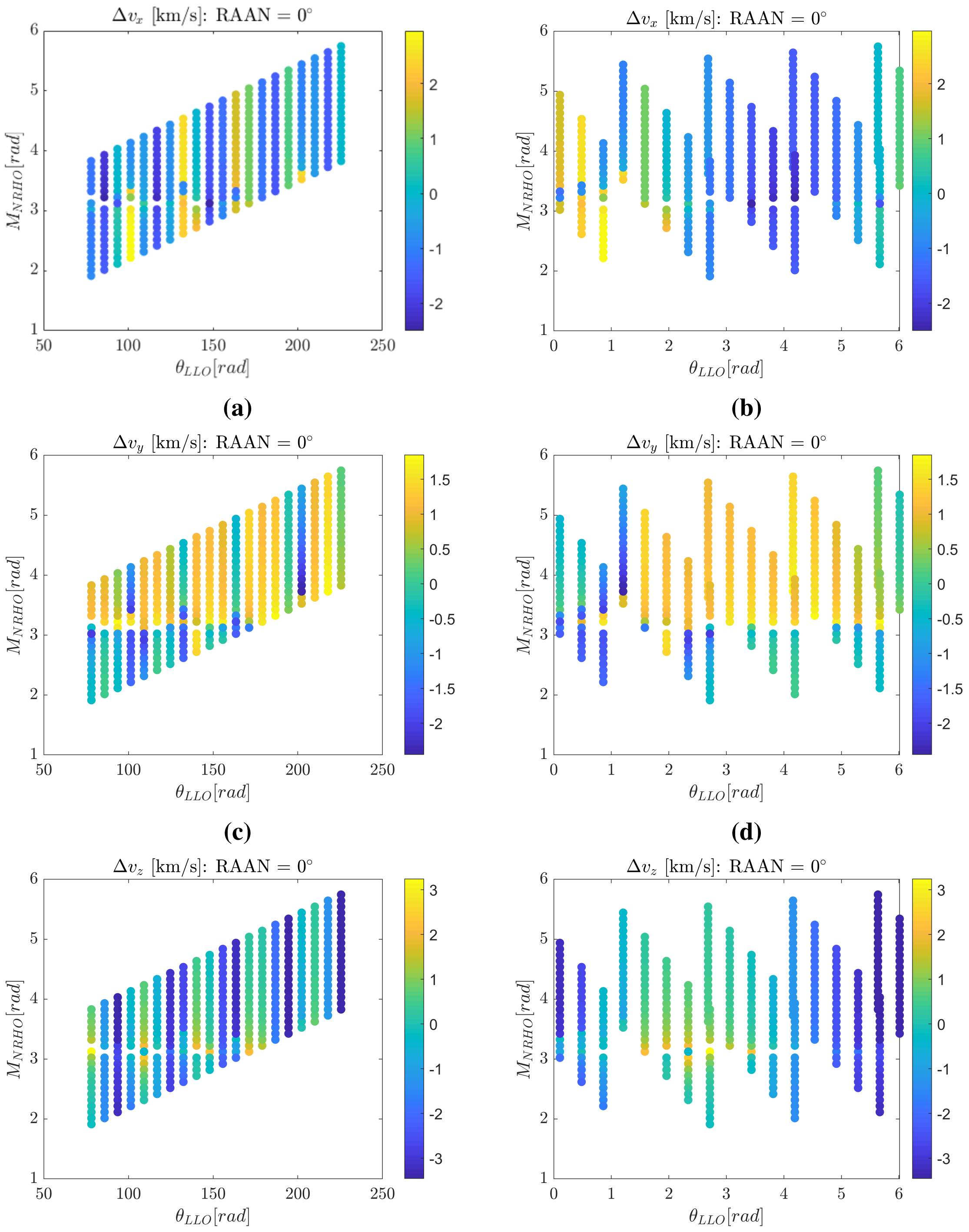
In this particular case, we considered a loitering time (∼TOF1) sufficiently long to allow the DSG to reach and surpass the perilune, while bounding the transfer time (TOF2) within a few hours. As can be observed, all guesses in the grid generated by the Lambert solver converge to a feasible solution in CR3BP model within a few iterations.

Surface Smoothing. As discussed in Sec. 2.7.2, the driver of the low performance achieved by the surrogate model can be identified into the complexity of the surface to be approximated. While a change of function approximator may offer a solution, an alternative may consist of regularizing (or smoothing) the surface of interest. In fact, as hinted in the previous section, the articulation of the surface is a derivative of the grid search formulation, which 1) employs two time of flight variables as design parameter, thus resulting agnostic to the geometrical characteristics of the problem, and 2) considers only feasible solutions, which may exist solely for either a prograde or retrograde settings, thus determining steep changes in the direction of the Δv vector. Therefore, benefits may derive from a change of variables, substituting the dependence of the Δv surface on the TOF1-TOF2 parameters with a dependence on angular quantities, which embed geometrical properties of the transfers. Due to the nature of the problem, we here preliminarily select the true anomaly (for the LLO) and $\sim$mean anomaly (for the NRHO) as angular measures to obtain a representation based on departure and terminal positions. Such quantities can be directly related to time through the following relations:

$$\theta_{LLO} = \theta_{inj} + n\tau_1 \tag{2.64}$$

$$M_{NRHO} = \frac{\tau_2}{T} 2\pi \tag{2.65}$$

where θ_{LLO} is the true anomaly on the LLO at the departure time, θ_{inj} is the true anomaly at the injection after the ascent from the lunar surface, n is LLO mean motion, τ_1 corresponds to TOF1 (reduced of the ascent time), M_{NRHO} represents the mean anomaly-like angular position of the rendezvous point on the NRHO, τ_2 is the total time from t_0, which is equivalent to TOF1 + TOF2, and T is the NRHO orbital period. Of note, the general definition for the mean anomaly would consider t_0 at perilune, while here we assume it to be at the passage of the DSG at the apolune.



(a) **(b)**

(c) **(d)**

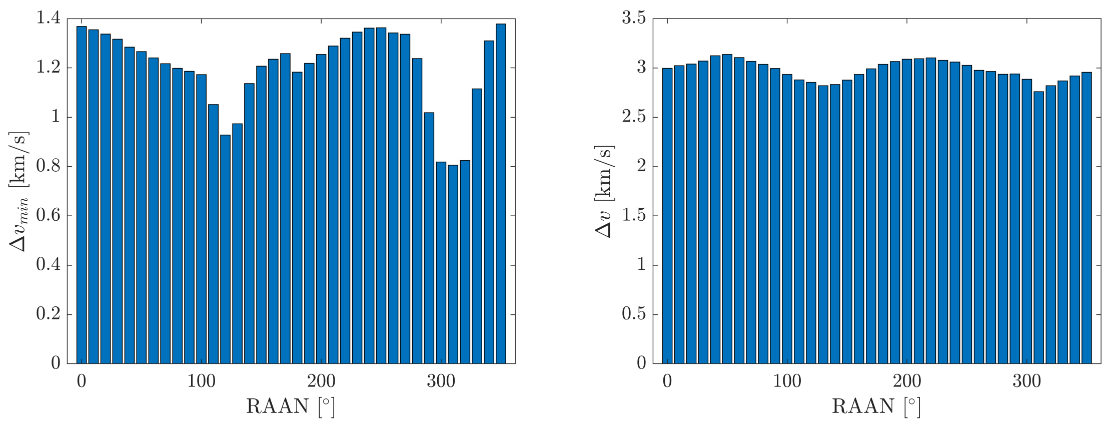
(e) **(f)**

Figure 2.43: Components of the Δv vector as a function of θ_{LLO} and M_{NRHO}: (a,c,e) continuous θ_{LLO} range and (b,d,f) wrapped range.

Fig. 2.43 displays the Δv vector components in terms of the angles θ_{LLO} and M_{NRHO}. All the images on the left can be seen as a direct analogous of the grid search in time, with the linear trend due to the law presented in Eq. (2.64). However, differently from the time-dependent representation, the angular representation allows for a re-organization of the surface through bounding the true anomaly in the canonical [0,2π] range. In particular, one can observe how similar regions which are distant in the continuous range representation (which directly mirrors the continuous time direction), get closer to each other in the wrapped case, suggesting that a representation of the surface in terms of more explicit geometrical quantities may be beneficial toward a smoother surface, and ultimately, may facilitate the training of a surrogate model.

RAAN Influence on the Δv Surface. Throughout the previous paragraphs, we emphasized how the geometry of the problem strongly influences the convergence structure and the Δv surface structure, either when considering particular transfers directions (such as the effect of prograde and retrograde guesses combined with orbit orientation), and the relative position of departure and target states. To conclude the analysis and gain a global understanding of the trajectory solution landscape, we here present the results associated with the execution of a grid search over the entire RAAN range. Specifically, the simulation considers a 20 × 20 grid size for each RAAN, spacing the range with a 10° step. Below, the key observations garnered from this search are reported.

- *Δv cost.* Based on our two-step optimization pipeline, we discover that within the CR3BP formulation transfers as cheap as $\sim$0.8 km/s in terms of total cost – here intended a summation of Δv at departure and rendezvous points - can be achieved. Specifically, as displayed in Fig. 2.44, any orientation of the LLO grants the possibility to identify relatively cheap transfers, which would result feasible according to current ascent module capability. Furthermore, we also observe how given orientations of the LLO may provide more favorable transfers overall, thus leaving higher flexibility in terms of segment duration once the LLO RAAN is fixed. The advantageousness of certain RAAN range can also be interpreted from a geometrical perspective in terms of relative LLO-NRHO orientation, coupled with the direction of the motion. In fact, as shown in Fig. 2.45, the cheapest solutions are associated with conditions where the LLO orientation in the rotating frame is somewhat aligned with the NRHO, while concurrently having the same direction of the motion.



(a) **(b)**

Figure 2.44: Total Δv cost for each 20 × 20 grid: (a) minimum cost and (b) average cost.

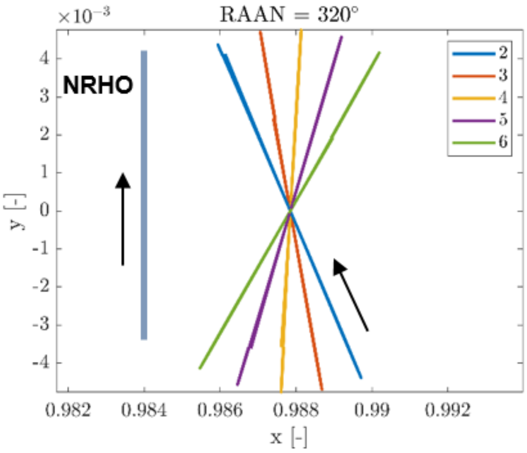
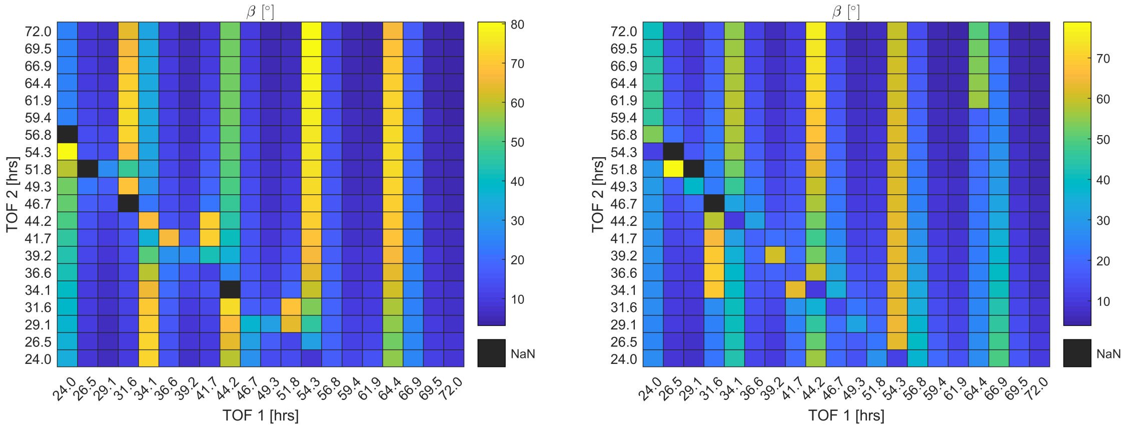


Figure 2.45: Example of relative LLO-NRHO orientation (top view) in the rotating frame. Black arrows indicating the motion direction, while the different colors represent an LLO with RAAN = 320° at different times (see legend, with days as unit).

- Δv *direction feasibility.* Further insight can be gained by considering the direction of the impulses, specifically for what concerns the velocity variation at the departure point from the LLO. Despite the components (and their corresponding signs) of the vector are directly available, they provide a less intuitive representation, especially in a three-dimensional space. Hence, we analyze the direction through a more convenient representation, mapping the Δv vector to an $\hat{R}$-$\hat{S}$-$\hat{W}$ frame (see Sec. 2.2.3), which enable to represent a vector in terms of its magnitude and two reference angles, here denoted as α and β, with α defined as the angle between the Δv vector projection on the $\hat{R}$-$\hat{S}$ plane and the $\hat{R}$ axis, while β is the angle between the Δv vector and its projection on the $\hat{R}$-$\hat{S}$ plane[3].

[3] In other words, α and β define the in-plane and out-of-plane directions of the impulse.



(a) **(b)**

Figure 2.46: β angle values expressed in degrees for two sampled right ascensions: (a) RAAN = 0° and (b) RAAN = 200°.

Fig. 2.46 reports the grid values for the β angle at two randomly selected RAANs. One can notice how, on average, the out-of-plane Δv component assumes low values, with the exception of some $\sim$uniformly distributed cases. Analysis of such cases reveals how such conditions correspond to cases where θ_{LLO} is close to $\pi/2$, while the rendezvous point is beyond the NRHO perilune. An example is reported in the figure below.

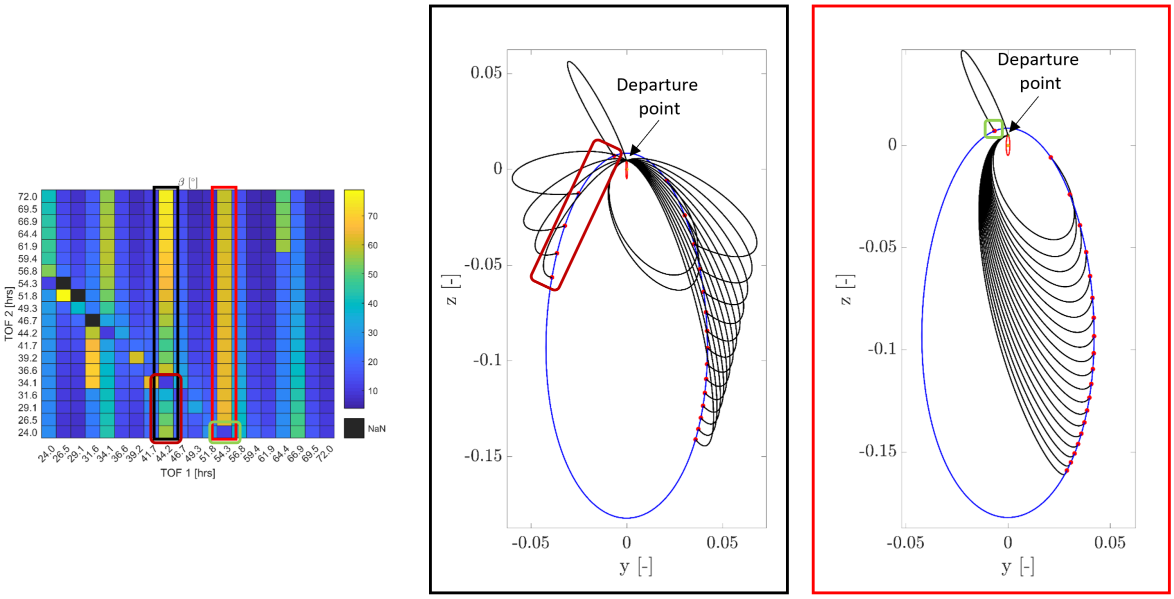


Figure 2.47: Sample trajectories for high β values.

It is worth mentioning that while the highlighted solutions in Fig. 2.47 converge numerically, they may result unfeasible from the practical perspective, thus potentially reducing the number of available solutions.

Chapter 3

A Framework for Anomalous Links Detection within P-LEO Satellite Networks

3.1 Introduction

P-LEO satellite constellations promise to offer seamless services for a variety of applications, including omnipresent connectivity. However, continuous monitoring challenges arise due to potential malfunctions and the vast size of these systems. Additionally, the growing reliance on P-LEO constellations makes them ideal targets for malicious actors [40, 41, 42], expanding the cyber-threat landscape to include space-based systems. Knowledge of satellites' location and network structure is generally of public domain [43]. Moreover, the low latency targeted by these networks reduces path diversity and route uncertainty, easing a malevolent actor's need to identify crucial network connections. These characteristics make space networks more vulnerable to certain network-oriented attacks like Denial of Service (DoS) and Distributed Denial of Service (DDoS) [44]. One such attack in the context of satellite constellations is ICARUS [45]. In this scenario, a malicious actor in control of a *botnet* is virtually capable of disrupting communications between two regions of the world by flooding a few selected connections. As P-LEO constellations underpin critical society's infrastructure, rapidly detecting these attacks becomes crucial to prevent catastrophic consequences. Unfortunately, discovery of such anomalous behaviors is hindered by the subtlety of these attacks, as malicious actors may hide their action by exploiting natural surges of network activity. Therefore, there is an urgency to identify effective methods capable of capturing the underlying complexities of these large systems.

Many algorithms have been developed to aid human operators in the identification and assessment of anomalous events, with traditional methods often relying on rule-based/threshold approaches [46]. For example, Jiang et al. [47] propose a method to analyze large volumes of telemetry data by conducting a pseudo-periodic analysis relying on a data compression technique based on symbolization, and on a similarity principle of eigenvectors obtained via a phase-plane analysis of the data stream. Instead, Li et al. [48] develop a method to detect electrical faults on satellites based on principal component analysis to extract salient data features, followed by a fuzzy c-means offline clustering algorithm and an approximate weighted proximal support vector machine (WPSVM) for online classification. To overcome the limitations faced by conventional techniques, more recent works explore the application of machine learning and deep learning, relying on their potential to capture intricate behaviors. Promising results have been obtained by methods based on recurrent neural networks such as Long Short-Term Memory (LSTM) [49, 50] and Gated Recurrent Unit (GRU) [51] networks, although such methods may be limited by computational resources and by the lack of extensively labeled datasets. Regardless of the approach, most of these methods have been developed for the detection of abnormal behaviors originating from natural causes, with little to no attention given to cybersecurity threats. Additionally, their application has commonly been limited to individual satellites rather than to an entire constellation.

Recent works demonstrate the potential of representing satellite systems as dynamic networks, which can capture the intricate connections and relations among objects in time-evolving systems. Particularly, graph-based methodologies are naturally available for dealing with problems related to dynamic networks, showing extraordinary promise when combined with the anomaly detection task. The literature teems of research devoted to the detection of unusual events in graph-represented systems – in particular, dynamic systems - with popular applications in fields such as computer security, e-commerce, and social media [52]. Traditional techniques rely on temporal behavior analysis and connectivity models. For example, the Commute-time based Anomaly detection in Dynamic graphs (CAD) [53] method targets the detection of anomalous edges by tracking variations in graph structures and changes in edge weights. GOutlier [54] builds over a structural connectivity model, handling network sparsity through dynamic partition, and designing a reservoir sampling to maintain structural summaries of the network. StreamSpot

[55] exploits a centroid-based clustering approach to model the behavior of graphs streams. As these methods struggle in capturing non-linear properties [56], more advanced techniques leveraging deep learning have been introduced. Some methods, such as NetWalk [57], employ a graph embedding approach to learn a dynamic representation of a graph, though often the embedding strategy is typically not aimed at detecting anomalies. Conversely, H-VGRAE [58] proposes a stochastic neural network specifically for detecting anomalies in dynamic networks through learning of a robust node representation in the form of random variables. More recently, the authors of TADDY [3] proposed a novel temporal-structural node encoding to encapsulate complex spatial dynamic correlations which are captured via a transformer attention mechanism, leading to superior performance.

While these approaches have gained an extraordinary popularity for conducting anomaly detection in interconnected systems, the literature presents few examples concerning their applicability to satellite networks'- especially for security applications. For instance, Zhang et al. [59] exploit a space-time graph model to represent a satellite constellation over which they simulate a DDoS attack; however, the authors establish a priori target satellites and focus on the security performance of the attack rather than on the detection of the attack itself. Instead, Guo et al. [60] introduce a blockchain-based distributed collaborative entrance defense (DCED) framework to detect a DDoS attack at the entrance of the satellite network; however, in this case graphs are not used to represent the system, and the orbital mechanics of the constellation network is largely ignored.

If research is in its infancy in regard to the detection of distributed attacks against space systems, the investigation of techniques applied to terrestrial networks is an active and mature field. Traditional methods involve traffic analysis and flow monitoring principles and are mostly based on signature and behavior-based methods [61]. Liu et al. [62] propose an isolation forests-based detection algorithm, testing its effectiveness on standard network intrusion datasets. Kiss et al. [63] develop a clustering algorithm based on a Gaussian mixture model specifically tailored for DoS detection. More recently, machine learning and deep learning techniques have rapidly gained popularity. Sabeel et al. [64] develop a Long Short-Term Memory -based architecture for the identification of unknown DoS/DDoS attacks. A similar approach is taken by Gogoi et al. [65], who focus on the detection of low and slow DoS attacks. Despite the popularity of these methods within terrestrial systems, limitations and challenges arise when it comes to space networks. In fact, common detection techniques often rely on powerful and costly hardware infrastructure, with similar considerations applying to mitigation techniques [45] such as in-network filtering [66]. Furthermore, these methods are generally tested on data collected from terrestrial networks, which are governed by their own dynamics. Conversely, space networks display unique spatial-temporal features regulating connections between nodes that differ from the behavior of terrestrial networks. This intrinsic topology dictated by the laws of orbital mechanics can then be leveraged for the development of ad-hoc, novel detection schemes.

In this chapter, we explore topological changes in the dynamic network structure representing a P-LEO constellation equipped with interlink technology to detect DDoS attacks. DDoS attacks may force the system to search alternative paths between communications origin and destination points or apply load balancing in routing schemes that would distribute the traffic load to avoid highly congested paths. Both changes may create topological variations in the network, such as the creation of new edges or a change of coloring. In this initial step, we assume coloring is fixed and only investigate the variation of new edges. As a starting point, this approach assumes centralized knowledge; that is, the network can leverage information about the whole graph. Drawing from the success demonstrated over several dynamic network problems, we select the TADDY [3] framework as a state-of-the-art baseline method for the detection of anomalous edges within an abstracted version of the ICARUS attack. However, as the framework is developed for significantly different systems, the performance of the algorithm is expected to decrease. Therefore, we modified the framework to adapt it to our physical P-LEO constellation system, thoroughly discussing how such changes affect the algorithm detection capability. The contributions of our work are 1) the introduction of a framework for anomalous connections detection in large satellite systems, exploiting a combination of graph-based representation and deep learning; 2) the identification of couplings between satellite dynamics and spatial ground node distribution, and the discovery of key features providing notable performance enhancements of the TADDY model on our physical system; 3) the collection of empirical evidence of algorithm generalization capability, with qualitative and quantitative analyses on systems parameters and their effect on performance.

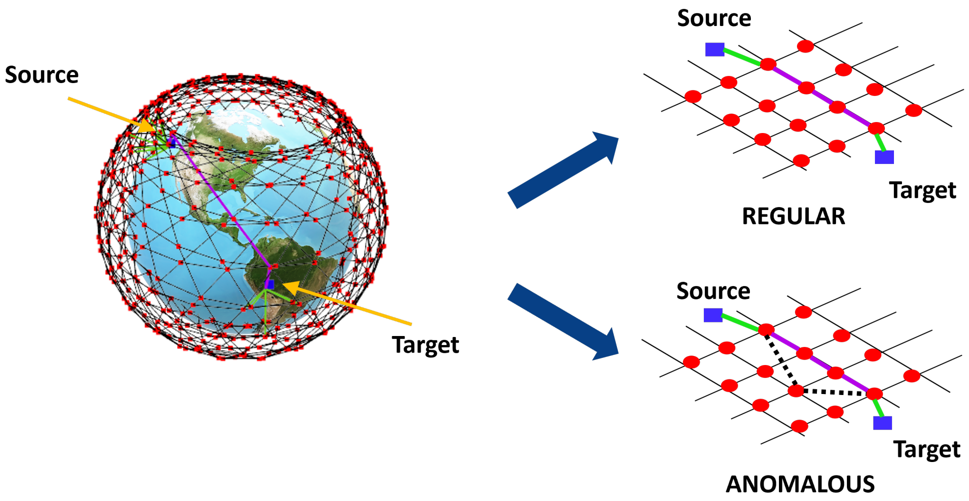


Figure 3.1: Schematic problem representation. Constellation image generated from a modified version of [1].

The remainder of this chapter is organized as follows. Sec. 3.2 provides background regarding the application of deep learning for anomaly detection on graphs. Sec. 3.3 briefly presents graph-related key concepts and deep learning theoretical background. Sec. 3.4 describes the problem formulation and modeling. Sec. 3.5 presents the methodology and the validation of the algorithm. Sec. 3.6 reports and discusses the results.

3.2 Related Work

3.2.1 Graph System Representation

System representation plays a key role in the successful identification of anomalies. Thanks to their capability of highlighting structural information, graphs constitute a suitable tool for representing large complex systems, where elements may exhibit inter-dependencies. Additionally, the nature of the anomalies themselves may hide dependencies, further underlining the usefulness of a graph-like representation.

Traditionally, graphs are applied to a variety of fields, such as e-commerce, social networks, and biology [67]. Thanks to their versatility, graphs have also been exploited in space-related research. For example, graphs can be utilized to represent individual spacecraft [68]: while nodes are representative of single subsystems (or components within a given subsystem), edges are representative of the relations among them. More intuitively, graphs are employed to represent connected systems of satellites, such as swarms or constellations. In [69], the authors conduct an observability analysis for satellite constellations, representing satellites as nodes and edges as measurements among spacecraft. Similarly, in [70], the authors utilize graphs for representing small constellations within the problem of multiple-satellite orbital control. In [71], the authors leverage a graph theoretical approach for maneuver planning in the context of satellite constellations, with edges representing maneuver options, and nodes correspond to a flayover of a target. More recently, graphs have been extensively employed to represent time-varying satellite networks [72, 73, 74, 75] for the investigation of routing performance of the emerging large satellite constellations.

3.2.2 Graph Anomaly Detection

Combination of the representation through graphs with the anomaly detection task generates the increasingly investigated field of *graph anomaly detection*. Different types of anomalies can be investigated [76], stemming from the structural components of the graph itself.

- **Node anomaly.** Anomalous nodes are typically individual vertices whose behavior differs from the one of the others. Such nodes may be either *isolated*, that is, they display no (or very few) connections with other elements in the graph, or *hubs*, that is, they display a high number of connections to other elements in the graph. As an example, in a social network context an anomalous node may be represented by a user sending thousands of messages to other users.

- **Edge anomaly.** Anomalous edges represent irregular links connecting nodes. For example, in a bank transaction network, anomalous edges may be representative of suspicious transactions between different accounts.

- **Sub-graph anomaly.** Anomalous sub-graphs are a group of nodes and/or edges that display a different behavior with respect to the remaining graph. In such a case, while individual nodes and/or edges may have a regular behavior, they become suspicious in an aggregate. For example, in an online review system a sub-graph anomaly may be represented by a set of users posting negative or misleading reviews simultaneously.

- **Graph anomaly.** An anomalous graph is defined by a graph whose structure significantly differs from the one of a given set of graphs. For example, borrowing from a biology context, an anomalous graph may be represented by a molecule whose atomic bonds are different from those displayed by molecules of the same group.

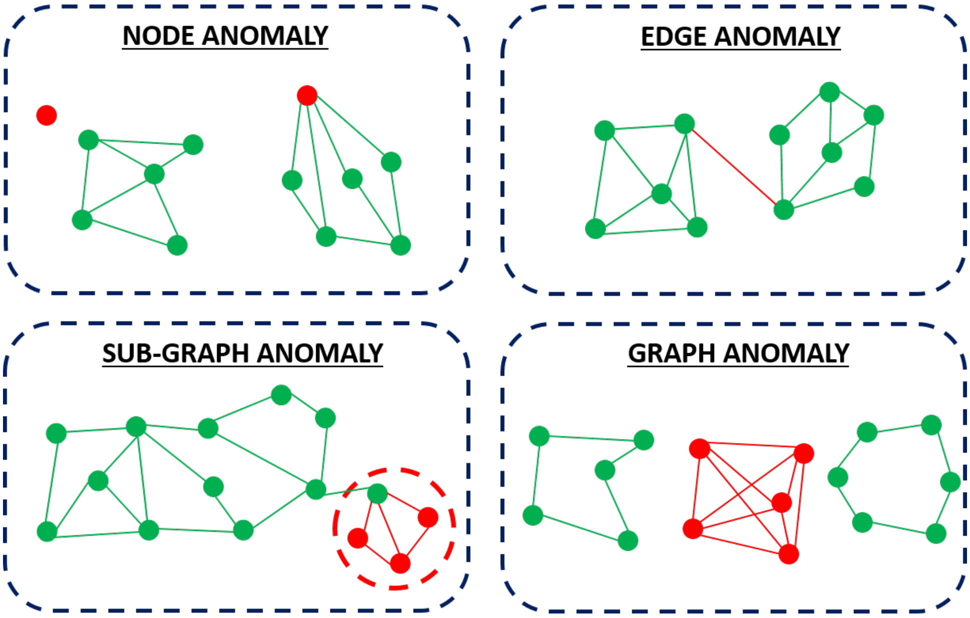


Figure 3.2: Schematic representation of graph anomalies.

Further connotation of the problem is introduced by the nature of the represented system, which might be stationary or time-variant. This attribute differentiates between the problems of *static* graphs and *dynamic* graphs anomaly detection. Identifying outliers on dynamic graphs is the most challenging between these two problems. In fact, dynamic graphs display structural/spatial characteristics and include temporal signals. As such, additional challenges in representing spatial and temporal information in complex evolving systems makes the anomaly detection task more daunting. Of note, the provided definitions for types of anomalies remains valid also for the case of dynamic graphs, with the nuance that for each case the anomalous behavior must be captured throughout the evolution of the graph (Fig. 3.3).

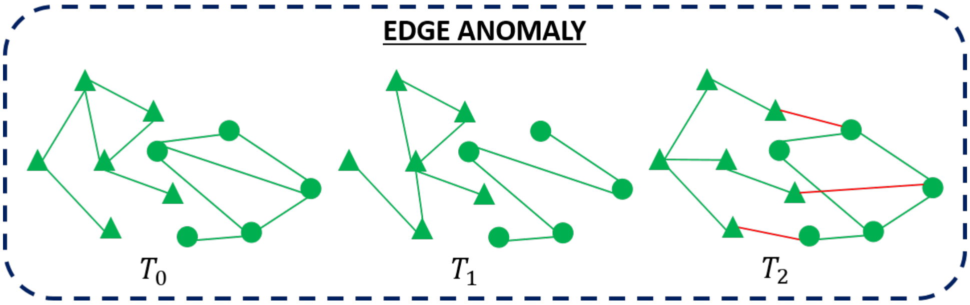


Figure 3.3: Example of anomalous edges in the context of dynamic graphs: anomalous connections appears between group of nodes without previous relations. T_i notation utilized to denote different time instants.

Static Graphs Anomaly Detection. Many algorithms have been developed to detect anomalies in graphs, often devoted to the identification of a single anomaly instance. In the realm of static graphs, traditional methods often relies on the extraction of graph statistical features. For example, the authors of oddball [77] propose to identify anomalous nodes by extracting meaningful statistical information associated with the neighbouring node set of a given node, while defining a set of laws to characterize regular behaviors. Instead, the authors of [78] propose SCAN, an algorithm aimed at the identification of anomalous nodes as outliers highlighted after the application of a graph clustering algorithm. However, many of these traditional methods are often limited in that extracted features and identification criteria are established by human expert knowledge, which may fail to capture meaningful hidden information. To overcome these limitations, more recently many machine learning algorithms have been developed. Fore example, the authors of DONE [79] present an unsupervised method for outlier nodes detection based on a network embedding approach which leverages deep autoencoders for generating a lower dimensional vector representation of the graph. To tackle the problem of anomalous sub-graph detection, the authors of [80] present DeepFD, a deep learning algorithm for fraud detection which models the users of a network in a latent space and classifies them based on the obtained latent space distribution.

Dynamics Graphs Anomaly Detection. For what concerns the task of anomaly detection on dynamic graphs, many recent techniques rely on the adoption of machine learning and deep learning algorithms [81]. Two main challenges are commonly addressed: 1) the lack of attributes and labeled anomalies in many available dynamic graphs datasets, which typically results in unbalanced data, and 2) the necessity of capturing both spatial and temporal information, as they both play a crucial role for detecting anomalies in dynamic systems. For example, in [57] the authors present NetWalk, where the anomaly detection task on a graph stream is carried out through an architecture exploiting a clique embedding for the graph representation, an autoencoder for obtaining the reconstruction error, and the application of *k*-mean clustering algorithm [82] to group nodes sharing the same timestamp. To address the issue of labeled anomalies, Zheng et al. introduce AddGraph [83], where a selective negative sampling technique is employed to generate anomalies artificially. In [84], the authors present a two-pronged approach based on the definition of two metrics to evaluate nodes' importance, aiming to identify sudden structural and attribute changes in graphs streams. A three sub-modules structure is instead adopted in the StrGNN [85] algorithm, where a *h-hop* substructure sampling is adopted to extract the sub-graph around an edge to be evaluated, followed by a node labeling function to determine the node role in the sub-graph; the extracted data are then passed through a Graph Convolutional Network (GCN) aimed to extract sub-graph features, followed by Gated Recurrent Units layers to capture temporal information. However, many of these techniques often decouple structural and temporal information processing, potentially missing relevant correlations. To mitigate this issue, Liu et al. introduce TADDY [3], an architecture featuring a novel structural-temporal node encoding and exploiting a transformer encoder network to accomplish the anomaly detection task. In this manuscript, the TADDY framework is employed to tackle the challenge of anomaly detection on P-LEO constellations. However, we introduce a series of necessary modifications so that the architecture can adapt to the different and faster satellite constellation dynamics.

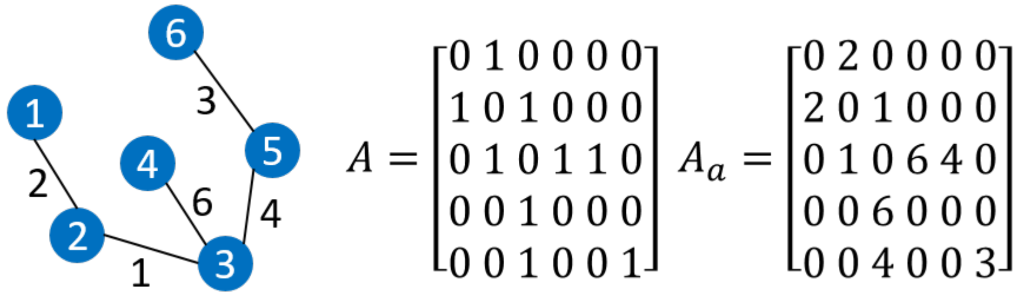
3.3 Theoretical Background

In this section, the theoretical background for the key topics discussed in this chapter - graphs and neural networks - are presented. At first, theoretical background on graphs is provided, together with some notation that will be employed throughout the reminder of this chapter. Next, a brief overview on neural networks is presented, followed by a more in depth description of the transformer neural network, here utilized as backbone architecture for the anomaly detection task.

3.3.1 Graph Theoretical Background

Definition 1. A graph is defined as a finite, non-empty set of entities called vertices (or nodes), which are gathered in a set denoted as $V(G)$, and a collection of nodes pairs - the edges - which belong to an edge set denoted as $E(G)$ [86]. Following this notation, the cardinality of the two sets is identified as $|V(G)|$ and $|E(G)|$, respectively. Given the definitions of nodes and edges sets, a pair of nodes v_1 and v_2 is considered adjacent if the pair $\{ v_1,v_2 \} \in E(G)$. In this way, a graph can be conveniently represented via an *adjacency matrix*, denoted as $\mathbf{A}$. Defining the total number of nodes as N, the adjacency matrix is a square matrix $\in \mathcal{R}^{N \times N}$. For a plain (or unattributed) graph, the adjacency matrix contains ones and zeros depending whether a connection exists between a pair of nodes. For the case of attributed graph - with attribute intended as a feature denoting some characteristic of the graph components - the ones in the adjacency matrix are typically replaced with the values of such attributes. An example for these concepts is provided in Fig. 3.4.

Definition 2. A graph is said to be dynamic (or temporal) if its nodes, edges, or attributes change over time. Given a finite time horizon, T, a dynamic graph can be represented as a sequence of static graphs, that is, $\mathbf{G} = \{G^t\}\ \forall t = 1, .., T$. Accordingly, the nodes and edges sets can be identified as V^t and E^t, such that a graph at each time step can be defined as $G^t = (V^t, E^t)$. It follows that, for each time stamp, an associated adjacency matrix, $\mathbf{A}^t$, must be defined.



$$A = \begin{bmatrix} 0&1&0&0&0&0 \\ 1&0&1&0&0&0 \\ 0&1&0&1&1&0 \\ 0&0&1&0&0&0 \\ 0&0&1&0&0&1 \end{bmatrix} \quad A_a = \begin{bmatrix} 0&2&0&0&0&0 \\ 2&0&1&0&0&0 \\ 0&1&0&6&4&0 \\ 0&0&6&0&0&0 \\ 0&0&4&0&0&3 \end{bmatrix}$$

Figure 3.4: Unattributed ($\mathbf{A}$) and attributed ($\mathbf{A}_a$) adjacency matrix example. $N = |V(G)| = 6$ and $|E(G)| = 5$.

3.3.2 Neural Networks Overview

Neural networks (NNs) can be categorized as a subset of machine learning (ML) techniques that are capable to learn and generalize patterns from some input data. A generic neural network consists of single processing units whose structure is designed to store experiential information from the input data in a latent manner [87]. They consists of a large number of interconnected units, referred to as *neurons*, that pass information among one another aiming to approximate the mapping existing between their input and the desired output. Neurons are organized in *layers*, with the number of the neurons constituting such layers commonly referred to as layer width. A basic network structure can then be represented as a sequence of input, hidden, and output layers. The number of hidden layers defines the depth of a network. As a rule-of-thumb, a network with more than three hidden layers is called *deep* neural network. Usage of deep networks defines the ML branch known as deep learning.

The mathematical foundation of neural networks' working principle is defined by the *universal approximation theory* [88]. According to such theory, any continuous function $f : [0,1]^n \rightarrow [0,1]$ can be approximated sufficiently well by a neural network, with at least a single hidden layer containing a finite number of weights and biases. These are learnable parameters of the network, and their value is optimized during the training process. As a consequence, the complexity of the function to be approximated strongly influences the design of the network itself.

Qualities such as the possibility to model non-linear relations and their generalization capability, with the latter being a necessary capability to perform inference on unseen or latent data, make neural networks ideal for carrying out complex tasks, finding applications in a wide variety of fields. Typical applications of neural networks include medical diagnosis via image processing, autonomous-driving, fraud-detection, visual recognition and language translation (though many other exist). In the following, a brief description of the most traditional networks classes and their applications is provided.

1. **Feed-forward Neural Networks.** Feed-forward neural networks (FFNNs) are the simplest type of network. Their name derives from the fact that information are passed through the layers in a forward manner, that is, they flow from input to output without any cycle. Typical architectures falling under the umbrella of FFNNs include:

   - Multi-layer Perceptron (MLP): Designed to provide an approximation of a given function, this is the simplest network architecture. When information is passed between layers, an activation function is applied to each neuron to introduce a non-linear transformation and enable to capture more complex information. Classical choices for such activation functions include the Sigmoid, the Hyperbolic Tangent, the Rectified Linear Unit (ReLU), and Softmax.
   - Radial Basis Function (RBF) NN: This is a shallow network whose neurons use the Gaussian function as their activation function. Unlike the MLP, the input to the layer is not the original input, but a transformation of the input through a Gaussian function. RBFNNs find similar applications to those of the MLPs architecture, though for some selected applications (such as the approximations of complex, non-linear functions with localized patterns) they tend to perform better than MLPs.
   - Convolutional NNs (CNNs): A more advanced architecture which includes convolutional layers (required to capture local patterns), an activation layer, pooling layers (used for downsampling of the data), and a fully-connected layer (utilized for prediction), this type of network becomes convenient when the dimensionality of the input data increases. In this type of network, each neuron is connected solely to the neighboring ones in the successive layers - with the number of connections,

commonly referred to as receptive field, defined by the filter/kernel size - thus preventing the number of parameters of the network to explode when highly dimensional data are provided. A typical example of such high dimensional data is an image, whose resolution strongly influences the total parameters of the network. Because of their natural predisposition to handle images, CNNs find most of their applications in computer vision tasks, such as image classification, object detection, semantic segmentation and pose estimation.

2. **Recurrent Neural Networks.** Unlike FFNNs, recurrent neural networks (RNNs) involve cycles or feed-back loops. Additionally, the internal connections of a RNN allow the network to process inputs with variable lengths. Once the input is passed to the network, the RNN generates an output and a so-called hidden state. At each time step, the hidden state is generated as a function of the current input and the hidden state from the previous time step. This hidden state is then passed as input to the next time step, allowing the network to maintain a "memory" of the past inputs. Thanks to this property, recurrent networks are commonly employed to handle sequential data. Traditional applications include time series prediction, image captioning, natural language processing (NLP) and video analysis. The most common RNNs architectures include:

   - Long Short-Term Memory NNs: This type of network is designed to resolve the *vanishing gradient* phenomenon that affects basic RNN architectures, which is accomplished by controlling the flow of information through a gating mechanism. Such mechanism includes an input gate, which determines which input values should be stored in the memory cell, a forget gate, which defines which values from the cell should be forgotten, and an output gate, which determines the next hidden state based on the updated cell state.

   - Gated Recurrent Units: Still making use of the gating mechanism for information processing, this is a simpler version of the LSTM architecture. Unlike the LSTM, GRUs only contain the update and reset gates, and do not feature a memory cell. Because of their simpler structure, GRUs are computationally less expensive than

LSTMs, hence faster to train. In comparison to the LSTMs, they generally perform better on tasks where long term memory is not strictly necessary.

3.3.3 Transformer Architecture

Since its introduction in 2017 with the seminal paper “Attention Is All You Need” by Vaswani et al. [2], the transformer neural network has gained high popularity in many fields, reaching and beating state of the art performance in a plethora of tasks, from NLP (text classification, generation, summarization and segmentation, question answering, translation etc.) [89, 90] to computer vision [91, 92] and time series forecasting [93]. The transformer architecture origins can be traced to inherent limitations of established models - namely, recurrent neural networks of different flavors - to tackle sequence-to-sequence modeling tasks. In fact, the sequential nature of such architectures prevents parallelization within training examples, which is essential when dealing with batching across example of long sequences when working under memory constraints. The success of the transformer model derives from the fact that recurrence is discarded in favor of the so called attention mechanism, which can efficiently capture the meaning of the input context, enabling to learn global dependencies between different elements, independently from their distance in the sequence.

The general transformer architecture (Fig. 3.5) is composed by encoder and decoder stacks. In this context, the encoder maps the symbol representations of the input sequence to continuous representations, which are then taken by the decoders block to generate the output. The model works in an auto-regressive manner, that is, at each step the output is concatenated to the input to generate the next one. In the next sections, a detailed description of the transformer architecture is provided, following the flow direction of information through the model.

Input Embedding. While the transformer network demonstrates to be very powerful on a plethora of tasks, performance would be poor if raw data were to be directly passed to the model. As such, the first step of processing data through the transformer is to create embeddings of the input. Embeddings can be described as a lower-dimensional, continuous representation of the

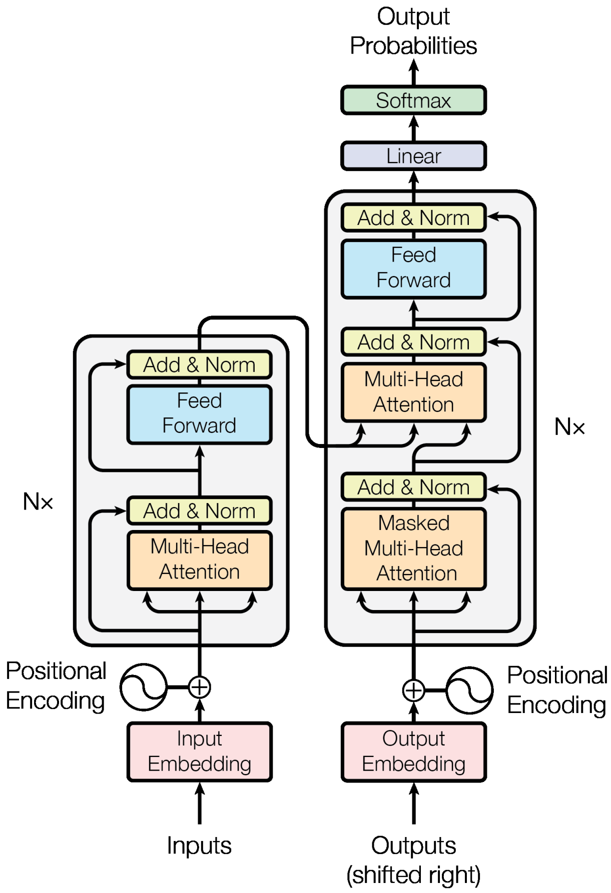


Figure 3.5: Transformer architecture. Original illustration from [2].

data, which encode semantic information. One of the main benefits of such a transformation stands into the additional generalization capability. In fact, embeddings capture semantic similarity, producing similar representations for similar inputs. In such a way, the model can generalize on unseen input, which may have a similar projection in the embedding space. Of note, the embeddings parameters are learnable, i.e., they are learned throughout the network training.

Positional Encoding. As the transformer model does not rely on recurrence, information about the sequence order must be injected. To this aim, positional encodings are added to the input embedding, allowing the model to have a knowledge about the input order. There are two main categories of positional encoding: fixed and learnable. In the original formulation of the transformer model, sine and cosine functions are utilized to inject the position information:

$$
\begin{aligned}
PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\
PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}})
\end{aligned} \tag{3.1}
$$

where pos is the position, i is the dimension and d_{model} is the size of the input embedding.

Encoder. The main purpose of the encoder is to process and encode input sequences to extract meaningful information, enabling the model to effectively capture complex patterns and dependencies in the data. In the original formulation, the transformer encoder is composed by a stack of six identical blocks, allowing the model to learn hierarchical representations of the input. Furthermore, by having a stack of consecutive blocks, representations are refined, granting the possibility to capture more abstract and high-level features. Each of these blocks comprises two sub-layers:

1. *Multi-head attention.* It constitutes the core learning step in the transformer architecture, and it is based on the fundamental concept of attention. In fact, thanks to the attention mechanism the transformer network is capable of learning dependencies within input sequences, independently from their distance in the sequence. Furthermore, it allows the model to focus solely on the most relevant parts of the input sequences, discarding irrelevant information. The multi-head attention is built upon the combination of multiple self-attentions, a mechanism based on the definition of three matrices: Q (the queries), K (the keys) and V (the values). Such matrices are constituted by learnable parameters, and are defined as:

$$Q = X_{embed} * W_Q \qquad K = X_{embed} * W_K \qquad V = X_{embed} * W_V \tag{3.2}$$

   with W_Q, W_K and W_V being learnable weights matrices, and X_{embed} being the input embedding matrix. In particular, the transformer presented in [2] utilizes the so called "scaled dot-product attention", which can be expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3.3}$$

where the $\mathrm{softmax}$ function is utilized to prevent singularities while constraining the dot product to assume values between 0 and 1, and d_k is the dimension of queries and keys (while V has a different dimension, d_v, though often times it is set equal to d_k). In Eq. (3.3), the multiplication between Q and K defines a linear transformation that improves the embedding representation to enhance the attention process. In particular, their combination is optimized to identify similarities between elements in the input sequence, as they define embeddings which still have knowledge of input features. Next, recalling the original language task of the transformer model, a new linear mapping is introduced via the matrix V, which generates a new projection of the embeddings toward a space more favorable to the prediction of the next token in a text sequence. This is due to the fact that this new embedding space better captures the input sequence context, thus knowing when two elements in the same sequence are more likely to appear together and be correlated. Of note, the formula presents a normalization by the queries/keys dimension, which is required to prevent large values, especially for very long sequences. Despite application of Eq. (3.3) alone would provide good results, the authors of [2] found that performance are improved if instead of utilizing a d_{model} dimensional queries, keys and values, such quantities are linearly projected h times, with dimensions d_k, d_k and d_v. Following this intuition, the multi-head attention can be formalized as:

$$\begin{gathered}\mathrm{MultiHead}(Q,K,V)=\mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O\\ \text{where}\quad \mathrm{head_i}=\mathrm{Attention}(QW_i^Q,KW_i^K,VW_i^V)\end{gathered} \tag{3.4}$$

In Eq. (3.4), $W_i^Q, W_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{model}\times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v\times d_{model}}$ are learnable parameters matrices.

2. *Position-wise Feed-Forward Networks.* The multi-head attention sub-layers is followed by position-wise feed forward layers, which mainly serve to introduce non-linearity in the model and to reduce the dimensionality of the attention layer output. Unlike the multi-head attention layer, where each position in the sequence is attended simultaneously, this sub-layer applies separate linear transformations, operating independently on each

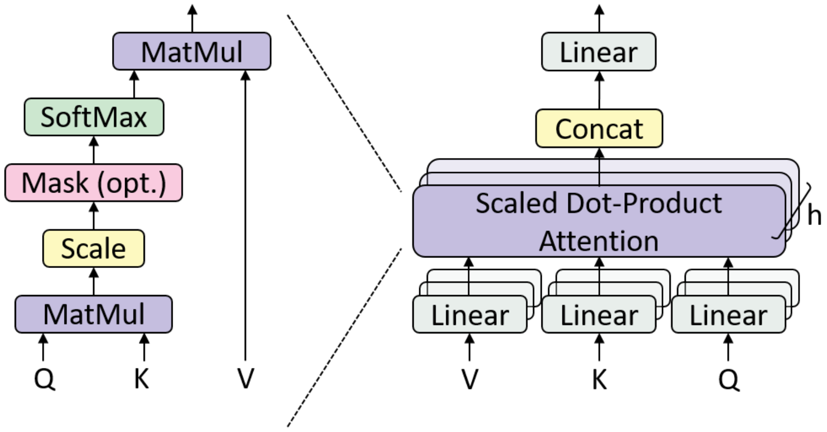


Figure 3.6: Scaled Dot-Product and Multi-Head Attention schematic representation. Image inspired from [2].

position. In such a way, the model processes different representations separately, thus capturing local patterns. The output of the feed-forward layer is then regularized to prevent overfitting. Formally, the feed-forward layer block can be described as:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3.5}$$

where x are the processed information, the $\max$ operator defines the ReLU activation function and W_1, b_1, W_2, b_2 are weights and biases of the layers.

As a conclusive remark, residual connections [94] are applied throughout each encoder block to address the vanishing gradient problem, thus allowing the possibility to define deeper architecture, and to preserve the original information as they flow through the network.

Decoder. The decoder has a structure similar to the encoder, and it consists of N identical blocks (six in the original formulation). Each decoder block presents two key differences with respect to the encoder ones. The first difference appears (Fig. 3.5) in the multi-head attention layer, which is here dubbed "masked". The reason is that the input presents a masking such that the attention does not attend positions further ahead in the sequence. By applying the masking, the $\mathrm{softmax}$ function assigns zero probability to the tokens in future positions, thus effectively preventing the network from looking ahead in the sequence. The second key difference is constituted by the presence of an additional multi-head attention layer, commonly referred to as “cross-attention” layer, which allows to share information with the encoder. Specifically, the input queries and keys in this layer originate from the encoder, while the values derive from the output of the previous multi-head attention layer in the decoder. Similarly to the encoder, regularization and skip connections are applied throughout the decoder blocks. Of note, the decoder displays a different behavior between training and inference in that during the prediction phase it works in auto-regressive fashion.

3.4 Problem Formulation and Modeling

In this work, we consider a simplified formulation of the ICARUS attack presented in [45]. Specifically, we assume the DDoS attack happens, but we do not model the attack itself, nor we specify the location of the botnet nodes. Rather, we model the effect of the attack, and we aim to identify anomalous satellite links resulting from re-routing of signals over paths connecting two arbitrary regions of the world targeted by the attack. Leveraging the connected nature of the problem, we model a large satellite constellation as a dynamic graph [95], assigning a direction to each edge belonging to a path connecting the two regions, here for simplicity, represented as two unique nodes. We employ a traditional "+ grid" structure, with each satellite linked to the preceding and successive ones on the same orbit, and with the two closest satellites in the adjacent right and left orbit planes [96]. Under these settings, following the methodology described in [1], each satellite of the constellation can be assigned a unique ID that will remain the same throughout the entire simulation. In order to maintain a simple node numbering, satellite vertices in the constellation are identified by first setting a reference orbit plane (for us, the one associated with the RAAN closest to zero as measured in a counterclockwise direction); next, we define a reference satellite (which, in our case, is the one closest to the equator with positive anomaly) and we start numbering the satellites on the reference plane following the direction of the orbital motion. Once the satellites on the reference plane are assigned their unique ID, those in the adjacent planes are numbered following the same procedure, starting from the last ID in the previous plane. The described procedure is summarized in Fig. 3.7.

Defining as n the number of satellites in a single orbital plane, once the spacecraft in the reference plane are identified, the reference satellites in the adjacent planes (taken in a counterclockwise direction) will be numbered as $m * n$+1, with m=1,..,M-1, being M the number of planes in the constellation. The same procedure followed in the first reference plane is then applied to all the satellites in the remaining ones. Defining as N the total number of satellites in the constellation, the ground nodes will have ID $N + 1$ and $N + 2$. According to this notation, edges belonging to the paths identified between the two ground nodes will be, in general, defined as $\{ v_i$,$v_j \}$, with i being the *tail* node and j the *head* node defined following the

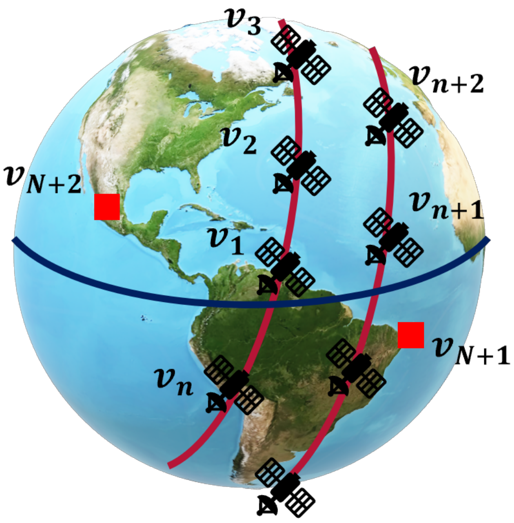


Figure 3.7: Definition for the initial constellation graph, with the two red square representing two ground locations.

chosen "+ grid" linking strategy. An exception is constituted by the two ground nodes, which will be the nodes v_i and v_j for the first and last edge in each path.

To generate the stream of graphs that will be processed by the neural network, we then employ the following procedure:

1. We select two ground nodes (representative of the two regions) that have to be linked following a satellite interlink path
2. We run the satellite dynamics, computing at each time step the top k-shortest paths using Dijkstra's algorithm [97], where the higher the value of k, the greater the path diversity is

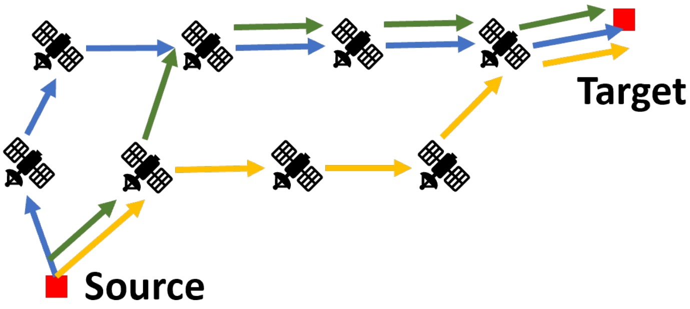


Figure 3.8: Schematic representation of multiple paths connecting a source and a target node.

3. We reduce the dataset by removing repeated edges throughout the dataset. This is equivalent to considering the graph in Fig. 3.8 without parallel edges between the same pairs of nodes

4. We aggregate graphs belonging to multiple time steps in a unique graph, thus creating a synthesis of the graph over a time span. Such a graph is referred to as *snapshot*

5. We group multiple snapshots into sequences of equal number of snapshots

Differently from the procedure followed in [3], anomalies are artificially created in the graph by connecting nodes solely within the snapshot under investigation; that is, the anomalous edges are created using the node set constituting the examined snapshot. In this way, we avoid the creation of anomalous edges that would violate the mechanics of the problem (i.e., we do not have edges between nodes that belong to the entire graph but not to the sampled sub-graph). As such, the new edge mimics a more plausible connection within the snapshot.

3.5 Methodology

The TADDY architecture can be applied to the problem of anomalous link detection on non-terrestrial networks, provided that key modifications are applied. In the following, we first describe the algorithm's core ideas; next, we briefly present the architecture validation over common benchmark datasets. Then, as the constellation problem displays significant underlying differences with respect to usually evaluated non-space scenarios, we discuss how the method can be adapted for our specific use case. Finally, we will show the application of the modified algorithm on the two ground points connection scenario, with extensive considerations on the main problem parameters.

3.5.1 Transformer-based Anomaly Detector

We perform the anomaly detection task by leveraging a transformer-based [2] neural network architecture inspired by the work of Liu et al. [3]. To represent the dynamic nature of the graph, a sequence of snapshots is passed to the neural network, with the goal of evaluating the legitimacy of edges in the last snapshot of the sequence based on past observations. Following the methodology described in [3], the anomaly detection process is divided into four consecutive steps: 1) an edge-based substructure sampling; 2) a novel spatial-temporal encoding; 3) a pass through the dynamic graph transformer; 4) a pass through a discriminative anomaly detector (Fig. 3.9). In the next paragraphs, salient aspects for each of these steps are summarized; for more details, the reader is invited to refer to the original work [3].

Edge-based Substructure Sampling. The behavior of an edge in a graph can be, at minimum, characterized by the structural relations with its surroundings. This statement is corroborated by existing works [85], which suggests anomalies are likely to appear in local graph sub-structures. For this reason, the first step in the algorithm pipeline is to extract a fixed-size, importance-aware contextual nodes set in the neighborhood of the edge to be evaluated, from now on referred to as "target edge" and denoted as e_{tgt} (while we will refer to the nodes defining such edge as "target nodes"). One traditional method to extract such sub-structure is the so-called h-hop

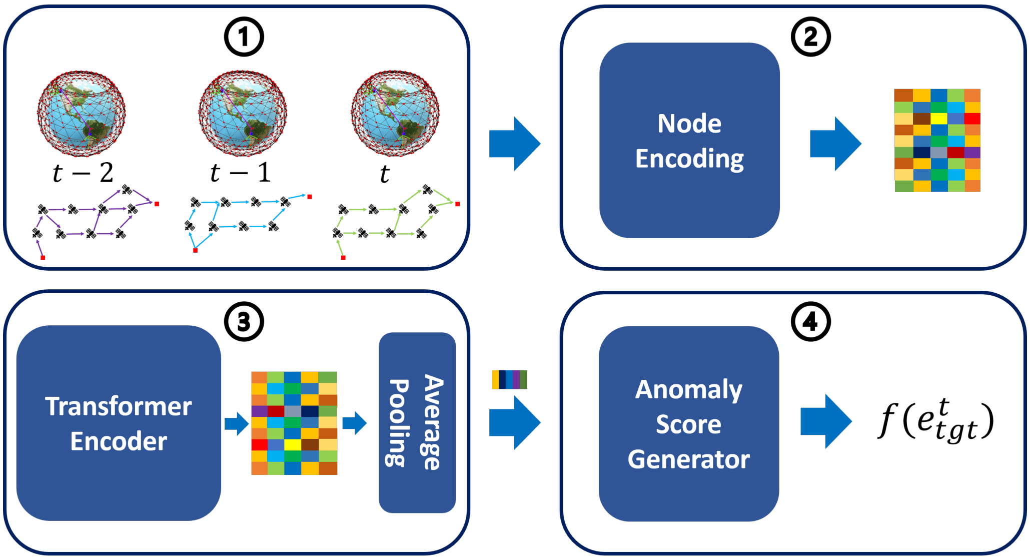


Figure 3.9: Schematic representation of the methodology for a snapshot sequence of length equal to three. The letter e in the first step denotes a generic edge at a given time step to be evaluated.

sampling, where h defines the "hopping distance[1]" between the target nodes and the surrounding nodes. However, despite such technique typically convenient, it is limited in that it assumes a fixed value for h, but it remains agnostic to the degree of each node and the role that each node assumes in the sub-structure, with the risk of introducing noisy data in the sampled structure, especially for graphs with high disparity in nodes degree. To overcome these limitations, this algorithm exploits a graph-diffusion technique derived from Personalized PageRank (PPR) [98], where a graph-diffusion matrix is computed according to:

$$\mathbf{S_{PPR}} = \alpha(\mathbf{I} - (1-\alpha)\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^{-1} \tag{3.6}$$

In Eq. (3.6), $\mathbf{I}$ is the identity matrix, $\mathbf{A}$ is the adjacency matrix, $\mathbf{D}$ is the graph degree matrix (diagonal), and $\alpha \in (0, 1]$ is the teleport (or restart) probability, where α is the probability of jumping to a uniformly random vertex of a graph, while 1-α is the probability to follow a random edge out of the currently considered node. Given the $\mathbf{S_{PPR}}$ matrix, each i-th row is representative of the degree of connectivity between the i-th node in the graph and every other node, thus capturing the connectivity of the nodes at the global level. Following this

[1] the hopping distance is quantified as the number of edges separating two nodes, that is, the length of the path between two nodes assuming an unweighted graph (or, equivalently, a weighted graph with each edge assigned a weight of 1).

consideration, the connectivity vector of the target edge can be computed by summation of the rows associated with the corresponding target nodes. Formally, considering e_{tgt} = $\{v_1, v_2\}$, the connectivity vector associated to the target edge is expressed as:

$$\mathbf{s}_{e_{tgt}} = \mathbf{s}_{v_1} + \mathbf{s}_{v_2} \tag{3.7}$$

Finally, the contextual nodes set entering the sampled substructure is obtained by sorting and extracting the highest values from the connectivity vector. Of note, the target nodes are excluded from the ranking[2]. For a stream of graphs, given a target edge, this procedure is applied to all the graphs within a pre-determined temporal window, also referred to as “look-back horizon”. Next, the node set for a target edge at a given time stamp is given by the union of all the nodes set within the look-back horizon window.

Spatial-temporal Encoding. In order to provide an informative representation of the dynamic graph, both spatial and temporal information must be encoded. To this aim, the TADDY architecture exploits a novel spatial-temporal encoding that is capable of capturing both local and global spatial information, as well as the temporal signal originating from the system (that is, three encoding terms are introduced). In particular, such encoding is generated through learnable parameters, which are found to be easier to train and more flexible in highlighting spatial-temporal correlations.

For the local information, a distance-based encoding is used. Specifically, as a local feature, the distance between each node $v_j^i \in V_{set}^i(e_{tgt})$, i.e., each j-th node at the i-th time step belonging to the vertex set of the target edge at the i-th time step, and the target nodes is computed, retaining the minimum value. If no connection exists, the distance value is automatically set to an arbitrarily high value. Hence, the distance-based spatial encoding for the node v_j^i can be expressed as:

$$\mathbf{x}_{distance}(v_j^i) = \mathrm{linear}(\min(\mathrm{dist}(v_j^i, v_1), \mathrm{dist}(v_j^i, v_2))) \in \mathbb{R}^{d_{enc}} \tag{3.8}$$

[2] At code level, this can be obtained by simply assigning an arbitrarily low value to the target nodes.

with linear, min and dist being the learnable linear mapping, the minimum value extraction function and the distance computing function respectively. Through this mapping, the local information is projected to an encoding space of dimension d_{enc}.

To extract global spatial information, the graph diffusion matrix can be exploited. In particular, only the values in the $\mathbf{S_{PPR}}$ associated with the nodes $v_j^i \in V_{set}^i(e_{tgt})$ are used. However, instead of directly using the diffusion values as a source of the encoding (whose values might be too close to each other to provide meaningful information of the structural role of each node), the TADDY algorithm exploits a rank-based encoding, that is, the rank position of each node in $V_{set}^i(e_{tgt})$ is used to represent the nodes relations in the substructure. The diffusion-based encoding can be then formally expressed as:

$$\mathbf{x}_{diffusion}(v_j^i) = \mathrm{linear}(\mathrm{rank}(\mathbf{s}_{\mathrm{e_{tgt}}}[\mathrm{idx}(\mathrm{v_j^i})])) \in \mathbb{R}^{d_{enc}} \tag{3.9}$$

with rank and idx being the the ranking operation and the index-inquiring function respectively.

Finally, to encode the temporal information, the relative time between the time step t to which the target edge belongs and the i-th time step in the considered snapshots sequence[3] is used, that is, $t - i$ is used as source value to be encoded. The formal expression is provided below.

$$\mathbf{x}_{temporal}(v_j^i) = \mathrm{linear}(||t - i||) \in \mathbb{R}^{d_{enc}} \tag{3.10}$$

Once all three encodings are generated, the spatial-temporal encoding for each node $v_j^i \in V_{set}^i(e_{tgt})$ is computed as the summation of the three encoding vectors, that is:

$$\mathbf{x}(v_j^i) = \mathbf{x}_{distance}(v_j^i) + \mathbf{x}_{diffusion}(v_j^i) + \mathbf{x}_{temporal}(v_j^i) \in \mathbb{R}^{d_{enc}} \tag{3.11}$$

Finally, the target edge encoding matrix $\mathbf{X}(e_{tgt}^t)$ is computed by concatenating the encoding vectors of all the nodes in the substructures of each time step in the considered graph stream.

[3] For example, the relative time distance between edges belonging to the same snapshot is 0; hence, the value 0 is encoded. For two consecutive snapshots, the relative distance is 1; therefore, the number 1 is encoded (and so on).

Transformer Module. While many works adopt stacking of spatial and temporal feature extractor and processing (such as discussed for the StrGNN [85] architecture), the TADDY algorithm exploits a transformer network using solely the encoder block, with the aim of projecting the encoding into a new feature space. The choice of adopting a transformer is here driven by the presence of the attention mechanism, which is leveraged to capture spatial-temporal correlations within the encoding. In this work, the `PyTorch`[4] library transformer encoder module is used, which is based on the original implementation proposed in [2]. As input to the encoder, the target edge encoding matrix, $\mathbf{X}(e^t_{tgt})$, is used, and no positional encoding is applied. The output of the transformer module is a new matrix, $\mathbf{Z}(e^t_{tgt})$, which is the embedding matrix of the target edge, with dimensions equivalent to those of the encoding matrix.

As head of the transformer, a pooling (or readout) layer is attached to obtain a dense vector representation of the edge embedding. While there exist more complex techniques such as DiffPool[99], we follow the original algorithm implementation by using a simpler MeanPooling[100, 101], which has the advantage of not introducing additional learnable parameters. The MeanPool layer is expressed as:

$$\mathbf{z}(e^t_{tgt}) = \text{MeanPool}(\mathbf{Z}(e^t_{tgt})) = \sum_{j=1}^{N_i} \frac{\mathbf{Z}(e_{tgt})_j}{N_i} \tag{3.12}$$

where $\mathbf{Z}(e^t_{tgt})$ is the j-th row of the embedding matrix $\mathbf{Z}(e^t_{tgt})$ and N_i is the size of the encoded substructure. Defining as τ the length of the look-back horizon, N_i corresponds to $\tau(k+2)$, being k the number of contextual nodes, with the "2" accounting for the target nodes.

Anomaly Detector. The last step in the algorithm pipeline is to produce an anomaly score, here generically denoted as $f(e)$, for each processed edge. To this aim, a linear layer taking as input the edge embedding vector is used; next, a sigmoid function is applied to constrain the output between 0 and 1: the closer the score is to 1, the higher the probability the processed edge is anomalous, while values closer to 0 more likely represent a regular edge. The whole network is then trained in a contrastive fashion [102]. This methodology is commonly used for training machine learning models when such models must be able to identify inputs that

[4] `https://pytorch.org/`

are similar/dissimilar from each other. Within our problem, the algorithm takes both positive and negative examples, which are both projected by the architecture onto the embedding space. From this perspective, positive edges will have projections into the embedding space which are closer to each other, and far from the representation of negative edges (and vice versa). The contrastive learning aspect is then conveyed through the binary cross-entropy (BCE) loss function, which has two "contrastive" terms: one term accounting for positive edges prediction and one for negative sample prediction. The expression for the loss function is given as:

$$L = -\sum_{i=1}^{m^t} \log(1 - f(e_{pos,i})) + \log(f(e_{neg,i})) \tag{3.13}$$

where m^t is the number of edges at timestep t. Given an input edge, the desired output will then be the one that minimizes both terms. Please note that, at the implementation level, the binary cross-entropy with logits loss function is used instead of applying a sigmoid followed by BCE for numerical stability purpose[5].

3.5.2 Algorithm Performance

Network performances are evaluated via the Area under the ROC[6] curve (AUC), which represents the probability that an anomalous edge will be recognized as such against the examination of a normal edge. Quantitatively, the AUC score is computed as the area subtended to the ROC curve, which is obtained by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR), for all possible thresholds, computed as it follows:

$$TPR = \frac{TP}{TP + FN} \tag{3.14}$$

$$FPR = \frac{FP}{TN + FP} \tag{3.15}$$

where TP means True Positive (that is, the number of predictions correctly labelled as positive), TN means True Negative (that is, the number of predictions correctly recognized as negative),

[5] https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

[6] Receiver Operating Characteristic

FP is False Positive (that is, the predictions labelled as positive but that belong to the negative class) and FN is False Negative (opposite of false positive). As a reference, a perfect AUC score has a value of 1, meaning that, given two classes (here corresponding to regular and anomalous edges), the network is capable of perfectly distinguishing between the two; instead, a value of 0.5 would mean that the network has no discriminative capability, thus being akin to a random classifier. An example of receiver operating curve is provided in Fig. 3.10.

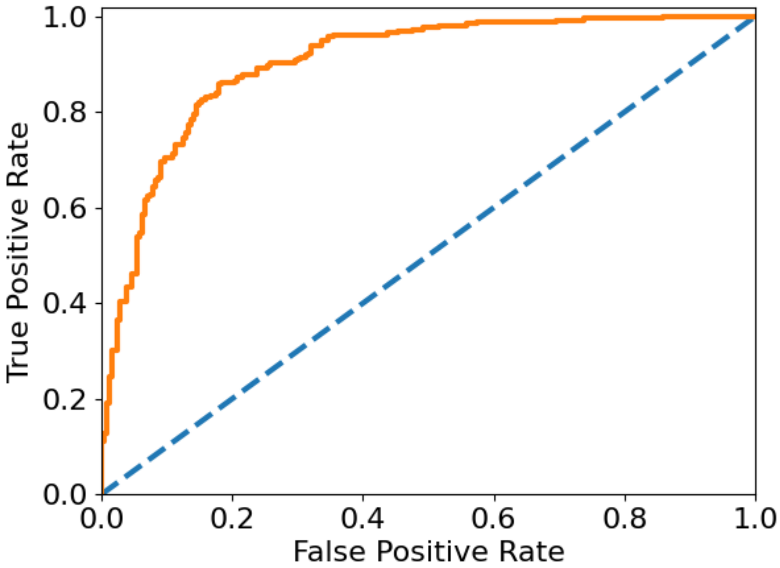


Figure 3.10: Example of ROC (orange, solid line). Dashed blue line representing a random classifier.

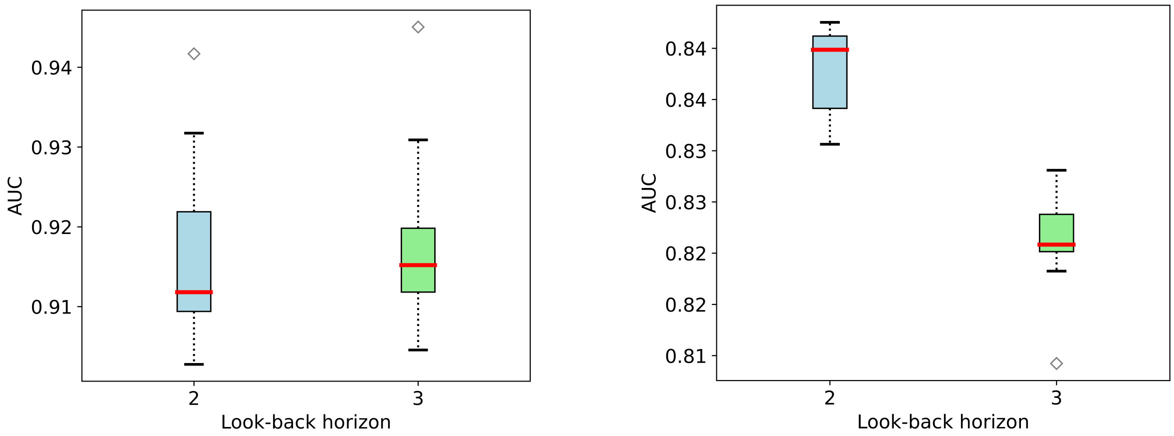
3.5.3 Algorithm Preliminary Evaluation

Before moving to the satellite constellation problem, it is crucial to validate our implementation over commonly adopted benchmark datasets against the performance declared in the original paper. To this purpose, we select two datasets: UCI messages[7] and Bitcoin-Alpha[8]. The first is a social network dataset gathered at the University of California, Irvine. Each node represents a student, while each edge is a message between two students. The second is a network of bitcoin users trading on an online platform; in this case, nodes are users, while an edge is created when a user rates another one. The choice of these two specific datasets was determined by the fact

[7] http://konect.cc/networks/opsahl-ucsocial

[8] http://snap.stanford.edu/data/soc-sign-bitcoin-alpha

that the size of the graph is comparable to the one of a large satellite network ($|V(G_{UCI})|$ = 1899 and $|E(G_{UCI})|$ = 59835, while $|V(G_{BTCAlpha})|$ = 3783 and$|E(G_{BTCAlpha})|$ = 24186). For both datasets, we consider two lengths for the look-back horizon, τ = 2 and τ = 3, and we train ten separate networks, each with random initialization of parameters and distribution for the anomaly generation. Next, we evaluate the highest-performing model on the same dataset and compare the performance in terms of AUC score.



(a) **(b)**

Figure 3.11: (a) Bitcoin-Alpha test results and (b) UCI messages test results. 10% of anomalous edges injected in each test set. Diamond shape representing outliers.

Fig. 3.11 displays the obtained result. In both cases, the performance aligns with those reported in [3], where, for analogous simulation settings, the AUC score for UCI messages is reported to be 0.8370, while for Bitcoin Alpha, it is 0.9423. Additionally, according to the findings in the paper, the Bitcoin Alpha dataset benefits from larger time horizons, while for the UCI messages, a longer look-back is detrimental as the dataset is characterized by short-term spatial-temporal dependencies. Both qualitative trends can be observed in our results, though with more emphasis in the UCI case. Given the obtained results, we believe that the observed differences in exact values have to be attributed to implementation aspects, such as the batch size utilized during the training and the transformer encoder architecture details.

3.5.4 Graph Anomalies Generation

Before presenting and discussing the results for our specific case study, it is crucial to clarify how anomalies are treated during training and testing (both in the above validation and our problem). At training time, the technique known as *negative sampling* is exploited. In this technique, the randomly sampled edges are processed by the network, but they do not belong to the graph itself. This can be formalized as it follows: given a regular graph, which we can denote as $G_+ = (V, E)$, random connections are generated among nodes $\in V$, creating a negative edge set, E_-, with $|E_-|$=$|E|$; next, a negative graph is defined as $G_- = (V, E_-)$. This method is used to ease the training while allowing the neural network to learn salient features representing an anomaly. However, a different, more realistic technique known as *anomaly injection* is applied during testing. In this case, randomly sampled edges are generated and injected into the raw data, that is, in the original dataset, before any pre-processing is applied. Referring to the previous formalism, this corresponds to creating a graph $G = (V, E \cup E_-)$, here with $|E_-| \ll |E|$. The relevance of this latter concept is twofold: firstly, while the goal of the algorithm is to classify edges belonging to the last snapshot in a sequence, anomalies may be present in the previous graphs of the sequence, hindering the detection task; secondly, the identification of irregular edges is more challenging, as the injected edges become part of the original dataset, that is, if the dataset do not provide sufficient spatial-temporal information, anomalous injected edges may go undetected. By employing negative sampling and anomaly injection, a network can be trained to identify hidden features characterizing anomalies, thus becoming capable of detecting irregular edges in the graph stream. These two concepts are schematically represented in Fig. 3.12. As a conclusive remark, it is important to re-iterate that both negative sampling and anomaly injection are performed in a slightly different manner with respect to the procedure utilized in [3]. In fact, while in the original work anomalous connections consider the entire graph node set (which is possible due to the nature of the utilized dataset such as UCI messages, where a link is simply a contact between two users), this operation is not directly applicable to our case due to the necessity to respect the constellation configuration, where a connection can be created, at best, with satellites in sight. As such, both anomaly generation procedures in our problem solely involve the nodes present at the considered time step.

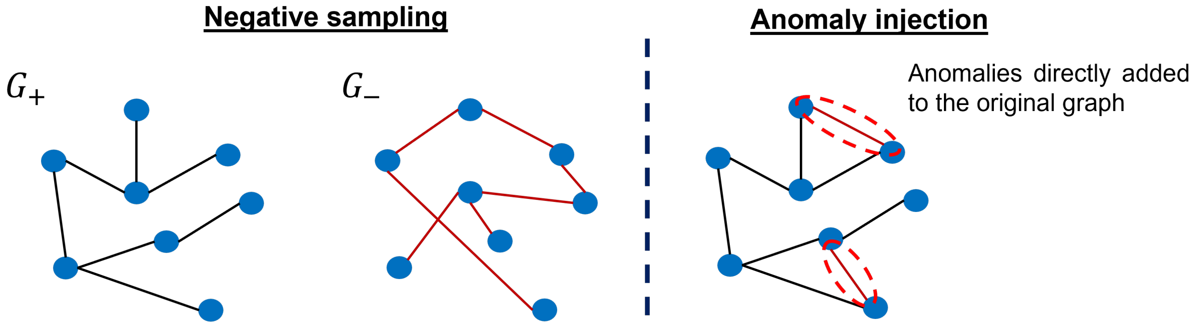


Figure 3.12: (left) negative sampling representation: a negative graph, G_-, is created from a regular graph, G_+, by randomly connecting nodes in the positive graph; (right) anomaly injection representation: anomalies become part of the positive graph.

3.6 Results and Discussion

In this section, we present and examine the outcomes garnered for the generated constellation dataset. Our study includes an analysis of the impact of simulation settings on performance and an investigation of the effect of the constellation configuration, specifically the number of satellites and ground nodes. While presenting the results, we additionally highlight inherent differences with respect to the datasets used for the validation, and we discuss how we bridge the gap through changes within the algorithm.

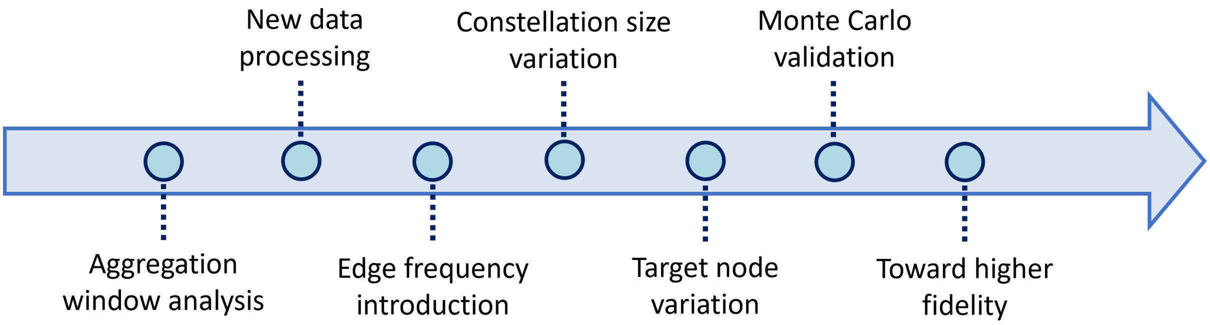


Figure 3.13: Results section graphical roadmap.

To test how simulation parameters affect performance, we freeze the constellation configuration, that is, the dynamic evolution of the constellation architecture parameters, such as a change of number of satellites, are omitted. In this study, we simulate a constellation with 40 orbital planes, 40 satellites per plane, and two ground nodes (whose location is arbitrarily fixed to correspond to the cities of Auburn (AL, USA) and Anchorage (AK, USA)), for a total of 1602 nodes in the entire graph. Each orbit is circular, with an orbit altitude set to 700 km and an orbit inclination fixed to 65°. We run the simulation for 24 hours, which is a time sufficiently long for all the nodes to be at least once part of one of the k-th paths, which turns out to be useful at the implementation level. We set the time step to 2 minutes (with each time step generating a graph), thus promoting diversity in successive snapshots while still allowing temporal information to be passed through graph sequences. In fact, such cadence allows edges to persist across multiple snapshots, thus granting the possibility to capture their behavior over a given time horizon. The choice of time step size is problem-dependent and an essential factor in the determination of the algorithm's behavior. For a time step size tending to zero, there would be little to no difference in consecutive snapshots; as such, for a sequence of graphs not sufficiently long, a graph may appear static. Conversely, a time step size significantly larger than the characteristic time of the underlying dynamics would lead to extreme differences in consecutive snapshots, thus missing relevant dynamical relations and preventing the network from learning meaningful edges' behavior.

All simulations are conducted using the same network architecture configuration. For the results that follow, the encoding dimension, d_{enc}, is set to 32; informal testing has been conducted with a dimension of 64 across multiple cases, which displayed little difference with respect to the other condition, and not sufficient to justify a larger network size (which would have required longer training time). This setting holds for both the encoding layer and the transformer network input dimension. Differently from the original work [2], the transformer is an encoder-only network (no decoder) with two encoder sub-blocks; two heads are used for the multi-head attention mechanism. For each case, the architecture is trained for 100 episodes, using Adam optimizer, with a learning rate of 1e-3 without decay. Training and testing have been conducted in parallel on two separate machines: 1) desktop Alienware Aurora R8 with 9^{th} Generation Intel Core processor (8 cores), and 2) a laptop Dell Inspiron 13 7380 equipped with an 8^{th} Generation Intel Core processor (4 cores).

3.6.1 Aggregation Window Analysis

The first parameter we investigate is the snapshot size. The definition of this parameter assumes particular relevance in a system with a fast dynamics as the one here considered, since it directly influences how spatial-temporal information are correlated across time steps. It is here crucial to emphasize how the definition of this parameter substantially differs from what is done in [3], due to the diverse nature of the dataset. In fact, in [3], the authors fix a snapshot size for the datasets (1000 and 2000 are chosen for the six datasets tested in the paper) without considering the true time stamp of each edge. However, this operation is not possible for a satellite constellation scenario, as edges sharing the same time stamp must be grouped to preserve the system, hence the graph, physical structure. Rather than defining a fixed snapshot size, we introduce an aggregation window (AGW): we first split the edge set into single time step snapshots; next, we gather multiple snapshots to create a temporal synthesis of the graph over a pre-determined time window. Given a sequence of these snapshots with length τ, anomalies are injected by considering the node set constituted solely by the nodes appearing in the last graph of the sequence. In this way, we avoid the unrealistic injection of connection between satellites not belonging (at minimum) to the same time window.

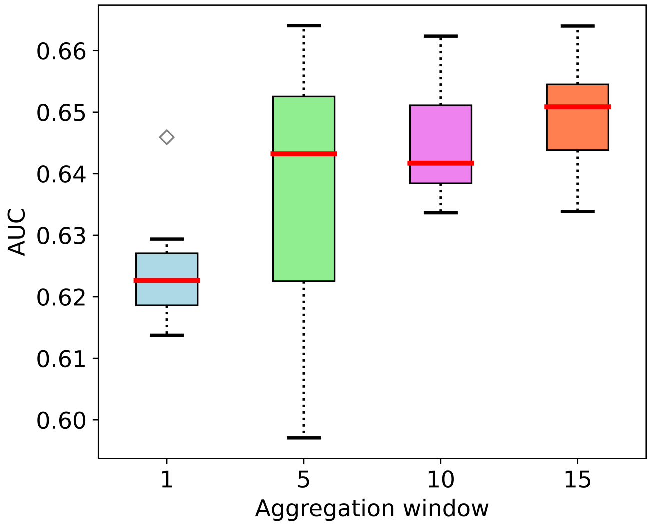


Figure 3.14: Aggregation window sensitivity. Look-back horizon fixed to τ = 2.

Fig. 3.14 reports the results obtained on the aggregation window parameter investigation. Analogously to what was discussed in Sec. 3.5.3, ten separate networks are trained for each of the selected aggregation window values. Next, the best model obtained from each training is compared against the other models belonging to the same AGW over the same evaluation edge set. Observation of Fig. 3.14 reveals how performance improves with increasing value of aggregation window; this could be interpreted as an indicator of the fact that longer temporal horizons may be beneficial for detecting anomalies. Please note how, in this case, the temporal horizon is influenced by two factors, that are, the aggregation window and the look-back horizon, here fixed to τ = 2 for all the simulations. In other words, the temporal window observed by the network can be interpreted as $\tau \times$AGW.

A closer look at the results shows how the performance is consistently degraded with respect to those obtained during our preliminary implementation evaluation, with the top-performing model reaching an AUC$\sim$0.66, against the AUC>0.80 in all the validation cases. We believe that the cause for such a gap is to be found in the dataset and, more specifically, in the structure of the resulting graphs. By the nature of our problem, each graph representing the constellation system is connected[9], meaning that it is a 1-component graph; conversely, the graph associated with the results in Fig. 3.11 is intrinsically disconnected, that is, there are multiple components. This factor assumes extraordinary importance when considering 1) the anomaly injection procedure in general, 2) the way spatial information is extracted from the graph, and 3) the injection of anomalies using solely the node set of the examined snapshot. In fact, since the graph is connected, we speculate that the additional edge will more likely be blended with regular edges, reducing the expressive power of the spatial encoding. This leaves the temporal encoding the burden to inform the network that the added edge constitutes an anomaly. However, as demonstrated by the ablation study conducted in [3], the temporal information plays a minor role in comparison to the structural encoding; consequently, since the spatial information is consistently weakened, the algorithm achieves lower performance for the constellation case.

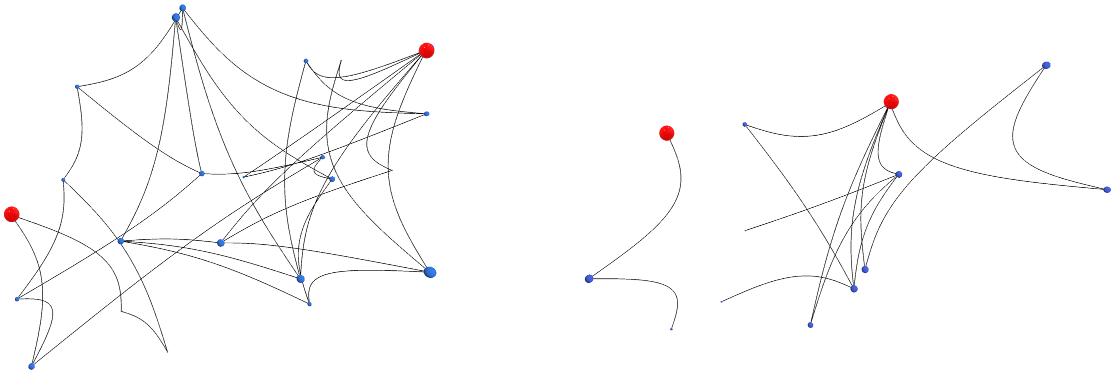

(a) (b)

Figure 3.15: (a) single component graph, 1 time step snapshot and (b) multi-component graph, 1 time step snapshot. Red nodes representative of the 2 ground points.

Fig. 3.15 portrays two graph snapshots considering a single time step, AGW = 1, while following the data processing exploited in [3]. While the claim of the graph being connected is

[9] this statement is true with an unprocessed dataset. The current processing technique may still lead to some graphs being disconnected, despite this is mitigated by the aggregation of multiple time stamps in a unique snapshot.

true in general, the edge processing procedure involves the elimination of double edges. As a consequence, when eliminating repeated edges throughout the whole edge set *before* grouping time steps through aggregation windows, there exists a chance that some single time step snapshot will result in a multi-component graph (such as in the graph on the right in Fig. 3.15). Intuitively, when grouping multiple time steps within a given time horizon, the likelihood of having disconnected graphs strongly reduces, and all the graphs processed by the network will be connected. An example is reported in Fig. 3.16, where snapshots for an aggregation window of 5 and 10 are reported.

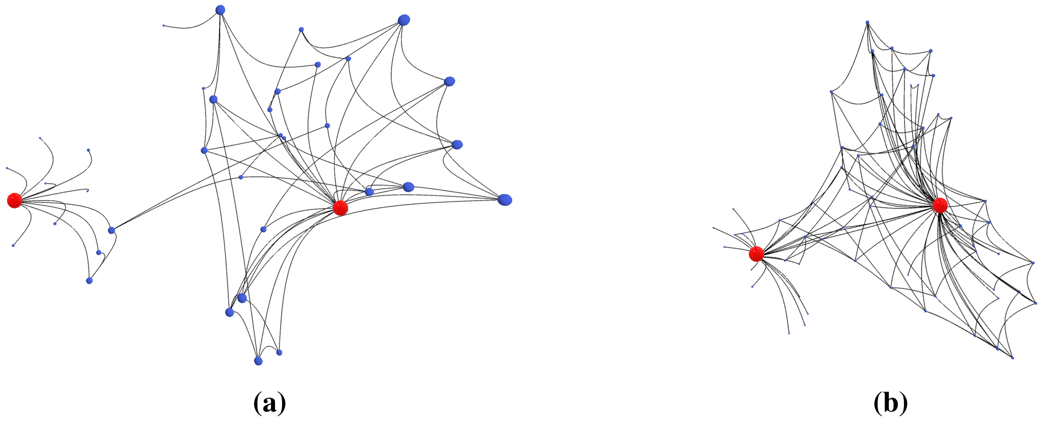


Figure 3.16: (a) snapshot for an aggregation window of 5 (10 minutes) and (b) 10 (20 minutes). Red nodes are representative of the two ground points.

Observation of Fig. 3.16 reveals that gathering multiple time steps into a single snapshot generates a "polarized" graph, with the two ground nodes displaying a high degree (that is, the number of edges incident to a vertex), while the nodes in between have typically a lower degree. We believe this apparent characteristic to be a byproduct of the combination of the constellation dynamics and the data reduction process: while intermediate links may persist across multiple time steps (which will result in keeping solely one of the repeated links according to the data reduction procedure), connections to the ground nodes are highly dynamic. As such, when aggregating multiple time steps, the two ground nodes will show a higher degree due to the numerous unique links. This strongly affects the values in the diffusion matrix [98] and, consequently, the set of nodes entering the sub-graph utilized to extract local graph spatial information, which oftentimes includes one of the target nodes. In particular, this influences the distance (spatial) information that is encoded by the network, thus having a direct impact on the algorithm's performance.

3.6.2 New Data Processing

As highlighted in the previous section, the direct elimination of repeated links *before* applying the aggregation window may negatively affect the structure of the processed graphs, which may result in being disconnected. However, as this contradicts the nature of the system, we believe the performance of the algorithm to be significantly affected. For this reason, to maintain the connected characteristic of the system, we modify the data processing procedure by switching the operations: at first, we separate edges according to their unique time step; next, for each time step, we remove the repeated edges; finally, we apply the aggregation window.

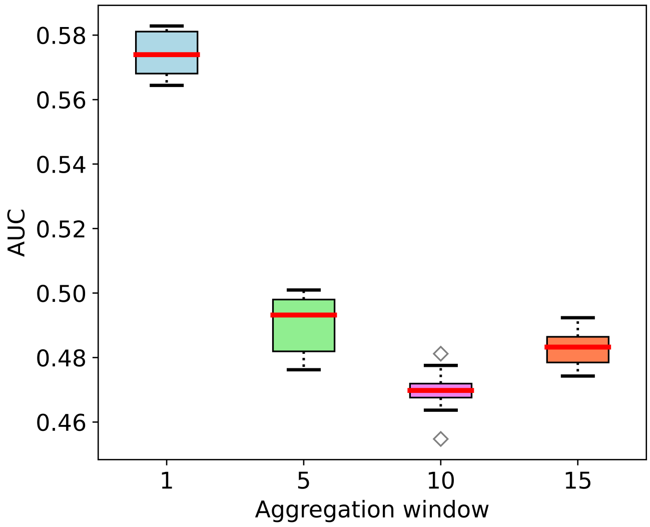


Figure 3.17: Aggregation window sensitivity: alternative data processing. Look-back horizon fixed to $\tau = 2$.

Fig. 3.17 displays the results for the experiments conducted through the alternative data processing strategy. One can observe how performance is significantly degraded with respect to those reported in Fig. 3.14, with the network performing on par with a random classifier for three out of four cases. Additionally, there appears to be an inverted trend, with better performance obtained using a shorter time horizon. Inspection of the results reveals how this is a consequence of the nature of the graph itself and the anomaly injection technique. In fact, the alternative processing technique strengthens the connectivity of the graph. This increased connectivity negatively couples with higher values of aggregation window, where the possibility of connections among nodes belonging to originally different time steps increases. As a consequence, the original constellation-graph physical nature is lost. Furthermore, with a more densely connected graph, anomalies are blended with regular edges with a higher degree, thus making the discrimination between regular and irregular edges even more challenging. Conversely, for AGW = 1, both the original temporal signal and the structure of the system are preserved, facilitating the detection task. Nonetheless, the coupling between injected anomalies and graph connectivity persists, thus leading to overall lower performance than those presented in the previous section.

3.6.3 Edge Frequency Introduction

The two previous sections demonstrate that purely structural (intended as graph topology) and temporal (here referred to the actual time step to which edges belong) information are insufficient to conduct the anomaly detection task on the constellation problem successfully. Consequently, additional information must be injected to enrich the encoded data. For example, we generated graph-constellation snapshots by computing k-shortest paths, which are not necessarily disjointed, meaning that the same link between two satellites can appear multiple times. Therefore, we re-introduce this information by using the frequency with which an edge appears as an additional feature to be encoded and processed by the network. While this can be directly done for regular edges, a “pseudo-frequency” is introduced for the anomalous ones, with the encoded value for anomalies randomly sampled between 1 and the maximum frequency in the same snapshot. This is applied both for the negative sampling, where the frequency is sampled in the range derived from the positive snapshot, and the anomaly injection, with the sampled value belonging to the range of frequency in the snapshot where the anomalies are injected. We tested the effect of edge frequency information injection both on the dataset processed following the procedure described in [3] and the modified procedure discussed in Sec. 3.6.2 (further insights regarding the effect of the frequency introduction can be found in Appendix B.1).

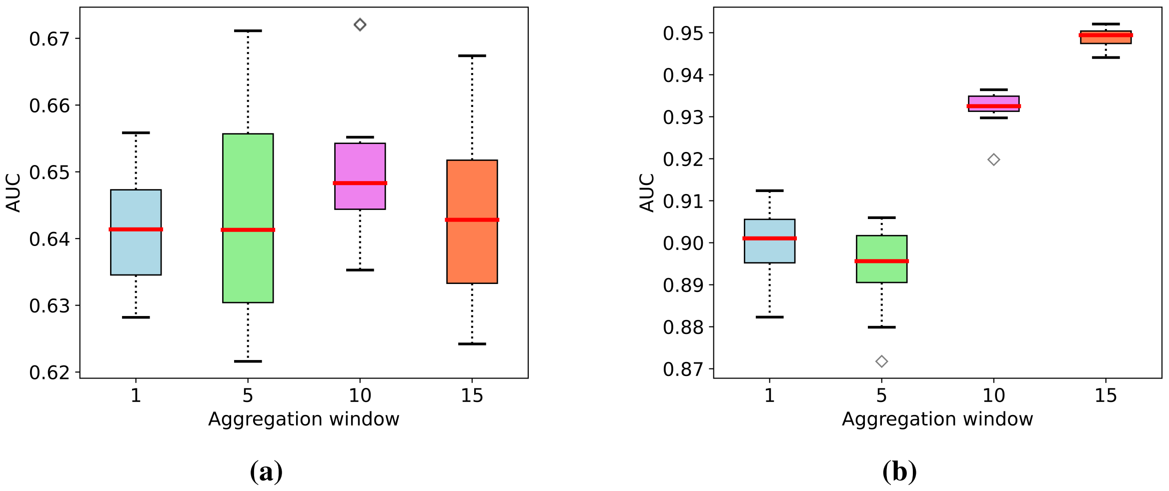
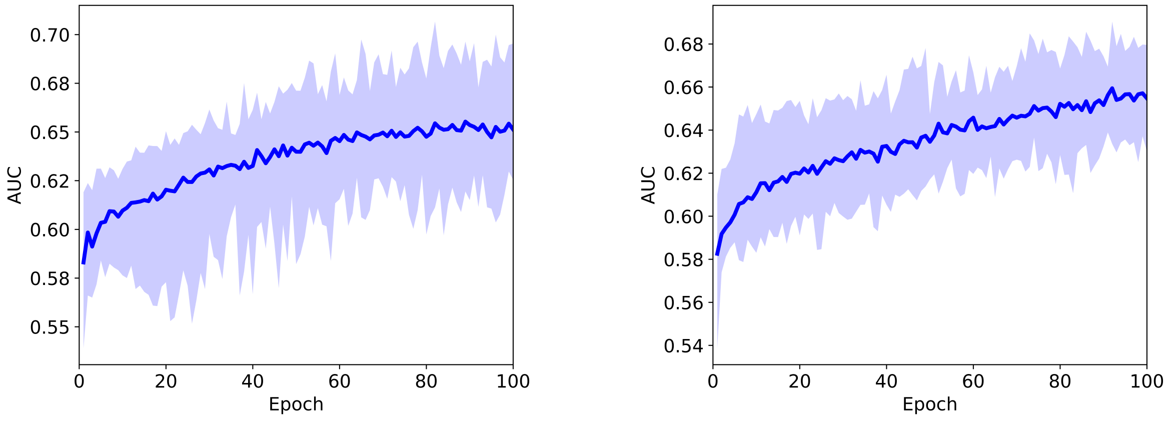


Figure 3.18: Frequency information injected: data processed as described (a) in the TADDY methodology [3] and (b) in Sec. 3.6.2.

Fig. 3.18 shows the result associated with the conducted analysis. To allow for a direct comparison with the results discussed in the previous sections, the look-back horizon τ is maintained to two while the same ranges of AGWs are investigated. Comparison of Fig. 3.18(a) with Fig. 3.14 reveals how there is no significant general improvement, with the AUC score being similar on average for all AGW values. However, when monitoring the AUC score improvement over training, we observed how typically the trend corresponding to the results reported in Fig. 3.14 displayed a flattening toward the end of the training; conversely, for the case with frequency injected, the AUC score displayed a positive trend, potentially indicating the possibility to achieve higher performances for longer training (see Fig. 3.19). A significant improvement with respect to both the results reported in Fig. 3.14, Fig. 3.17 can instead be observed in Fig. 3.18(b), with AUC score reaching up to $\sim$0.95 for AGW = 15.

3.6.4 Constellation Configuration Investigation

Having proven the effectiveness of the algorithm, we want to investigate how the constellation configuration influences the algorithm's performance. In particular, we are interested in observing the effect of the constellation size and of the variation of the ground nodes, as they both impact the encoded information. For example, the number of satellites influences the number of times each satellite appears in multiple time steps, thus affecting the temporal information;



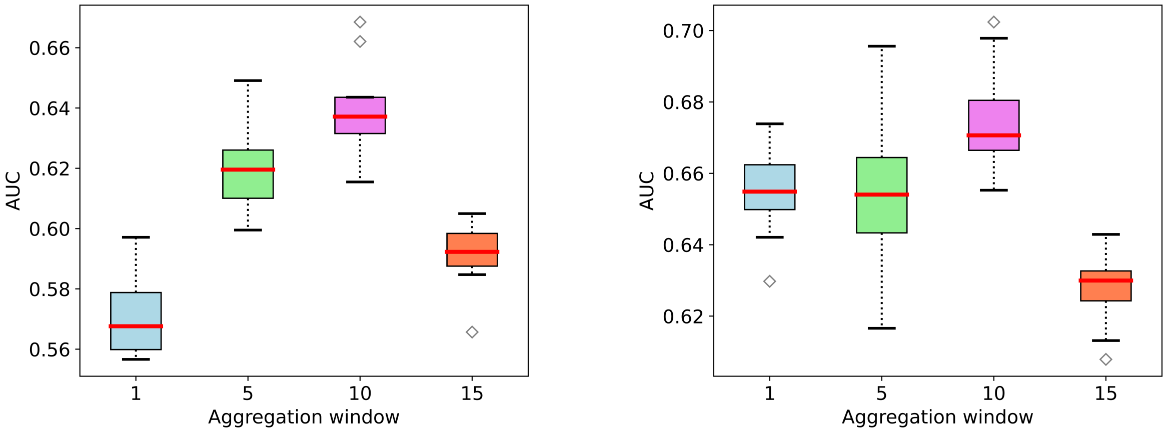
(a) **(b)**

Figure 3.19: AUC score improvement along training: (a) without and (b) with frequency injection. Solid blue line representing the average curve across 10 networks training.

similarly, a variation of source and target nodes modifies the graph structure, hence affecting the spatial information.

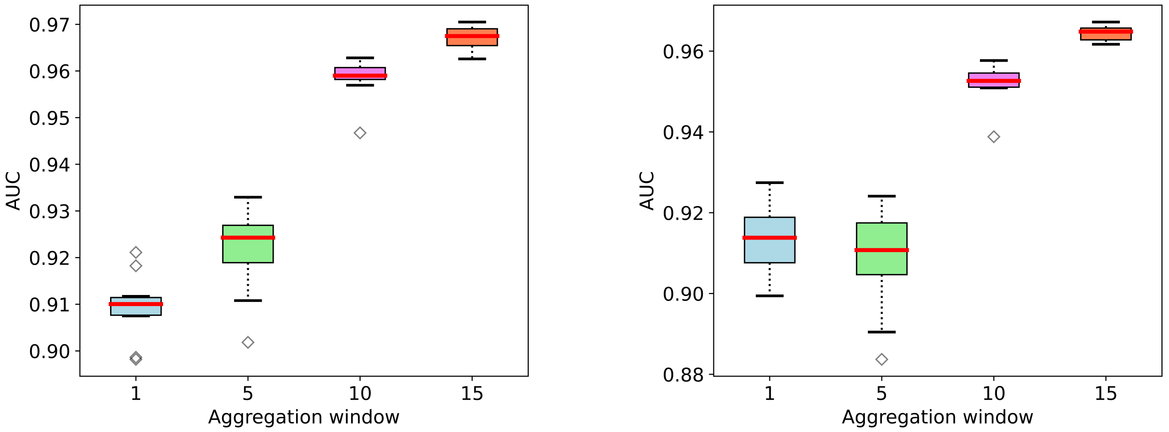
Constellation Size Variation. Two additional constellation sizes are used in this study: 900, with 30 orbital planes and 30 satellites per plane, and 1200, with 30 planes and 40 satellites per plane. In order to allow a more direct comparison with the network performance over the original scenarios, the target nodes are here selected to be Auburn and Anchorage; additionally, orbit altitude and inclination are left unchanged. Similar to the previous section, we test the pre-trained network over these two new graphs to understand whether the networks have specialized over the constellation configuration or have learned to recognize some underlying graph features which are not specialized on a specific graph.

Figs. 3.20 and 3.21 display the results for the new constellation sizes for both data processing approaches. Direct comparison with Fig. 3.18 shows that, while the single distributions themselves are different (as expected), the AUC scores reached are somewhat aligned. In particular, Fig. 3.20(a) denotes worse performance than those in Fig. 3.20(b). We believe this to be given by a higher difference of temporal signal derived from the satellite dynamics with respect to the nominal case (1600 satellites). In fact, the different satellite distributions influence the time horizon over which a given satellite is 1) in sight of one of the two ground points and 2)



(a) **(b)**

Figure 3.20: Frequency information injected: (a) 900 satellites and (b) 1200 satellites comparison. Data pre-processed as described in the TADDY methodology [3].



(a) **(b)**

Figure 3.21: Frequency information injected: (a) 900 satellites and (b) 1200 satellites comparison. Data processed as described in Sec. 3.6.2.

belonging to paths across multiple time steps. Consequently, for the case with 900 satellites, the data processing corrupts the spatial information, while the temporal information is significantly different from the nominal case. It follows that one can expect performance to be degraded to some extent. In opposition to this, the case with 1200 satellites appears, on average, to be more aligned with the original performance and more consistent among the different aggregation windows. Conversely, the AUC scores reached are qualitatively unaltered for both cases in Fig. 3.21.

Target Nodes Variation. The analyses presented in the previous sections were all conducted considering the same pair of source and target nodes. Consequently, as an analysis confined to the same conditions may result myopic and unable to generalize to unseen scenarios, we here investigate the performance of the network when the two target nodes change. Specifically, we consider a case where one of the two is changed and where both are changed. For this verification, no new training is conducted; that is, the same networks used for the previous cases are tested on newly generated datasets. Hence, our goal is to verify whether a model trained over specific conditions is merely specializing on them, without potentially being transferable to a different scenario. As previously mentioned, in the discussed scenarios source and target nodes were arbitrarily chosen to correspond to the geographic location of the cities of Auburn (AL, USA) and Anchorage (AK, USA). For the case involving the change of a single node, the city of Auburn has been replaced with Milan (Italy), while for the case of both nodes changed, the source and target nodes have been assumed to correspond to Tokyo (Japan) and New Delhi (India) respectively. While the locations themselves have been $\sim$ randomly chosen, we selected them with the intent to obtain a different graph topology.

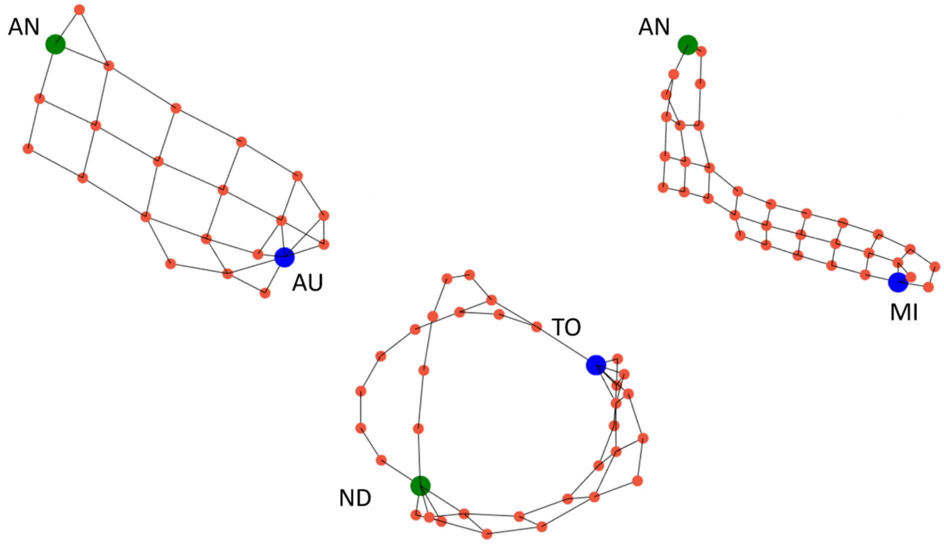
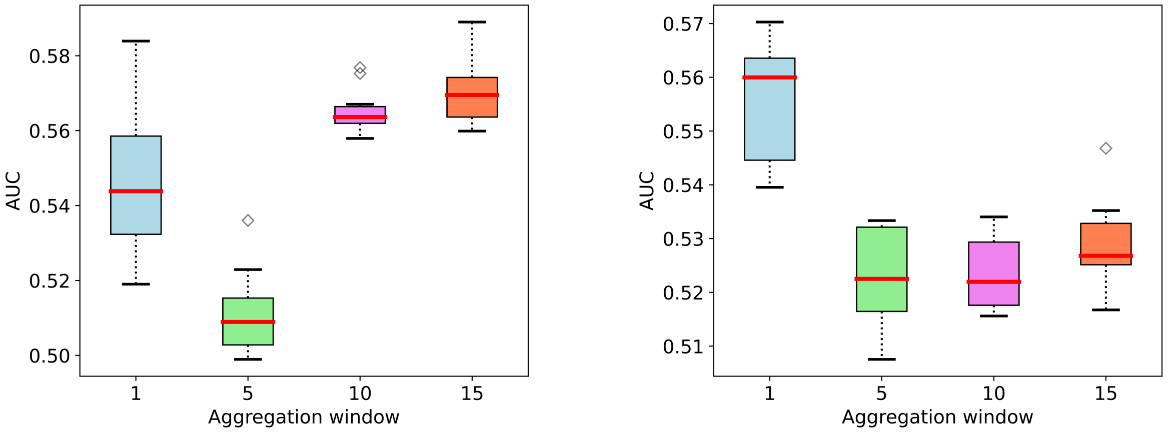


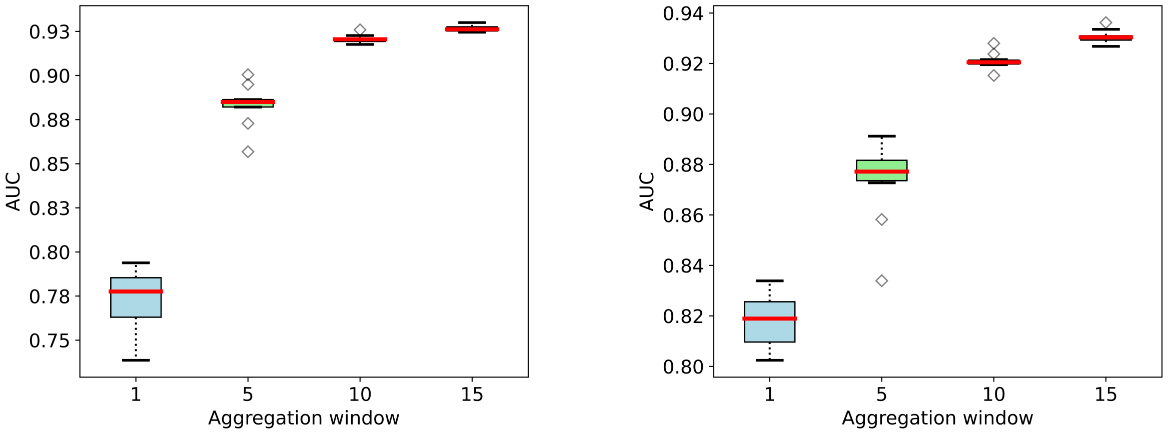
Figure 3.22: Graph representation for the three cases; randomly selected snapshot using AGW = 1. Data processing from Sec. 3.6.2. (AN:Anchorage, AU: Auburn, MI: Milan, TO: Tokyo, ND: New Delhi).

Fig. 3.22 displays a snapshot of a graph for each of the three tested cases, obtained using AGW = 1 applied to the dataset processed as discussed in Sec. 3.6.2. While the shape itself of the graph does not mirror the node distribution in the actual constellation (NetworkX library plot tool adopting "spring layout" with automatic settings used), one can observe how structural differences exist for the three cases. In particular, for the Tokyo to New Delhi connection, paths appear to be more disjointed than in the other two cases. We believe that this difference, in addition to the fact that data undergoes some processing, is a consequence of a coupling between the constellation configuration and the geographic location of the two target nodes.



(a) **(b)**

Figure 3.23: Frequency information injected: (a) Milan to Anchorage (b) Tokyo to New Delhi. Dataset pre-processed as described in the TADDY methodology [3].



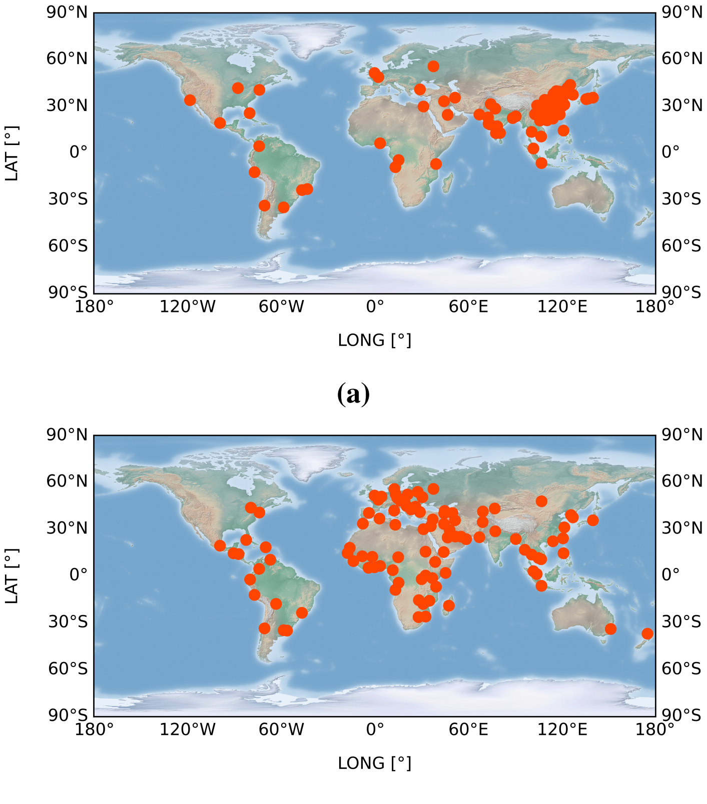
(a) **(b)**

Figure 3.24: Frequency information injected: (a) Milan to Anchorage and (b) Tokyo to New Delhi. Data processed as described in Sec. 3.6.2.

Figs. 3.23 and 3.24 show our test results for the Milan to Anchorage and Tokyo to New Delhi scenarios. Please note that solely the networks trained with the frequency information injected have been used. Overall, one can observe how performance has decreased in both cases, with a stronger degradation in Fig. 3.23. Upon inspection of the results, different motivations can be identified. To begin with, the data processing has a strong impact on the structure of the graphs and may lead to a loss of original graph properties (such as connectivity). It follows that, for cases where the already modified graph over which the network was trained is further changed, the network loses its ability to discern anomalies, behaving almost like a random classifier. Conversely, for the other case, the original characteristics of the corresponding graphs are unaltered despite the graph structure being somewhat different (Fig. 3.22). As such, the network is still effective in the anomaly identification task. On top of this, one can notice how the performance in Fig. 3.24(b) is better than those in Fig. 3.24(a). We argue that the reason is to be found in the typical structure of the graph for this specific case. As previously highlighted, the graphs for the Tokyo to New Delhi scenario frequently display disjoint paths. This aspect, combined with the anomaly injection procedure, makes anomaly identification easier. In fact, as we discussed in Sec. 3.5.4, the anomaly injection is performed by randomly sampling connections in the node set of the associated snapshot. Therefore, having disjointed paths, there exists a chance that connections are created between nodes belonging to such independent paths. This affects the structural encoded information, resulting into an ease of the detection task. Nonetheless, the performance still remains worse than those obtained on the nominal case, which is expected considering that these new graphs differ from those over which the networks have been trained.

3.6.5 Monte Carlo Validation

To further corroborate the validity of the method and its capability to generalize on unseen scenarios, we conducted an extensive analysis by testing the networks over a wide set of node pairs. To sample such pairs, we considered the most populated cities in the world according to the "World Cities Database" (`https://simplemaps.com/data/world-cities`). As such a database includes about 41000 cities, we first reduced the data by sampling the top 100 cities; however, this operation resulted in unbalanced distribution and variety of connections, as most of such cities are cluttered in the southern-east Asia region. For this reason, instead of directly sampling the top 100 cities in the database, we considered the top 100 with the constraint of having a single city per country (Fig. 3.25).



(b)

Figure 3.25: Top 100 most populated cities sampling: (a) original data and (b) country uniqueness constraint applied.

Next, to better categorize the connections, we divided the world map into four asymmetric quadrants, each containing 25 cities. Finally, for each quadrant, we randomly sampled a subset of 10 cities, which are ultimately used to create the connections (Fig. 3.26).

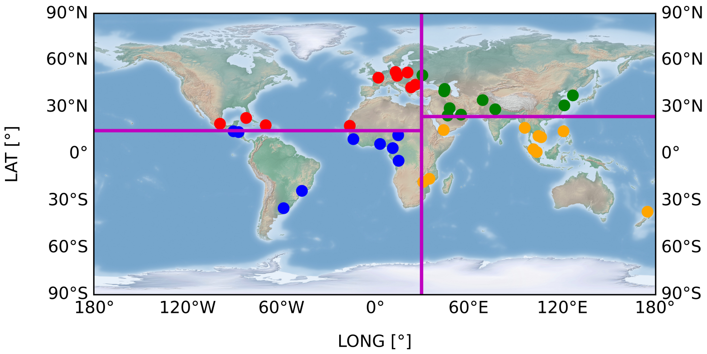


Figure 3.26: Final set of cities utilized to generate the analysis dataset. Different colors identify nodes belonging to different quadrants.

To introduce variety and classify the connections, we considered multiple possibilities, including links between node pairs within the same quadrant or with adjacent ones. Additionally, in order to assess whether the direction of the connections influences the performance, we considered connections in opposite directions (e.g., west-to-east and vice versa, or north-to-south and vice versa). This procedure identifies ten different classes for a total of 960 node pairs. A schematic representation of the directions for creating the node pairs dataset is depicted in Fig. 3.27.

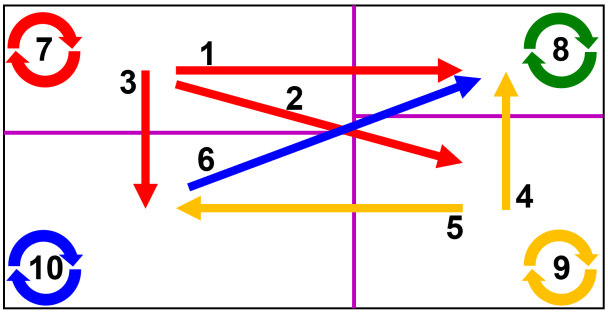


Figure 3.27: Schematic representation of the connection groups. Ten separate classes were considered (looping arrows representing connections within the same quadrant).

Each node pair is then used to generate the dataset to be analyzed by the neural network. We maintain the simulation settings analogous to those described at the beginning of the section, fixing the constellation size to 1600 satellites. After a dataset is generated, we process the edges following the procedure described in Sec. 3.6.2. Finally, we conduct again an analysis on the aggregation window, testing for each case the ten best network models used in the previous sections.

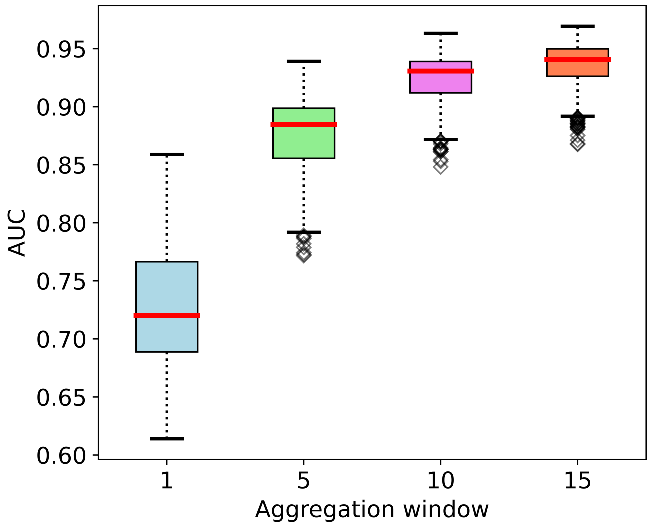


Figure 3.28: Monte Carlo analysis results for the selected 960 ground node pairs.

Fig. 3.28 displays the results of our analysis. The average AUC score obtained by the ten networks on each aggregation window is shown for each dataset. Differently from the previous cases, where the distribution over different trained models was displayed, the image represents the distribution of the average performance over all the considered node pairs. One can observe a similar trend to the one in Fig. 3.24, though the distribution shows a much higher dispersion. To assess what factors are the most impactful on performance, for each aggregation window, we isolated the node pairs associated with the top whisker for the highest performing cases and with the bottom outliers for the worse performance (in the absence of outliers, for AGW = 1 the lower and upper whiskers are considered).

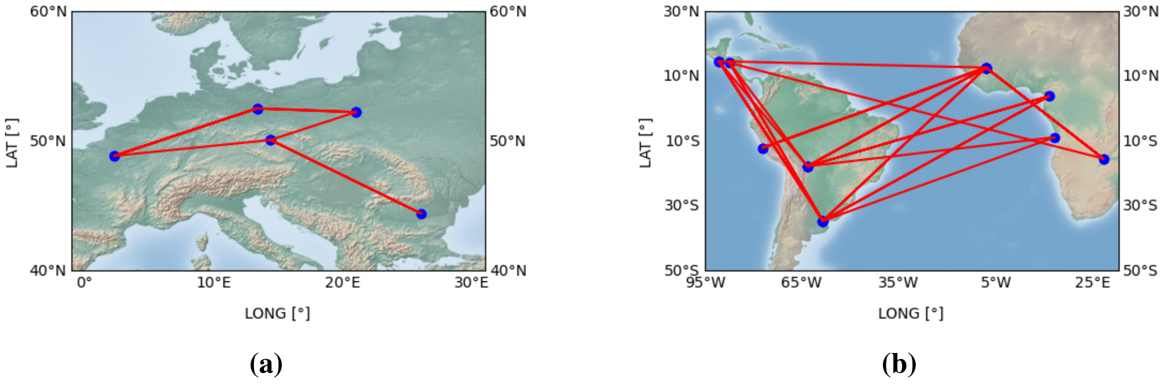


Figure 3.29: Examples of ground nodes pairs connection: (a) AGW = 5, group 7, and (b) AGW = 5, group 10. Best performance typically feature nodes pairs with larger variability in latitude and longitude.

Visual inspection of the node pairs locations (Fig. 3.29) associated with whiskers/outliers reveals how the geographic position of the target nodes in each pair plays a crucial role. Specifically, for each node pair, we consider the difference between the latitudes and longitudes of the nodes, as well as the slope of a virtual line connecting each node (assuming a 2D view of the world map). For each of these quantities, Tabs. 3.1 to 3.3 report the average values for worst and best scenarios for all the aggregation windows values. Overall, one can observe a common trend, with best performance associated with higher values and worst performance associated with lower ones, with an exception constituted solely by the case AGW = 1 in Tab. 3.2.

Table 3.1: Average latitude difference for worst and best performing cases for each AGW.

AGW	1	5	10	15
Worst performance	14.7435°	2.1421°	3.8790°	2.6925°
Best performance	36.7379°	38.9620°	34.2670°	32.8848 °

Table 3.2: Average longitude difference for worst- and best-performing cases for each AGW.

AGW	1	5	10	15
Worst performance	82.0297°	10.4081°	12.0533°	36.1152°
Best performance	60.4272°	97.5383°	95.5841°	91.5470 °

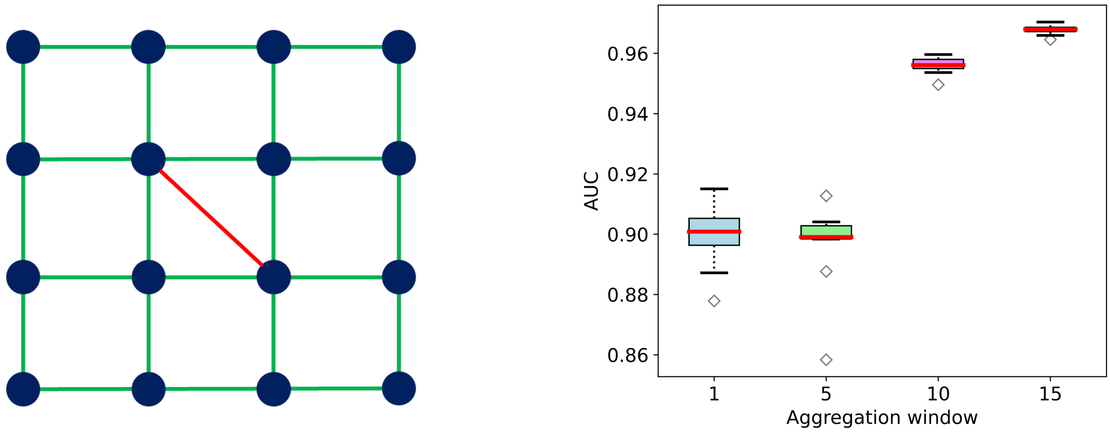
Table 3.3: Average slope of a virtual line on a 2D map connecting node pairs for worst- and best-performing cases for each AGW.

AGW	1	5	10	15
Worst performance	11.6302°	11.2431°	22.5622°	13.3775°
Best performance	46.4224°	30.6881°	27.3098°	27.9639 °

Based on the observations made in the previous sections and the results from this analysis, we believe the cause of the behavior reported in the tables is a combination of satellite dynamics, temporal resolution of the snapshots, and anomaly injection technique. As mentioned in the description of the simulation setup, a resolution of two minutes is used to capture constellation graphs. However, considering the inclination of the orbits, such a frequency may be too low for cases where the difference between latitudes or longitudes between connected pairs is lower, as subsequent snapshots would display higher satellite-nodes variability. Consequently, similarly to injected anomalous edges, nominal edges will have a lower temporal footprint, thus hindering the detection capability of the neural network. Conversely, for cases where the difference in latitude or longitude is higher, it is more likely that the same edge will appear in multiple subsequent snapshots, thus easing the discrimination between it and spurious anomalous connections. Similar considerations apply when considering the slope of a line connecting the nodes in each pair; qualitatively, as the slope of the virtual line connecting any ground nodes pair gets closer to the orbit inclination value (65° set for all simulations), the performance is expected, on average, to improve. This consideration derives from the fact that a slope closer to the orbit inclination will more likely ensure a stronger temporal footprint of regular edges, thus again increasing the network discriminative capability.

3.6.6 Toward Higher Fidelity: +grid Interlink Model

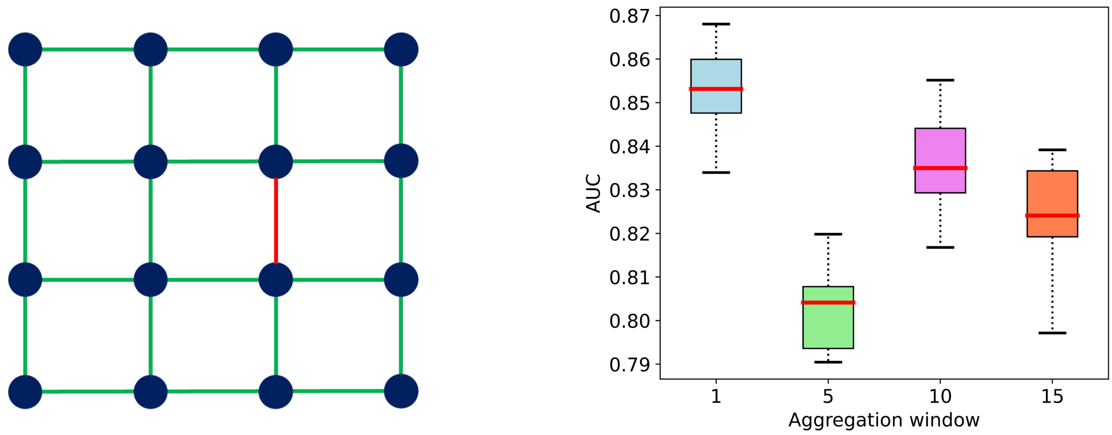
As a final analysis for this work, we investigate the architecture behavior when we increase the similarity between regular and anomalous links. Specifically, in the previous analysis, irregular links were created between a node and any other node in the associated snapshot node set; this is closer to the original formulation of the TADDY framework, but it does introduce a deviation from the +grid interlink model. Hence, we here explore cases where the second node of a new connection is sampled within the surrounding substructure of the origin node. First, we consider diagonal connections, then nominal +grid connections.



(a)

(b)

Figure 3.30: (a) Graphical representation of possible negative connections and (b) modified framework performance.



(a) (b)

Figure 3.31: (a) Graphical representation of possible negative connections and (b) modified framework performance.

Fig. 3.30, 3.31 display the results of this analysis. In both cases, we can observe how pre-trained models display high performance even under unseen conditions. Note that the results reported in Fig. 3.31 are generated by modifying the data processing. Throughout the previous sections, when merging multiple graphs based on a given aggregation window, each snapshot was generated by shifting the node set by a number of time steps equivalent to the aggregation window itself. Conversely, merged graphs are generated in this case by shifting the node set of a single time step. In such a way we preserve the full nature of the original dataset while completely respecting the problem dynamics. A visual representation of such a difference is reported in Fig. 3.32.

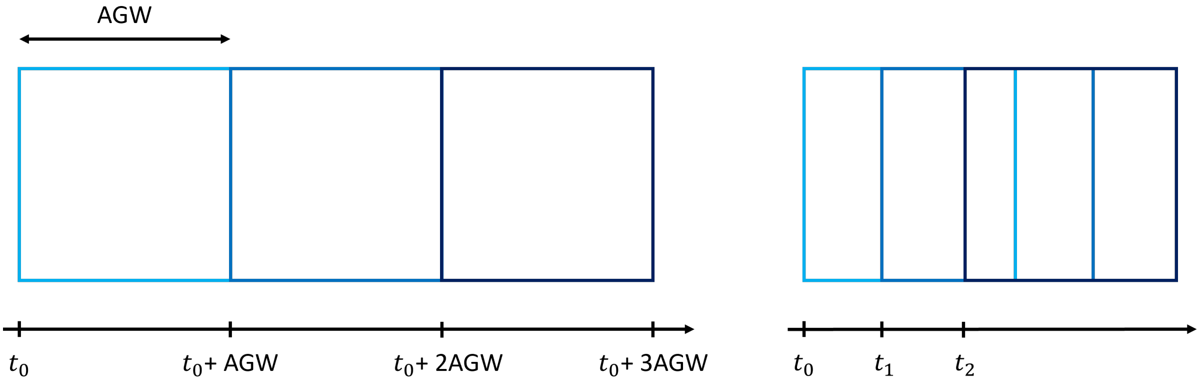


Figure 3.32: On the left, consecutive merged graphs are generated by shifting the node set by a number of time steps equivalent to AGW. On the right, the merged graphs are generated by shifting each node set by a single time step.

Chapter 4

Evolving Strategies for Reaction to Adversarial Attacks Against P-LEO Constellations

4.1 Introduction

Throughout the last decade, the increasing demand for high-quality ubiquitous services has driven the development of Proliferated Low Earth Orbit constellations. One of the leaders in the sector, Starlink, is currently operating more than 5,000 satellites, with about 12,000 approved by the Federal Communication Commission (FCC), and the ambitious goal of an extension up to 42,000. However, while modern society benefits from the countless services enabled by these large space systems – which find application in fields such as public safety, civil and military communications, weather forecasting and Earth observation to name a few – challenges arise in the management of such a wide infrastructure. In particular, the increased satellite network capacity and the necessity of monitoring the status of many space assets demand an efficient link to the ground. To this end, an optimal ground segment distribution, which enables frequent contact with the satellites, thus allowing on-board data offloading and constant monitoring while minimizing costly redundancy, becomes crucial. In this context, a fundamental aspect often neglected in the open literature is the security of such a large space infrastructure. In fact, historically satellite developers and operators have relied on the principle of *security by obscurity* [103], assuming the whole constellation infrastructure inaccessible to external unauthorized entities. However, thanks to technological advancements and the possibility of flaws in satellite constellation chain management, capable malicious actors may now be able to target all the levels of the constellation infrastructure with a variety of attacks, including command intrusion, payload control, malware, denial of service, and hacking [40] (Fig. 4.1).

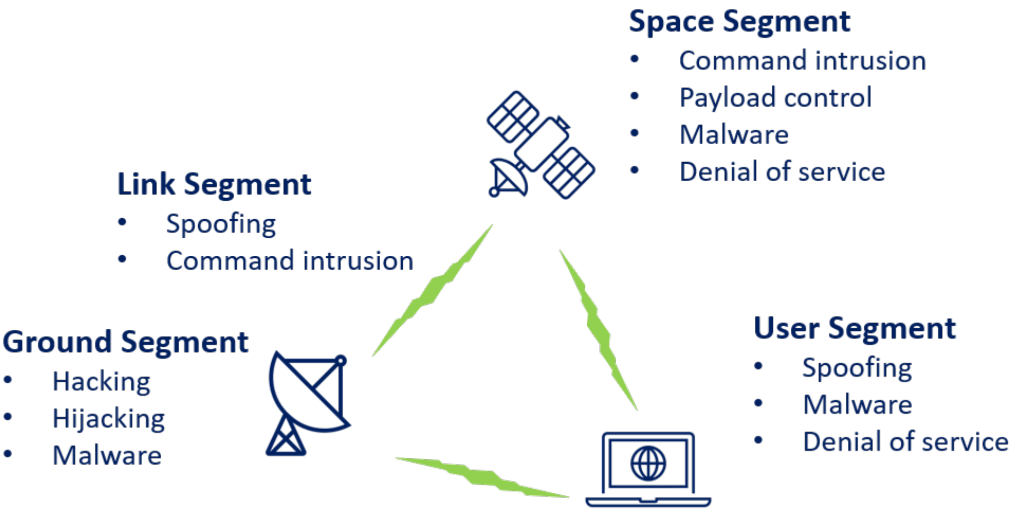


Figure 4.1: General overview of a satellite constellation infrastructure (ground, space, user and link segments) in association with potential threats.

As an example, in February 2022 a cyber-attack was launched against Viasat's KA-SAT network [41], causing partial interruption of satellite broadband service in Ukraine, with spillover effects registered in other European regions. In cases such as these, the attack signal was disseminated by the space segment, though the point of access was the ground segment. The concern about cyber-attacks against satellite systems gains particular importance when considering P-LEO constellations, as they constitute an enlarged surface of attack potentially susceptible to actions of malevolent actors, such as criminal organizations, terrorist organizations, and adversarial nation-states (threat tier III-VII, as identified in the Space Attack Research and Tactic Analysis (SPARTA)[1] threat model by the Aerospace Corporation [104]). Following the Viasat event, in March 2022 Russia attempted again to disrupt Ukrainian communications through jamming SpaceX's Starlink terminals [42]; while the attack was unsuccessful, such events demonstrate how the increasingly important P-LEO constellations may become an ideal target during critical situations, emphasizing the necessity to introduce effective countermeasures. Within this framework, strategic placement of terrestrial stations and contact times become of paramount relevance for both satellite operators and potential malicious actors. Considering the possibility of undesired actions on their assets, satellite operators may distribute their ground segment not only to conduct routine operations, but also to provide timely counter for adversarial

[1] https://sparta.aerospace.org/

attacks. Similarly, a malevolent actor may look at strategic locations to threaten the satellite constellation, while reducing its detectability.

The problem of optimal ground stations distribution is an active field of research and can be tackled from different angles. One possibility consists of directly optimizing the ground stations' locations through the definition of objective functions dependent on the specific case study. For example, Non-dominated Sorting Genetic Algorithm-II (NSGA-II) is used by del Portillo et al. [105] for finding the trade-off surface for minimizing the number of ground stations while maximizing the performance in terms of data rate and coverage. Christos and Panagopoulos [106] introduce site diversity as a technique to improve satellite communication availability, handling in two steps the optimization problem aimed at minimizing the number of required ground stations while subject to the given constraints. First, they formulate the problem as a binary-integer-linear-programming (BILP) problem, which is proven to be NP-hard. Then, they propose a solution via the design of a branch-and-bound algorithm, which achieves global optimality, based on linear-programming (LP) relaxation and on the development of a greedy method. Alternatively, ground station placement can be approached as a scheduling problem [107], where the goal is typically to maximize the total amount of downloaded data, while dealing with constraints on energy, data requirements, and line of sight availability. Traditional solution methods for this problem involve the definition of a single weighted objective function or the formulation of a multi-objective optimization problem, minimized through evolutionary algorithms (such as genetic algorithms, GAs) or other optimization techniques [108]. For example, Petelin et al. [107] discuss and test multiple methods to solve a multi-objective ground station scheduling problem for optimizing contacts with satellites, including algorithms such as NSGA-II, NSGA-III, the Strength Pareto Evolutionary Algorithm (SPEA2) and the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D). Their work demonstrates that multi-objective optimization typically performs as well as, if not better than, single objective problems solved via GA, with MOEA/D outperforming all the other methods in most of the tested problem settings. However, all these methods assume nominal operations, without considering possible adversarial actions capable of compromising ground nodes (or satellites).

To include the possibility of hostile actions against a satellite constellation within the context of optimal ground station distribution, we investigate the problem of strategic node placement for an attacker-defender transit time game [109]. Inspired by the game theoretic model called FlipIt [110] where two adversaries iteratively take control of a resource, balancing certainty of control and cost of taking control, we assume a simple threat model which considers two opponents competing for disrupting/maintaining a satellite constellation. The two opponents are deemed to be capable of directly influencing the operational status of orbiting satellites. If this hypothesis is reasonable for the defender, we acknowledge it constitutes a strong assumption for the attacker. However, in reality there exist multiple ways through which a malicious actor may accomplish such goal, including kinetic-physical attacks, non-kinetic physical attacks, electronic attacks and cyber-attacks. Depending on the type of attack, the consequences may be either temporary or permanent. In this work, we *assume* the attacker to possess all the capabilities to deliver a non-kinetic attack that induces satellites into entering safe mode, which is a state recoverable by the defender upon contact with the impaired satellites. To trigger satellites into entering safe mode, an attacker may use a combination of actions, such as GPS spoofing and communication interference, or it may exploit vulnerabilities in the satellite's onboard software or command processing system. Given this assumption, our focus is shifted from the problem of preventing an attack, to the problem of reacting to one, should a malevolent actor be successful in its intents. Unlike the simpler FlipIt framework, we elevate the realism of our model by introducing satellite constellation dynamics and real-world geography. As these elements are introduced, the complexity of the problem increases, and leads to a higher dimensionality of the solution space. The two opponents have to act with only partial information regarding each other, while having to deal with the dynamics introduced by the satellite motion, which demands precise action timing. Because of this, while the consequences of even a single action become hard to predict, the agents must also plan how to deal with a variety of opponent strategies. As a result, the problem becomes significantly more challenging to solve.

In this work, we examine the use of competitive coevolutionary algorithms [111] with genetic programming (GP) [112] as a generative hyper-heuristic [113] for developing reactive ground station placement strategies for the adversarial ground station transit time game. Unlike similar work in ground station placement problems, rather than directly searching for a solution of ground station placements, our search method finds a *policy* for placing ground stations based on the observed situation, able to respond to a variety of attacks. Once evolved, these reactive strategies can immediately provide solutions to unknown threats without running a new search, an important component in the future development of complex, real-time agents. In addition, our method of operating on geospatial observation data with GP serves as a useful framework more broadly applicable to problems on satellite networks. We prove our solution to be: (1) nontrivial, beating both random behavior and a minimal, deterministic reactive strategy, here used as baselines; (2) versatile, adapting to different conditions and outperforming static solutions found with previous evolutionary methods; (3) robust, maintaining good performance over a wide pool of scenarios with varying level of complexity; and (4) better than heuristic human-expert solutions.

In contrast to previous work [109], the solution to the problem involves a reactive defender, that is, a defender aware of the effects of the adversary actions on the satellite constellation performance. We explore reactive agents with both partial and complete knowledge of the attacker's effects. Additionally, we expand the orbital model to allow simulations of inclined orbit, and we introduce geographic constraints to prevent unrealistic solutions. Finally, we examine a human hand-crafted strategy, compare it with the solution found via competitive coevolution, and discuss its limitations for increasing problem complexity.

This chapter is organized as follows. Sec. 4.2 provides information regarding the orbital mechanics behind our orbital model and the theoretical background concerning competitive coevolution and genetic programming. Sec. 4.3 presents the description of our ground station transit time game. Sec. 4.4 describes the methodology utilized to compute satellite performance and provides details on the specific implementation of the algorithm employed to solve the transit time game. Sec. 4.5 presents adversarial scenarios with increasing level of complexity and analyzes the results. Finally, Sec. 4.6 discusses observed algorithm performance.

4.2 Preliminaries

In this section, the theoretical background for the main topics and methodologies discussed in this chapter are presented. At first, a brief description for the orbital mechanical model is presented. Next, the two key tools utilized to identify a solution to the explored problem - competitive coevolution and genetic programming - are introduced, focusing on the main concepts relevant for the comprehension of the work.

4.2.1 Elements of Constellation Design

Orbital Model. For the sake of simplicity, this work considers satellites revolving on circular orbits, following an ideal (i.e., non perturbed) dynamical model. In such a way, each satellite orbit can be uniquely characterized through a set of parameters, commonly referred to as classical (or Keplerian) orbital elements. There are six parameters, each describing a particular feature of the orbit: a, e, i, Ω, ω, θ. In the following, an explanation for each of them is provided.

- *Semi-major axis (a).* This first element offers information about the size of an orbit, and it is mathematically defined as half the sum of the periapsis and apoapsis radii. Considering a generic elliptical orbit, the semi-major axis represents the line that goes from the center of the ellipse to one of its extreme points along the major dimension.
- *Eccentricity (e).* This parameter defines the shape of an orbit, quantifying how much such orbit is elongated with respect to a circular one. For a closed orbit, $0 \leq e < 1$ (while it assumes values equal or higher than 1 respectively for parabolas and hyperbolas). It can be seen as the norm of the eccentricity vector, which is a vector pointing toward the periapsis, defined as:

$$\mathbf{e} = \frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{|\mathbf{r}|} \tag{4.1}$$

where $\mathbf{h}$ is the angular momentum vector given by $\mathbf{h} = \mathbf{r} \times \mathbf{v}$, μ is the primary body gravitational parameter, $\mathbf{r}$ is the position vector, and $\mathbf{v}$ is the velocity vector.

- *Inclination (i).* It determines the tilting of an orbit with respect to a reference plane. It is formally measured between the $\hat{K}$ unit vector direction (normal to the reference plane in an $\left\{\hat{I}, \hat{J}, \hat{K}\right\}$ tern) and the angular momentum $\mathbf{h}$ vector direction, which instead is normal to the orbit plane. It goes from 0° to 180° and can be computed from:

$$\cos i = \frac{\hat{K} \cdot \mathbf{h}}{|\hat{K}||\mathbf{h}|} \tag{4.2}$$

- *Right Ascension of the Ascending Node (RAAN, Ω).* Given a reference plane, this parameter represents the angle spanning from the reference plane vernal axis direction, that is, a reference direction, and the direction pointing toward the ascending node (where a node here is the point where the orbit crosses the reference plane). Following this definition, a generic ellipse features two nodes, with the ascending one corresponding to the point where a satellite crosses the reference plane from south to north. It goes from 0° to 360° and is measured through:

$$\begin{cases} \Omega = \arccos\left(\frac{\hat{I}\cdot\mathbf{n}}{|\hat{I}||\mathbf{n}|}\right) & \\ \Omega = 2\pi - \Omega & if\ \ n_J < 0 \end{cases} \tag{4.3}$$

being $\mathbf{n}$ the normal to the orbital plane.

- *Argument of the periapsis (ω).* Also known as periapsis anomaly, it is the angle, measured on the orbit plane, between the ascending node direction and the periapsis line direction. It goes from 0° to 360° and is measured as:

$$\begin{cases} \omega = \arccos\left(\frac{\mathbf{n}\cdot\mathbf{e}}{|\mathbf{n}||\mathbf{e}|}\right) & \\ \omega = 2\pi - \omega & if\ \ e_K < 0 \end{cases} \tag{4.4}$$

where $\mathbf{e}$ is the eccentricity vector.

- *True anomaly (θ).* This parameter defines the position of a satellite on an orbit and it represents the angle spanning between the position of a satellite and the pericenter line. Also in this case, the range is between 0° and 360° and the computation reads:

$$\begin{cases} \theta = \arccos\left(\frac{\mathbf{e}\cdot\mathbf{r}}{|\mathbf{e}||\mathbf{r}|}\right) \\ \theta = 2\pi - \theta \qquad\qquad if\ \ \mathbf{r}\cdot\mathbf{v} < 0 \end{cases} \tag{4.5}$$

By leveraging these definitions, assuming an ideal spherical model the components of position vector of a satellite at each time can be conveniently expressed as:

$$\begin{cases} x(t) = R_E(\cos\Omega\cos(\theta(t)) - \sin\Omega\sin(\theta(t))\cos i) \\ y(t) = R_E(\sin\Omega\cos(\theta(t)) + \cos\Omega\sin(\theta(t))\cos i) \\ z(t) = R_E\sin(\theta(t))\sin i \end{cases} \cdot \tag{4.6}$$

with R_E being the radius of the Earth. Of note, expressing the position of a satellite in this form becomes extremely convenient at implementation level, as it allows for vectorization of the operations, allowing to compute the position of all the satellites, over all the orbital planes, for all time steps of a given simulation at once. In such a way, large scale simulations can be enabled.

Coverage Analysis Fundamentals. A common problem in space mission geometry is to determine whether and when a satellite will have contact opportunities (either for communication, data downlink, or observation purposes) with a point on the ground. To this aim, one can consider a geometric approach which relies on the relative position of the spacecraft and the point of interest. From the satellite perspective, a vector originating from the satellite which is tangent to the Earth surface defines the true (or geometric) horizon. Given this definition, the surface within the horizon is referred to as access area, that is, the entire area with which a satellite has contact opportunity at any time. Assuming then a spherical Earth model, the angular radius spanned by the small circle defining the geometric horizon is referred to as maximum central angle, typically denoted as λ_{max}, when measured from the center of the planet, while it is referred to as angular radius of the Earth, ρ, when measured from the satellite. For a given target, one can then define two additional angles, namely the nadir angle, η, which is the angle spanning from the subsatellite point and the target location (measured from the satellite perspective), and the elevation angle, ϵ, which is the angle between the spacecraft and the local horizon. A representation for these concepts is depicted in Fig. 4.2.

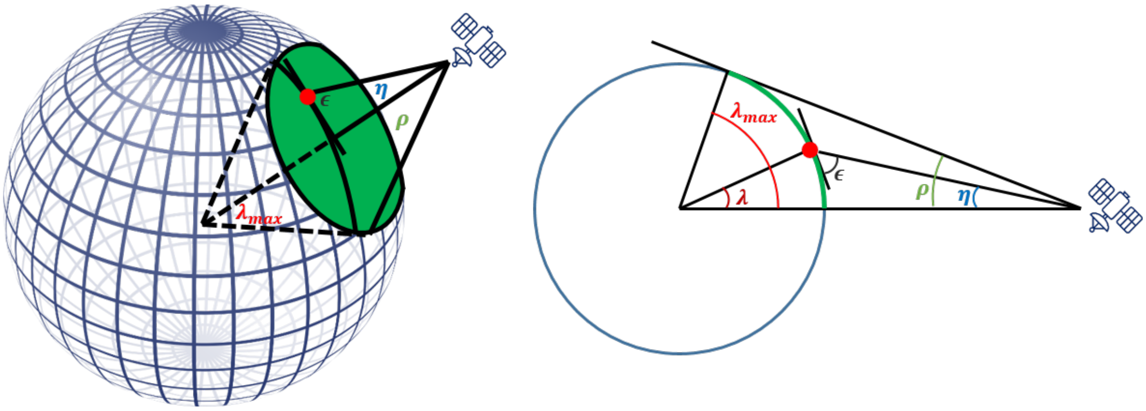


Figure 4.2: Schematic representation of the angular relationships between satellite and target positions.

Constellation Architectures. When it comes to the design of the constellation configuration, two main approaches are commonly utilized.

- *Streets of coverage constellation.* This constellation architecture is based on the concept of “streets of coverage”, which involves n satellites distributed over m ∼polar orbit planes with the goal of providing continuous global coverage. In this type of constellation, the number of satellites in each orbital plane is such that the area covered by to adjacent satellites overlaps. However, as the covered area can be represented as a circle, there exist blind regions, referred to as “dip” areas. Consequently, in order to ensure continuous coverage, satellites in adjacent orbit planes needs to be distributed such that these regions are covered. Introducing λ_{street} as the angle given at the intersection of the covered areas of two consecutive satellites on the same orbit plane, continuous coverage is guaranteed if the following condition is satisfied:

$$(m+1)\lambda_{street} + (m-1)\lambda_{max} \geq 180^\circ \tag{4.7}$$

- *Walker constellation.* This second architecture provides a symmetric constellation configuration. In this architecture, satellites are distributed over multiple orbit planes, all with the same inclination. Commonly, the architecture can be uniquely identified through the notation i:T/P/F, where i is the common inclination, T is the total number of satellites, P is the number of planes and F is a value between 0 and P-1 identifying the phase difference between two satellites in adjacent planes at the crossing of the ascending node. Based on the spacing in RAAN of the planes, there exist two different configurations, namely Walker Delta, which considers planes spacing between 0 and 2π, and Walker Star, which instead considers spacing between 0 and π.

4.2.2 Evolutionary Algorithms

Evolutionary algorithms are a type of search algorithm inspired by natural evolution. In this class of algorithms, a population of candidate solutions to a problem undergoes an iterative stochastic process where the evaluated quality of a solution (its “fitness”) affects the probability that it will be chosen to reproduce (creating new, modified copies of itself in the population) or to survive to the next iteration. This evolutionary pressure, where stronger solutions reproduce and weaker solutions are removed, a phenomenon commonly referred to as “survival of the fittest”, results in a gradual improvement in the population’s solution quality over time.

Fundamentals. Two main concepts drive the evolutionary cycle: 1) variation operators (recombination and mutation), which promote diversity within a population, thus fostering novelty, and 2) selection, which pushes towards the increment of the mean quality of solutions in the population. The combination of these two concepts can then be seen as the evolutionary cycle optimising the fitness function, approaching the optimal value throughout iterations. When dealing with evolutionary algorithms, many aspects must be considered [114]. In the following, details for the key components and concepts is provided.

- *Representation.* This can be intended as the first step to take when solving a problem via an evolutionary algorithm. This operations defines the mapping between the object representing the solution of a problem, called the “phenotype”, and its encoding, that is, its representation within the algorithm, referred to as “genotype”. As an example, suppose the problem to be solved involves the optimization of a constellation configuration in terms of parameters such as number of planes, satellite per planes, orbit inclination and orbit altitude. In this case, the admissible values for these quantities, which for simplicity may all be considered as integers, constitute the phenotype, whereas they may be represented as binaries (the genotype) for their manipulation within the algorithm.

- *Fitness function.* Also referred to as evaluation function, it represents the criterion for selection, defining what the improvement is in the context of a given problem. Technically, this function assigns a quality metric to the genotype; however, typically this function is designed from an inverse representation, with a quality metric evaluated in phenotype space. Continuing the previous example, the goal of the optimization may be to maximize the coverage of the constellation over a certain region of the planet. In such a case, the fitness may coincide with such desired metric, which can be more easily correlated to the original problem parameters (i.e., the phenotype).

- *Population.* It is defined by the representations of all possible solutions, and it can be seen as a multiset of genotypes, that is, a set which also allows multiple copies of the same element. The population is composed of individuals, whose number determines the size of the population. During an evolutionary process, individuals alone are static entities, while the population as a whole is the one that changes and adapts. Typically, the population size is constant throughout evolution, thus promoting competition between individuals. An additional relevant concept related to the population is its diversity, which can be defined as the number of different solutions.

- *Parent selection mechanism.* This operation refers to the selection of individuals to become parents for the next generation. Selection of individuals is typically performed based on their quality, with the goal of improving the solution throughout iterations. Note that this process is probabilistic, with higher performing individuals being selected with a high probability, while a small probability is left to low-quality individuals. This prevents the optimisation to become too greedy and potentially converge to a local optimum.

- *Variation operators.* After parents are selected, variation operators are applied for the generation of new offspring. In general, variation operators are classified based on their arity, that is, the number of inputs provided to the operator. For a single input, that is, arity = 1, the operator is referred to as "mutation". This is a stochastic operator, whose outcome derives from a series of random choices. It applies at genotype level, where to each allele (the building unit of a genotype) is associated a non-zero probability to assume a different value. If the variation operator is binary, that is, arity = 2, it is referred to as "recombination" or "crossover". In this case, genes from two parents are combined to generate an offspring, with the idea that desirable qualities form both parents may enter the genetic pool and passed to the next generations. Similarly to mutation, recombination is a stochastic operator, and the traits from both parents to be combined are randomly selected. A representative example is provided in Fig. 4.3.

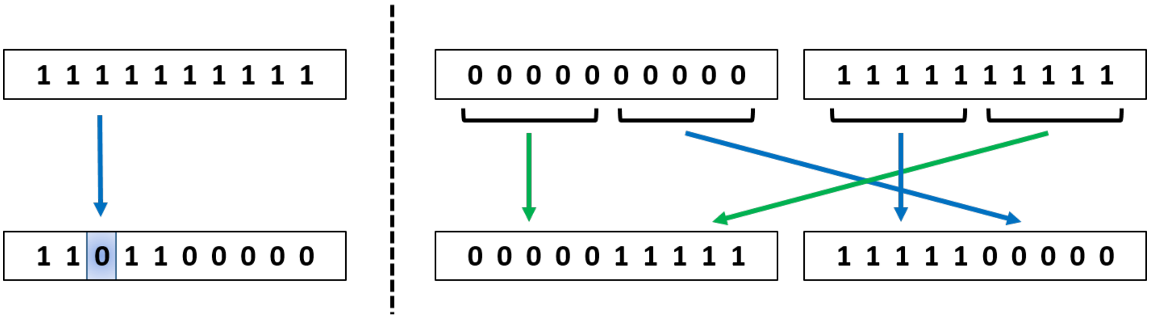


Figure 4.3: Schematic representation of mutation (left) and recombination (right) operations.

Please notice that different types of random changes may be applied, such as randomly switching or replacing different/multiple segments of the genome.

- *Replacement.* Also referred to as survivor selection mechanism, it aims to assess individuals based on their quality, after the generation of the offspring. Differently from previous steps, this process is typically deterministic, aiming to identify and promoting individuals with the best traits to remain part of the genetic pool. As the population size is typically constant at the end of each iteration, the selection keeps or replaces previous individuals on a quality metric base.

Competitive Coevolution. In many adversarial problems, such as the game studied in this work, the quality of a solution is dependent on its interaction with other solutions. Competitive coevolutionary algorithms accommodate this by evaluating each solution against a sample of others, to estimate overall fitness [115]. Often, when the roles of solutions in a problem are not symmetric, multiple separate populations of candidate solutions are used. Competitive coevolution operates by encouraging a "coevolutionary arms race" in which solutions continually evolve under selection pressure to exploit weaknesses in their opponents, providing an adaptive evolutionary pressure towards strong, robust solutions that can function against a wide variety of adversaries. However, it is necessary to ensure that the solutions produced constitute a genuine improvement in quality, rather than a narrow focus on defeating the current opponents without improving global performance. Demonstrating this requires comparing solutions against new opponents that were not involved in their evolution, to function as a validation set [116].

4.2.3 Genetic Programming

Genetic programming is a broad technique used in the field of evolutionary computation which allows computers to evolve dynamically-structured solutions to problems, sometimes described as "programs", though the form of these solutions can range anywhere from symbolic equations, to electrical circuits, to literal computer programs [117]. These solutions are most commonly structured as trees, such as a syntax tree or decision tree, but many other representations exist. In this work, we use tree-based GP to evolve syntax trees representing the strategy for a defender agent (the details of our implementation are provided in Section 4.4.2).

Trees in GP are composed of a variety of different node types, called "primitives", each performing a certain function. When representing syntax trees in GP, each node in the tree takes inputs from its children, and outputs a new value to its parent, with the value at the root node being the output of the whole tree. Terminal nodes of the tree serve as input variables, sensor values, or constants, while the internal nodes are functions or operators that modify these values. For example, a tree representing an arithmetic function on the variables x might have the operators $+$, $-$, $\times$, and $\div$ as internal primitives, with x and integer constant nodes as terminals. An example of a tree representation of a program is reported in Fig. 4.4 .

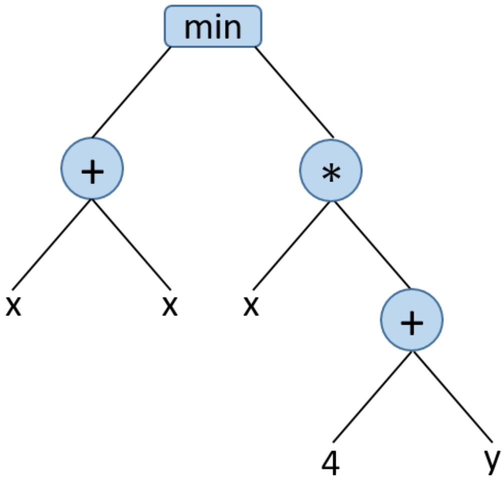


Figure 4.4: GP tree representation for the program `min(x+x,x*(4+y))`.

Trees are initialized at the start of evolution by randomly selecting primitives to build a tree of a given depth, that is, the levels of a tree, choosing a uniform distribution of depths to form the population. Similarly to what discussed in Sec. 4.2.2, trees evolutionary process involves the generation of new, modified trees through the mutation and recombination operators, where mutation modifies a single parent, and recombination combines the genes of multiple parents. Typically, for tree-based GP, mutation is performed by copying the parent's tree, and then randomizing a randomly-chosen subtree to create a local change. Recombination is similar, combining two parents by copying the first parent's tree, then replacing a randomly chosen subtree with a random subtree from the second parent. This allows the spread of potentially meaningful subunits of a program across the population.

In this work, we use a variant of tree-based GP called strongly-typed GP [118], in which every primitive has a specific type (such as integer, point, or matrix) associated with its output and each of its inputs. A constraint is applied such that the output type of a child node must match the corresponding input type of the parent. This constraint means that during recombination, the donated subtree must have the same output type as the one that is being replaced. Typing allows GP to more easily represent complex environments where multiple types of data matter, and simplifies the use of primitive functions that need parameters of different types, such as scaling a matrix by a scalar. Trees in GP are commonly initialized using a method called "Ramped Half-and-Half", which tries to ensure diverse initial trees by using a mix of two algorithms, “Grow” and “Full”. In both cases, individuals are initialized such that their depth is bounded by a user-defined maximum. The distinction between the two methods stand in the fact that, while in the Grow approach trees can be initialized with various sizes and shapes, in the Full approach all leaves are at the same depth. When conducting the experiments, we have observed that the restrictions of strongly-typed GP particularly bias the Full algorithm due to competing constraints on tree structure. As such, in the results later discussed solely the Grow method is utilized.

4.3 The Ground Station Transit Time Game

In an adversarial setting, the impact of an attack may be correlated to its detection time. This is one of the key lessons learned during the Hack-a-Sat[2] competition promoted by the Air Force Research Lab (AFRL). Throughout the first two years of the competition, now in its third year, the challenges faced by the participants highlighted how timing constitutes a crucial aspect for success, especially when it comes to attacking and defending satellites that allow for a brief time window for action. This underlying idea constitutes the foundations of what we called an adversarial ground station transit time game [109], with transit time here intended as the time that passes during the transit of a satellite between a hostile ground station and one controlled by the constellation operator. Our general modeling is comprised of a constellation operated by a single agent, which we label as defender, while a competing agent, here referred to as attacker, attempts to disrupt the constellation service (Fig. 4.5).

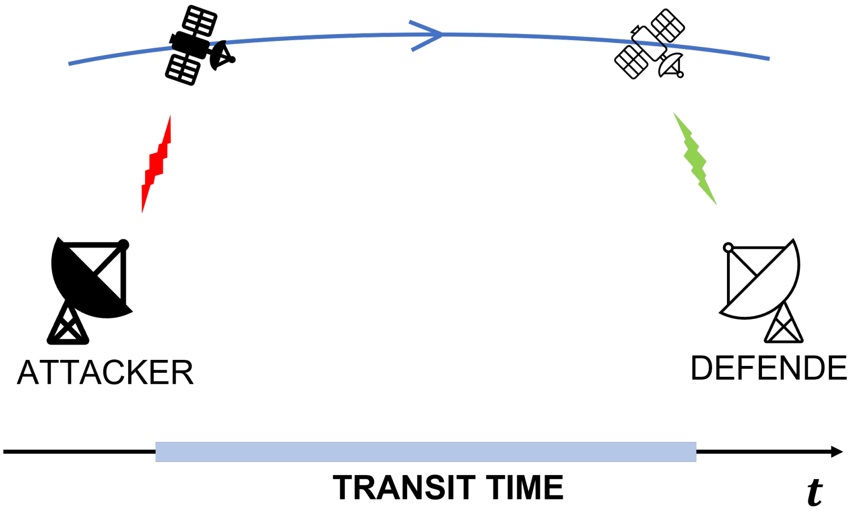


Figure 4.5: Transit time concept schematic representation.

In this work, the constellation service is defined as the coverage provided by the satellites, which is guaranteed when the constellation is fully operational under the control of the defender. Within this framework, we assume the attacker to be able to compromise one or more individual satellites when line of sight (LoS) exists between them and the attacker's ground node; the defender is capable of re-establishing nominal satellite operations when LoS exists between a

[2] https://hackasat.com

compromised satellite and one of its own stations, partially or entirely recovering the original service level. We assume the defender checks the status of its satellites as soon as one of its stations enters the access area of a satellite, that is, the surface of the Earth visible by a spacecraft. Similarly, we assume the attacker compromises the satellites as soon as one of its stations enters the access area. To compute the condition of visibility, we follow a classic unit vector formulation [119], which involves the determination of the angle between the position vector of a station and a satellite. As discussed in Sec. 4.2.1, in our modeling we assume the Earth to be a perfect sphere and the satellites orbits to be circular. Following the previously introduced notation, the unit position vectors, expressed as a function of time, of a satellite and a generic ground station (GS) can be conveniently expressed as:

$$\hat{\mathbf{r}}_{SAT}(t) = \begin{bmatrix} \cos\Omega\cos(\theta(t)) - \sin\Omega\sin(\theta(t))\cos i \\ \sin\Omega\cos(\theta(t)) + \cos\Omega\sin(\theta(t))\cos i \\ \sin(\theta(t))\sin i \end{bmatrix} \tag{4.8}$$

$$\hat{\mathbf{r}}_{GS}(t) = \begin{bmatrix} \cos(lon_{GS} + \omega_E t)\cos(lat_{GS}) \\ \sin(lon_{GS} + \omega_E t)\cos(lat_{GS}) \\ \sin(lat_{GS}) \end{bmatrix} \tag{4.9}$$

where Ω is the right ascension of the ascending node (here assumed to have the same range as the longitude) of a generic orbital plane, $\theta(t)$ represents the time evolution of the true anomaly, i is the orbit inclination, ω_E is the Earth angular velocity, and lon_{GS} and lat_{GS} represent the longitude and the latitude of a ground station. Defining then the maximum central angle, λ_{max}, we consider the LoS to exist when the condition $\cos(\lambda) \geq \cos(\lambda_{max})$ is met, with λ being the angle between a spacecraft and ground station position vectors.

Given these abstracted attack and defense actions, the objective of the attacker is to identify positions on the globe where deploying ground nodes such that the degrading effect on the constellation is maximized. Once deployed, the attacker's stations are considered to be fixed for the entire duration of the simulation, throughout this work limited to two days. On the other hand, the defender aims to identify the best location(s) to position an additional ground node such that the effect of the malevolent actor's actions are maximally reduced. In our current formulation, we let the defender's node deployment to happen after one day, a time window deemed to be long enough for the defender to perform such action, while collecting sufficient observational information about the effects of the attacker's action (we investigate varying this reaction time in section 4.5.9).

As an indicator of the constellation status, we consider the cumulative satellite up-time, defined as the total time during which each satellite is operative under the defender's control; from this perspective, the goal of the attacker is then to maximize the satellite downtime (opposite of the up-time), which is then reflected on the coverage provided by the constellation. To quantify the up-time, assuming N as the number of time steps and M as the total number of satellites in the constellation, we define the Boolean variable o_{nm} (with $n \in [1, N]$ and $m \in [1, M]$), representative of the operational status of the m-th satellite at the n-th time step as follows:

$$o_{nm} = \begin{cases} 1 & if\ satellite\ m\ under\ defender's\ control \\ 0 & otherwise \end{cases}$$

Consequently, the goal of the attacker is to minimize the following objective function, while the goal of the defender is to maximize it:

$$f = \sum_{n} \sum_{m} o_{nm}$$

For a given simulation duration, we split its length in two equal parts: at the beginning of the first half, the attacker is granted the possibility to deploy its stations, while at the beginning of the second half the defender adds its nodes. We will refer to the additional defender nodes that are deployed at the beginning of the second episode half as mobile ground stations. The quality of the defender behavior is then measured as the relative improvement in up-time at the end of the simulation with respect to up-time measured after the first half. As a final remark, it is relevant to emphasize that our formulation ultimately involves two distinct classes of actions: 1) satellite disabling/re-enabling, which happens under satisfaction of the line-of-sight condition previously described, and 2) node deployment, which is the output of our solution method

(Sec. 4.4). Therefore, while the action of deploying nodes is conducted at the beginning of the first half (for the attacker) and second half (for the defender), throughout the entire simulation, at each step, both deployed attacker nodes and defender's initial fixed ground nodes are constantly acting on the satellites.

4.4 Methodology

Previous work on adversarial ground station transit time games [109] determined that competitive coevolutionary algorithms [111, 115] are well-suited to the problem of adversarial ground station placement. Population-based stochastic search algorithms such as evolutionary algorithms stand out for their effectiveness in complicated, high-dimensional solution spaces, such as those present when numerous ground stations and satellites are interacting. In addition, the solution representation of genetic programming [112, 117] is available as a natural method of encoding and evolving complex, dynamic behaviors for agents when using an evolutionary algorithm. Because of these benefits, in this work we employ a competitive coevolutionary algorithm for the generation of attacker and defender agents, using GP trees to represent the defender's reactive strategy. If we were only searching for solutions to a single problem instance, alternative techniques like simulated annealing, Monte Carlo methods, or direct evolution of solutions might be substantially faster than a coevolutionary search. However, competitive coevolution with GP results in the creation of more broadly-applicable policies which themselves run quickly, and is a technique which can be more effectively extended to more complex scenarios. Alternative methods of policy search such as reinforcement learning on neural networks may also be applicable to this problem, although we have not yet investigated this in detail. In the remainder of this section, details of our implementation and experiment methodology are provided.

4.4.1 Competitive Coevolutionary Algorithms

We employ competitive coevolution with two populations, one representing attacker positions and one representing defender strategies. In our model the attacker acts first, and it is not reactive; hence, attackers are evolved as a simple list of coordinates as in existing work on this problem [109]. The defender strategies are evolved as GP trees, which represent a computer program as a parse tree of functions (for details about our GP tree implementation, please refer to 4.4.2). The evolutionary parameters used for competitive coevolution are listed in Table 4.1. Due to the high computational cost of performing experiments, only minimal hand tuning of parameters was performed. Of note is that our coevolutionary experiments are configured to be somewhat smaller in population size than is ideal for GP [120] in order to facilitate a large number of runs of the algorithm for statistical purposes. Runs with these parameters typically last up to two hours on our highly-parallelized hardware, with 30 runs per experiment performed (where one “experiment” corresponds to a single configuration of scenario and parameters). Informal testing found that runs configured with larger population sizes or evaluation limits produced somewhat increased complexity of agents and fitness scores, but insufficient to justify the significantly longer runtime for our experiments, as runtime of evolution is proportional to the number of evaluations. In experiments where the number of evaluations was increased to $\sim$1,000,000 from our default $\sim$100,000 (and with a correspondingly larger population), a single evolutionary run was observed to last up to a day. A schematic of the coevolutionary cycle for this work is presented in Figure 4.6.

Table 4.1: Parameters used in evolution. The total number of evaluations (runs of the simulation) needed is proportional to the number of individuals per generation ($\mu + \lambda$), the number of opponents sampled per individual, and the number of generations.

Evaluations	98,500
Generations	50
Population size (μ)	50
Number of children (λ)	150
Mutation/recombination ratio	50%
GP tree initialization	Ramped Grow, depth ≤ 7
Fixed position initialization	Uniform random
Parent selection	(k=2)-tournament
Survival selection	Truncation
Opponents sampled per individual	10

4.4.2 GP: Defender Strategy Trees

The action performed by the defender agent is to select a location to place an additional ground station, once half of the simulation time has passed. As small changes in the positioning of the attacker ground stations can have large effects on which satellites are compromised and when, the defender needs to be able to choose this location dynamically based on observations of the current state of the constellation.

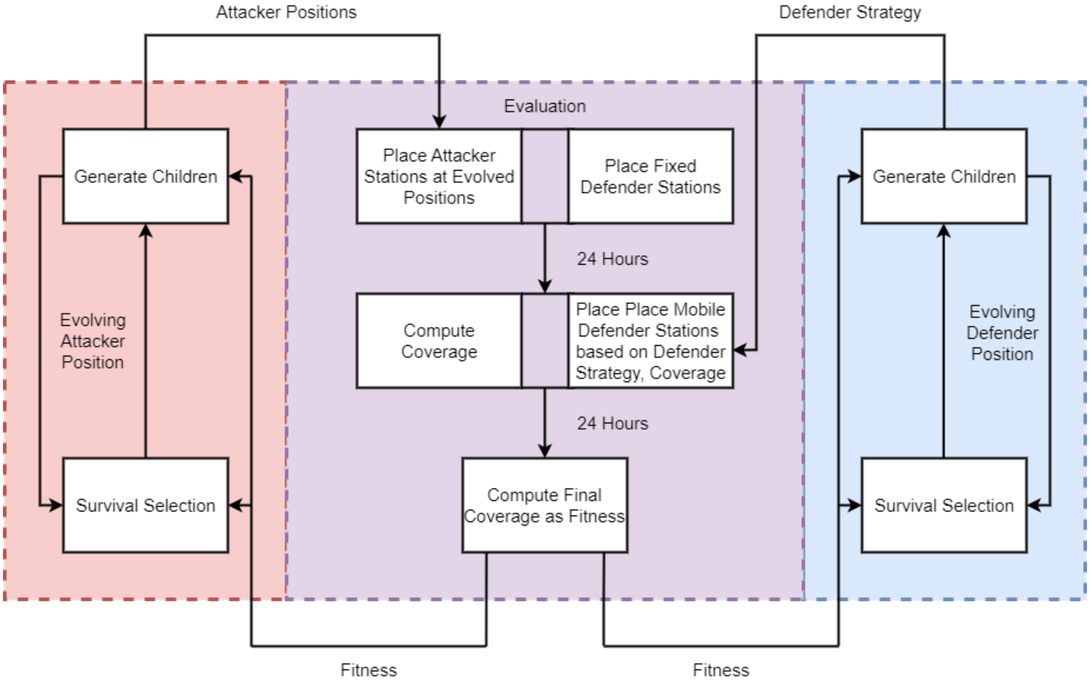

Attacker Positions
Defender Strategy
Evaluation
Generate Children
Place Attacker Stations at Evolved Positions
Place Fixed Defender Stations
Generate Children
24 Hours
Evolving Attacker Position
Compute Coverage
Place Place Mobile Defender Stations based on Defender Strategy, Coverage
Evolving Defender Position
24 Hours
Survival Selection
Compute Final Coverage as Fitness
Survival Selection
Fitness
Fitness

Figure 4.6: Diagram of the competitive coevolutionary cycle and its interaction with the constellation simulation. The defender receives a map of the coverage calculated after 50% of the simulation time.

Our defender agents are strongly-typed GP trees which primarily operate on two-dimensional arrays described as “heatmaps”. These heatmaps divide the world into latitude-longitude grid cells with a given resolution, and store a real number value for each cell. In order to select a location to place a ground station, a tree outputs a heatmap based on its sensor inputs describing which locations it considers more or less desirable to place its mobile ground station, placing it at the coordinates corresponding to the lowest-valued grid cell (with ties broken randomly). Figure 4.7, top-right, gives an example of such an output map. Optionally, a mask can be applied at this step, such that the defender instead uses the lowest-valued grid cell excluding the masked region, for example, prohibiting the placement of ground stations on water. These heatmaps are used as the primary data type because they provide an intuitive spatial representation of data over the surface of the planet, and can be recombined arithmetically or manipulated in complex ways while preserving the geographic correlation of the data. The grid resolution is a parameter which can be varied independently from a given strategy; a low resolution can be used during the expensive evolutionary process to decrease computation time (we use a 2° resolution), and then increased during application of a finalized agent to give more precise results, though we don't use this in our analysis.

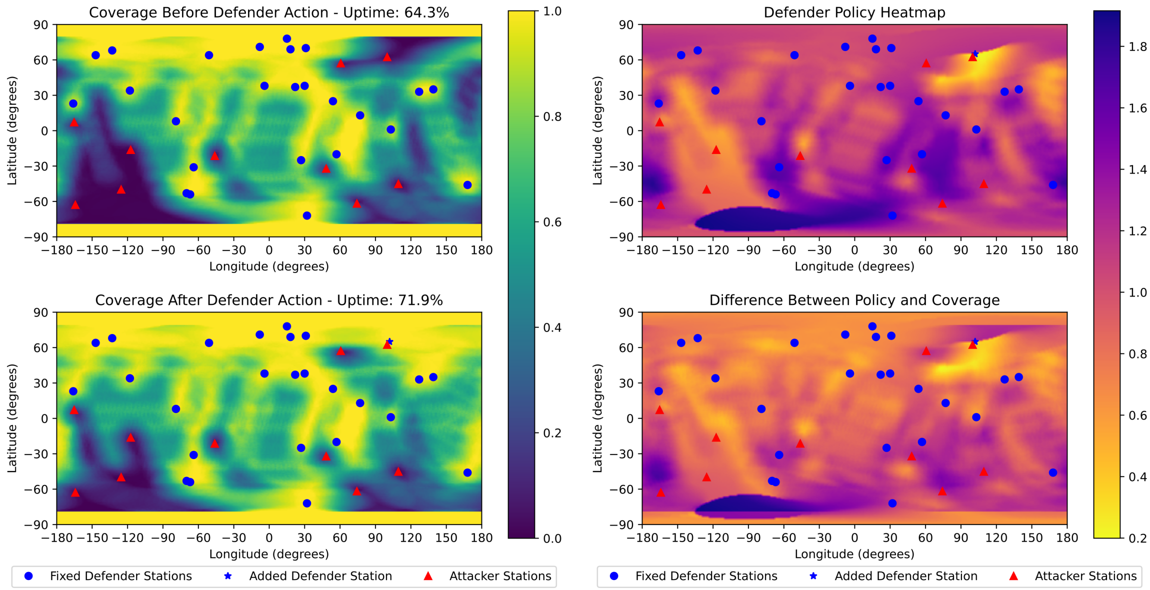


Figure 4.7: Visualizations of a game between an attacker and a defender agent. **Top left:** Map of relative coverage before defender action. "Relative coverage" refers to the mean service coverage at a given location during an attack, as a fraction of mean service coverage in a situation without an attacker. **Bottom left:** Map of relative coverage after defender action. **Top right:** Policy map showing regions the defender values as a site to place a ground station (lower is better). **Bottom right:** Normalized difference between defender policy and the coverage map, highlighting nonlinear complexities in its behavior.

As the primary input to the defender, we provide a heatmap of satellite up-time which we refer to as the "coverage map". This coverage map serves to inform the agent of the degree to which each location on the planet has observed fewer working satellites than normal. The coverage map is determined through the following procedure: for each latitude-longitude cell of the heatmap, at each time step from the start of the simulation to the present, we determine whether any satellites have sight of that cell, and whether any are defender-controlled (that is, not attacker-controlled). We define the relative coverage value of that cell to be the number of timesteps in which a defender-controlled satellite is visible, divided by the number of timesteps where any satellite is visible. In cells where no satellites at all are visible, such as near the poles, the value is treated as 100%. The process is summarized in Figure 4.8, and an example coverage map is shown in Figure 4.7, top-left. If an area on the coverage map reports low coverage, that indicates that an attacker is able to compromise satellites "upstream" of that area

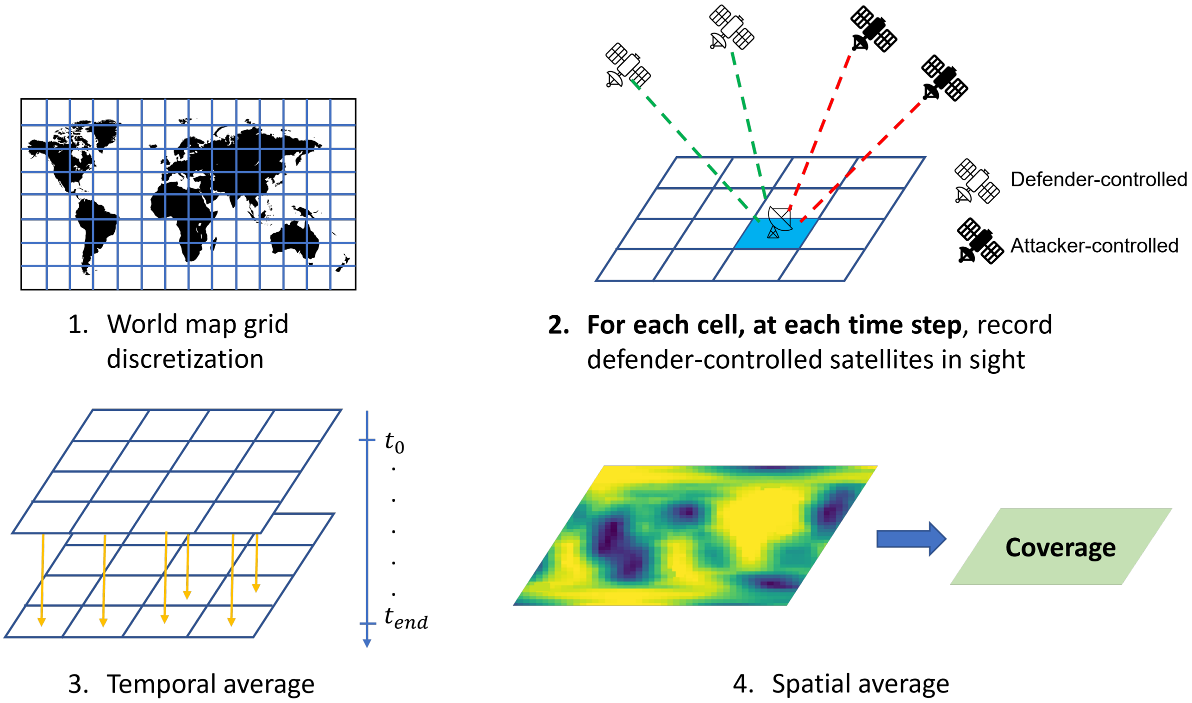

Defender-controlled
Attacker-controlled
1. World map grid discretization
2. **For each cell, at each time step**, record defender-controlled satellites in sight
t_0
t_{end}
3. Temporal average
Coverage
4. Spatial average

Figure 4.8: Coverage map computation steps. Color scale is representative of the temporally averaged coverage over the corresponding grid cell based on actual satellite operational status.

without a defender ground station being able to respond. These low-coverage regions can then be suggestive of good locations to place the defender's additional ground station. The defender is also given a heatmap with the "base coverage" at each cell, which gives the percentage of timesteps where any satellite was visible. Since a defender knows the location of its own ground stations, we also provide a heatmap giving the distance to the nearest defender station at each cell (the defender has no information about attacker station locations except what can be ascertained from the coverage map). This map can be useful to the agent in avoiding the selection of locations that are already well-covered, or in more complex behaviors such as identifying regions that should be well-covered but are not, which might indicate the location of an attacker. Finally, a set of fixed gradient heatmaps are given for use as constants during computation. The fact that some of these inputs suggest simple strategies on their own is advantageous in the design of this GP representation, because it means that simple trees developed early in evolution can still achieve minor success. This is a helpful property in evolutionary algorithms, as it allows the algorithm to discover a better search gradient early on.

In addition to the heatmap data type, defender GP trees operate on latitude-longitude pairs (the "point" data type), as well as real-valued "scalar" and "angular" values, with the latter between -180 and 180 degrees. Points can be created by selecting the maximum- or minimum-valued point from an input heatmap. Scalar values are simply constant parameters, and angular values are either constants or a sensor value that gives the inclination of the constellation's orbits. The full set of primitive nodes used to construct a defender strategy tree is given in Table 4.2. Most of the primitive functions provided to the defender exist for manipulating and combining heatmaps. These are primarily arithmetic functions, such as adding two heatmaps together elementwise, or multiplying each element of the heatmap by a scalar. We also provide a set of functions for "rotating" a heatmap, by mapping the heatmap's grid to points on a sphere, performing a rotation transformation on the grid points based on the input parameters, and then resampling the transformed points in the original coordinate system to create a new output heatmap. This resampling sets the value of each heatmap cell to the average value of the transformed heatmap at each of the cell's corners. These rotation functions exist primarily so that the agents can make decisions based on the orbital paths of satellites, such as positioning a ground station "upstream" of a low-coverage region. A more detailed explanation of this rotation operation is provided in Appendix C.1.

A graphical example of a GP tree is given in Figure 4.9, demonstrating how heatmaps can be combined into a policy. This tree gives priority to regions 90° east from a region of low coverage (left branch), and far from existing defender stations (right branch). Figure 4.10 shows a different, more complex GP tree for a defender strategy created through evolution. This tree notably makes use of several different sensor heatmaps as inputs, utilizing the rotation functions and arithmetic operations in order to combine these sources. The primitive set chosen for this implementation is intended to enable a wide variety of ways to combine and manipulate input data, allowing for complex and unexpected strategies to emerge.

Table 4.2: Primitives used by the GP trees of the reactive defender agents. The types used are: Angular (A), Scalar (S), Point (P), and Heatmap (H).

Name	Type	Inputs	Description
angle literal	A	None	A specific angle, initialized at uniform random
scalar literal	S	None	A specific scalar between -10 and 10, initialized at uniform random
coverage	H	None	The coverage map resulting from the attack after one day
base coverage	H	None	The coverage map when no attackers are present
latitude gradient	H	None	A gradient from -1 to 1, from south to north pole
longitude gradient	H	None	A gradient from -1 to 1, 0° to 180° longitude
defender stations	H	None	The distance to the nearest defender station, normalized between 0 and 1
inclination	A	None	The inclination of the constellation
add	H	H, H	Adds two heatmaps
subtract	H	H, H	Subtracts one heatmap from another
multiply	H	H, H	Multiplies two heatmaps
negate	H	H	Negates the values of a heatmap
normalize	H	H, H	Normalizes a heatmap between -1 and 1
add scalar	H	H, S	Adds a value to all cells in a heatmap
multiply scalar	H	H, S	Multiplies all cells in a heatmap by a value
mask greater than	H	H, S	Create a binary heatmap with 1 where the original exceeded a value and 0 elsewhere
mask less than	H	H, S	Create a binary heatmap with 1 where the original was less than a value and 0 elsewhere
rotate horizontal	H	H, A	Rotate a heatmap around the north pole by an angle
rotate azimuth	H	H,P,A,A	Rotate a heatmap in 3D space such that a given point moves along a given compass direction
rotate inclination	H	H,P,A,A	Rotate a heatmap in 3D space such that a given point moves along an orbital plane with given inclination
maximum point	P	H	Get the maximum-valued point from a heatmap
minimum point	P	H	Get the minimum-valued point from a heatmap

4.4.3 Experiment Procedure

To compare the quality of individual defender agents against each other, we measure the degree to which constellation up-time has increased in the second half of the simulation (after the defender has placed its additional ground station), compared to the first half of the simulation. A score of 50%, for example, would indicate that placing the mobile ground station increased the global service up-time by 50% compared to service up-time during the first half of the simulation.

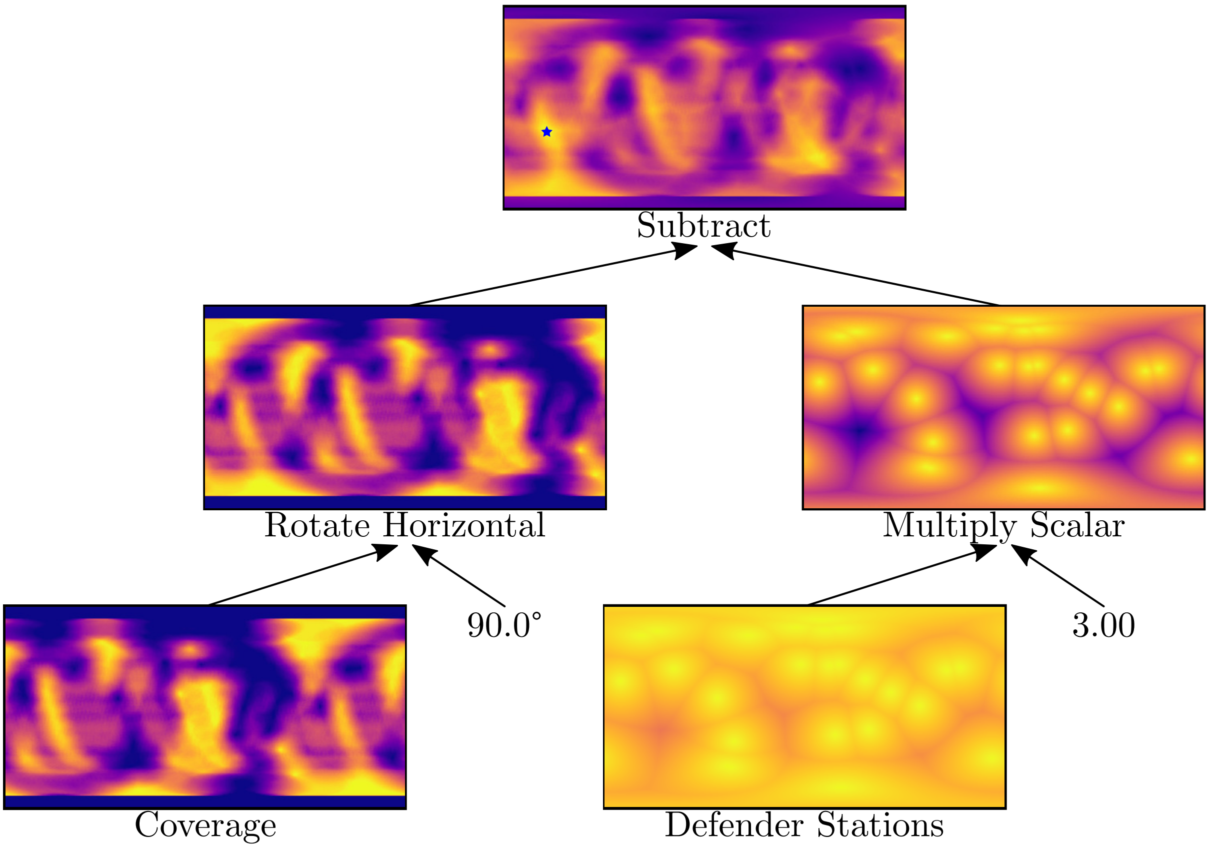


Figure 4.9: Graphical toy example of a GP tree showing how heatmaps are operated on within a tree, referencing primitives defined in Table 4.2. This tree returns the policy heatmap returned by the "subtract" node at the root, which takes its left and right child nodes as operands, and so on throughout the tree. Numbers represent literal values. Brighter yellow colors are prioritized by the policy for placing the mobile node, with the preferred location marked with a blue star.

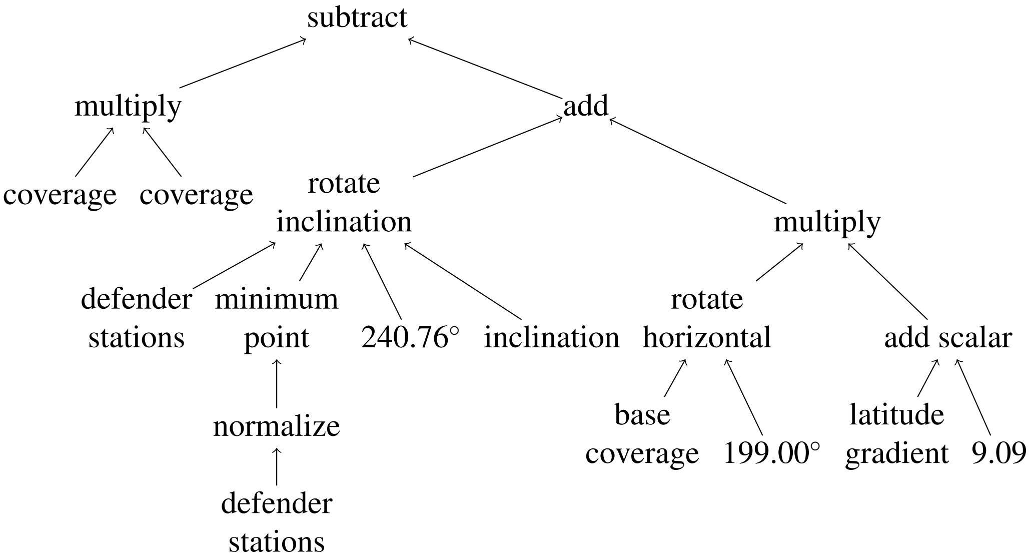


Figure 4.10: Example GP syntax tree from an evolved defender agent referencing primitives defined in Table 4.2, with parameters passed from left to right.

In order to compare the defenders produced by the two different methodologies, we perform 30 evolutionary runs of both methodologies, and take the defender agent which scored the highest against its coevolutionary opponents in the final generation of each run as a representative. We then produce a test set of attacker agents in the same way, from 30 separate evolutionary runs, to ensure that none of the defenders or attackers have directly adapted to one another during evolution. This results in two sets of 30 defenders, and a test set of 30 attackers. Each defender is evaluated against this common test set to obtain an average improvement in constellation up-time per defender. We then perform a Student's t-test to determine if the defenders generated by one methodology produced significantly different performance scores to those from the other; a significance level α = 0.05 is used in this study. We also report the best, mean, and worst scores from that set of 30 defenders.

The parameters used to construct and simulate the constellation are given in Table 4.3. Throughout this work, we consider a 2π-constellation, with orbital planes equally spaced in RAAN between 0° and 360°. The computational cost of the simulation is proportional to the total number of satellites, the temporal resolution, and the square of the coverage map resolution. Note that decreasing the coverage map resolution will decrease the precision with which the reactive defender agent can act, while decreasing the temporal resolution increases the chance of a fast-moving satellite skipping over a ground station in between time steps. Temporal resolutions of at most four minutes were observed to be accurate to within a small margin of error for these parameters.

Table 4.3: Parameters of the constellation simulation.

Number of orbital planes	10
Satellites per orbital plane	10
Inclination	70°
Altitude	1000 km
Central angle	10°
Min. elevation angle	34.7°
Coverage map resolution	2°
Temporal resolution	2 minutes
Simulation duration	48 hours

4.4.4 Visualizing Results

While statistical tests can measure the numerical performance of these evolved strategies, it does not provide much insight into the underlying behavior of the attacker and defender agents. Furthermore, although one can study an individual agent and its behavior in detail, it is often difficult for humans to interpret the logic driving an AI agent (see, for example, Fig. 4.10), and this gives little information about the general characteristics of agents produced by evolution in a given scenario. In order to make qualitative comparisons of agents' behavior in aggregate, we can visualize the set of geographic points chosen by all attacker or defender agents for a given scenario against a variety of opponents. To this end, we use Kernel Density Estimation (KDE)[3] as a visualization technique. KDE is a method which takes a discrete collection of points, and estimates a smooth probability density function for the random variable producing those points. The resulting function allows for a visualization of agent actions that displays density more clearly than simply plotting those points. From these visualizations, one can see locations which highly-evolved attackers and defenders tend to choose frequently or more rarely, giving a sense of the dynamics of the game. It is also easier to compare visualizations between different cases or configurations to see how those dynamics change as a result of that difference. An example of this visualization technique is provided in Fig. 4.11.

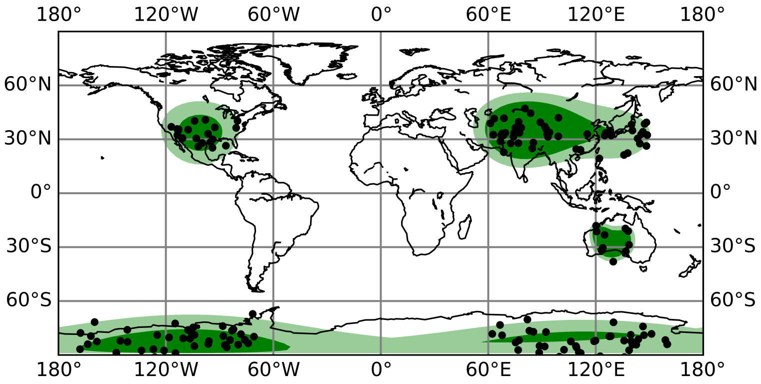


Figure 4.11: An example KDE (green regions) generated from a set of random points (black dots).

[3] Using the *spherical_kde* Python library by William James Handley, `https://github.com/williamjameshandley/spherical_kde`

To evaluate the quality of the evolved attacker locations and defender strategies, we conducted an extended test campaign over a variety of scenarios. Throughout this section, capabilities are shown incrementally, first demonstrating the performance of the algorithm over simpler cases, then moving toward more realistic conditions.

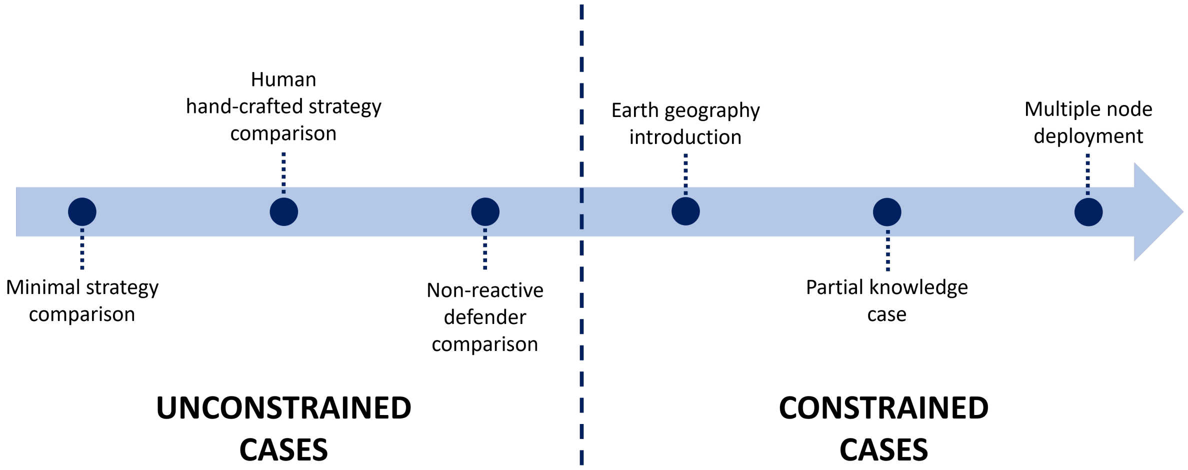

Human hand-crafted strategy comparison
Earth geography introduction
Multiple node deployment
Minimal strategy comparison
Non-reactive defender comparison
Partial knowledge case
UNCONSTRAINED CASES
CONSTRAINED CASES

Figure 4.12: Results section graphical roadmap.

Fig. 4.12 summarizes the general organization of the section, with the arrow direction mirroring the increasing complexity of the simulated cases. In the following, complexity encompasses the increasing number of ground nodes (for both the attacker and the defender) and the introduction of constraints. We consider both fictitious scenarios where the fixed defender ground station placement is arbitrarily assigned, regardless of the corresponding location on the real-world map, as well as scenarios including an existing ground station distribution. For the latter, the KSAT Global Ground Station Network[4] is selected. To clearly identify the scenarios, we label them as follows: "Simple2" — two fixed defender stations and two attacker stations; "Simple4" — four fixed defender stations and four attacker stations; "KSat5" — five attacker stations and 25 defender stations belonging to the KSAT ground network; "KSat10" — ten attacker stations and, again, 25 defender stations belonging to the KSAT network. Please note that when geographic constraints are introduced, only the "KSat5" and "KSat10" are considered.

[4] `https://www.ksat.no/ground-network-services/` (our modeled network does not include some recently added ground stations)

With the exception of the "multiple node deployment" case, in all the cases the defender is allowed to deploy one single node.

4.5.1 Case 1: minimal strategy comparison

Two simple strategies are implemented and compared against the policy evolved through our new method: the first one, labelled as "Reference", involves placing the additional defender station in the location with lowest relative coverage, thus defining a minimally complex, reasonable strategy; the second one, labelled as "Random", simply places the additional node at a random location, providing a baseline performance level.

Table 4.4: Performance of 30 evolved defenders against a simple reference agent, and a random agent. Best, average, and worst scores among those 30 are given. Percentages representing the relative improvement in up-time during the second half with respect to the first half, following the deployment of the defender's mobile node

.

Scenario	Best	Avg.	Worst	Reference	Random
Simple2	119.89%	85.57%	48.44%	26.42%	23.05%
Simple4	108.36%	70.32%	29.07%	27.35%	19.75%
KSat5	17.64%	13.40%	8.29%	4.01%	1.41%
KSat10	15.37%	11.70%	5.45%	4.30%	1.68%

Table 4.4 presents, for the four scenarios described above, the performance of the representative defender agents from 30 evolutionary runs and the two simple strategies against a test set of 30 evolved attackers. Every one of the evolved defender strategies is able to outperform the simple reference strategy, oftentimes by a substantial margin. The comparatively weak performance of the random agent confirms the effectiveness of both the evolved agents and the simple reference strategy, particularly on the more complex scenarios.

4.5.2 Case 2: human hand-crafted strategy comparison

To preliminarily evaluate the quality of our solution method, we developed a simple hand-crafted strategy to be compared with the policy evolved by the defender. In fact, knowledge of the problem modeling, availability of the coverage map, and human intuition may lead to significant improvements with respect to the "Random" (completely agnostic with respect to the coverage map) and "Reference" (which simply selects the locations with lowest coverage for the additional defender station) strategies. To develop such a heuristic strategy, the following approach has been adopted: 1) we generated several coverage maps associated with different distributions of fixed defender locations and attackers to gain insight in the relation between node placement and coverage maps; 2) we observed how the addition of a defender node in each of those maps affected the coverage, mimicking the process followed by the algorithm described so far; 3) we encoded our observations into a decision tree. The strategy encoded in the tree was carried out by a human in order to evaluate its performance, providing them with the same information given to our defender agents. We evaluated the hand-crafted strategy against the same attackers used to evaluate the evolved strategies, and compared the resulting scores against the same "Best" and "Median" agents used for the defender robustness experiments (shown in Fig. 4.15).

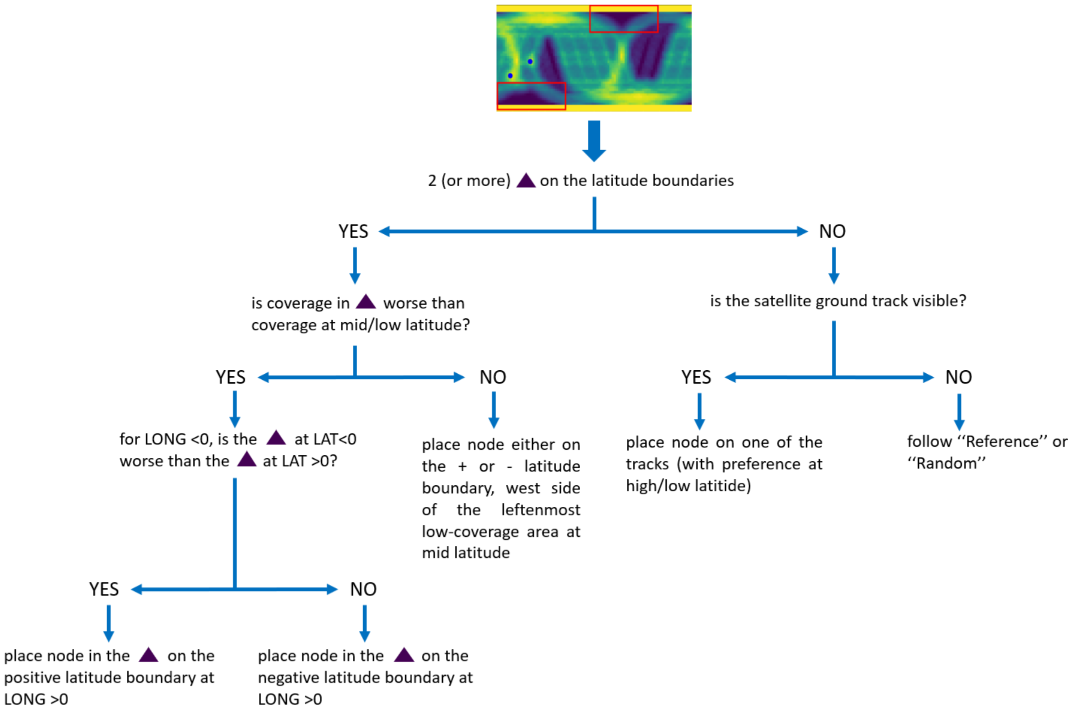

2 (or more) ▲ on the latitude boundaries
YES
NO
is coverage in ▲ worse than coverage at mid/low latitude?
is the satellite ground track visible?
YES
NO
YES
NO
for LONG <0, is the ▲ at LAT<0 worse than the ▲ at LAT >0?
place node either on the + or - latitude boundary, west side of the leftenmost low-coverage area at mid latitude
place node on one of the tracks (with preference at high/low latitide)
follow "Reference" or "Random"
YES
NO
place node in the ▲ on the positive latitude boundary at LONG >0
place node in the ▲ on the negative latitude boundary at LONG >0

Figure 4.13: Heuristic strategy diagram. LONG and LAT abbreviation for longitude and latitude respectively.

Figure 4.13 shows the developed strategy: the blue triangles in the decision-making process are representative of analogous shape, down-coverage area features frequently appearing in the coverage map after 50% of the simulation time has passed. Such features turn out to play a key role in the decision process: characteristics such as their location and extension are recognized as the major factors contributing to the final decision for the additional node positioning.

Table 4.5: Performance of the hand-crafted strategy ("Human") against the best and median evolved strategies for each scenario. Underlined results performed statistically significantly better than the hand-crafted strategy.

Scenario	Best	Median	Human
Simple2	119.89%	83.84%	107.56%
Simple4	108.36%	69.04%	74.45%
KSat5	17.64%	13.84%	11.61%
KSat10	15.37%	11.50%	12.83%

Table 4.5 reports the results obtained over four of the previously presented scenarios for the unconstrained case, compared against the best and median evolved strategies for each scenario. The simple logic depicted in Figure 4.13 is comparable in performance to the median evolved strategy, but is significantly less effective than the best evolved strategies for all but the simplest scenario. In addition, the hand-crafted strategy took much more time and human labor to create and evaluate than the evolved strategies, and required subjective human decision-making to operate, so the ability for evolution to typically produce strategies of a similar or greater quality is a promising result.

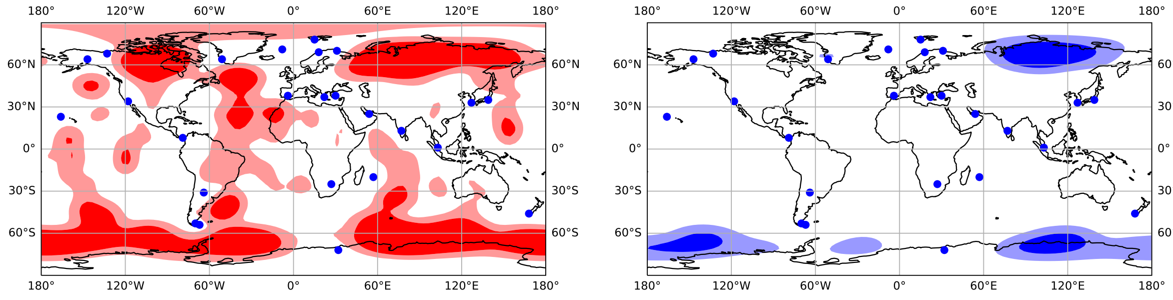
4.5.3 Case 3: non-reactive defender comparison

An initial approach to this problem [109] involved the direct evolution of coordinates for the defender's mobile ground station. While this methodology allows for fine-tuning of a defender's action against a single attack, it is much less effective against a wide variety of attackers, since such a defender does not represent a general-purpose strategy. This work hypothesizes that a reactive defender strategy which responds to observations resulting from its current opponent will be able to select actions which are more effective against individual attackers, in a wider variety of situations. To test this hypothesis, we coevolve fixed attacker and defender locations using the methodology from a first implementation of this work [109], for each of the previously mentioned four scenarios. We then evaluate the performance of the evolved defender locations against a test set of high-performing attackers that they did not evolve against, and compare the resulting scores with the performance of the previously-evolved reactive defender strategies on those attackers. The attacker agents used here are the best attackers from each of 30 separate evolutionary runs against evolving defender positions.

Table 4.6: Comparison of the performance of 30 evolved defender positions (Best, Avg., Worst) against 30 evolved reactive defender strategies (R. Best, R. Avg., R. Worst). Underlined results indicate statistically significant improvements for the reactive defender over the fixed defender.

Scenario	Best	Avg.	Worst	R. Best	R. Avg.	R. Worst
Simple2	67.63%	29.59%	7.24%	119.89%	85.57%	48.44%
Simple4	53.15%	31.22%	15.36%	108.36%	70.32%	29.07%
KSat5	18.05%	11.18%	0.11%	17.64%	13.40%	8.29%
KSat10	17.59%	10.30%	0.63%	15.37%	11.70%	5.45%

Our results in Table 4.6 demonstrate that, for the simpler scenarios, the reactive defender strategies are substantially more versatile than fixed defender locations at responding to a variety of attacks. In these scenarios, no fixed defender location is able to outperform the average defender strategy. For the more complex KSAT scenarios, however, the evolved defender positions perform unexpectedly well, and are comparable to the reactive defender strategies in average performance, though they are much less robust in terms of worst-case performance. For the Simple2, Simple4, and KSat5 scenarios, statistical testing finds the improvement significant, while the result for the KSat10 scenario is inconclusive.



(a) Attackers

(b) Defenders

Figure 4.14: Distributions of the ground station placements of evolved attackers and position-evolving defenders in the KSat10 scenario estimated by KDE. Fixed defender ground stations are shown as blue dots.

Upon examining the actual solutions proposed, the reason for the unusually strong performance of evolved defender positions on KSAT scenarios turns out to be the geography of the KSAT network, based on real Earth geography. Since no real-world geographic constraints are being applied at this stage to the placement of attacker or defender stations, a dominant strategy emerges (Fig. 4.14) in which attackers place nearly all of their ground stations in the ocean, where the land-bound KSAT network has limited reach (similar considerations apply also to stations located in Siberia and northern Canada). As a result, there is little meaningful variety in behavior between different attackers, resulting in a very narrow set of dominant defender positions that counter this single strategy. Because of this, a non-reactive defender which cannot adapt to different attacks is not strongly disadvantaged. In Section 4.5.5, we address these dynamics by constraining agents to land-based ground stations.

4.5.4 Defender strategy robustness

The effectiveness of an evolved defense strategy is measured by its robustness against a variety of attacks. To investigate this aspect, we examine the quality of several specific defenders, i.e., the evolved defenders with the best and median average performance against opponents for each scenario, against the best attacker positions evolved over 30 runs on the corresponding scenario.

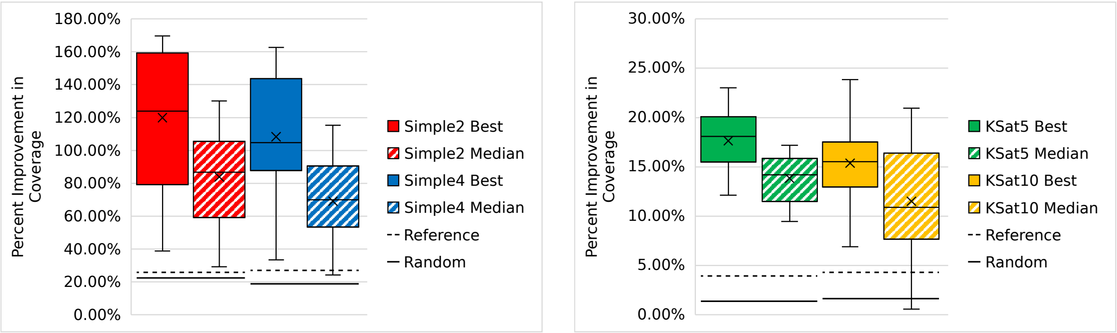


Figure 4.15: Unconstrained case: box plots showing distribution of defender strategy performance over 30 attackers. Reference and Random agent performance are shown as horizontal lines.

Figure 4.15 portrays the distribution of fitness scores received by eight selected agents over a case without geographic constraints. While all agents show variability in performance against different attack situations, they generally remain effective compared to the simple reference strategy and random agent.

4.5.5 Case 4: Earth geography introduction

In the cases described so far, ground node placement for both attacker and defender agents was allowed for every grid point of the Earth map, regardless of the real-world geographical boundaries. In lower-fidelity representations, dominant, less realistic solutions (such as the ocean-based strategies we observed) are possible, and evolution tends to identify and converge to them. However, these dominant solutions may disappear when complex, real-word constraints are introduced in the game dynamics. In order to bridge the gap with reality, we add an additional layer of complexity to the problem by introducing land and water attributes to the node coordinates, imposing that only land-based sites are admissible for node positioning. Note that our methodology is agnostic to the fidelity of the simulated game, provided that the GP tree function nodes are updated accordingly, and sufficient computational resources are spent on tree evolution.

To enforce land-based coordinates for attackers and defenders, we apply a repair function: in the event the evolved coordinates are water-based, this function "repairs" them to be the nearest land-based location. Throughout this process, we do not modify the stored values (the genotype), but solely the ground station locations interpreted from them (the phenotype). Allowing the genotype to retain coordinates located in the ocean makes it easier for mutation to cross oceans. We here need to specify that this operation has a stronger effect on the attacker. In fact, since the reactive defender strategy generates a priority value for each grid-cell, we can simply enforce the constraint by selecting the land-based coordinate featuring the highest score.

Table 4.7: Comparison of the performance of 30 evolved defender positions (Best, Avg., Worst) against 30 evolved reactive defender strategies (R. Best, R. Avg., R. Worst), all constrained to land on Earth. Underlined results indicate statistically significant improvements for the reactive defender over the fixed defender.

Scenario	Best	Avg.	Worst	R. Best	R. Avg.	R. Worst
KSat5	12.35%	1.06%	-0.89%	17.60%	13.96%	10.26%
KSat10	10.27%	0.83%	-0.70%	16.83%	13.38%	9.99%

Table 4.7 displays the results associated with this case, comparing 30 evolved defender positions against 30 evolved defender strategies. We use a set of 30 high-performing attackers evolved with this land constraint which these defenders did not encounter during evolution, analogous to those in Section 4.5.3. Constraining the agents to locations on land eliminated the consistent strategy available to fixed defender locations, reducing the average performance of these agents against attackers they did not evolve against; in fact, we can observe how the performance is consistently degraded even when compared to the simple reference strategy,

being more comparable to a random agent. To the contrary, the reactive defender strategy is not significantly affected by the application of the constraint, and performs comparably to the unconstrained case.

In all of the following cases, we keep the land constraint in place to prevent the emergence of these dominant and unrealistic strategies. As these cases do not change anything for the attacker, the same attacker set used for the land constraint experiments is reused, making fitness values directly comparable.

4.5.6 Case 5: partial knowledge case

Throughout the cases presented so far, we have always assumed the defender to have full knowledge about the coverage state, here intended as information about actual satellite service. This was possible thanks to the coverage map provided at evaluation time, which contained coverage information for all latitude-longitude grid cells, including all water-based positions. However, while the constellation coverage information, in its orbital mechanics meaning, can be always computed, information about satellite service would depend on the actual constellation operational status; as such, in reality knowledge about degraded constellation performance would likely originate primarily from populated areas. To model such a condition, we consider a simplified model where the defender has access to coverage information for all land-based locations (with the simplified assumption that these correspond to inhabited areas), but it has no information about ocean-based cells. To this purpose, the coverage map provided at halftime gives accurate information for land-based coordinates; however, the value reported for ocean cells is just the average coverage of all the land-based cells. We use an average here to ensure that the agents do not interpret these cells as having particularly high or low coverage.

Table 4.8: Comparison of the performance of 30 evolved reactive defender strategies with full information of ocean coverage (F. Best, F. Avg., F. Worst) against 30 evolved reactive defender strategies with partial information (P. Best, P. Avg., P. Worst), all constrained to land on Earth. Results showed no statistically significant difference.

Scenario	F. Best	F. Avg.	F. Worst	P. Best	P. Avg.	P. Worst
KSat5	17.60%	13.96%	10.26%	19.67%	15.46%	7.15%
KSat10	16.83%	13.38%	9.99%	18.08%	13.53%	7.78%

Table 4.8 summarizes the performance of 30 reactive defenders evolved over the full information case (from Section 4.5.5) against the performance of 30 strategies evolved with partial information. Analysis of the results reveals that no statistically significant difference exists between the performance of strategies evolved with the two simulation settings, meaning that the defenders are able to evolve successful strategies even when incomplete information is provided. However, comparing the locations chosen by defenders between these two cases (Figure 4.16(b) and Figure 4.16(c)) shows that the evolved defenders behave very differently under these constraints, placing fewer ground stations in North America and Europe, and more in Australia and the Antarctic Peninsula. This indicates that while under these conditions limiting the defender's information did not decrease performance, it forces the defender to find alternative strategies, which might not always be of identical quality in different situations.

4.5.7 Case 6: multiple node deployment

One of the main assumptions in the previously discussed cases is that the defender has the capability to deploy only a single auxiliary node. While this hypothesis may be reasonable for the most trivial cases like Simple2 and Simple4 (as an excessively unbalanced node deployment between defender and attacker may negatively impact coevolution and simultaneously hinder interpretability of the strategies), when moving to more realistic models it is reasonable to assume a more capable defender. Within our framework, such enhanced capability naturally translates into granting the defender the possibility to deploy multiple auxiliary nodes at once. To enable the defender strategies to represent taking multiple actions, each defender's GP tree is run iteratively, adding a ground station in the location chosen during each iteration, and augmenting the input with data about the ground stations added in previous iterations. These placements occur simultaneously in the simulation. The defender has a "defender stations" terminal node which reports the distance of each cell to the nearest defender station; therefore, by using this node in its tree, the defender can avoid placing its nodes in the same place as previous iterations.

Table 4.9 reports the performance of a reactive defender allowed to deploy three additional nodes. Any number of nodes could be used here, but we can not allow the defender agent itself to choose a number without requiring the addition of a cost minimization objective, which is

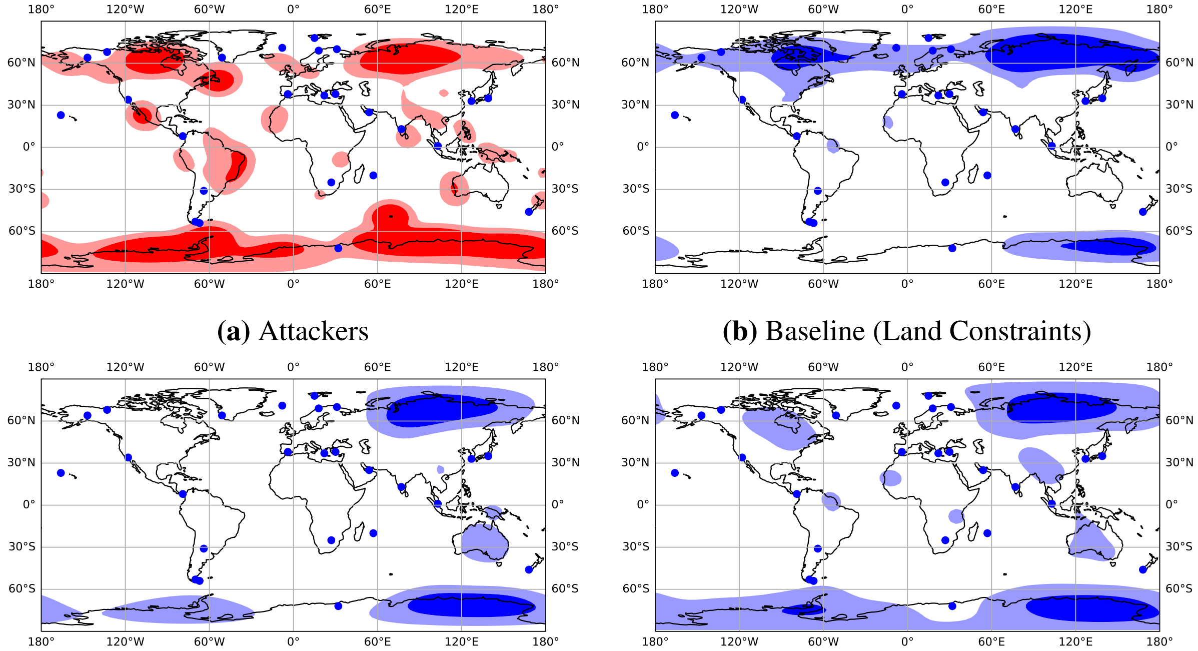
Table 4.9: Performance of 30 evolved reactive defender strategies allowed to deploy three additional ground stations (Best, Avg., Worst).

Scenario	Best	Avg.	Worst
KSat5	32.67%	24.96%	14.44%
KSat10	35.29%	30.64%	19.95%

outside of the scope of this experiment. Since these defenders have an inherent advantage in being able to place multiple additional ground stations, they can not be directly compared to the previous agents. However, the fact that we observe a substantial improvement with respect to the corresponding single-node deployment cases indicates that these defenders are correctly utilizing this advantage.

4.5.8 Strategy visualization and comparison

As discussed in Section 4.4.4, while the statistical analysis provides information about the quality of the results, it does not provide an intuitive understanding of the logic behind the attacker nodes placement and the defender evolved strategies. To overcome this limitation, a KDE-based technique is here employed to visualize for different scenarios what areas are more frequently (or rarely) selected as good regions for nodes distribution. An example of this for several of the KSat10 scenarios is shown in Figure 4.16. In all cases, the evolved attackers and defenders tended to choose locations far from existing defender stations, showing a particular preference for high latitude (both positive and negative) regions. Such a general trend is coherent with 1) the constellation orbital configuration, as the inclination of the orbits leads to a higher satellite density in the proximity of those regions and 2) the absence of cost modeling or other types of restriction within the simulation settings. Thanks to this visualization approach, we can observe that the partial information and multiple deployment cases result in the development of substantially different defender strategies to the baseline (land constraints) case, adapted to the specific characteristics of their environment. In particular, this visualization method highlights that the defender agents in the multiple deployment case are correctly evolving the capability to spread out their ground stations, leading to wider diversity of locations being covered. This allows them to exert control of regions that the attacker would otherwise dominate.



(a) Attackers

(b) Baseline (Land Constraints)

(c) Partial Information

(d) Multiple Deployments

Figure 4.16: Distributions of the ground station placements of evolved attackers and defenders on the KSat10 scenario, estimated by KDE. Fixed defender ground stations are shown as blue dots. The set of attackers displayed in Figure 4.16(a) is used as the opponent for all three sets of defenders.

As a conclusive remark, it is relevant to emphasize that while the general trend of deploying nodes at higher latitudes is interpretable, the particular solutions themselves remain nontrivial. As an example, Fig. 4.17 displays the improvement in relative up-time for the same distribution of fixed defender's ground segment and attacker's nodes, and two distinct locations for the additional defender's mobile node. In such a case, the solution involving the higher up-time improvement (bottom right) is not the one at higher latitude (top right). Concurrently, as observed during the development of the hand-crafted strategy, there exists a strong variability in up-time increment when choosing different longitudes at higher latitudes, thus corroborating the complexity of the solution space.

4.5.9 Examination of defender response time

For all the experiments presented in this work, we applied a 24-hour delay between the beginning of the attack and the defender's deployment action. This was done primarily to ensure that the simulation has had time to converge to stable "before" and "after" coverage states, allowing

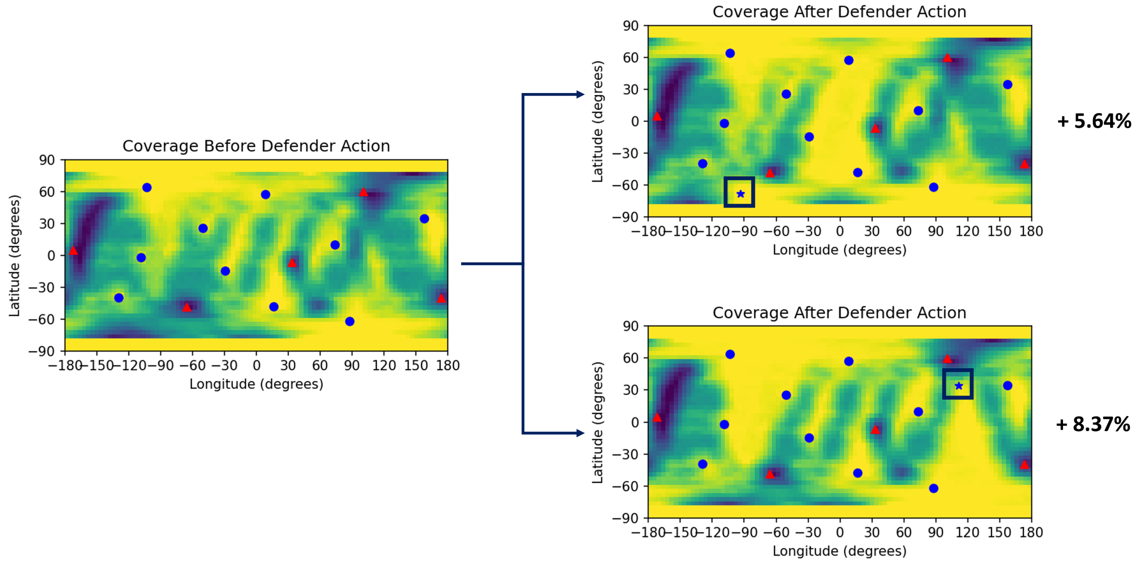


Figure 4.17: Up-time improvement example for two different defender's mobile node coordinates. Attacker and defender's fixed nodes randomly distributed.

us to best characterize the impact of the defender's ground station placement. However, in a hypothetical real-world deployment of such an agent, this large lead time is not necessary. To evaluate the effects of varying the defender's reaction time, we reuse the evolved defender agents from Section 4.5.6 for the KSat10 scenario, and we repeatedly re-run the simulation while varying the response time between 5 minutes and 24 hours. For shortened lead times, it is no longer accurate to measure fitness by comparing the coverage values before and after the defender action, as the attack may not fully propagate until after the defender acts, resulting in artificially low measured improvements. Instead, we compare the coverage that would have occurred after 24 hours with no defender action to the coverage at the end of the simulation. Plotting the resulting fitness values in Figure 4.18 shows that these agents are able to find good ground station locations almost immediately, reaching their maximum performance after approximately 6 hours of observing the attack. With an orbital period of around 1 hour and 45 minutes, this means that only a few orbits are needed to collect enough data for the defender to deploy its mobile ground station. Fewer than that, the global impact of the attack is less evident on the coverage heatmap. Notably, we do not evolve new agents to utilize these shorter response times; defender agents evolved for 24-hour times are robust enough to usually succeed in these substantially different conditions. It is likely that defender strategies specifically evolved to act with shorter response times could better handle such incomplete observation data.

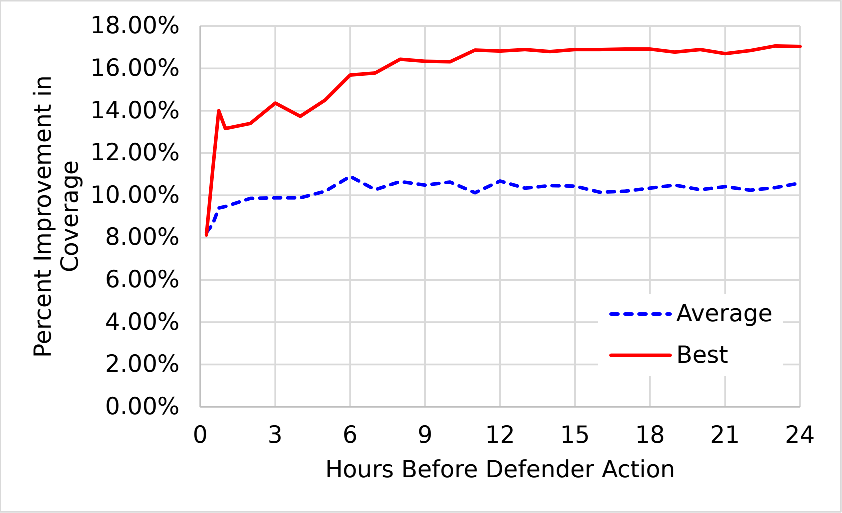


Figure 4.18: Variation in mean performance and the best agent's performance given different lead times before the defender is allowed to place its mobile node

4.6 General Considerations

The experiments presented here demonstrate that the combination of a competitive coevolutionary algorithm with reactive, genetic programming-based defender strategies solves the adversarial ground station transit time game effectively compared to previous work. In particular, the technique of combining spatial heatmaps in the GP tree is a novel approach that provides a convenient representation for this problem, enabling the evolution of strategies that apply high-level information originating from the dynamics of the constellation. The individually-meaningful building blocks provided by this representation facilitate early steps in evolution; simultaneously, they also encourage the development of advanced strategies which combine data in complex ways. We observed the emergence of such complex strategies in practice, which corroborates our assessment of the complexity of this problem and the solutions required for it, emphasizing how successful strategies are not intuitive and demand the proper combination of signals originating from different sources. These experiments show that our GP design is robust to the specifics of the problem, and can produce strong, distinct results under many different scenarios and constraints. These evolved strategies are themselves robust to a wide variety of attacks, which was not generally possible in previous methodologies.

Chapter 5

Conclusions

In this work we discussed the problems of anomaly resilience, detection, and reaction in various astrodynamics problems, with a focus on trajectory design in cislunar regime and P-LEO satellite constellations. In the following, key contributions and findings for each individual problem are summarized, and future avenues of work are discussed.

5.1 Resilience Problem

The resilience problem has been discussed in the context of trajectory design in cislunar space through the exploration of two case studies: descent trajectory abort, which assumed the occurrence of an anomaly along the descent trajectory from the Gateway to a Low Lunar Orbit, and surface-stay abort, which assumed the ascent module had to return to the Gateway in advance with respect to the nominal mission duration. Hence, we characterized the convergence and dynamical structures of the NRHO-to-LLO abort trajectory design space associated with these two scenarios, employing direct and flow-informed approaches.

We first discussed the descent problem from a numerical perspective, leveraging a traditional two-step pipeline at first, and then introducing a three-step optimization routine relying on surrogate modeling. Our analyses conducted via the two-step pipeline revealed the complexity of the convergence structure, which is found to be characterized by sparse regions of catastrophic collapse where the pipeline fails to provide a converged solution. The conducted analysis revealed how the presence of such regions can be directly correlated to a plethora of factors, including the inherent formulation of the correction algorithms, the dynamics of the problem itself, and geometrical properties, overall underscoring the susceptibility of any optimization pipeline to the existence of these "singular" conditions. In the attempt to reduce computational cost and improve the convergence landscape, we then applied a surrogate-based optimization pipeline. The obtained results demonstrated our approach to be promising, suggesting how surrogate modeling may constitute a viable solution to accelerate large-scale simulations. Finally, we gained insight into the dynamical structure of the problem by leveraging a flow-informed method based on FTLE maps. Our analysis revealed how families of trajectories can be identified, with lower FTLE values associated with states rapidly escaping the system, and higher FTLE values corresponding to a motion persisting in the proximity of the Moon (within the considered time horizon). Knowledge of such dynamical features may then be leveraged for maneuver planning, thus resulting informative for the potential identification of cheap trajectory solutions. We then explored the ascent problem through a similar analysis, first employing a two-step optimization pipeline, then applying the surrogate-based method. For the former analysis, we highlighted how again regions of stiff convergence appear throughout the considered design space, shifting and morphing based on the settings of the optimization pipeline. Then, we observed how the surrogate-based pipeline performed significantly worse with respect to the first case study, providing a converged solution solely within a handful of conditions. This prompted an extensive investigation, which resulted in the following conclusions:

- The geometry of the problem strongly influences the shape of the surface to be approximated, here corresponding to the components of the Δv at the departure point on the LLO as a function of the time of flight. In fact, the difference in dynamics between the LLO and the NRHO determines sampling of departure-rendezvous pairs which are highly variable from a geometrical perspective, thus resulting in steep changes in the Δv direction, even for points which are close in the time domain.

- While the time of flight is generally convenient to utilize as a variable to explore the solution space, a representation that more directly relates to the geometry of the problem may result in a smoother surface. In particular, direct mapping from time to angles demonstrates how to improve the regularity of the surface, showing how spatially similar sample points which are distant in the time domain get closer in angular space.

- The most challenging condition is determined by transfers from the LLO to positions in the perilunar region. In such cases, we discovered how convergence can be accomplished through either temporally short transfers, or through longer trajectories (both in space and time) in the form of multi-revolution transfers.

- Specific orientations of the LLO provide, on average, a lower total Δv cost. Hence, if such orientations are selected, multiple low-cost transfers can be available for different trajectory profiles, thus promoting flexibility of operations. In particular, the most convenient conditions are associated with cases when the LLO and the NRHO are $\sim$coplanar, while concurrently having the same direction of motion.

- Particular relative geometries (such as those involving departure from the LLO with a true anomaly close to $\pi/2$) coupled with long time of flight may require infeasible maneuvers from a practical perspective.

5.1.1 Application

While we acknowledge potential limits of our study (see Sec. 5.1.2), it is here relevant to emphasize how our discoveries find immediate applications at different levels of the mission and trajectory design process. To begin with, the investigation on the convergence structure may inform designers in the development of an optimization pipeline, specifically in terms of flexibility of correction schemes and options for initial guess generation. Furthermore, the presented research suggests two paths for accelerating large-scale simulations. The first one (more direct) relies on the introduction of approximate algorithms within a traditional optimization architecture, as they may accurately provide initial guesses in substitution of heavier exact models, thus lowering the overall computational burden. The second one (more indirect) derives from the findings of the investigation, which can be leveraged either by pruning in advance regions of the search space where an optimizer will struggle to converge/will provide an impractical solution, and/or to inform of more manageable and interpretable design space representations.

5.1.2 Future Directions

The investigation conducted on the resilience problem in the context of trajectory design opens several research avenues, either to mitigate the highlighted limitations, to improve upon observed performance, and expand the line of work. In the following, a few potential directions are suggested.

- For what concerns the direct search via the offline three-step pipeline, a relevant aspect is defined by the distribution of the initial set of sample points. In this study, Latin hypercube has been adopted to generate the initial set, resulting in a near-random distribution of points. While this is a valid approach, especially for cases where the input/output mapping is unknown, our analyses demonstrated the complexity of the convergence structure of the problem, which may have solely a few sparse and isolated regions where an optimizer may struggle if poor initial conditions are provided. As such, despite the adoption of adaptive sampling techniques that rely on a combination of exploitation and exploration criteria, an ill-sampled initial set may result in a poorly-initialized surrogate model. Therefore, future work can focus on exploring methodologies to sample the initial set of points smartly.
- The analyses conducted in the second case study highlighted how the complexity of the surface strongly limits the performance of the selected metamodel, opening to either exploration of different surrogate models or to regularize the surface. In this work, steps have been taken toward smoothing of the surface, underscoring how an angular representation which better captures the geometrical features of the problem may be beneficial. Hence, future work may either explore different types of representation, or investigate alternative function approximators. In particular, other Kriging methods, such as Universal Kriging, are available. Furthermore, neural networks may offer additional flexibility, though one should be conscious of their inherently larger number of parameters and the longer training time required with respect to a simpler model.
- To generate initial guesses, a Lambert solver has been utilized throughout our investigation. Though this provides a fast guess generator which often time can lead to a converged

solution in higher fidelity (provided that the pipeline is robust enough and flexible in terms of optimizer settings), it may limit the type of available transfers. Specifically, it can be observed how the obtained converged solutions are strongly influenced by a two-body-like type of transfer, which hardly leverages the dynamics of the environment. As such, a more robust initial guess generator may be investigated directly in CR3BP, potentially leading to alternative and more convenient transfers. Additionally, it is worth emphasizing that we limited our investigation to two-impulse maneuvers, which granted a low-cost pipeline from a computational stand point. Future works may consider the introduction of intermediate maneuvers, which may be beneficial in terms of both Δv cost and convergence structure improvement. Finally, modifications can be introduced in the correction algorithm formulation. Specifically, we kept the formulation simple by imposing constraints exclusively on the terminal position. In this regard, additional constraints may be introduced based on mission requirements (total Δv, impulse direction, flight path angle, etc.), which would introduce an a priori pruning of solutions which are feasible in a simplified model, but that would likely violate feasibility in higher fidelity.

5.2 Detection Problem

The detection problem has been discussed within the context of P-LEO satellite constellations. Thanks to their development, the size of the space infrastructure is rapidly increasing; however, challenges are expected for the successful and effective management of such a large space system. The elevated number of satellites makes the constant monitoring of individual units a complex task for the limited human personnel; additionally, existing algorithms dedicated to the detection of anomalous events typically focus on individual satellites and pay little to no attention to potential cyber-threats. Toward improving management of large space networks, in this work we developed an anomaly detection framework to effectively and automatically identify irregular behaviors on P-LEO systems.

To begin with, we modeled a satellite constellation as a dynamic graph, with nodes representing satellites and ground points, and edges representing connections among them. Next, we employed the TADDY deep learning architecture for the classification of edges belonging to graphs generated from multiple shortest paths connecting two arbitrary ground nodes that abstractly represent two regions of the world. To investigate the performance of the framework, we initially considered different time horizons (throughout the analyses referred to as aggregation windows) in the range of minutes to tens of minutes to assess how the extension of the observation window influences the capability of the model to capture the temporal behavior of the network. Our initial analysis revealed how direct application of the original methodology struggles to reach satisfactory performances, reaching AUC scores lower than 0.7 for all considered cases. In particular, we identified graph structure (connectivity), constellation dynamics, and anomaly injection technique to be the key parameters affecting the classification task. Furthermore, we observed how the traditional data pre-processing technique employed in the original work was detrimental for the constellation study, thus requiring a different approach. After introducing a variation in data processing, we presented an extensive investigation which emphasized how topological and temporal information alone are insufficient to capture the nature of the constellation graph, thus needing to be enriched to enhance the algorithm's performance. Notably, we discovered the transformative impact of injecting frequency information on the baseline TADDY framework performance. Finally, we conducted a large variety of tests over modified constellation configurations, varied sets of terminal nodes and different negative sampling connection strategies which showcased the ability of the network to potentially learn underlying graph structures, being capable of maintaining good performance even over unseen scenarios.

5.2.1 Application

Despite that the implemented framework has to be considered as only a first step toward the development of a comprehensive tool for monitoring a space network, the lessons learned throughout our investigation may be immediately applied by satellite operators. In particular, the identified coupling between satellite dynamics and spatial ground node distribution (and the corresponding algorithm performance) can be directly leveraged by human developers to inject expert knowledge in the design of future detection algorithms, and by current operators to evaluate situations requiring attention.

5.2.2 Future Directions

While the obtained results are encouraging, it is also relevant to highlight potential limitations of the method. In particular, thanks to the Monte Carlo analysis, we emphasized how the physical nature of the system strongly couples with the algorithm parameters, possibly negatively impacting the performance. Furthermore, the method is based on the main assumption that the constellation architecture is static, that is, the number of nodes in the constellation (particularly satellite nodes) remains constant. As such, any dynamic evolution of the constellation, such as decommissioned/replaced satellites, is disregarded. To mitigate these shortcomings and bring further improvements in model performance, multiple opportunities exist.

- Firstly, immediate steps can be taken to increase the model fidelity, which may provide sufficiently rich information to capture the complexity of the problem, boosting the algorithm's performance.
- Secondly, the significant improvement introduced by injecting the frequency attribute may indicate the potential to incorporate other sources of information to boost the detection capability. Such additional features may be related to attributes of the physical constellation, as well as to properties of the graph itself.
- Thirdly, the Monte Carlo analysis revealed how the configuration and dynamics of the constellation coupled with the relative distribution of the ground points constitute a key factor in determining the performance of the model. Hence, future work may look at ways to specialize the algorithm parameters toward improving performance for specific application cases.
- Next, the model can be extended to handle non-static constellations by introducing a methodology to dynamically handle constellation changes.
- Finally, it is worth remarking that the developed framework considers a centralized view, that is, it assumes that global information on the network is available. However, in reality such a perspective may not be immediately feasible, thus delaying reactions should an

anomaly occur. As such, the framework could be repurposed to conduct the anomaly detection task directly on the edge, that is, to operate directly at node/satellite level, which may be a more beneficial perspective to promptly react to irregular events.

5.3 Reaction Problem

The reaction problem has also been discussed in the context of satellite P-LEO constellations, with a higher focus on the security of such systems. In fact, as P-LEOs provide support to critical infrastructures, they also constitute a broader surface of attack for malevolent actors, who may see individual satellites as a potential point of access for malicious intents. As a proxy to study large-scale attacks against P-LEO satellite networks, we first introduced what we defined as an adversarial ground station transit time game, which features an attacker and a defender competing against each other to disrupt or maintain the constellation service, in this work quantified through the satellite up-time. Next, we employed competitive coevolution and genetic programming as solution methods to the problem. Specifically, we granted the adversaries the capability to deploy ground nodes, with the defender initially capable of deploying a single node in addition to an existing ground segment, and we employed these methods to evolve locations for the attacker where to deploy rogue nodes from where to launch its attack, while reactive strategies encoded through genetic programming trees were evolved for the defender. To evaluate the validity of the method, we tested our approach over a wide range of scenarios with incremental complexity. Our initial evaluation considered an unconstrained environment, where both players were allowed to deploy their nodes in any region of the world. Within this formulation, we tested the approach against minimally complex strategies, smart strategies expressed as static solutions, and a human-crafted policy. Results obtained over these scenarios demonstrated the superiority of our novel approach, and emphasized the complexity of the solution space, which caused the hand-crafted strategy to rapidly decrease in effectiveness as the complexity of the scenarios increased. Next, we refined the model via the introduction of geographic constraints, bounding the deployment of nodes to be land-based-only. Furthermore, we considered scenarios which included partial information and the possibility of deploying multiple nodes for the defender. Again, the novel approach demonstrated to be superior against alternative methods, with evolved reactive policies showcasing robustness against a variety of unseen opponents. Finally, we also analyzed the effect of the response time on up-time improvement, observing how defender policies evolved through previously discussed scenarios (that is, over a 24-hour horizon) demonstrate to be robust even at different time scales.

5.3.1 Application

While the presented research aims to develop reactive strategies against adversarial actions, the proposed methodology can more immediately be leveraged from a resilience perspective. In particular, our approach can be utilized for the design of the ground segment, as it enables identification of a ground node distribution which is resilient to adversarial attacks. In fact, though we acknowledge this approach to necessitate a trade-off in the design as it implies to sacrifice optimality for nominal operations, the resulting ground node distribution may allow to maintain sub-optimal behavior even in the case of unforeseen, adversarial events.

5.3.2 Future Directions

The results of this work suggest that our methods can be extended to a wider variety of scenarios, as well as to higher-fidelity environments and agents. In particular, our attacker aims to degrade the constellation service on a global scale; we observed how this typically results in placing their nodes at high-latitudes, as these locations grant frequent contacts with satellites. Many real-world actors might also desire to disrupt service over populated areas, or over a specific region. Such localized objectives might dramatically change the observed behaviors. The introduction of a cost objective would further impact the dynamics of the game, forcing agents to consider a trade-off between cost and effectiveness when selecting locations. A cost objective has serious game-theoretic consequences over the outcome of a scenario; for example, an agent which can not achieve dominance over an opponent's strategy could still make that strategy prohibitively expensive to execute.

Strategies can also be improved by refining our GP tree formulation. In particular, we observed how the geographic nature of the selected primitives (Table 4.2) allowed the defender to identify meaningful robust strategies even when the complexity of the environment increases.

However, as the quality of the solution derives in part from the selected primitives, one may be interested in exploring different primitive choices. For example, evolution of a reactive strategy may benefit from providing primitives that inform the agent about temporal dynamics of the system. In such a way, the defender agent may be able make inference about potential locations of the attacker, thus selecting more strategic positions for its mobile nodes' placement. Nonetheless, it must be noted that, in general, adding more primitives may render evolutionary search inefficient. To prevent this, it could be helpful to investigate the primitive set statistically to identify functions which evolution is failing to effectively exploit. Additionally, it may be worth exploring more flexible representations for agent strategies like neural networks, either as a component to the tree or as complete replacement.

A complementary factor to consider when evaluating complex strategies is their interpretability. Through our results, we showed how KDE was helpful to interpret the aggregate behavior observed in our experiments. However, while the visualization of placed ground stations between two agents allows one to view differences in behavior, it does not explain the cause of those differences. While GP constructs trees out of primitive functions with known behavior, the resulting strategies combine these functions in ways that are difficult to understand despite being effective. As our GP tree formulation primarily relies on heatmaps, one possibility may include viewing the intermediate stages of these heatmaps during tree execution, in case certain subtrees have evolved with a meaningful granular function. This can be then combined with techniques like frequent subtree mining of evolved trees, which would allow one to extrapolate which factors appear to be determinants for the development of a successful strategy.

Finally, any speedup in evaluation would allow for either faster evolution of strategies, or enable the use of larger populations and longer evolution to produce higher-quality strategies. One promising idea is to modify the fidelity of the simulation over time, since the resolution has an extreme impact on runtime. As the reactive defender's strategy is agnostic to the spatial-temporal granularity of the simulation, a strategy could be rapidly evolved at a low resolution for most of an evolutionary run, only increasing towards the end to ensure that the evolved strategies are effective in a more realistic environment. Additionally, the attacker and defender could be evolved independently of each other at the start of evolution, allowing them to first learn how to respond to the orbital dynamics of the simulation at a much faster rate without the high cost of coevolution, before later facing intelligent opponents. Improvements in the methods used by coevolution to select opponent pairs could also greatly reduce the number of evaluations needed overall, with similar benefits to increasing evaluation speed [121].

References

[1] B. S. Kempton, “A Simulation Tool to Study Routing in Large Broadband Satellite Networks,” Master’s thesis, Christopher Newport University, 2020.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” *NIPS’17: Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach (CA, USA)*, December 2017.

[3] Y. Liu, S. Pan, Y. G. Wang, F. Xiong, L. Wang, C. Qingfeng, and V. C. Lee, “Anomaly Detection in Dynamic Graphs via Transformer,” *IEEE Transactions on Knowledge and Data Engineering*, November 2021. DOI: 10.1109/TKDE.2021.3124061.

[4] M. Indaco, E. Taheri, and D. Guzzetti, “Structure of the NRHO-to-LLO Abort Trajectory Design Space,” *AIAA Scitech 2024 Forum, Orlando (FL, USA)*, January 2024. DOI: https://doi.org/10.2514/6.2024-1452.

[5] M. Indaco and D. Guzzetti, “Transformer-based anomaly detection in P-LEO constellations: A dynamic graph approach,” *Acta Astronautica*, vol. 218, pp. 177–194, 2024. DOI: 10.1016/j.actaastro.2024.02.019.

[6] M. Indaco, S. N. Harris, D. Seals, S. Mulder, D. R. Tauritz, and D. Guzzetti, “Coevolving Defender Strategies Within Adversarial Ground Station Transit Time Games via Competitive Coevolution,” *The Journal of the Astronautical Sciences*, vol. 70, pp. 1–32, 2023. DOI: 10.1007/s40295-023-00411-w.

[7] "The Global Exploration Roadmap (Lunar Surface Exploration supplement update)," tech. rep., International Space Exploration Coordination Group (ISECG), October 2022.

[8] Z. D. May, M. Qu, and R. Merrill, "Enabling Global Lunar Access for Human Landing Systems Staged at Earth-Moon L2 Southern Near Rectilinear Halo and Butterfly Orbits," *AIAA Scitech 2020 Forum, Orlando (FL, USA)*, January 2020.

[9] R. Whitley and R. Martinez, "Options for Staging Orbits in Cis-Lunar Space," *IEEE Annual Aerospace Conference, Big Sky (MT, USA)*, March 2016.

[10] D. C. Davis, S. Bhatt, K. C. Howell, J.-W. Jang, R. J. Whitley, F. Clark, D. Guzzetti, E. M. Zimovan, and G. Barton, "Orbit Maintenance and Navigation of Human Spacecraft at Cislunar Near Rectilinear Halo Orbits," *AAS/AIAA Space Flight Mechanics Meeting, San Antonio (TX, USA)*, February 2017.

[11] S. Trofimov, M. Shirobokov, A. Tselousova, and M. Ovchinnikov, "Transfers from near-rectilinear halo orbits to low-perilune orbits and the Moon's surface," *Acta Astronautica*, vol. 167, pp. 260–271, 2020. DOI: `https://doi.org/10.1016/j.actaastro.2019.10.049`.

[12] "Human Landing System Concept of Operations," tech. rep., NASA, 2019.

[13] G. L. Condon, C. C. Esty, C. F. Berry, S. P. Downs, C. Ocampo, B. Mahajan, and L. M. Burke, "Mission and Trajectory Design Considerations for a Human Lunar Mission Originating from a Near Rectilinear Halo Orbit," *AIAA Scitech 2020 Forum, Orlando (FL, USA)*, January 2020.

[14] C. R. Short, K. C. Howell, and X. M. Tricoche, "Lagrangian coherent structures in the restricted three-body problem," *AAS/AIAA Astrodynamics Specialist Conference, Girdwood (AK, USA)*, August 2011.

[15] C. R. Short, D. Blazveski, K. C. Howell, and G. Haller, “Stretching in phase space and applications in general nonautonomous multi-body problems,” *Celestial Mechanics and Dynamical Astronomy*, vol. 122, pp. 213–238, July 2015. DOI: 10.1007/s10569-015-9617-4.

[16] E. S. Gawlik, J. E. Marsden, P. C. D. Toit, and S. Campagnola, “Lagrangian coherent structures in the planar elliptic restricted three-body problem,” *Celestial Mechanics and Dynamical Astronomy*, vol. 103, pp. 227–259, March 2009. DOI: 10.1007/s10569-008-9180-3.

[17] L. Bucci, A. Kleinschneider, M. Lavagna, and F. Renk, “Optimal Escape Manifold for Cis-lunar Halo Orbits,” *69th International Astronautical Congress, Bremen (Germany)*, October 2018.

[18] A. Fossà, G. Bucchioni, E. Blazquez, E. Canalias, S. Lizy-Destrez, R. Bertrand, A. Lamy, and J.-F. Goester, “Two and three impulses phasing strategy with a spacecraft orbiting on an Earth–Moon NRHO,” *Acta Astronautica*, vol. 198, pp. 669–679, 2022. DOI: 10.1016/j.actaastro.2022.06.042.

[19] E. M. Zimovan, “Characteristics and Design Strategies for Near Rectilinear Halo Orbits within the Earth-Moon System,” Master’s thesis, Purdue University, 2017.

[20] T. A. Pavlak, “Mission Design Applications in the Earth-Moon System: Transfer Trajectories and Stationkeeping,” Master’s thesis, Purdue University, 2010.

[21] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*. Hawthorne (CA, USA): Microcosm Press, 4 ed., 2013.

[22] R. J. Whitley, D. C. Davis, L. M. Burke, B. P. McCarthy, R. J. Power, M. L. McGuire, and K. C. Howell, “Earth-Moon near Rectilinear Halo and Butterfly Orbits for Lunar Surface Exploration,” *2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird (UT, USA)*, August 2018.

[23] C. Howard D., *Orbital Mechanics for Engineering Students*. Boston (MA, USA): Butterworth-Heinemann, 3 ed., 2014.

[24] C. Thangavelu, "Transfers between Near Rectilinear Halo Orbits and Low Lunar Orbits," Master's thesis, University of Colorado, 2018.

[25] S. M. Clarke, J. H. Griebsh, and T. W. Simpson, "Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses," *Journal of Mechanical Design*, vol. 127, pp. 1077–1087, November 2005. `DOI: 10.1115/1.1897403`.

[26] J. Park and I. Sandberg, "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation*, vol. 3, pp. 246–257, June 1991. `DOI: 10.1162/neco.1991.3.2.246`.

[27] J. P. C. Kleijnen, "Regression and Kriging metamodels with their experimental designs in simulation: A review," *European Journal of Operational Research*, vol. 256, pp. 1–16, January 2017. `DOI: 10.1016/j.ejor.2016.06.041`.

[28] D. G. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand," *Journal of The South African Institute of Mining and Metallurgy*, vol. 52, pp. 201–203, 1951.

[29] J. P. C. Kleijnen, "Kriging metamodeling in simulation: A review," *European Journal of Operational Research*, vol. 192, pp. 707–716, February 2009. `DOI: 10.1016/j.ejor.2007.10.013`.

[30] H. Wackernagel, *Ordinary Kriging*, pp. 74–81. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. `DOI: 10.1007/978-3-662-03098-1_11`.

[31] J. N. Fuhg, "Adaptive surrogate models for parametric studies," Master's thesis, Leibniz University of Hannover, 2019.

[32] F. Bachoc, "Cross Validation and Maximum Likelihood estimations of hyper-parameters of Gaussian processes with model misspecification," *Computational Statistics & Data*

Analysis, vol. 66, pp. 55–69, October 2013. DOI: 10.1016/j.csda.2013.03.016.

[33] J. N. Fuhg, A. Fau, and U. Nackenhorst, "State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging," *Archives of Computational Methods in Engineering*, vol. 28, pp. 2689–2747, June 2021. DOI: 10.1007/s11831-020-09474-6.

[34] J. L. Loeppky, J. Sacks, and W. J. Welch, "Choosing the Sample Size of a Computer Experiment: A Practical Guide," *Technometrics*, vol. 51, pp. 366–376, November 2009. DOI: 10.1198/TECH.2009.08040.

[35] D. Canales and K. C. Howell, "Leveraging Finite-Time Lyapunov Exponent Maps to Design Tours Incorporating Three Moons," *ASCEND 2021 (virtual), Las Vegas (NV, USA)*, November 2021.

[36] D. Guzzetti and H. Baoyin, "Time-Varying Shadows of Quasi-Periodic Motion Across Sections of the Flow Within Nearly Time-Periodic Three-Body Dynamics," *The Journal of the Astronautical Sciences*, vol. 68, pp. 855–890, September 2021. DOI: 10.1007/s40295-021-00284-x.

[37] C. R. Short and K. C. Howell, "Lagrangian coherent structures in various map representations for application to multi-body gravitational regimes," *Acta Astronautica*, vol. 94, pp. 592–607, February 2014. DOI: 10.1016/j.actaastro.2013.08.020.

[38] Z. Zhangming, L. Haiyang, and W. Xugang, "An adaptive sampling method for Kriging surrogate model with multiple outputs," *Engineering with Computers*, vol. 38, pp. 277–295, April 2022. DOI: 10.1007/s00366-020-01145-1.

[39] C. R. Short, "Flow-informed Strategies for Trajectory Design and Analysis," Master's thesis, Purdue University, 2016.

[40] "Competing in Space," tech. rep., NASIC Public Affairs Office, December 2018.

[41] N. Boschetti, N. G. Gordon, and G. Falco, “Space Cybersecurity Lessons Learned from The ViaSat Cyberattack,” *Ascend 2022, Las Vegas (NV, USA)*, October 2022.

[42] “Space Threat Assessment 2023,” tech. rep., Center for Strategic & International Studies, April 2023.

[43] P. T. J. Kon, D. Barradas, and A. Chen, “Stargaze: A LEO Constellation Emulator for Security Experimentation,” *CCS ’22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles (CA, USA)*, November 2022.

[44] M. Jia, Y. Shu, Q. Guo, Z. Gao, and S. Xie, “DDoS Attack Detection Method for Space-Based Network Based on SDN Architecture,” *ZTE Communications*, vol. 18, pp. 18–25, December 2020. DOI: 10.12142/ZTECOM.202004004.

[45] G. Giuliari, T. Ciussani, A. Perrig, and A. Singla, “ICARUS: Attacking low Earth orbit satellite networks,” *USENIX Annual Technical Conference (virtual)*, July 2021.

[46] Y. Gao, T. Yang, M. Xu, and N. Xing, “An Unsupervised Anomaly Detection Approach for Spacecraft Based on Normal Behavior Clustering,” *2012 Fifth International Conference on Intelligent Computation Technology and Automation, Zhangjiajie, Hunan (China)*, January 2012.

[47] H. Jiang, K. Zhang, J. Wang, X. Wang, and P. Huang, “Anomaly Detection and Identification in Satellite Telemetry Data Based on Pseudo-Period,” *Applied Sciences*, vol. 10, pp. 103–108, December 2020. DOI: 10.3390/app10010103.

[48] K. Li, Y. Wu, S. Song, Y. Sun, J. Wang, and Y. Li, “A novel method for spacecraft electrical fault detection based on FCM clustering and WPSVM classification with PCA feature extraction,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 231, pp. 98–108, January 2017. DOI: 10.1177/0954410016638874.

[49] Y. Wang, J. Gong, J. Zhang, X. Han, and P. Castaldi, “A Deep Learning Anomaly Detection Framework for Satellite Telemetry with Fake Anomalies,” *International Journal*

of Aerospace Engineering, vol. 2022, pp. 1–9, January 2022. DOI: `10.1155/2022/1676933`.

[50] M. A. M. Sadr, Y. Zhu, and P. Hu, “An Anomaly Detection Method for Satellites Using Monte Carlo Dropout,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, pp. 2044 – 2052, April 2023. DOI: `10.1109/TAES.2022.3206257`.

[51] G. Xiang and R. Lin, “Robust Anomaly Detection for Multivariate Data of Spacecraft Through Recurrent Neural Networks and Extreme Value Theory,” *IEEE Access*, vol. 9, pp. 167447–167457, December 2021. DOI: `10.1109/ACCESS.2021.3136505`.

[52] Z. Zhang, P. Cui, and W. Zhu, “Deep Learning on Graphs: A Survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, pp. 249–270, January 2022. DOI: `10.1109/TKDE.2020.2981333`.

[53] K. Sricharan and K. Das, “Localizing anomalous changes in time-evolving graphs,” *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, Snowbird (UT, USA)*, June 2014.

[54] C. C. Aggrawal, Y. Zhao, and P. S. Yu, “Outlier detection in graph streams,” *IEEE 27th International Conference on Data Engineering, Hannover (Germany)*, April 2011.

[55] E. Manzoor, S. M. Milajerdi, and L. Akoglu, “Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs,” *KDD ’16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco (CA, USA)*, August 2016.

[56] Z. Yuan, M. Shao, and Q. Yan, “Motif-Level Anomaly Detection in Dynamic Graphs,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2870–2882, May 2023. DOI: `10.1109/TIFS.2023.3272731`.

[57] W. Yu, W. Cheng, C. Aggarwal, K. Zhang, H. Chen, and W. Wang, “NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks,” *KDD ’18:*

The 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London (UK), August 2018.

[58] C. Yang, L. Zhou, H. Wen, Z. Zhou, and Y. Wu, "H-VGRAE: A Hierarchical Stochastic Spatial-Temporal Embedding Method for Robust Anomaly Detection in Dynamic Networks," 2020.

[59] Y. Zhang, Y. Wang, Y. Hu, Z. Lin, Y. Zhai, L. Wang, Q. Zhao, K. Wen, and L. Kang, "Security Performance Analysis of LEO Satellite Constellation Networks under DDoS Attack," *Sensors*, vol. 22, pp. 1–10, September 2022. DOI: 10.3390/s22197286.

[60] W. Guo, J. Xu, Y. Pei, L. Yin, C. Jiang, and N. Ge, "A Distributed Collaborative Entrance Defense Framework Against DDoS Attacks on Satellite Internet," *IEEE Internet of Things Journal*, vol. 9, pp. 15497–15510, September 2022. DOI: 10.1109/JIOT.2022.3176121.

[61] H.-Y. Kwon, T. Kim, and M.-K. Lee, "Advanced Intrusion Detection Combining Signature-Based and Behavior-Based Detection Methods," *Electronics*, vol. 11, pp. 867–886, March 2022. DOI: 10.3390/electronics11060867.

[62] F. T. Liu, K. M. Ting, and Z.-H. Zhouu, "Isolation Forest," *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa (Italy)*, December 2008.

[63] I. Kiss, P. Haller, and A. Beres, "Denial of Service Attack Detection in Case of Tennessee Eastman Challenge Process," *Procedia Technology*, vol. 19, pp. 835–841, December 2015. DOI: 10.1016/j.protcy.2015.02.120.

[64] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of Deep Learning in Detecting Unknown Network Attacks," *2019 International Conference on Smart Applications, Communications and Networking (SmartNets), Sharm El Sheikh (Egypt)*, December 2019.

[65] B. Gogoi and T. Ahmed, “HTTP Low and Slow DoS Attack Detection using LSTM based deep learning,” *2022 IEEE 19th India Council International Conference (INDICON), Kochi (India)*, November 2022.

[66] D. Gong, M. Tran, S. Shinde, H. Jin, V. Sekar, P. Saxena, and M. S. Kang, “Practical Verifiable In-network Filtering for DDoS Defense,” *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas (TX, USA)*, July 2019.

[67] Z. Ming, L. Liu, S. Zhou, and Z. Tian, “Survey on security issues of routing and anomaly detection for space information networks,” *Scientific Reports*, vol. 11, pp. 1–18, December 2021. `DOI: 10.1038/s41598-021-01638-z`.

[68] R. Boumghar, R. N. N. Madeira, A. Donati, I. Angelis, J. F. M. D. Silva, J. A. M. Heras, and J. Schulster, *Enhanced Awareness in Space Operations Using Web-Based Interactive Multipurpose Dynamic Network Analysis*, pp. 795–810. Cham: Springer International Publishing, 2019. `DOI: 10.1007/978-3-030-11536-4_31`.

[69] Y. Hu, I. Sharf, and L. Chen, “Distributed orbit determination and observability analysis for satellite constellations with angles-only measurements,” *Automatica*, vol. 129, pp. 1–11, July 2021. `DOI: 10.1016/j.automatica.2021.109626`.

[70] H. Zhang and P. Gurfil, “Cooperative Orbital Control of Multiple Satellites via Consensus,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, pp. 2171–2188, October 2018. `DOI: 10.1109/TAES.2018.2808118`.

[71] C. N. McGrath, R. A. Clark, and M. Macdonald, “Novel concept of satellite manoeuvre planning using graph theoretical techniques,” *Advances in Space Research*, vol. 67, pp. 3775–3784, June 2021. `DOI: 10.1016/j.asr.2020.06.008`.

[72] F. Tang, “Dynamically Adaptive Cooperation Transmission among Satellite-Ground Integrated Networks,” *2020 - IEEE Conference on Computer Communications (virtual)*, July 2020.

[73] Y. Zhang, Q. Wu, Z. Lai, and H. Li, "Enabling Low-latency-capable Satellite-Ground Topology for Emerging LEO Satellite Networks," *2022 - IEEE Conference on Computer Communications, London (UK)*, May 2022.

[74] X. Xu, Z. Gao, and A. Liu, "Robustness of satellite constellation networks," *Computer Communications*, vol. 210, pp. 130–137, October 2023. DOI: 10.1016/j.comcom.2023.07.036.

[75] F. Jiang, Q. Zhang, Z. Yang, and P. Yuan, "A Space–Time Graph Based Multipath Routing in Disruption-Tolerant Earth-Observing Satellite Networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, pp. 2592–2603, October 2019. DOI: 10.1109/TAES.2019.2938447.

[76] M. X, W. Jia, S. Xue, J. Yang, C. Zhou, Q. Sheng, H. Xiong, and L. Akoglu, "A Comprehensive Survey on Graph Anomaly Detection with Deep Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, pp. 1–32, October 2021. DOI: 10.1109/TKDE.2021.3118815.

[77] L. Akoglu, M. McGlohon, and C. Faloutsos, "oddball: Spotting Anomalies in Weighted Graphs," *Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hyderabad (India)*, June 2010.

[78] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "SCAN: a structural clustering algorithm for networks," *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, (CA, USA)*, p. 824–833, August 2007.

[79] S. Bandyopadhyay, L. N, S. V. Vivek, and M. N. Murty, "Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding," *Proceedings of the 13th International Conference on Web Search and Data Mining, huston (TX, USA)*, pp. 25—33, January 2020.

[80] H. Wang, C. Zhou, J. Wu, W. Dang, X. Zhu, and J. Wang, “Deep Structure Learning for Fraud Detection,” *2018 IEEE International Conference on Data Mining (ICDM), Singapore (Singapore)*, November 2018.

[81] H. M. Al-Ammal, “A Review of Machine Learning Techniques for Anomaly Detection in Static Graphs:,” in *Advances in Computational Intelligence and Robotics* (Y. A. Albastaki and W. Awad, eds.), pp. 146–162, IGI Global, 2020.

[82] N. Ailon, R. Jaiswal, and C. Monteleoni, “Streaming k-means approximation,” *Proceedings of the 22nd International Conference on Neural Information Processing Systems, Vancouver (Canada)*, pp. 10—-18, December 2009.

[83] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, “AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN,” *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao (China)*, August 2019.

[84] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, “Fast and Accurate Anomaly Detection in Dynamic Graphs with a Two-Pronged Approach,” *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage (AK, USA)*, August 2019.

[85] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen, “Structural Temporal Graph Neural Networks for Anomaly Detection in Dynamic Graphs,” *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (virtual)*, October 2021.

[86] J. McDonald, “Lecture Notes: Graph Theory,” *Auburn University, Auburn (AL, USA)*, 2018.

[87] S. Haykin, *Neural Networks and Learning Machines*. Upper Saddle River, New Jersey 07458: Pearson, Third Edition, 2009.

[88] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, pp. 303–314, December 1989.

[89] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, eds.), (Online), pp. 7871–7880, Association for Computational Linguistics, July 2020. `DOI: 10.18653/v1/2020.acl-main.703`.

[90] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (J. Burstein, C. Doran, and T. Solorio, eds.), (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019. `DOI: 10.18653/v1/N19-1423`.

[91] A. Kolesnikov, A. Dosovitskiy, D. Weissenborn, G. Heigold, J. Uszkoreit, L. Beyer, M. Minderer, M. Dehghani, N. Houlsby, S. Gelly, T. Unterthiner, and X. Zhai, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” *International Conference on Learning Representations (virtual)*, 2021.

[92] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” *2021 IEEE/CVF International Conference on Computer Vision (ICCV), (virtual)*, March 2021.

[93] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting,” *Proceedings of The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, (virtual)*, vol. 35, no. 12, pp. 11106–11115, 2021.

[94] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas (NV, USA)*, June 2016.

[95] M. Indaco and D. Guzzetti, "Transformer-based Anomaly Detection on Dynamic Graphs: Application to Satellite Constellations," *33rd AAS/AIAA Space Flight Mechanics Meeting, Austin (TX, USA)*, January 2023.

[96] Q. Chen, L. Yang, D. Guo, B. Ren, J. Guo, and X. Chen, "LEO Satellite Networks: When Do All Shortest Distance Paths Belong to Minimum Hop Path Set?," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, pp. 3730–3734, August 2022. DOI: 10.1109/TAES.2022.3143090.

[97] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[98] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then Propagate: Graph Neural Networks meet Personalized PageRank," *International Conference on Learning Representations, New Orleans (LA, USA)*, May 2019.

[99] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal (Canada)*, December 2018.

[100] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," *ICML'20: Proceedings of the 37th International Conference on Machine Learning (virtual)*, July 2020.

[101] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 2378–2392, June 2022. DOI: 10.1109/TNNLS.2021.3068344.

[102] P. H. Le-Khac, G. Healey, and A. F. Smeaton, "Contrastive Representation Learning: A Framework and Review," *IEEE Access*, vol. 8, pp. 193907–193934, October 2020. DOI: 10.1109/ACCESS.2020.3031549.

[103] J. Willbold, M. Schloegel, M. Vögele, M. Gerhardt, T. Holz, and A. Abbasi, "Space Odyssey: An Experimental Software Security Analysis of Satellites," *Computer Science - 2023 IEEE Symposium on Security and Privacy (SP), San Francisco (CA, USA)*, May 2023.

[104] B. Bailey, "Cybersecurity Protections for Spacecraft: A Threat Based Approach," Tech. Rep. TOR-2021-01333-REV A, Aerospace Corporation, April 2021.

[105] I. del Portillo, B. Cameron, and E. Crawley, "Ground Segment Architectures for Large LEO Constellations with Feeder Links in EHF-bands," *2018 IEEE Aerospace Conference*, pp. 1–14, March 2018. DOI: 10.1109/AERO.2018.8396576.

[106] C. N. Efrem and A. D. Panagopoulos, "Globally Optimal Selection of Ground Stations in Satellite Systems with Site Diversity," *IEEE Wireless Communications Letters*, vol. 9, pp. 1101–1104, July 2020. DOI: 10.1109/LWC.2020.2982139.

[107] G. Petelin, M. Antoniou, and G. Papa, "Multi-objective approaches to ground station scheduling for optimization of communication with satellites," *Optimization and Engineering*, pp. 1–38, 2021. DOI: 10.1007/s11081-021-09617-z.

[108] A. Globus, J. Crawford, J. D. Lohn, and A. Pryor, "Scheduling Earth Observing Satellites with Evolutionary Algorithms," *International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, July 2003.

[109] M. Indaco, S. N. Harris, D. Seals, D. R. Tauritz, and D. Guzzetti, "Toward Co-evolving Solutions of Adversarial Ground Station Transit Time Games for P-LEO Constellation Management," *2021 AAS/AIAA Astrodynamics Specialist Conference (virtual)*, September 2021.

[110] M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "FlipIt: The game of "stealthy takeover," *Journal of Cryptology*, vol. 26, pp. 655–713, Oct. 2013. DOI: 10.1007/s00145-012-9134-5.

[111] W. D. Hillis, “Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure,” *Phys. D*, vol. 42, pp. 228—234, June 1990. DOI: 10.1016/0167-2789(90)90076-2.

[112] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, December 1992.

[113] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, “Hyper-heuristics: A survey of the state of the art,” *Journal of the Operational Research Society*, vol. 64, pp. 1695–1724, 2013. DOI: 10.1057/jors.2013.71.

[114] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer, 2015. DOI: 10.1007/978-3-662-44874-8.

[115] K. Krawiec and M. Heywood, “Solving Complex Problems with Coevolutionary Algorithms,” *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pp. 832—858, July 2020. DOI: 10.1145/3377929.3389874.

[116] T. Miconi, “Why Coevolution Doesn’t “Work”: Superiority and Progress in Coevolution,” *Proceedings of the 12th European Conference on Genetic Programming*, pp. 49—60, Apr. 2009. DOI: 10.1007/978-3-642-01181-8_5.

[117] R. Poli, W. B. Langdon, and N. F. McPhee, *A field guide to genetic programming*. Lulu Enterprises, UK Ltd, 2008.

[118] D. J. Montana, “Strongly Typed Genetic Programming,” *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, 1995. DOI: 10.1162/evco.1995.3.2.199.

[119] J. R. Wertz., ed., *Orbit and Constellation Design and Management*. El Segundo, USA: Springer Dordrecht, 2002.

[120] D. Schweim, D. Wittenberg, and F. Rothlauf, “On sampling error in genetic programming,” *Natural Computing*, vol. 21, pp. 1–14, June 2022. DOI: 10.1007/s11047-020-09828-w.

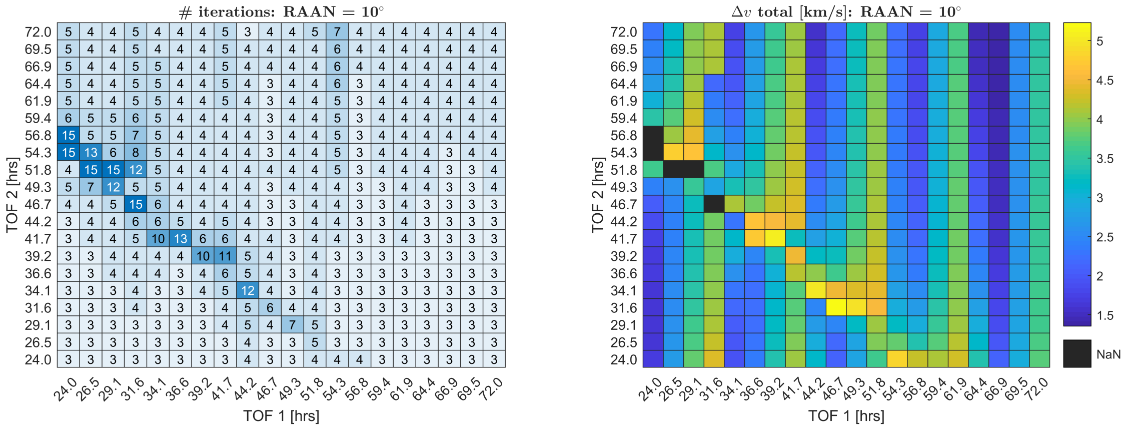
[121] S. N. Harris and D. R. Tauritz, “Competitive Coevolution for Defense and Security: Elo-Based Similar-Strength Opponent Sampling,” *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1898—1906, July 2021. DOI: 10.1145/3449726.3463193.

Appendices

Appendix A

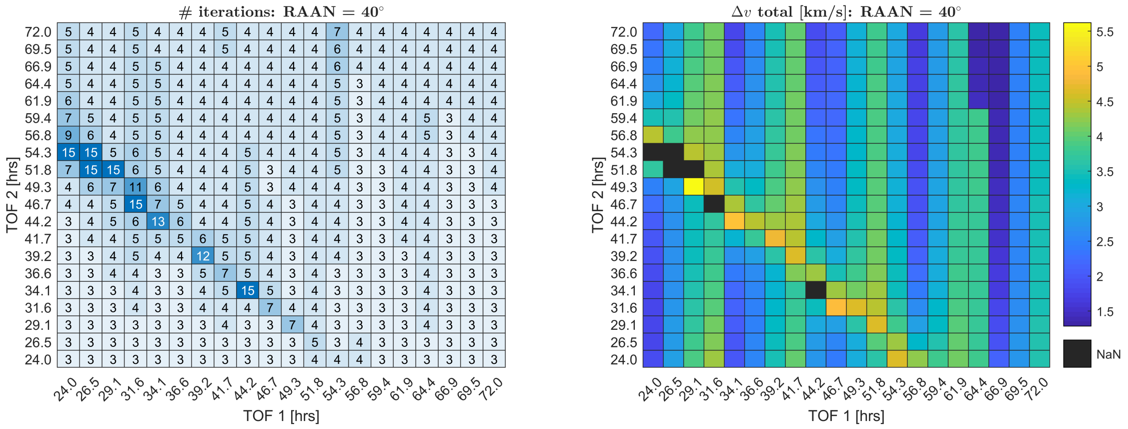
Resilience

A.1 RAAN Scan Supplementary Results



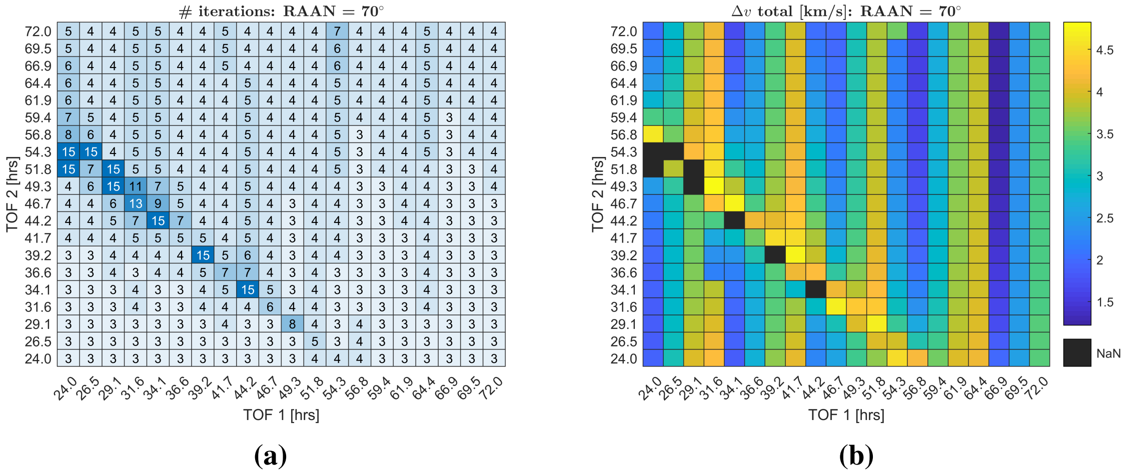
(a) **(b)**

Figure A.1: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 10°.



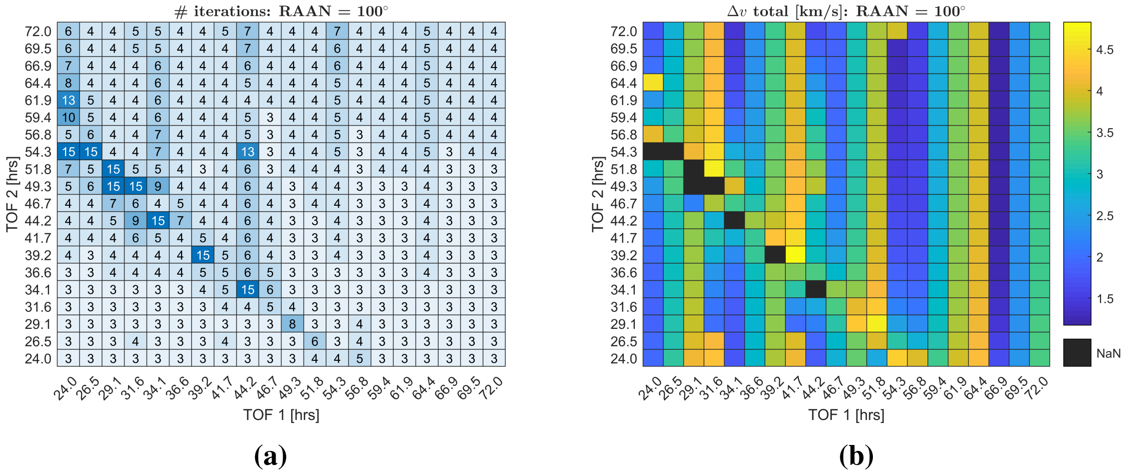
(a) **(b)**

Figure A.2: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 40°.



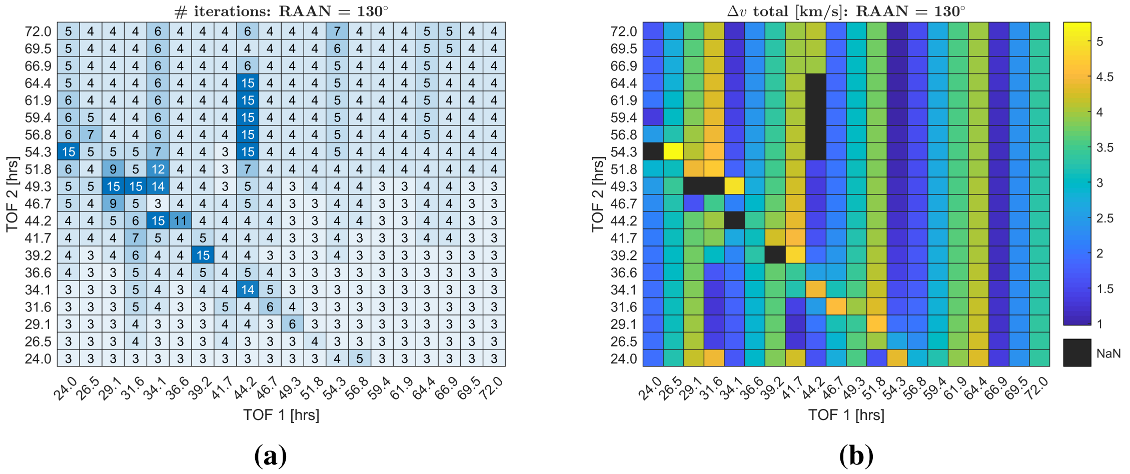
(a) (b)

Figure A.3: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 70°.



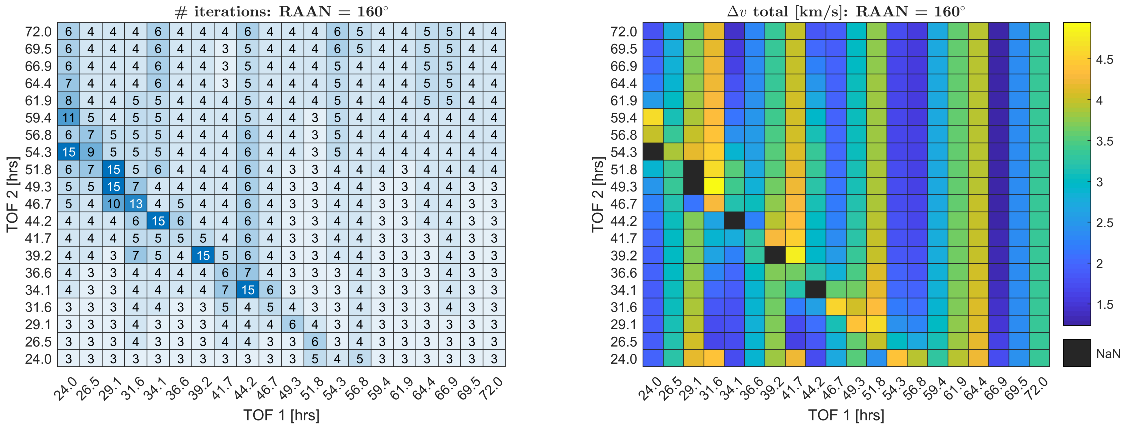
(a) (b)

Figure A.4: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 100°.



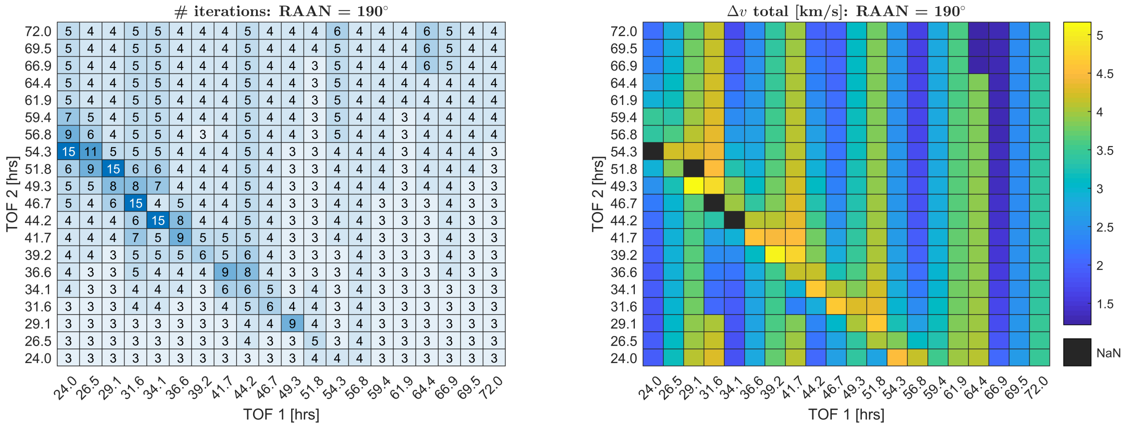
(a) (b)

Figure A.5: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 130°.



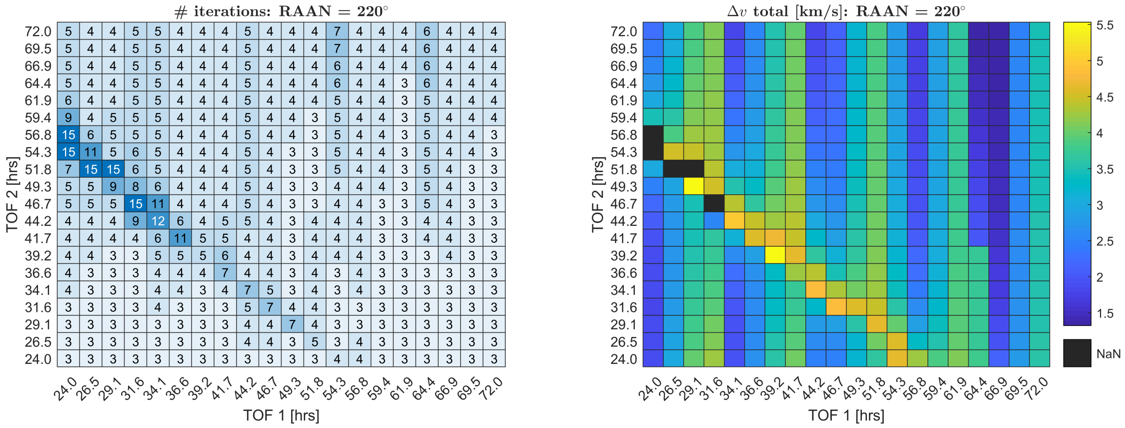
(a) (b)

Figure A.6: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 160°.



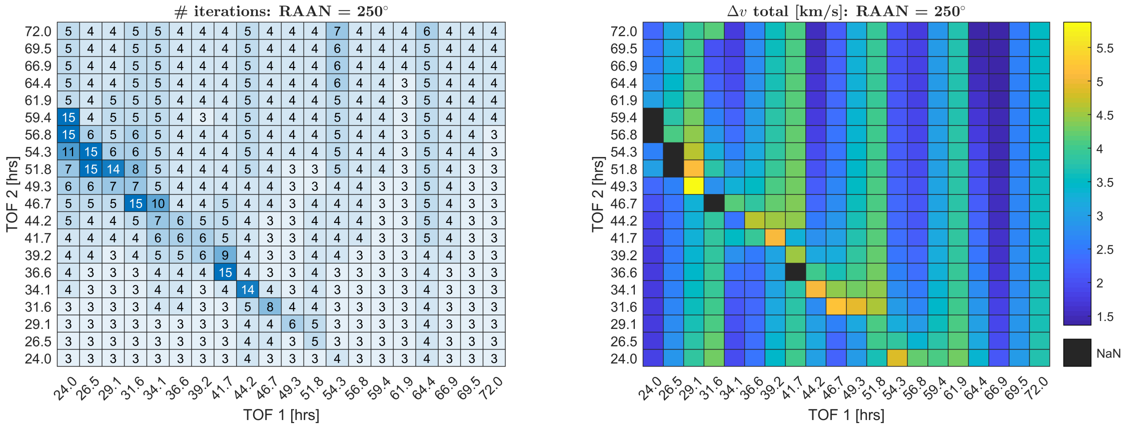
(a) (b)

Figure A.7: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 190°.



(a) (b)

Figure A.8: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 220°.



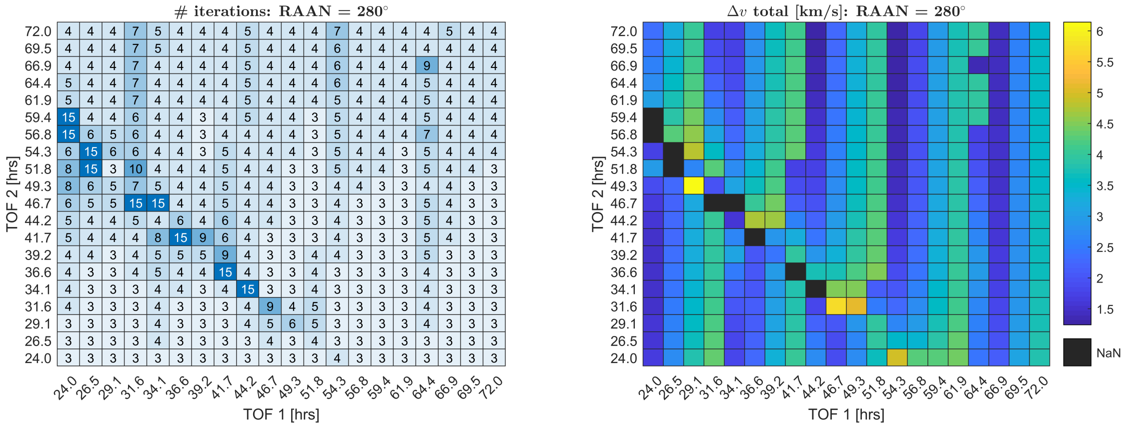
iterations: RAAN = 250°

	24.0	26.5	29.1	31.6	34.1	36.6	39.2	41.7	44.2	46.7	49.3	51.8	54.3	56.8	59.4	61.9	64.4	66.9	69.5	72.0
72.0	5	4	4	5	5	4	4	4	5	4	4	4	7	4	4	4	6	4	4	4
69.5	5	4	4	5	5	4	4	4	5	4	4	4	6	4	4	4	5	4	4	4
66.9	5	4	4	5	5	4	4	4	5	4	4	4	6	4	4	4	5	4	4	4
64.4	5	4	4	5	5	4	4	4	5	4	4	4	6	4	4	3	5	4	4	4
61.9	5	4	5	5	5	4	4	4	5	4	4	4	5	4	4	3	5	4	4	4
59.4	15	4	5	5	5	4	3	4	5	4	4	4	5	4	4	3	5	4	4	4
56.8	15	6	5	6	5	4	4	4	5	4	4	4	5	4	4	3	5	4	4	3
54.3	11	15	6	6	5	4	4	4	5	4	4	4	5	4	4	3	5	4	4	3
51.8	7	15	14	8	5	4	4	4	5	4	3	3	5	4	4	3	5	4	3	3
49.3	6	6	7	7	5	4	4	4	4	4	3	3	4	4	3	3	5	4	3	3
46.7	5	5	5	15	10	4	4	5	4	4	3	4	4	4	3	3	5	4	3	3
44.2	5	4	4	5	7	6	5	5	4	3	3	4	4	4	3	3	5	4	3	3
41.7	4	4	4	4	6	6	6	5	4	3	3	4	4	4	3	3	5	4	3	3
39.2	4	4	3	4	5	5	6	9	4	3	3	4	4	3	3	3	4	4	3	3
36.6	4	3	3	3	4	4	4	15	4	3	3	4	3	3	3	3	4	3	3	3
34.1	4	3	3	3	4	4	3	4	14	4	3	4	3	3	3	3	4	3	3	3
31.6	3	3	3	3	4	4	3	3	5	8	4	4	3	3	3	3	4	3	3	3
29.1	3	3	3	3	3	3	3	3	4	4	6	5	3	3	3	3	4	3	3	3
26.5	3	3	3	3	3	3	3	3	4	4	3	5	3	3	3	3	4	3	3	3
24.0	3	3	3	3	3	3	3	3	4	3	3	3	4	3	3	3	4	3	3	3

(a)

(b)

Figure A.9: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 250°.



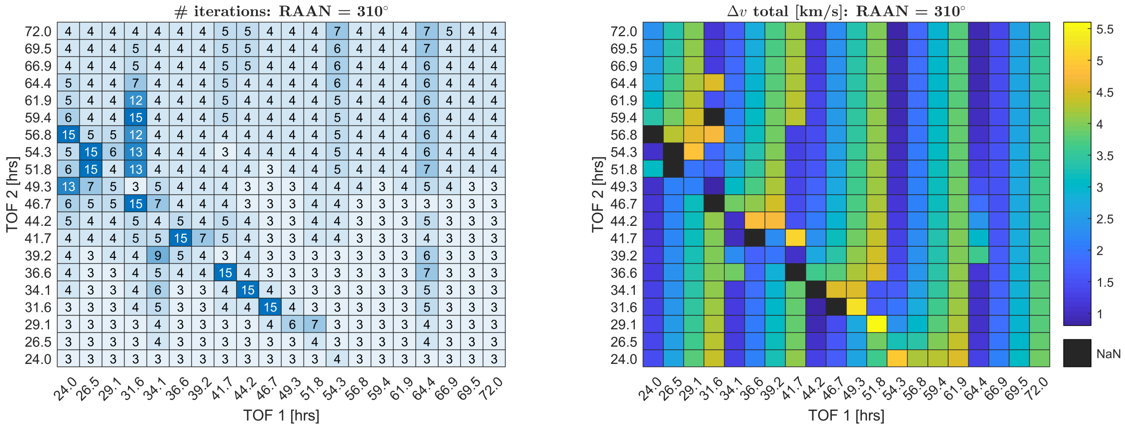
iterations: RAAN = 280°

	24.0	26.5	29.1	31.6	34.1	36.6	39.2	41.7	44.2	46.7	49.3	51.8	54.3	56.8	59.4	61.9	64.4	66.9	69.5	72.0
72.0	4	4	4	7	5	4	4	4	5	4	4	4	7	4	4	4	4	5	4	4
69.5	4	4	4	7	5	4	4	4	5	4	4	4	6	4	4	4	4	4	4	4
66.9	4	4	4	7	4	4	4	4	5	4	4	4	6	4	4	4	9	4	4	4
64.4	5	4	4	7	4	4	4	4	5	4	4	4	6	4	4	4	5	4	4	4
61.9	5	4	4	7	4	4	4	4	5	4	4	4	5	4	4	4	5	4	4	4
59.4	15	4	4	6	4	4	3	4	5	4	4	3	5	4	4	4	5	4	4	4
56.8	15	6	5	6	4	4	3	4	4	4	4	3	5	4	4	4	7	4	4	4
54.3	6	15	6	6	4	4	3	5	4	4	4	3	5	4	4	3	5	4	4	4
51.8	8	15	3	10	4	4	4	5	4	4	3	3	5	4	4	3	5	4	4	3
49.3	8	6	5	7	5	4	4	5	4	4	3	3	4	4	3	3	4	4	3	3
46.7	6	5	5	15	15	4	4	5	4	4	3	3	4	3	3	3	5	4	3	3
44.2	5	4	4	5	4	6	4	6	4	4	3	3	4	3	3	3	5	4	3	3
41.7	5	4	4	4	8	15	9	6	4	3	3	3	4	3	3	3	5	4	3	3
39.2	4	4	3	4	5	5	5	9	4	3	3	4	4	3	3	3	5	3	3	3
36.6	4	3	3	4	5	4	4	15	4	3	3	4	3	3	3	3	4	3	3	3
34.1	4	3	3	3	4	4	3	4	15	3	3	4	3	3	3	3	4	3	3	3
31.6	4	3	3	3	4	3	3	3	4	9	4	5	3	3	3	3	4	3	3	3
29.1	3	3	3	3	4	3	3	3	4	5	6	5	3	3	3	3	4	3	3	3
26.5	3	3	3	3	4	3	3	3	3	4	3	4	3	3	3	3	3	3	3	3
24.0	3	3	3	3	3	3	3	3	3	3	3	3	4	3	3	3	3	3	3	3

(a)

(b)

Figure A.10: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 280°.



iterations: RAAN = 310°

	24.0	26.5	29.1	31.6	34.1	36.6	39.2	41.7	44.2	46.7	49.3	51.8	54.3	56.8	59.4	61.9	64.4	66.9	69.5	72.0
72.0	4	4	4	4	4	4	4	5	5	4	4	4	7	4	4	4	7	5	4	4
69.5	4	4	4	5	4	4	4	5	5	4	4	4	6	4	4	4	7	4	4	4
66.9	4	4	4	5	4	4	4	5	5	4	4	4	6	4	4	4	6	4	4	4
64.4	5	4	4	7	4	4	4	5	4	4	4	4	6	4	4	4	6	4	4	4
61.9	5	4	4	12	4	4	4	5	4	4	4	4	5	4	4	4	6	4	4	4
59.4	6	4	4	15	4	4	4	5	4	4	4	4	5	4	4	4	6	4	4	4
56.8	15	5	5	12	4	4	4	4	4	4	4	4	5	4	4	4	6	4	4	4
54.3	5	15	6	13	4	4	4	3	4	4	4	4	5	4	4	4	6	4	4	4
51.8	6	15	4	13	4	4	4	4	4	3	4	4	5	4	4	4	7	4	4	4
49.3	13	7	5	3	5	4	4	4	3	3	3	4	4	4	3	4	5	4	3	3
46.7	6	5	5	15	7	4	4	4	3	3	3	4	4	3	3	3	3	3	3	3
44.2	5	4	4	5	4	5	4	5	4	3	3	4	4	3	3	3	5	3	3	3
41.7	4	4	4	5	5	15	7	5	4	3	3	4	4	3	3	3	4	3	3	3
39.2	4	3	4	4	9	5	4	3	4	3	3	3	3	3	3	3	6	3	3	3
36.6	4	3	3	4	5	4	4	15	4	3	3	3	3	3	3	3	7	3	3	3
34.1	4	3	3	4	6	3	3	4	15	4	3	3	3	3	3	3	5	3	3	3
31.6	3	3	3	4	5	3	3	4	4	15	4	3	3	3	3	3	5	3	3	3
29.1	3	3	3	3	4	3	3	3	3	4	6	7	3	3	3	3	4	3	3	3
26.5	3	3	3	3	4	3	3	3	3	3	3	4	3	3	3	3	4	3	3	3
24.0	3	3	3	3	3	3	3	3	3	3	3	3	4	3	3	3	3	3	3	3

(a)

(b)

Figure A.11: Multi-revolution Lambert guess exploitation: (a) iteration map, and (b) total Δv associated with each transfer. NaN cells refers to non-converged solutions. RAAN = 310°.

Appendix B

Detection

B.1 Edge Frequency Impact Explanation

In Sec. 3.6.3 we discussed the impact of edge frequency, showcasing how enriching with it structural and temporal information significantly improves the network discriminative capability. We believe that the observed behavior is the consequence of two factors:

1. graph diversity: the baseline algorithm relies on detecting anomalies on pure spatial and temporal information. While the graphs themselves are highly dynamic due to the satellite motion, which ensures a change of nodes and paths in subsequent snapshots, the actual structure of individual snapshots may appear somewhat similar in multiple sequences. Hence, it is possible that the dataset does not include a sufficiently high number of examples to tune the network parameters properly. However, introducing the frequency attribute provides an additional dimension to the graph, such that even if two graphs have an identical structure, they will remain different objects due to the presence of the frequency information.

2. frequency distribution: despite the Earth's rotation, with the two ground nodes fixed, the dynamics of the graphs are mostly dominated by the satellite motion. While it is true that this determines changes in the graph (trivially, the node set changes in time), as previously mentioned, the structure is not expected to change significantly. As such, having a fixed number of paths, while the frequency of individual edges changes, the overall distribution is expected to follow a certain behavior. However, anomalies present a random behavior, which may move away from the regularity of nominal conditions. Therefore, even if the network does not have knowledge of the concept of "paths", it can leverage the frequency

information combined with structural and temporal signals to infer the regularity (or not) of the edges.

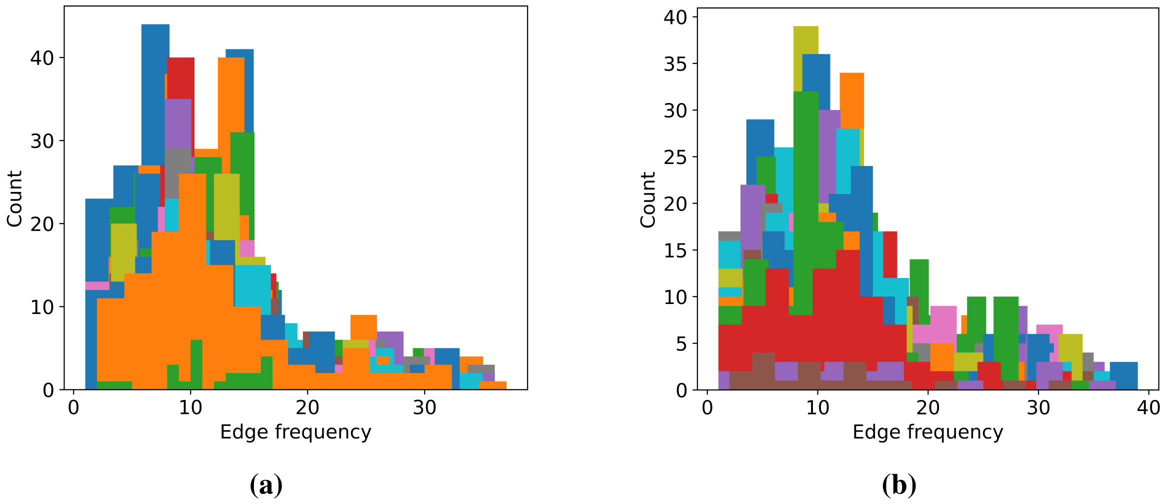


Figure B.1: Regular edge frequency distribution for the case AU to AN: (a) training set and (b) test set. Colors associated with different snapshots.

Fig. B.1 displays the edge frequency count for training and test sets for the scenario the Auburn to Anchorage scenario, for a constellation of 1600 satellites. Despite the two images display distinct frequency counts for different snapshots (as expected), the overall distribution presents a similar trend, with a skew toward lower frequency. This is reasonable, as many paths are expected to share similar edges, while having a few edges less utilized.

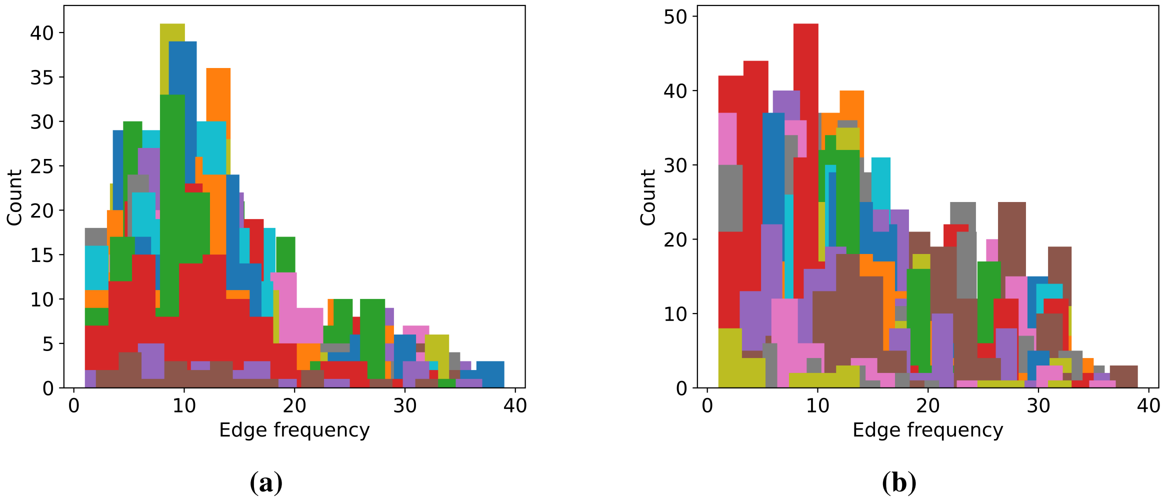


Figure B.2: Edge frequency distribution for the case AU to AN for two percentages of injected anomalies: (a) 10% o and (b) 80%. Colors associated with different snapshots.

However, when injecting anomalies, one can notice a variation of this behavior (Fig. B.2). While for the case of a 10% injection ratio, the effect is less evident (which is expected considering the few anomalies are distributed in many snapshots), the extreme case with an injection rate of 80% displays a visible change.

Appendix C

Reaction

C.1 Heatmap Rotation

The set of GP primitives for the defender includes several functions to "rotate" a heatmap and produce a new one. This can be conceptualized as rotating a spherical globe with the heatmap's data on its surface, while fixing the grid of the heatmap in place without rotation. The output maps each cell of the heatmap to the new data in that location after the rotation. Points on the heatmap are mapped to latitude, longitude coordinates, so these rotations can be performed by converting to Cartesian coordinates and multiplying by an appropriate rotation matrix. However, heatmaps are made up of discrete data cells, which generally won't line up after a rotation, so these need to be resampled from the original heatmap.

For this to work, the heatmap must be accessible by a continuous function on the sphere. Let $Hcell(i, j)$ be the accessor function for the heatmap array at row i and column j. When mapped to the sphere, for a given grid resolution res, the centers of these grid cells are located at $-\frac{\pi}{2} + i \cdot res$ latitude and $-\pi + j \cdot res$ longitude. We will define $Hmap(lat, lon)$ as a function mapping a latitude, longitude point to a heatmap value, which gives the value of the heatmap cell with the nearest center to that point:

$$Hmap(lat, lon) = Hcell(round\left(\frac{lat + \pi/2}{res}\right), round\left(\frac{(lon + \pi)\ mod\ 2\pi}{res}\right))$$

where $round$ rounds to the nearest integer, and mod is the Euclidean modulo operator.

Let $Rot(lat, lon)$ be the rotation of a point (lat, lon) following a given transformation, with $Rot^{-1}(lat, lon)$ as the inverse transformation. Then the value of the rotated heatmap at (lat, lon) is $Hmap(Rot(lat, lon))$. In order to define the cells of the rotated heatmap, we take this value at the four corners of each cell, and average them to get the value of that cell. The inner boundaries of the cell have the coordinates:

$$\begin{aligned}
top &= min(lat + \frac{res}{2} - \epsilon, \frac{\pi}{2}) \\
bottom &= max(lat - \frac{res}{2} + \epsilon, -\frac{\pi}{2}) \\
right &= lon + \frac{res}{2} - \epsilon \\
left &= lon - \frac{res}{2} + \epsilon
\end{aligned}$$

where ϵ is a value much smaller than the resolution to ensure that the ties are broken in the direction of the current cell, so that the identity rotation does not modify the heatmap. From these, the values of the rotated heatmap are:

$$\begin{aligned}
Hmap_{rot}(lat, lon) &= (Hmap(Rot^{-1}(top, right)) \\
&+ Hmap(Rot^{-1}(top, left)) \\
&+ Hmap(Rot^{-1}(bottom, right)) \\
&+ Hmap(Rot^{-1}(bottom, left)))/4 \\
Hcell_{rot}(i, j) &= Hmap_{rot}(-\frac{\pi}{2} + i \cdot res, -\pi + j \cdot res)
\end{aligned}$$