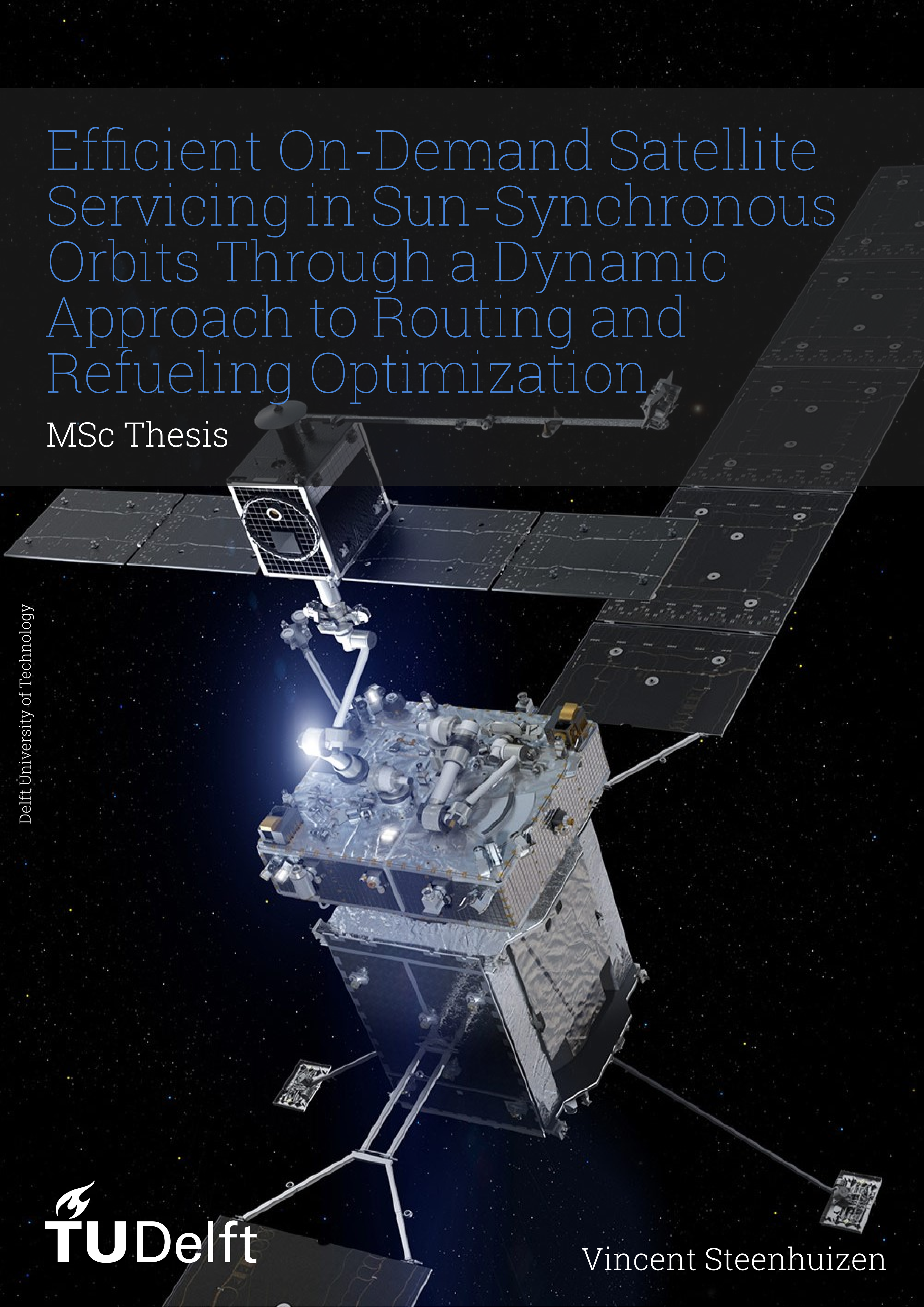


Efficient On-Demand Satellite Servicing in Sun-Synchronous Orbits Through a Dynamic Approach to Routing and Refueling Optimization

MSc Thesis

Vincent Steenhuizen

Delft University of Technology

TU Delft

Efficient On-Demand Satellite Servicing in Sun-Synchronous Orbits Through a Dynamic Approach to Routing and Refueling Optimization

MSc Thesis

by

Vincent Steenhuizen

Student Name	Student Number
Steenhuizen	4672372

Supervisor:	Marc Naeije
Thesis Duration:	Januari, 2024 - September, 2024
Faculty:	Faculty of Aerospace Engineering, Delft

Cover:	Northrop Grumman, Satellite Services in Space - An Image of a Satellite in Space
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Preface

This thesis addresses the optimization of on-demand satellite servicing in Sun-Synchronous Orbits through dynamic routing and refueling strategies. Traditional satellite operations, limited by finite on-board fuel, lead to premature mission termination. This research focuses on optimizing a refueling infrastructure comprising fuel stations, a service satellite, and client satellites, specifically for Sun-Synchronous Orbits. The algorithm is designed to efficiently manage fuel resources by determining the most effective servicing routes to minimize mission time and fuel consumption. A multi-objective optimization framework is developed to model these operations, utilizing a custom algorithm that refines transfer trajectories through high-precision numerical simulations that account for perturbations. Robustness analysis demonstrates that the algorithm can effectively handle uncertainties, consistently converging to the same optimal solutions across varying conditions. The developed algorithm efficiently identifies optimal servicing paths and is scalable to more complex missions involving multiple satellites and refueling stations.

Vincent Steenhuizen
Delft, September 2024

Summary

This thesis examines the changing landscape of satellite operations and space missions, influenced by advances in reusable launch vehicles and the need for more sustainable and cost-effective solutions. Traditional single-use satellite and launch vehicle designs are becoming less practical due to their short lifespans and the growing problem of space debris. The research focuses on on-orbit servicing as a solution to extend satellite lifespans through in-space refueling, repairs, and upgrades, reducing the need for frequent replacements. The purpose of this paper is to develop an algorithm for an infrastructure involving fuel stations, a servicer, and client satellites for on-orbit refueling in a Sun-synchronous orbit. This system uses passive fuel stations and active servicing satellites to refuel client satellites when needed, addressing challenges related to fuel use, orbital mechanics, and real-time decision-making.

The research starts by optimizing the path for a servicer satellite to move from a fuel station to a client satellite in the refueling system. It uses numerical methods to predict trajectories while considering forces such as gravity, atmospheric drag, and thrust. The transfer model, built with Tudat-Py, allows for a realistic simulation of the transfer trajectory. The optimization aims to minimize transfer time, fuel consumption, and orbit matching errors. After testing different approaches, the Cowell propagator with a Runge-Kutta-Fehlberg 8(7) variable step integrator and the MOEA/D-Gen optimizer are selected for their balance of accuracy and computational efficiency. The initial results show difficulties in finding the global optima, indicating the need for further refinement.

To improve the optimization, an analytical method is presented to calculate transfer paths between a servicer and a client satellite. Using the vis-viva equation and modeling maneuvers as impulsive shots, the analysis focuses on the main gravitational effects by using the oblateness of the Earth for passively shifting the right ascension of the ascending node. This setup allows for a more systematic calculation of possible transfers, where changes in the satellite's path align with the target orbit's key parameters. A step-by-step method adjusts the decision variables, using local search techniques to fine-tune maneuver timings, directions, and coasting phases for better orbit matching. This refined method provides more reliable results for complex servicing missions, avoiding the problems of earlier methods that often converged to local minima.

The thesis subsequently explores optimizing the route of a servicing satellite visiting multiple client satellites in Sun-synchronous orbits, framed as a Dynamic Traveling Salesman Problem. This is based on the single transfer optimization and requires real-time adjustments for changing satellite positions, fuel needs, and mission priorities. The model parameters include the sequence of satellite visits, refueling stops, and transfer strategies. These parameters need to be adjusted so that fuel use and transfer time are minimized. Due to the complexity of evaluating many paths, the model uses simplified calculations, such as estimates for satellite states and pre-determined routes. The NSGA-II optimization algorithm is adapted to efficiently find the best solutions, balancing exploration of new paths with refinement of known ones. The results show effective servicing sequences and demonstrate the model's ability to adapt to changing mission needs.

Verification of the optimization results is carried out to ensure that the solutions are accurate. This process includes comparing the Pareto optimal fronts obtained from a brute-force analysis of the design space with those generated by the NSGA-II algorithm. In addition, specific points along the Pareto front are examined to evaluate their choice of path and fuel consumption. Finally, the optimization results are recalculated without the earlier simplifying assumptions to verify the reliability and robustness of the proposed approach.

The thesis finally evaluates the sensitivity and robustness of the model to changes in parameters, which is crucial for practical application. This assessment involves varying the random seed to test the consistency of the solutions, altering the available fuel to observe the model's response to changes in resources, and introducing uncertainties in the initial satellite states to analyze the impact of uncertainties. The findings demonstrate that the model consistently generates reliable solutions in different scenarios, which shows its effectiveness even with uncertainties or changes in mission conditions.

Acknowledgments

I want to thank Marc Naeije for accepting my research topic and for his great supervision and support throughout this thesis. I also want to thank Josh Rodenbauch from Dawn Aerospace for the helpful monthly meetings, which were key in making sure the research was relevant to the industry.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
CS	Client Satellite
DFT-port	Docking and Fluid Transfer port
DTSP	Dynamic Traveling Salesman Problem
EO	Earth Observation
FS	Fuel Station
GEO	Geostationary Orbit
HPC	High Performance Computing
LEO	Low Earth Orbit
MINLP	Mixed-Integer Non-Linear Programming
OOR	On-Orbit Refueling
OOS	On-Orbit Servicing
OTV	Orbital Transfer Vehicle
RAAN	Right Ascension of the Ascending Node
RPOD	Rendezvous, Proximity Operations, and Docking
RK	Runge-Kutta
RKF	Runge-Kutta-Fehlberg
RSc	Resupply Spacecraft
SS	Service Satellite
SSC	Service Spacecraft
SSO	Sun-Synchronous Orbit
TLE	Two-Line Element
TSP	Traveling Salesman Problem
WBS	Work Breakdown Structure

Symbols

Symbol	Definition	Value	Unit
a	Semi-major axis		m
γ	Burn Direction in the Local Horizontal Plane		rad
e	Eccentricity		$-$
ΔV	Velocity Change		m/s
G	Gravitational Constant	$6.6743 \cdot 10^{-11}$	$m^3 kg^{-1} s^{-2}$
g_0	Gravitational Acceleration on Earth	9.81	ms^{-2}
i	Inclination		rad
I_{sp}	Specific Impulse		s
J_2	Earth's Oblateness Factor		$-$
k	Any Arbitrary, Positive Integer		$-$
Λ	Mass Fraction		$-$
M	Mean Anomaly		rad
M_0	Initial Satellite Mass		kg
M_E	Earth's Mass	$5.972 \cdot 10^{24}$	kg
M_e	Mass of Satellite after a Burn		kg
M_f	Fuel Mass of Satellite		kg

Symbol	Definition	Value	Unit
μ	Earth's Gravitational Parameter	$3.986 \cdot 10^{14}$	$m^3 s^{-2}$
n	Mean Angular Motion		rad/s
ω	Argument of Periapsis		rad
Ω	Right Ascension of the Ascending Node		rad
$\frac{d\Omega}{dt}$	RAAN Drift Rate		rad/s
θ	True Anomaly		rad
r	Orbital Radius		m
R	Orbital Radius		m
R_E	Radius of Earth	$6.371 \cdot 10^6$	m
t	Time		s
T	Orbital Period		s
u	Argument of Latitude		rad
V	Velocity		m/s
V_f	Final Velocity		m/s
V_i	Initial Velocity		m/s
z	Number of Transfer Phases		$-$

1 Introduction

Rapid advances in space exploration have catalyzed a fundamental shift in how we approach satellite design, mission planning, and space operations. Traditionally, satellites and launch vehicles were engineered for single-use missions, and their operational lifespans were tightly constrained by their onboard fuel capacity. Once this fuel was depleted, the satellites were rendered inoperable and left to drift in orbit as space debris. This approach, while initially effective, has increasingly revealed its limitations, particularly in terms of sustainability, cost-efficiency, and the long-term viability of space missions.

In response to these challenges, the space industry has witnessed significant innovation, particularly in the development of reusable launch vehicles. SpaceX's Falcon 9, for example, has dramatically lowered the cost of access to space by introducing a model in which launch vehicles are recovered and reused [1]. This reduction in cost has democratized space access, enabling a broader range of entities, including smaller companies and nations, to engage in space missions.

As launch costs decrease, there is increasing emphasis on optimizing the use of in-orbit assets, particularly satellites. One of the most promising solutions to this challenge is on-orbit servicing (OOS), which involves extending the operational life of satellites through in-space refueling, repairs, and upgrades. This capability has the potential to transform satellite mission planning by reducing the need for new satellite launches, thus contributing to more sustainable space operations.

The importance of these developments is particularly evident in orbits such as the Sun-synchronous orbit (SSO), which is highly favored for Earth observation missions because of its consistent lighting conditions. As the number of satellites in SSO increases, driven by more affordable launch options, the need for efficient orbital management and servicing solutions becomes paramount. In other words, orbital servicing technologies represent a critical advancement in maintaining the viability and functionality of these satellites over extended periods.

However, the implementation of OOS is not without its challenges. The dynamic nature of satellite orbits, the need for precise fuel management, and the requirement for real-time decision making in mission planning are significant hurdles that must be overcome. This thesis focuses on addressing these challenges by developing and optimizing an algorithm that can dynamically adjust the routing of a servicer satellite within an on-orbit refueling (OOR) infrastructure. This algorithm is designed to operate on demand, adapting to the unpredictable nature of space operations and ensuring the efficient use of resources in refueling missions.

This research contributes to the growing field of space operations by providing innovative solutions that enhance the sustainability and cost-efficiency of satellite missions.

2 Background Information

This chapter provides the necessary background information for understanding the development and challenges of satellite servicing and refueling in space. It explores the historical evolution of satellite and launch vehicle design, the significance of Sun-Synchronous Orbits (SSO), and the state of current on-orbit servicing technologies.

Section 2.1 reviews the historical context of satellite and launch vehicle design, focusing on the shift from single-use to reusable technologies. Section 2.2 discusses the importance of SSO for Earth Observation (EO) and its advantages for OOR. Section 2.3 examines current OOR technologies and strategies. Section 2.4 addresses the challenges in extending satellite longevity and ensuring sustainability in space operations. Section 2.5 introduces optimization models for efficient satellite servicing. Section 2.6 covers the economic implications of OOR. Section 2.7 presents Dawn Aerospace's approach to enabling satellite refueling and overcoming current barriers. Finally, the chapter culminates in the formulation of the problem in Section 2.8, setting the stage for the research question to be addressed in subsequent chapters.

2.1. Historical Context of Satellite and Launch Vehicle Design

Since the dawn of space exploration, satellites and launch vehicles have traditionally been designed for single-use missions. Satellites, once launched, operate with a finite amount of fuel, which dictates the life expectancy of their mission. Once fuel is depleted, these satellites become inoperable, effectively becoming space debris. Similarly, launch vehicles have historically been single-use, with stages that are jettisoned during ascent and often left to burn up in the atmosphere or crash into the ocean.

The inefficiencies of this approach are stark compared to other industries, such as aviation, where aircraft are designed to be reused across thousands of flights. The idea of building a new aircraft for each journey would be considered economically and environmentally unsustainable, yet this has been the standard in space operations for decades.

The turn of the century saw an increase in interest in reusable space technologies driven by the potential to significantly reduce the cost of access to space. SpaceX, a private aerospace manufacturer and space transportation company, has been at the forefront of this revolution. The company's Falcon 9 rocket, which first achieved a successful landing and recovery of its first stage in 2015 [2], has proven the feasibility of reusability in space launch vehicles. This innovation has dramatically reduced the cost of sending payloads to space, with prices falling by almost an order of magnitude compared to previous standards [1]. The success of Falcon 9 has made space more accessible, enabling a new wave of small satellite operators and expanding the market for space-based services.

However, while reusable launch vehicles have addressed the cost and frequency of launches, the challenge of extending the operational life of satellites remains. The current paradigm still relies on single-use satellites, leading to a continuous cycle of replacement and launch. This inefficiency presents an opportunity for innovation in the form of OOS, where satellites can be refueled, repaired, or upgraded while in orbit, potentially extending their operational lives by years or even decades.

2.2. Sun-Synchronous Orbit and Its Importance

Sun-synchronous orbit is a near-polar orbit in which a satellite passes over any given point of the Earth's surface at the same local solar time. This consistency in lighting conditions makes SSO particularly valuable for EO satellites, as it ensures that images are taken with the same solar illumination angle each time, minimizing shadows and variations caused by changing sunlight, thus enabling more reliable comparisons of environmental changes, land use, and other variables over time.

The popularity of SSO for EO missions has only increased with the advent of more cost-effective launch options, particularly SpaceX's rideshare and transporter missions [3]. These missions have significantly reduced the barrier to entry for smaller companies and organizations, allowing them to place their satellites into SSO at a fraction of the cost previously required. This trend is likely to continue as more small satellite constellations are planned for EO and communication purposes.

In addition to its operational benefits, SSO offers unique advantages for OOS. The oblateness of the Earth causes a natural precession of the orbital plane, known as the J2 effect, which can be exploited to change the inclination of a servicing satellite's orbit passively. This reduces the fuel needed for orbital adjustments, making SSO an ideal target for the deployment of servicing satellites (SS).

2.3. Current State of On-Orbit Servicing Technologies

The concept of OOS is not new, but its implementation has been limited to specific missions and scenarios. The first notable OOS mission was NASA's Hubble Space Telescope repair in 1993, where astronauts aboard the Space Shuttle Endeavour installed corrective optics to rectify a flaw in the telescope's primary mirror [4, 5]. This mission demonstrated the feasibility and value of on-orbit servicing, paving the way for future missions.

Currently, the most straightforward form of OOS is the one-to-one servicing strategy, where a single servicing satellite visits and services a single target satellite. This approach, while effective for specific high-value missions like Hubble, is not scalable for broader applications, particularly in the commercial sector, where cost-efficiency is of great importance.

Advances in technology and the growing demand for satellite services have led to the exploration of more complex servicing strategies. One such strategy is one-to-many servicing, where a single servicing satellite is designed to visit multiple client satellites (CS) during its mission. NASA's Restore-L mission, which aims to refuel the Landsat-7 satellite, is an example of this approach. This mission could potentially extend to servicing other satellites in the future, demonstrating the scalability of the one-to-many model [6].

In academic research, much attention has been given to the many-to-many servicing strategy, where multiple SS operate within a constellation of CS. This model is particularly relevant for geostationary orbit (GEO) and low Earth orbit (LEO), where satellite constellations are becoming more common. The many-to-many approach is seen as a way to maintain and extend the life of expensive GEO satellites or to manage large constellations in LEO, such as those used for global Internet coverage.

Another innovative approach is the peer-to-peer strategy, in which satellites within a constellation can refuel each other, thus eliminating the need for dedicated SS. This strategy is particularly appealing for constellations where all satellites are equipped with mission-specific instruments and the capability to act as servicers. This approach has been shown to optimize fuel usage in the constellation, minimizing the need for new launches [7].

The most recent development in OOS was made by Zhu et al. [8] and Han et al. [9], who introduce a new servicing mode called the FS-servicer-CS On-Orbit Refueling mode, where passive fuel stations (FS) are deployed in orbit, and active SS move between these FS and CS. This infrastructure allows for greater flexibility in mission planning and operations, as the SS can refuel and continue their missions without returning to Earth.

2.4. Challenges in Satellite Longevity and Sustainability

Despite advances in OOS technologies, several challenges remain in extending the longevity of satellites and ensuring the sustainability of space operations. The most significant challenge is fuel management. Satellites are launched with a finite amount of fuel, which limits their operational life. Once the fuel is depleted, the satellite becomes non-functional, contributing to the growing problem of space debris.

The deployment of OOR infrastructure addresses this challenge by allowing satellites to be refueled in space. However, optimizing the use of fuel for both SS and CS remains a complex problem. The SS must balance the need to minimize its own fuel consumption with the need to efficiently service multiple clients, each with its own fuel constraints and mission requirements.

In addition to fuel management, the dynamic nature of space environments poses another challenge. Satellites in LEO, for example, are subject to perturbations from atmospheric drag, solar radiation pressure, and gravitational influences from the Earth and Moon. These factors can cause deviations from planned orbits, making it difficult to predict the exact location of a satellite at any given time. This uncertainty complicates the planning and execution of servicing missions, requiring robust optimization models that can adapt to changing conditions.

2.5. Optimization Models for Satellite Servicing

Optimization models play a crucial role in the planning and execution of OOS missions. These models are designed to find the most efficient way to service a set of CS while minimizing fuel consumption and mission time. The complexity of these models varies depending on the servicing strategy employed.

For example, in a one-to-many servicing scenario, the optimization model must determine the optimal sequence of satellite visits that minimizes the total mission time and fuel usage. This problem is akin to the Dynamic Traveling Salesman Problem (DTSP), where the goal is to find the shortest possible route that visits a set of locations and returns to the starting point. It differs from the regular Traveling Salesman Problem (TSP) in the following way: Consider a postman who needs to deliver a number of packages to different houses in the city. Solving the question of what the most efficient route is that the postman can take is considered as the regular TSP. Throughout the entire route that the postman is traveling, the locations of the houses he needs to visit do not change, meaning that the effort that is required to travel between any two houses remains equal. Therefore, the distance that the postman needs to travel between two houses does not depend on the choices he makes before starting this leg. In other words, the problem is static.

Now, if the postman is replaced by a SS, the houses are replaced with CS and the city is replaced by a group of satellites in SSO, the problem changes. Due to perturbations that act on all satellites in orbit, the relative position between all satellites constantly changes. Therefore, the 'distance' that a servicing satellite must travel between any two CS does depend on the choices it makes before starting this specific leg. This dynamic nature of the problem is what adds the largest amount of complexity to finding a solution to the problem.

In even more complex scenarios, such as many-to-many servicing, the optimization model must account for multiple servicing satellites, each with its own set of clients. This adds layers of complexity, as the model must not only determine the optimal sequence of visits for each SS, but also coordinate the activities of multiple satellites to avoid conflicts and ensure efficient use of resources.

Finally, the FS-servicer-CS OOR mode introduces a final level of complexity, as the optimization model must now include the locations of fuel stations and the need for SS to refuel periodically. This adds another component to the optimization problem, where the model must balance the trade-offs between mission time, fuel consumption, and the frequency of refueling stops.

2.6. Economic Implications of On-Orbit Servicing

The development of OSS capabilities has significant economic implications for the space industry. The ability to extend the operational life of satellites through in-space refueling and repairs can lead to substantial cost savings, as it reduces the need for frequent satellite replacements and the associated launch costs.

For commercial satellite operators, the potential to generate revenue over a longer period makes OOS an attractive investment.

2.7. Dawn Aerospace

One of the main reasons why it has proven to be so difficult to introduce a commercial OOR service is due to the 'chicken-and-the-egg' problem: An OOR service is only viable in the event that there are satellites that have the capability to be refueled. Because almost all satellites that have been launched have no such capabilities, there is no market for this yet. However, because there is no OOR service to make use of, it makes no sense for a company to put the extra effort into implementing a refueling port in their already very expensive satellite. Therefore, there will also be no increase in the market for potential OOR companies.

Currently, there are a couple of companies worldwide that are developing technologies that will help solve this problem. One of these companies is Dawn Aerospace, located in Delfgauw, The Netherlands. One of their main departments is the In-Space Propulsion department, which has the goal of delivering of-the-shelf propulsion systems for different sizes of satellites ranging from CubeSats to small satellites, and even larger in the near future [10]. They are working towards a potential solution to the chicken-and-the-egg problem by implementing a fuel port to their propulsion systems that can be used both on the ground for fueling the satellite before launch and on-orbit whenever an OOR service is commissioned. Since a ground fueling port is already necessary, the new Docking and Fluid Transfer Port (DFT-port) will facilitate the cost-effective addition of an OOR port for commercial and government satellites, thereby reducing the drawbacks of added mass and expense [11].

2.8. Problem Formulation

Sections 2.1 to 2.7 culminate in the formulation of the research question. The cheapest option for a company or government to launch their satellite into LEO is by launching it on a Falcon 9 from SpaceX. Because SpaceX has dedicated Transporter missions that launch dozens of satellites simultaneously to SSO, there will be a large cluster of satellites that are relatively close to each other after launch. Because many of these satellites will not make use of an orbital transfer vehicle (OTV) to place themselves in another orbit, this cluster will be maintained over time, with only some slight drift effects causing them to spread apart slowly. Because the satellites will be placed in a relatively low orbit of about 500 kilometer, the mission lifetime is on the order of a couple months to a maximum of a few years. Therefore, if an FS-servicer-CS OOR is launched together with the rest of these satellites on the Transporter mission, this refueling infrastructure can be used by the client satellite to increase their mission lifetime. With companies like Dawn Aeropsace that are already including refueling ports to some of these satellites, the demand for such an infrastructure will only increase over time. The added benefit of deploying the refueling infrastructure in SSO is that the oblateness of the Earth can be used to passively match the right ascensions of the ascending node of the SS and the CS, saving fuel and thus cost.

How such an FS-servicer-CS OOR infrastructure would be implemented is what will be investigated further in this report. Specifically, the following research question needs to be answered:

- Given an on-demand FS-servicer-CS OOR infrastructure in SSO, what is the most efficient path a servicer can take to refuel multiple client satellites?

To answer this question, the problem is divided into two subquestions that will be answered in Chapters 3 and 4 respectively:

- What transfer trajectory offers the highest efficiency when moving from a single fuel station to any given client satellite?
- What sequence of visits to multiple arbitrary client satellites optimizes efficiency for the servicing spacecraft, akin to solving the Dynamic Traveling Salesman Problem?

By answering these subquestions, the main research question can also be answered.

3 What transfer trajectory offers the highest efficiency when moving from a single Fuel Station to any given Client Satellite?

This chapter explores the most efficient transfer trajectories between a single FS and any given CS. It introduces the concepts of numerical propagation and optimization necessary for calculating optimal transfer paths, discussing the trade-offs between accuracy and computational efficiency in Section 3.1. The chapter also presents two local optimization methods in Section 3.2, one of which is custom designed for the transfer problem.

3.1. Propagation and Optimization

In satellite mission planning and orbital maneuvering, accurately predicting the future state of a satellite is crucial - comprising its position, velocity, and other relevant parameters. This process, known as satellite state propagation, allows mission designers to ensure that a satellite remains on its intended trajectory, performs the necessary maneuvers, and avoids potential hazards. The methods used to achieve this propagation are broadly categorized into analytical and numerical approaches, each with distinct advantages and limitations.

Analytical methods are based on closed-form equations derived from classical celestial mechanics and orbital dynamics. These methods are grounded in well-established mathematical formulas, such as those describing Keplerian orbits, where the satellite's motion is simplified to conic sections under the influence of a central gravitational force, typically that of the Earth. The appeal of analytical methods lies in their simplicity and computational efficiency. By reducing the problem to a set of equations, these methods allow for quick calculations, providing insights into the relationships between orbital elements and the forces acting on the satellite. However, this simplicity comes at a price. Analytical methods often depend on idealized assumptions, such as neglecting atmospheric drag, solar radiation pressure, and the gravitational influence of other celestial bodies, that limit their accuracy, especially over extended periods or in environments with significant perturbations.

To address the limitations of analytical methods, numerical propagation techniques have been developed. Unlike their analytical counterparts, numerical methods do not rely on simplified models but instead perform step-by-step integration of the satellite's equations of motion. This approach allows for a comprehensive simulation of the satellite's trajectory, taking into account all relevant forces at each time step. The numerical method implemented in this study leverages the Runge-Kutta-Fehlberg (RKF) integration technique to simulate the satellite's state under the influence of a range of forces, including gravitational interactions with the Earth, Sun, and Moon; atmospheric drag, which is particularly impactful in low Earth orbits; solar radiation pressure; and thrust forces applied during specific maneuver phases.

The primary advantage of numerical methods lies in their accuracy and flexibility. By accounting for all relevant forces, numerical propagation provides a highly realistic simulation of the satellite's trajectory, capable of handling complex scenarios that involve multiple perturbative forces and non-Keplerian effects. This level of detail is critical for precision missions and for scenarios where accurate predictions are necessary to ensure mission success. In addition, numerical methods offer significant customizability, allowing engineers to tailor simulations to include specific forces, model various thruster profiles, and adapt to changing mission requirements.

However, these benefits come with certain trade-offs. Numerical methods are computationally intensive and require significant processing power, especially for long-duration missions or simulations that require high precision. The setup and tuning of numerical models are also more complex compared to analytical approaches, demanding careful consideration of integration methods, time steps, and termination conditions to ensure both stability and accuracy. Additionally, while powerful, numerical methods are not immune to errors. Truncation and round-off errors can accumulate over time, potentially affecting the accuracy of long-term predictions. Therefore, managing these numerical errors is an essential aspect of implementing a successful numerical propagation model.

In summary, while analytical methods provide a quick and efficient means of predicting satellite trajectories, their reliance on simplified models limits their application to idealized scenarios. Numerical propagation, as employed in this study, offers a more accurate and flexible alternative, capable of simulating the complex dynamics of satellite motion in real-world conditions. This section will explore the implementation of the numerical propagation method used in this research, highlighting its advantages in precision and flexibility, as well as the challenges associated with its computational demands and complexity. Through this detailed examination, the effectiveness of numerical propagation in satellite mission planning will be thoroughly demonstrated.

3.1.1. Transfer model with Tudat-Py

This subsection describes how the transfer model is set up by making use of the open source Tudat-Py software.

Objectives of the transfer model

The goal of the transfer model is that it will be used by an optimization algorithm to optimize one or more specific values, called objectives. It does this by altering specific input values that are called decision variables. Finally, there could be constraints that are imposed on the model. A clear example of a constraint in this context would be the minimum altitude that a satellite is allowed to have during its transfer, as it would be very bad to have a satellite that reenters through the atmosphere.

The first step in deciding what the model will look like is to determine what the objectives will be. Almost always in spaceflight, the most important variable that needs to be optimized is the fuel mass, as this is directly related to the mission cost and mission lifetime. However, in the case of the refueling infrastructure considered, fuel deficiency is by definition not the primary objective, as it is assumed that there is enough fuel in orbit already. Therefore, because of the perspective of the business side, the most important objective is the transfer time. That is, how fast can I, as a refueling company, be able to refeul a satellite from another company?

Despite the assumption that fuel is readily available in space, it can still be rewarding to take it as an objective. Again, from a business perspective, it still costs money to get the fuel to orbit. So, if one can use less fuel for a transfer, more is available for selling to customers.

Ideally, these are the only two objectives needed to optimize the transfer between two satellites in space. However, because of the nature of numerical propagation, it is impossible to design a transfer model that guarantees the servicer will reach the target spacecraft. This is because the final state of each satellite depends on all its previous states, and the purpose of numerical propagation is to more accurately predict the future state of the satellite. Given the need to account for these subtle perturbations, it is not feasible to predefine the arrival of the SS at the CS. Therefore, the last objective is a cost function that evaluates the precision of the final orbit alignment between the two satellites.

Determination of the decision variables

The next step in determining what the model will look like is determining what the decision variables will be. As mentioned above, the decision variables are the 'nobs' that the optimization algorithm can tune in order to minimize the set objectives.

Generally, a satellite in orbit is capable of changing its velocity in all three dimensions and its attitude. It can do this at any given moment in time, and sometimes even with changing rates. That is, it can vary its thrust or it can vary the rate of rotational acceleration. Having the ability to incorporate these dynamics into the model is the most accurate way to represent the actual transfer. However, this would be infeasible since this would require one to somehow specify at each moment in time, what specific thrust the satellite would have in what specific direction. In order to solve this, a couple of assumptions can be made. First, the thrust magnitude produced by the thrusters on satellites is often relatively constant. Therefore, the thrust magnitude is assumed to be constant and equal to 80 Newtons, which corresponds to four B20 thrusters from Dawn Aerospace that are often used in similarly sized spacecraft [12]. In addition, the thrusters could theoretically be fired an unlimited number of times. Although it is possible to build the model with this capability, it does increase the complexity of the problem significantly, and because the single-to-single transfer optimization will only be a small part of the final DTSP that will be discussed in Chapter 4, it is best to have this model as simple as possible while maintaining feasible and practical results. Therefore, the number of burns was initially set at two: The first burn to enter the transfer trajectory and the second one to perform the orbit matching burn. However, it was later discovered that this approach limits the satellite's ability to match the target orbit. This is because it is impossible to first change to an elliptical orbit for phase matching and then return to a circular orbit with a semi-major axis different from the original circular orbit. This problem is solved by including a third burn that can be used to raise or lower the orbit. In conclusion, the servicer will thus have three burns in total: The first is to initialize the transfer, the second is to raise or lower the semi-major axis, and the final burn is the insertion burn, after which the servicer should have matched the orbit of the target. Again, it is good to emphasize that slightly more optimal transfers could be achieved by performing each of these three phases in multiple burns.

Now that the simplifying assumptions have been made, the decision variables can be deduced. Per burn, there are a total of four variables that can be changed: The first variable is the time the satellite is coasting before firing its thrusters. The second variable is the duration of the burn, and the final two variables are the two polar directions in which the satellite can fire its thrusters. This results in a total of twelve decision variables and thus inputs for the transfer model.

Finally, the model needs to know what the state of the servicer and target is at the start of the simulation. This results in the fact that three more inputs are required for the setup of the optimization: The starting time of the simulation, the starting state of the servicer, and the starting state of the target.

Tudat-Py

Because the goal of this chapter is to find which transfer trajectory offers the highest efficiency when moving from a single FS to any given CS, it is required per definition that the trajectory actually matches the orbit of the CS. As discussed in the introduction of this section, analytical methods often do not provide this required accuracy, as it is almost impossible to correctly implement the relevant perturbations. Therefore, the method of choice for determining the transfer trajectory is numerical simulation.

There are many different open source software tools available such as NASA's GMAT [13] and Orekit [14]. Although both are widely used, the software that will be used for this research is Tudat-Py [15]. The primary reason for this is its familiarity and the freedom it provides for changing nearly all settings, making it very versatile.

In summary, Tudat-Py lets the user first set the environment that is relevant for the body to be modeled. The different propagation and integration settings can then be set. Finally, the software automatically propagates the state of the body and saves all relevant data during the process.

Environment setup

The simulation environment is a critical component in the setup and execution of numerical simulations in Tudat-Py. This environment encompasses the physical and dynamical parameters that govern the interactions between various celestial bodies within the simulation framework. The accurate definition and configuration of this environment is essential for achieving realistic and reliable simulation results. The main elements of the simulation environment are listed below.

- **Celestial Bodies and System Configuration:** The environment is centered around the definition of celestial bodies, which include planets, moons, spacecraft, and other objects of interest. Each body within the environment is characterized by a set of physical properties such as mass, gravitational parameters, rotational attributes, and, where applicable, atmospheric properties. These properties are fundamental in determining the gravitational and non-gravitational forces acting upon the bodies throughout the simulation.

In typical space mission simulations, a central body is identified around which other bodies orbit. For instance, in simulations of missions around Mars, the planet Mars often serves as the central body, while the spacecraft and its accompanying moons or satellites are treated as secondary bodies.

- **Ephemerides:** The environment incorporates ephemerides to provide the positional and velocity data of celestial bodies over time. Tudat-Py supports a range of ephemeris models, including both analytical approaches and numerical integration methods. The choice of ephemeris model directly impacts the accuracy of the simulation, especially in long-duration missions where precise trajectory prediction is crucial.
- **Gravitational Models:** Additionally, the simulation environment includes models of gravitational fields, which may vary in complexity depending on the requirements of the simulation. While simple point mass approximations may suffice for basic simulations, more complex models incorporating zonal and tesseral harmonics are available for higher fidelity simulations. These models account for the non-uniform distribution of mass within celestial bodies, thereby providing a more accurate representation of the gravitational forces at play.
- **Atmospheric Models:** For simulations involving celestial bodies with atmospheres, such as Earth or Mars, the environment can include detailed atmospheric models. These models simulate atmospheric drag and other relevant atmospheric effects on spacecraft, which are particularly important during phases of atmospheric entry, descent, and landing.
- **Radiation Models:** The environment also accounts for radiation pressure, particularly from solar radiation, which can have a significant impact on spacecraft trajectories. The inclusion of solar radiation pressure models is vital for missions involving extended exposure to the sun, such as interplanetary missions or operations near the solar corona.

The environment in Tudat-Py is highly customizable, allowing users to define specific properties for each body, as well as custom gravitational fields and atmospheric models if the default configurations do not meet the requirements of the simulation. This flexibility is particularly important in research-oriented simulations where non-standard conditions or specific mission scenarios must be modeled accurately.

The fidelity of the simulation is highly dependent on the precision with which the environment is defined. By carefully selecting and configuring the components of the environment, a high degree of accuracy can be achieved in the simulations, thus improving the reliability of the predicted outcomes.

In Section 3.1.1 a careful selection of the environment will be made.

Thrust and Rotation Models

The thrust magnitude and direction are crucial components in the accurate simulation of the trajectory of the servicer satellite. They dictate how the servicer's velocity changes over time, directly influencing the satellite's path through space. In the transfer model, these are handled with particular care to ensure smooth transitions and accurate alignment with the spacecraft's velocity vector.

In many orbital mechanics simulations, the thrust is applied in discrete intervals, often switching on and off abruptly. However, this can pose significant challenges for numerical integrators and propagators, particularly those that rely on smooth functions for accurate and stable performance. Discontinuous changes in thrust can lead to numerical instability and inaccurate results, which is undesirable in precise orbital simulations.

To address this, the transfer model function utilizes Hermite spline interpolation to model the thrust magnitude. Hermite splines are a type of piecewise polynomial function that can be used to create smooth transitions between different points as shown in Figure 3.1.

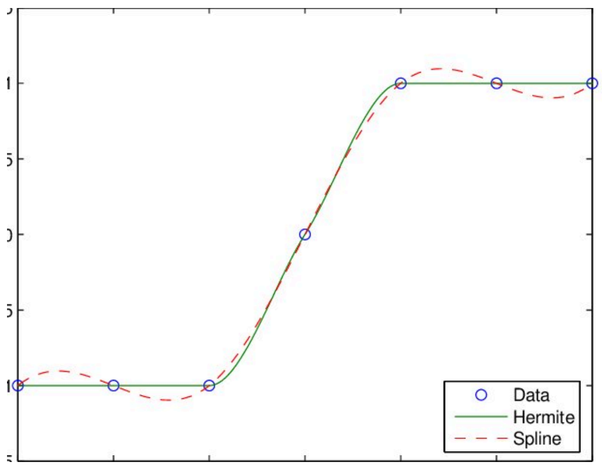


Figure 3.1: Piece wise cubic Hermite interpolation [16]

By applying Hermite splines, the thrust function ensures that the thrust magnitude varies smoothly over time, avoiding the pitfalls of abrupt changes. This smooth variation is not only beneficial for integrator and propagator performance, but also more realistically represents how real spacecraft engines might ramp up or down their thrust.

This approach provides the dual benefit of improved numerical stability and a more accurate representation of real-world thrust dynamics, where engines typically cannot switch instantaneously between different power levels.

The direction in which this thrust is applied is as critical as the magnitude. In the model, the thrust direction is carefully managed to ensure it is defined in respect to the servicer's velocity vector in what is known as the aerodynamic reference frame.

The thrust direction is initially specified in the body-fixed reference frame of the servicer. In this frame, the axes are fixed relative to the satellite's structure. However, for orbital maneuvers, it is often necessary to align the thrust direction with the velocity vector of the spacecraft, which requires transforming the body-fixed reference frame into the aerodynamic reference frame.

To be more precise, the aerodynamic reference frame is defined as follows:

- The x-axis aligns with the velocity vector of the spacecraft.
- The y-axis points out of the orbital plane.
- The z-axis points towards the center of the Earth, completing the right-hand rule.

One of the challenges with Tudat-Py is that it does not provide a straightforward method to propagate the attitude of the spacecraft while forcing the body-fixed x-axis to remain colinear with the velocity vector. This creates a need for a custom approach to maintain the correct orientation of the spacecraft during thrust maneuvers. To achieve this, the following steps are performed:

1. **Defining Constant Orientation in the Global Reference Frame:** The x-axis of the spacecraft is defined in the global reference frame to point in the same direction as the angular momentum vector of the orbit, which by definition points out of the orbital plane. This ensures that the x-axis of the spacecrafts body-fixed reference frame in the global reference frame is consistently aligned with the y-axis of the aerodynamic reference frame.
2. **Rotational Rate Matching Orbital Rate:** In Tudat-Py, when defining the constant orientation of the body-fixed reference frame with respect to the global reference frame, the rotation angle around the bodies x-axis as a function of time can be specified. This is done in such a way that the rotation rate of the spacecraft around its x-axis is set to match its orbital rate around the Earth. This forces the spacecraft to maintain its orientation relative to the Earth, ensuring that one of its axes always points towards the Earth, much like the z-axis in the aerodynamic reference frame.
3. **Transformation to Aerodynamic Reference Frame:** After establishing this pseudo-aerodynamic reference frame, the final step is to transform it into the actual aerodynamic reference frame,

where the x-axis is aligned with the velocity vector. This transformation is performed using a simple rotation matrix, which adjusts the orientation of the spacecraft to ensure that thrust is applied in the correct direction relative to its motion.

This series of transformations allows the simulation to accurately model the application of thrust in the desired direction, overcoming the limitations of the simulation framework and ensuring that the spacecraft's maneuvers are both realistic and precisely controlled.

Iterative Propagation Setup

With all the essential ingredients of the transfer model in place, namely the environmental setup, satellite properties, and thrust and acceleration models, the transfer model proceeds to simulate the servicer's trajectory through an iterative propagation process. This process is key to modeling the sequential phases of the transfer, where each phase represents either a coasting or a thrusting period. In total, there are seven separate propagations performed back-to-back.

During each iteration, the model performs the following tasks: First of all, the program reinitializes the servicer's mass and state. At the start of each phase, the servicer mass is reset to its initial value, and both the state vectors of the servicer and target are updated based on the final state of the previous phase. This ensures that the propagation reflects the correct initial conditions for each subsequent segment of the mission. Next, depending on the current phase, the thrust magnitude and direction are recalculated. These updated values must be incorporated into the propagation settings, so the integration and propagation settings are also updated during each iteration. Finally, for each phase, the model establishes termination conditions that determine when the propagation should end. These conditions could be based on reaching a specific altitude, exhausting a certain amount of fuel, or simply the passage of time. The flexibility in termination criteria allows the simulation to accurately represent different mission scenarios, ensuring that propagation only continues while the conditions for that phase are met.

3.1.2. Validation of the environment settings

To ensure the accuracy of the propagation environment selected for the servicer's mission, a validation process was undertaken. This process involved comparing the propagated states from a single propagation to actual ephemeris data of a satellite in SSO, operating at approximately the same altitude where the servicer will be deployed. The ephemeris data was provided by Dawn Aerospace, with whom monthly meetings were held to keep the project relevant to the industry.

The validation process necessitated careful tuning of the environment settings to closely match the real-world conditions experienced by the satellite. After extensive adjustments and refinements, the following settings were chosen for the propagation environment.

- **Point Mass of the Sun:** The gravitational influence of the Sun was modeled as a point mass, contributing to the perturbations experienced by the satellite.
- **Spherical Harmonics of the Earth (Degree and Order 10):** The Earth's gravitational field was represented using spherical harmonics up to degree and order 10. This provided a detailed approximation of the Earth's non-uniform gravitational field, which is crucial for accurately simulating satellite orbits in LEO and SSO.
- **Radiation and Aerodynamic Pressure:** Both radiation pressure from the Sun and aerodynamic drag were included in the model. These forces play a significant role in the orbital evolution of satellites, especially in SSO where sunlight and atmospheric drag are prominent factors. The reference area of the satellite was assumed to be square, and its dimensions were estimated from images of the reference satellite taken before launch. These dimensions were crucial for calculating the effects of radiation and aerodynamic pressure on the satellite's trajectory.

The inclusion of these settings result in an as accurate as possible propagation result, without introducing too much insignificant complexity. That is, no significant gain in propagation accuracy was found when the accuracy environment models were included.

Using the first epoch of the satellite's recorded data as the initial state for propagation, the simulation was run under the chosen environmental settings. The propagated states were then compared to the actual ephemeris data provided by Dawn Aerospace. The difference between the propagated state and the actual state over time is illustrated in Figure 3.2. It should be noted that the ephemeris data was provided only for short time intervals, which is why the ’kinks’ are present in the graph.

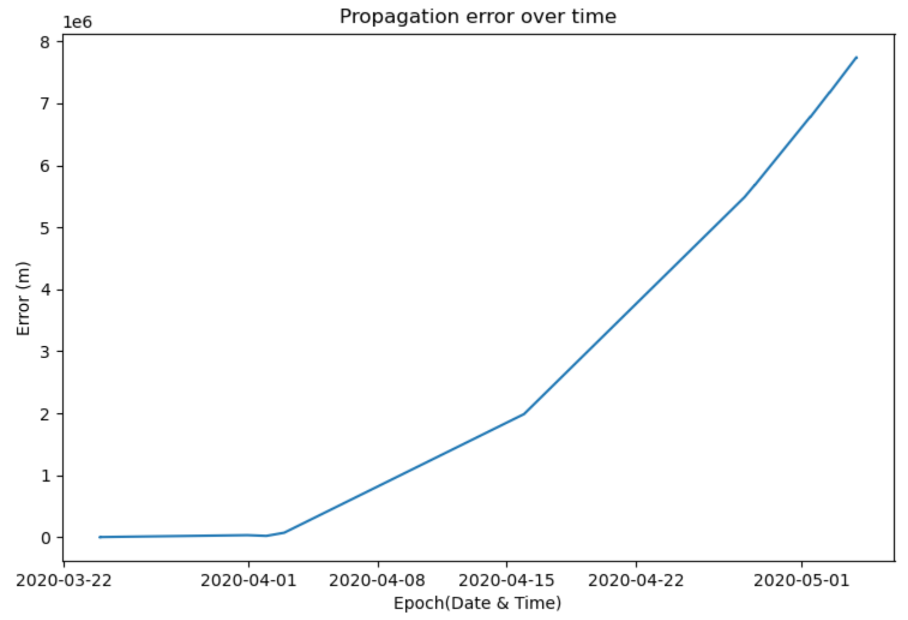


Figure 3.2: State difference as a function of time between the actual ephemeris data and the propagated state.

Zooming in on the first two time intervals gives Figure 3.3 below.

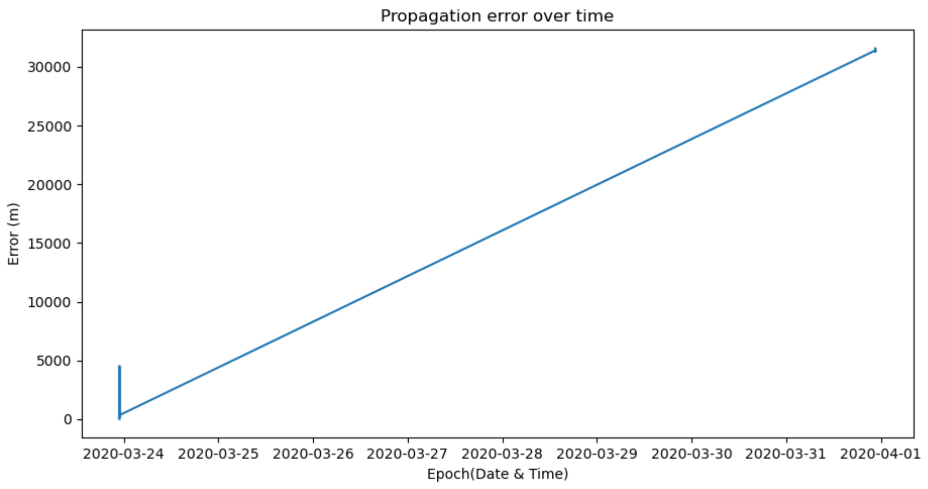


Figure 3.3: State difference as a function of time between the actual ephemeris data and the propagated state, zoomed in on the first two intervals.

As shown in this figure, over a period of approximately 8 days, the difference between the propagated state and the actual satellite state grew to about thirty kilometers. Although a discrepancy of thirty kilometers might seem substantial, it is important to consider several factors that influence this outcome.

First, at the altitude where the servicer will operate, atmospheric drag is quite substantial and exhibits significant temporal variations. These fluctuations are driven by changes in solar activity, atmospheric density, and other environmental factors that are inherently difficult to predict with precision

[17]. As a result, it is impossible to perfectly match the orbit over an extended period solely through propagation. Even the most carefully tuned model will encounter limitations due to these unpredictable atmospheric conditions.

Secondly, during actual satellite operations, minor adjustments and corrections are continuously made to refine the orbit, especially during orbit matchng. Therefore, achieving a propagated state within 30 kilometers of the target satellite after 8 days, without any in-flight corrections, is considered acceptable because the burns that are required to correct such errors are negligible compared to the transfer insertion burn.

Furthermore, interpreting a state difference that extends more than a thousand kilometers after more than a month of propagation, as illustrated in Figure 3.2, can be somewhat misleading. A more insightful approach is to examine the differences in the Keplerian elements between the ephemeris and the propagated state. These results are presented in Figure 3.4.

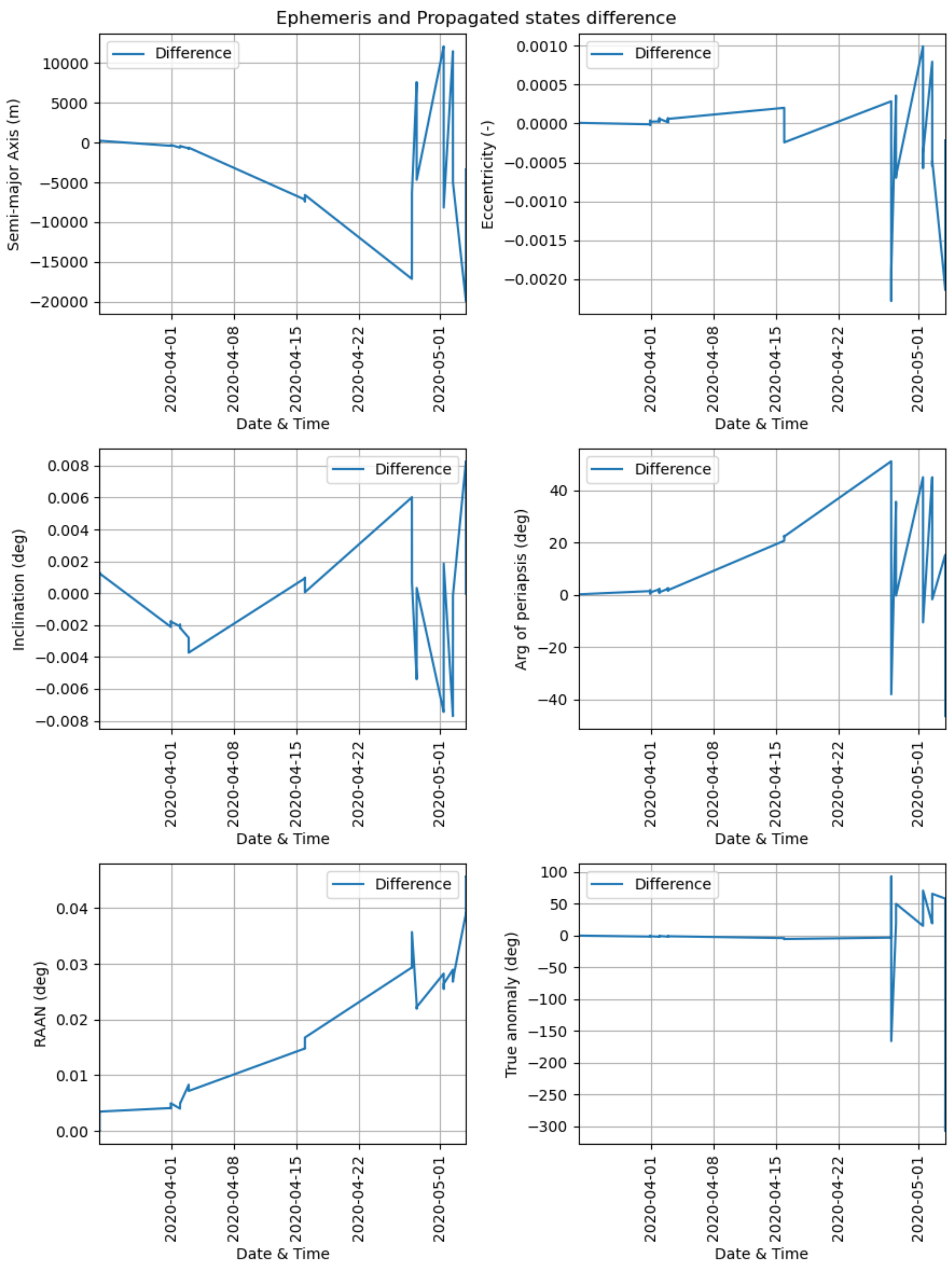


Figure 3.4: Difference in Kepler elements between the actual ephemeris data and the propagated state of the satellite as a function of time

This figure reveals that the large state error is primarily due to a phase difference between the two datasets. Since phase differences can be corrected with minimal fuel expenditure when sufficient time is available, this error is not particularly concerning. However, what is critical is the observation that the differences in inclination and right ascension of the ascending node (RAAN) are minimal. These parameters are crucial because correcting for significant errors in them would require a substantial amount of fuel.

Finally, the accuracy of the propagation model was further verified by the contact person at Dawn Aerospace. Their review and approval of the model provide additional confidence in the chosen environment settings and in the overall reliability of the simulation for mission planning and execution.

This validation process underscores the robustness of the selected environment settings and confirms that the propagation model is capable of simulating the servicer's trajectory to an acceptable accuracy, despite the inherent challenges posed by atmospheric drag and other environmental factors.

At this stage of the process, all necessary physical models and environmental settings have been incorporated into the transfer model, providing a framework to simulate the transfer. However, to complete the simulation, it must be addressed how the servicer's equations of motion are integrated over time and how the state propagation is handled numerically. These critical steps involve selecting an appropriate integrator and propagator, which will be discussed in the following subsection.

3.1.3. Determination of integration and propagation settings

In numerical simulations of orbital mechanics, the accuracy of the results and the efficiency of the computations are crucial to achieving reliable outcomes within reasonable time frames. The selection of an appropriate combination of propagator and integrator is a key factor in balancing these two often conflicting requirements.

The propagator dictates how the state of an orbiting body is updated over time, considering the forces acting upon it, while the integrator determines how the underlying differential equations are numerically solved. Different propagators are suited for different dynamical environments and orbital regimes, and their performance can vary widely depending on the specific characteristics of the orbit being modeled. Similarly, the choice of integrator influences the precision of the numerical solution and the amount of computational resources required to achieve the desired level of accuracy.

A more accurate propagator and integrator combination can yield precise results, but this often comes at the cost of increased computational complexity, leading to longer run times and higher demands on computing resources. Conversely, a more efficient combination may reduce computation time but at the risk of introducing larger errors, which can accumulate and significantly affect the accuracy of the simulation's outcome.

Thus, the selection of a suitable propagator and integrator is a trade-off between computational load and accuracy. The optimal combination depends on the specific goals of the simulation, the acceptable error margins, and the available computational resources. In this chapter, a comprehensive analysis is conducted to evaluate various combinations of propagator and integrator. The goal is to identify a combination that meets the accuracy requirements set by previous validation results while maintaining computational efficiency. This is particularly important in scenarios where multiple simulations need to be performed, such as in parameter studies or Monte Carlo analyses, where the total computational cost can become a limiting factor.

Propagators

As mentioned above, propagators are mathematical models that describe how an object's state evolves over time under the influence of forces such as gravity. The choice of a propagator depends on the specific dynamics of the system being modeled and the desired level of accuracy. Tudat-Py offers a total of seven different propagators, which are listed below:

- **Cowell's Method:** This method directly integrates the equations of motion, considering all perturbative forces acting on the object. It is versatile but can be computationally expensive, especially when modeling complex dynamical systems.
- **Encke's Method:** This method propagates the difference between a reference trajectory (typically a Keplerian orbit) and the actual trajectory. It is particularly useful for cases where the orbit is nearly Keplerian, as it reduces numerical errors by focusing on the smaller deviations.
- **Gauss-Keplerian:** This propagator is based on the Gauss variational equations, which provide a way to describe the evolution of orbital elements over time. It is useful for understanding how perturbations affect the shape and orientation of the orbit.
- **Gauss-Modified Equinoctial:** An extension of the Gauss method, this propagator uses modified equinoctial elements, which avoid singularities at zero eccentricity and inclination. This makes it suitable for a wider range of orbital configurations.
- **Unified State Model with Quaternions:** This method uses quaternions and velocity hodographs to propagate orbits, offering singularity-free rotation representation and improved numerical sta-

bility. It is particularly effective for low-thrust propulsion scenarios, providing higher accuracy and lower computational costs than Cartesian coordinates.

- **Unified State Model with Modified Rodrigues Parameters:** By replacing quaternions with Modified Rodrigues Parameters, this method reduces the state elements to six and enhances computational efficiency.
- **Unified State Model with Exponential Map:** This approach uses a three-parameter exponential map for rotations, avoiding normalization and providing high accuracy in orbit propagation. It excels in scenarios with continuous low-thrust propulsion and complex perturbations, making it highly robust for precise orbital simulations.

Given the high dynamic nature of the transfer model, characterized by multiple high-thrust phases, it is expected that the three unified state models do not perform well, as they are most suitable for low-thrust scenarios [18]. Additionally, because Encke's method is based on modeling the perturbations, it is only more accurate than Cowell's method in case the perturbing accelerations are much less than the acceleration of the central body [19]. Since the servicer uses high-thrust, Encke's method is also expected to not perform well. Finally, because the Gauss-Keplerian method becomes singular for circular orbits, this method is also not deemed suitable for this case [20].

Therefore, the Cowell and Gauss modified equinoctial propagators are expected to be the most promising propagators.

Integrator

An integrator is a numerical method that is used to solve the differential equations governing the system's dynamics. The choice of integrator affects the balance between computational efficiency and accuracy. The eight different types of integrator that Tudat-Py provides are listed below [21, 22].

- **Runge-Kutta Fixed Step:** A family of explicit integrators where the state is advanced using fixed time steps. This method is straightforward and widely used but requires careful selection of the step size to balance accuracy and computational load.
- **Runge-Kutta Variable Step:** Similar to the fixed-step version but with adaptive step sizes. This method adjusts the step size dynamically based on the local error estimate, offering better efficiency by taking larger steps when possible and smaller steps when necessary.
- **Bulirsch-Stoer Fixed Step:** An extrapolation method that improves accuracy by combining results from multiple fixed step sizes. It is particularly effective for stiff equations but can be computationally intensive.
- **Bulirsch-Stoer Variable Step:** This variant of the Bulirsch-Stoer method uses adaptive step sizes to further improve efficiency while maintaining accuracy, especially in cases with highly variable dynamics.
- **Adams-Bashforth-Moulton:** A predictor-corrector method where the next state is first predicted using previous states (Adams-Bashforth) and then corrected using the current state (Adams-Moulton). It is well-suited for smooth, non-stiff problems.
- **Adams-Bashforth-Moulton Fixed Order:** A variation of the above method where the order of the predictor-corrector pair is fixed, offering a balance between complexity and performance.
- **Adams-Bashforth-Moulton Fixed Step:** Another variant where both the order and the step size are fixed. This method is less flexible but can be optimized for specific applications.
- **Adams-Bashforth-Moulton Fixed Step Fixed Order:** The most constrained variant, where both the step size and the order are fixed. This method is the least flexible but can be efficient when the dynamics are well-understood and do not vary significantly.

The Runge-Kutta variable step integrator is expected to perform best for the transfer model due to its ability to handle fast and slow dynamics while maintaining accuracy during a long integration period [23]. It is important to note that there are several coefficient sets available for Runge-Kutta integrators. In summary, the selection of a coefficient set involves a trade-off between computational complexity and accuracy: larger coefficient sets provide greater accuracy for larger time steps, but they also demand more computations per time step.

Combined results

To effectively determine the best-performing combination of propagator, integrator, and integration settings, all possible combinations were implemented and compared against a single baseline. This baseline uses the same environment and simulation setup as the validation conducted in Section 3.1.2 and employs a fixed step Runge-Kutta integrator with the set of RKF78 coefficients and a very small timestep, ensuring that integration errors are practically negligible.

For each combination of propagator, integrator, and settings, the final position error was plotted against the number of computations required. The results are presented in Figure 3.5 below. It should be noted that this figure is only for showing the overall trend between the computational cost and the accuracy of the propagation. Figure 3.6 shows the important region of this plot, including an extensive legend indicating all propagator and integrator settings.

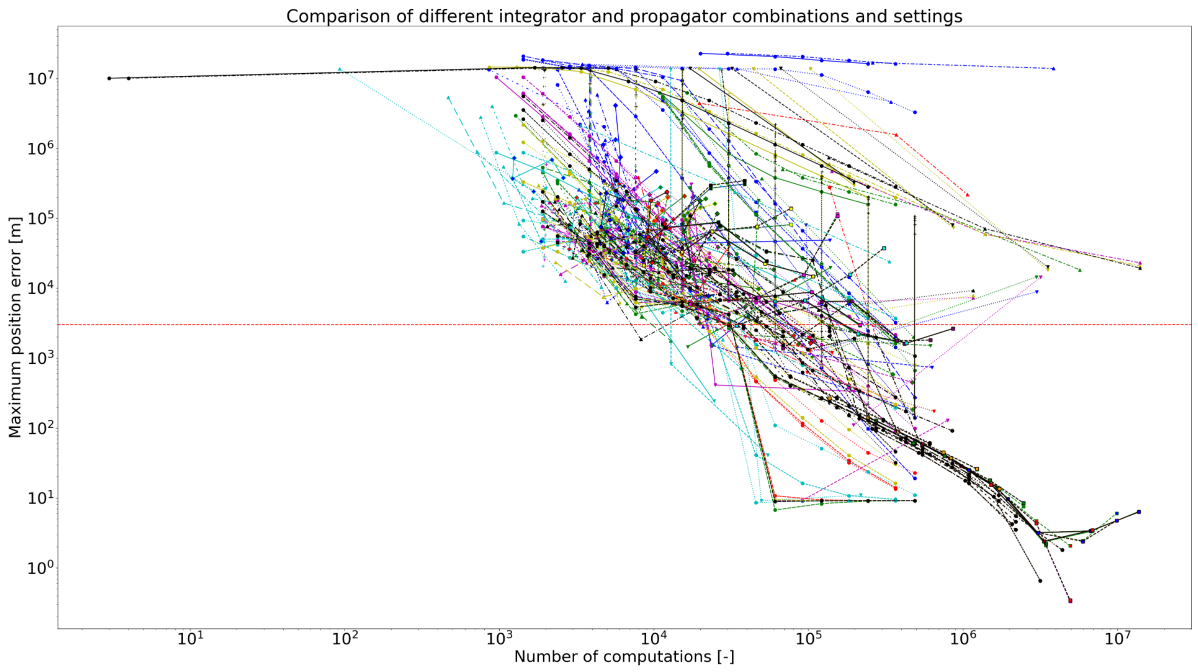


Figure 3.5: Final state error plotted against the computational load for all combinations of propagator, integrator, and settings

As shown in the figure, the data exhibit a diagonal trend, which is as expected since a greater number of computations generally leads to higher accuracy.

In addition, a threshold of 3 km is indicated by the dashed horizontal red line. This value was selected based on the validation results of the simulation model discussed in Section 3.1.2. In this section, the model accuracy was determined to be approximately 30 km after a week of propagation. This value represents the cumulative error introduced by the simulation over time, which includes both modeling inaccuracies and numerical errors from the propagators and integrators used.

To ensure that the choice of propagator and integrator does not introduce a significant additional error, a threshold of an order of magnitude lower was chosen, corresponding to 3 km. This conservative threshold ensures that the propagation method contributes minimally to the overall error budget, preserving the accuracy of the simulation. By setting this threshold, a propagator and integrator combination that provides a good trade-off between computational efficiency and accuracy can be selected.

Zooming in on the section of Figure 3.5 where the first lines intersect the threshold line reveals the details shown in Figure 3.6:

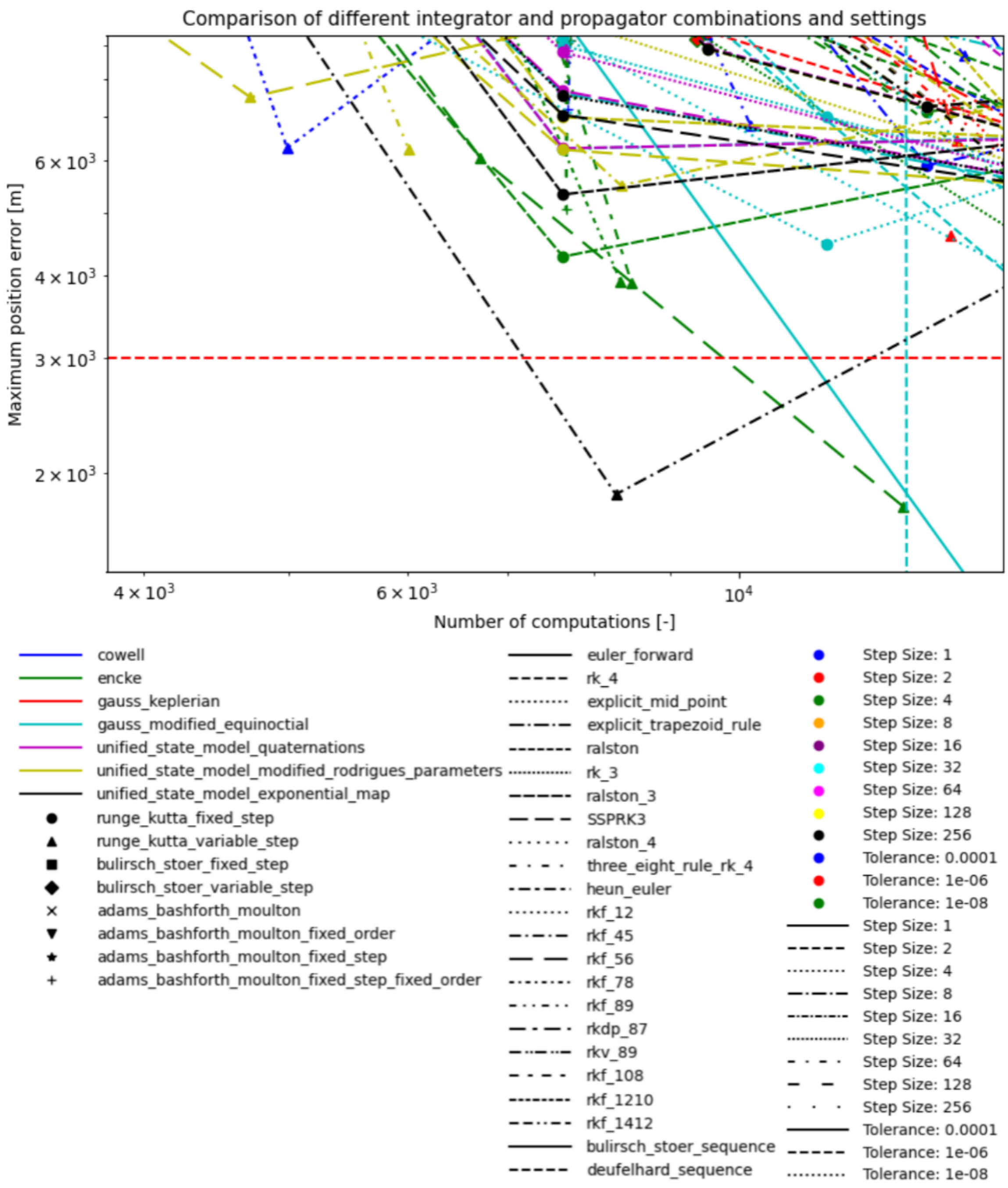


Figure 3.6: Final state error plotted against the computational load for all combinations of propagator, integrator, and settings, zoomed in on the most promising part of the plot

In this figure, the leftmost point below the 3 km threshold is represented by a black triangle. This point corresponds to the following combination of propagator, integrator, and settings:

- **Propagator:** Unified State Model Exponential Map
- **Integrator:** Runge-Kutta variable step
- **Setting:** Runge-Kutta-Fehlberg 4(5)

Although this appears to be the most suitable choice, it is important to note that this analysis does not account for the presence of thrust. The Unified State Model Exponential Map propagator may not perform optimally when thrust is introduced, making this combination potentially suboptimal. Similarly, the three Encke propagator points located just above the threshold line are also considered infeasible for the same reasons.

The most suitable option, therefore, is the blue triangle in the top left of Figure 3.6, which corresponds to the following combination of propagator, integrator, and integration settings:

- **Propagator:** Cowell
- **Integrator:** Runge-Kutta variable step
- **Setting:** Runge-Kutta-Fehlberg 7(8)

This choice aligns with the expectations presented in the literature, as discussed in Section 3.1.3. Although the final state error exceeds the previously set threshold of 3 km, this combination is selected due to the reliability of the propagator and integrator, as well as its computational efficiency while maintaining a reasonable accuracy that is approximately five times better than the accuracy achieved in the environment validation discussed in Section 3.1.2.

3.1.4. Optimizer selection

Now that the transfer model has been fully set up and validated, the optimization itself can be set up. The library that will be used for this is called Pygmo, which is an in Python wrapped version of the original package called Pagmo [24]. Pygmo provides a variety of different optimization algorithms for single- and multi-objective problems. Since the transfer model that is set up in Section 3.1 has three objectives, only multi-objective optimizers can be used. The five algorithms provided by Pygmo are:

- Multi-Objective Evolutionary Algorithm with Decomposition (MOEA/D)
- Multi-Objective Evolutionary Algorithm with Decomposition Generational (MOEA/D-Gen)
- Non-dominated Sorting Genetic Algorithm II (NSGA-II)
- Multi-Objective Hypervolume-based Ant Colony Optimization (MACO)
- Non-dominated Sorting Particle Swarm Optimization (NSPSO)

Determining which algorithm is best for the transfer model will be done in a similar way to the one used to determine the combination of propagator, integrator, and integrator settings in Section 3.1.3. Ideally, for each optimization algorithm, multiple combinations of optimization settings should be analyzed as well in order to make a confident selection for the best optimizer and settings combination. However, since the propagation of a single transfer takes a couple of seconds, even with parallel computation, a single optimization run to a decent level of convergence takes over an hour. Therefore, it is infeasible and out of the scope of this thesis to perform a thorough optimization analysis.

Fortunately, for this research it is sufficient to find an optimizer that functions reliably and shows decent levels of convergence rates. To determine which optimizer is the most suitable in this case, the five optimizers are analyzed with their standard settings.

However, before this analysis can be performed, two final decisions must be made. The limits of the decision variables must be set, and a cost function for the level of orbit matching must be formulated. Determining the limits values of the decision variables is not difficult. Six of the twelve decision variables are the three polar coordinate pairs, which by definition have limits from zero to 2π and from zero to π. Moreover, the minimal duration of a coasting phase or a burn cannot be less than zero seconds. The upper limits of these values are somewhat debatable and are discussed individually below.

- One might be inclined to assume that it might be beneficial to have a long initial coasting phase, as this could cause the two satellites to move closer together by themselves due to their orbits moving closer together. However, as all satellites that are considered are in SSO, they will per definition not move closer together, or at least not at a significant rate. Therefore, it is best to initialize the transfer as soon as possible. The maximum time for the initial coasting phase is therefore set at about one orbital period, or 6000 seconds.
- The duration of the second coast phase, which is the transfer time required to reach the targets orbit, can be a lot longer. However, based on the expected lifetime of the relevant missions [17], reaching a target within a couple of weeks is set to be the requirement. Therefore, the second coasting phase limit is set to one month.

- The duration of the third coasting phase, the coast between the orbit raising burn and the final insertion burn, is also quite set in stone. This is because the orbit raising burn is planned to be performed half an orbit before the final insertion burn. Therefore, the duration of this final coasting phase is set to 4000 seconds which is slightly more than half an orbital period at the operational altitude.
- Finally, the duration of the three burns must be determined. From Equation 3.20, which will be further discussed in Section 3.2, the burn time can be determined for a specific amount of fuel that is being burned. The first and last burns can be set to these values, as the model will terminate anyways before reaching higher burn times. The duration of the second burn, however, can be made shorter because it is only a small burn for increasing or decreasing the semi-major axis, and is therefore set to a maximum of 100 seconds.

Finally, the cost function must be determined to determine the level of orbit matching. An initial thought one might have is to simply take the final position error as was also done in the analysis of the propagation model in Section 3.1. However, this leads to the possibility that the algorithm does bring the position error to zero, but this does not guarantee that the velocities of both satellites are matched. Therefore, the velocity difference must also be taken into account in some way. This can be done in multiple ways. The most logical way of achieving this is to also calculate the final three-dimensional velocity difference between the two satellites and add this to the three-dimensional position error. Because both the position and velocity have different units and scales, a scaling factor must be introduced such that both errors have about the same effect. Since the radii of the considered orbits are approximately 7000 kilometers, and the velocity of the satellites are about 7 kilometers per second, the scaling factor is chosen to be 1000.

Another method would be to integrate the position error over time for some time after the final insertion burn. The benefit of this method is that it eliminates the requirement of a scaling factor.

The final method would be to set up a cost function based on the difference in Kepler elements of both satellites. However, this would require multiple scaling factors, as all Kepler elements have different scales.

Because no assumptions are made regarding a scaler factor in the second method, this method is chosen. A disadvantage of this method seems to be that it requires an extra propagation period after the orbits have been matched. However, this extra propagation is required anyways due to the rendezvous and proximity operations that take time as well.

After these final properties have been established, the transfer can be optimized by all five algorithms. The initial states of the servicer and the target were taken from two-line element (TLE) data from two satellites that were launched on the Transporter-9 mission from Space-X. The results of the optimization are shown in Figures 3.7 and 3.8 below:

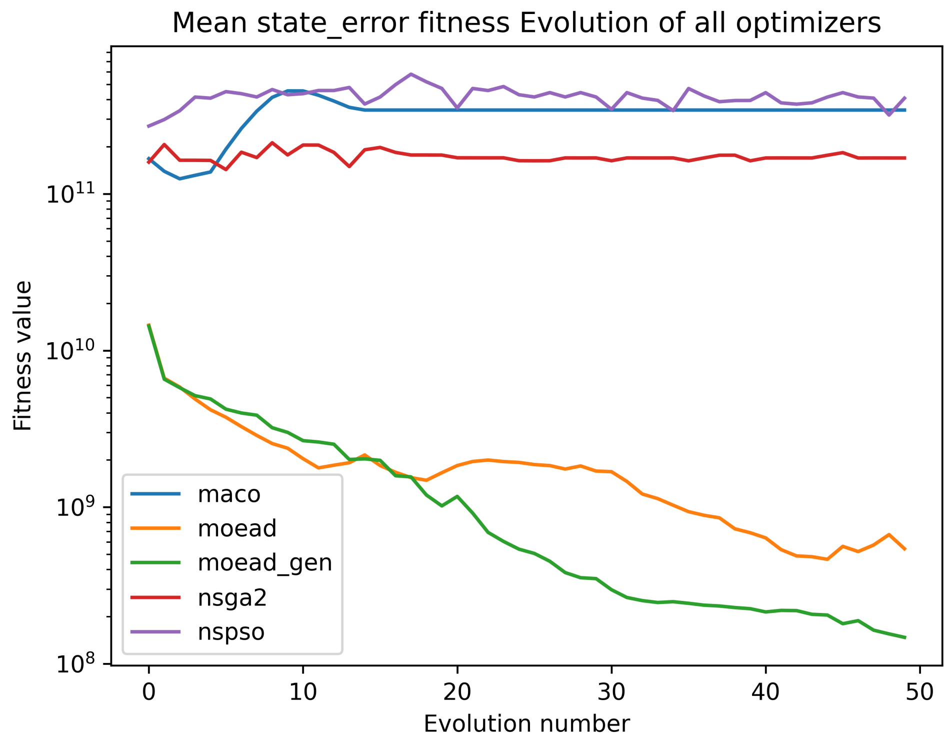


Figure 3.7: Mean integrated position fitness of the population as a function of the number of evolutions for all optimization algorithms.

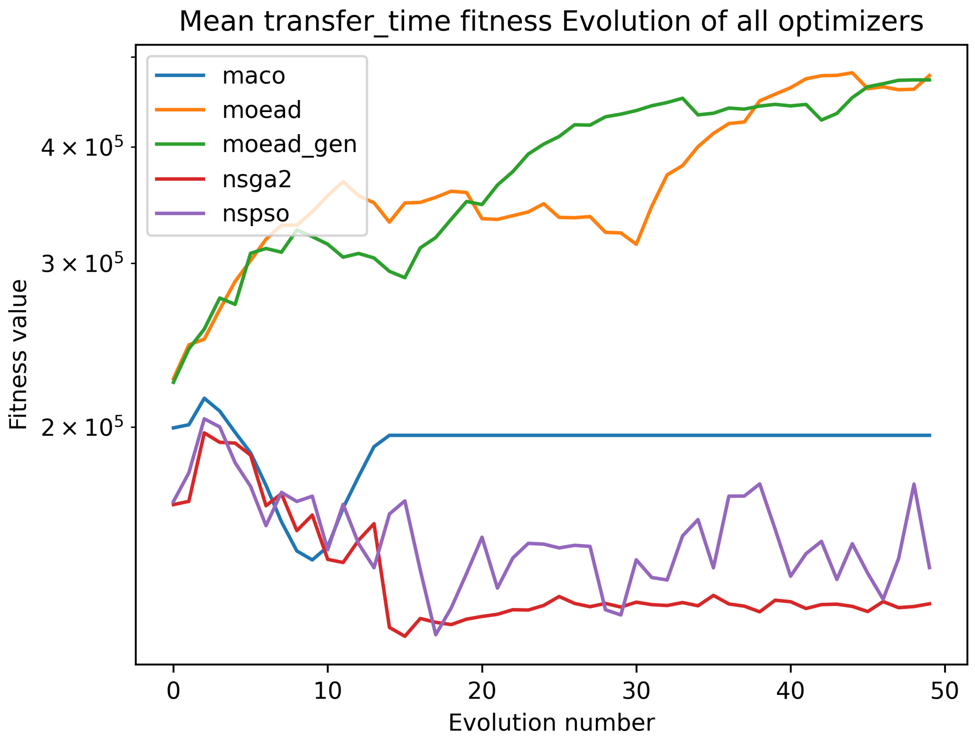


Figure 3.8: Mean transfer time fitness of the population as a function of the number of evolutions for all optimization algorithms.

In Figure 3.7 it is observed that both of the MOEA/D optimizers are the only two that noticeably decrease the state error fitness value as the number of generations increases. In contrast, Figure 3.8 shows that these are the only two optimizers that show a worsening behavior as the number of generations increases, and the NSGA-II, MACO, and NSPSO algorithms even show a slight decrease in the transfer-time fitness value as the number of generations increases.

This can be explained by the fact that NSGA-II, MACO, and NSPSO are all not able to decrease the state error fitness, as indicated in Figure 3.7, but are capable of decreasing the travel time fitness. For both MOEA/D optimizers, which are actually able to decrease the mean value of the state error fitness, it is clear that to achieve this, they must increase the mean transfer time fitness.

In conclusion, based on the more important state error fitness value shown in Figure 3.7 - it does not matter how fast the transfer is if the orbits are not matched - the MOEA/D-Gen algorithm is chosen to be the most suitable optimization algorithm. In addition, for the sake of potential tuning of the algorithm, this is also ideal, since this algorithm has no parameters to tune.

3.1.5. Initial results

Now that the optimizer has been chosen, the optimization can be run for a longer period such that it can fully converge. The results of this optimization are shown in Figure 3.9 below:

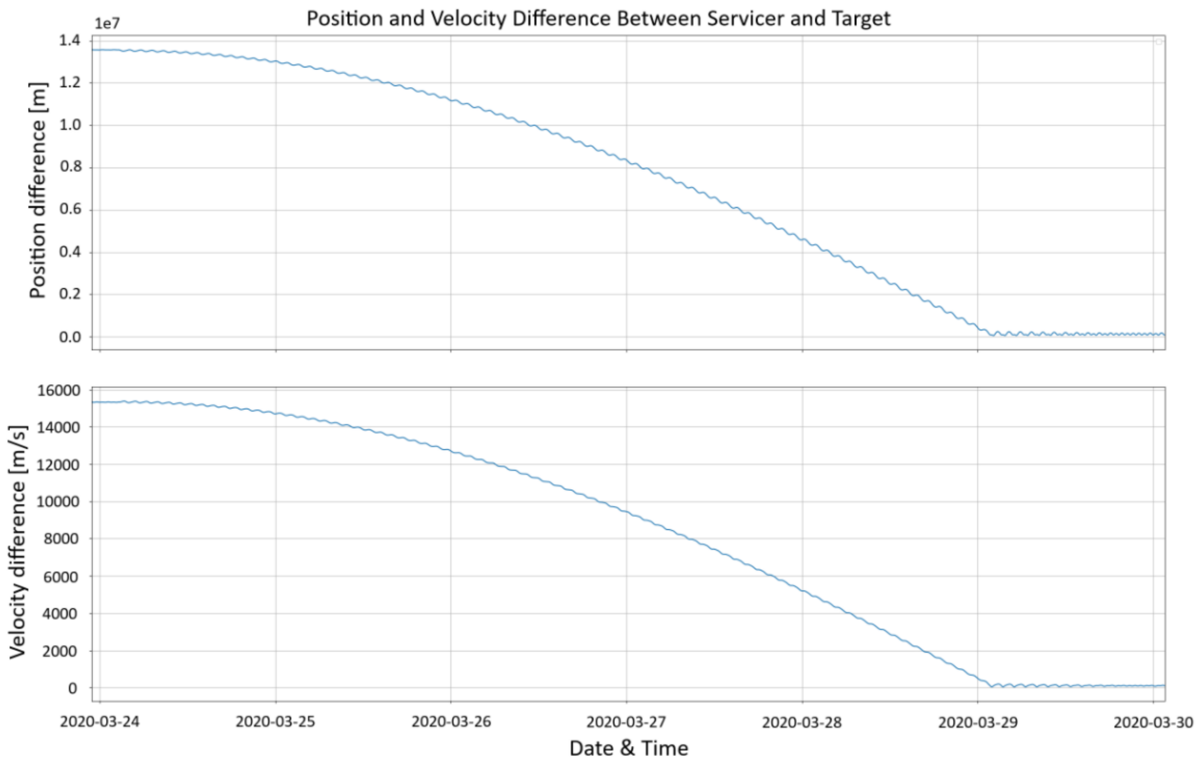


Figure 3.9: Position and velocity error as a function of time

Zooming in on the right side of the figure where the orbit is 'matched', results in Figure 3.10 below.

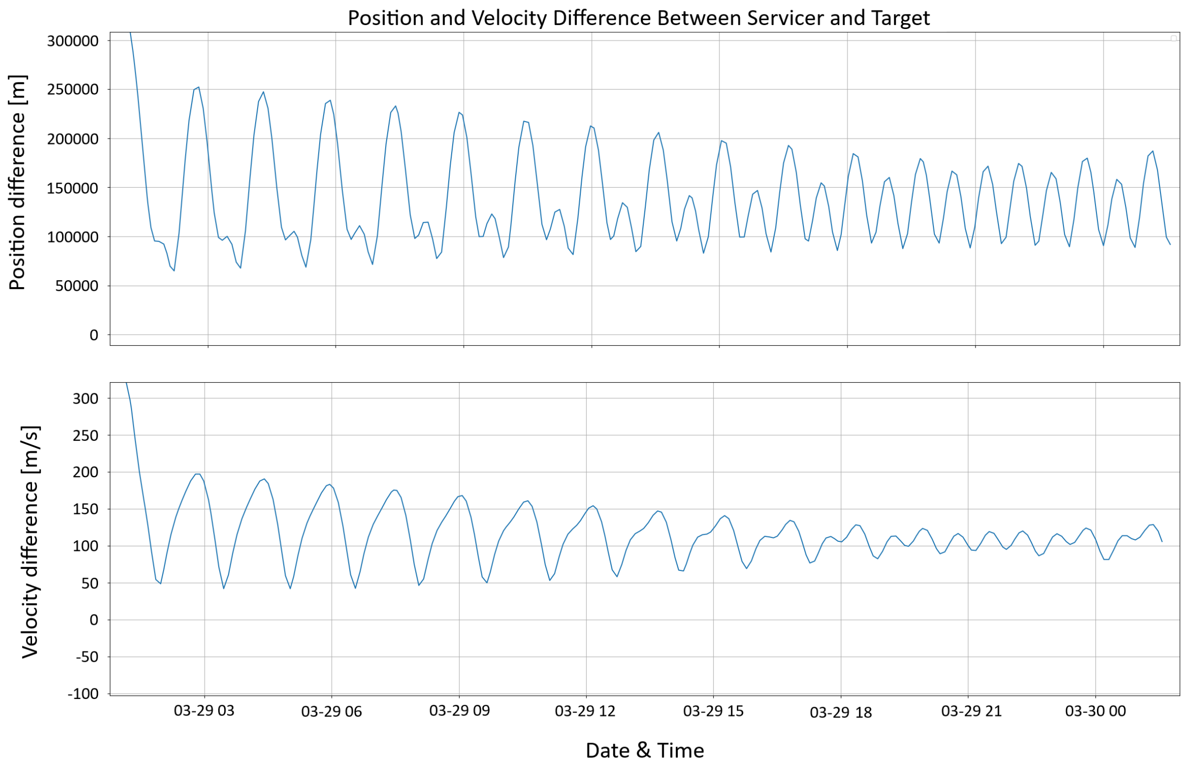


Figure 3.10: Mean transfer time fitness of the population as a function of the number of evolutions for all optimization algorithms.

As can be seen from this figure, the servicer has definitely moved closer to the target satellite. Unfortunately, they are still far from the required 10 kilometer position error that was provided by Dawn Aerospace as a reasonable distance to start the rendezvous, proximity operations, and docking (RPOD). To find out why this occurs, the Kepler elements of both satellites as a function of time are plotted and shown below in Figure 3.11.

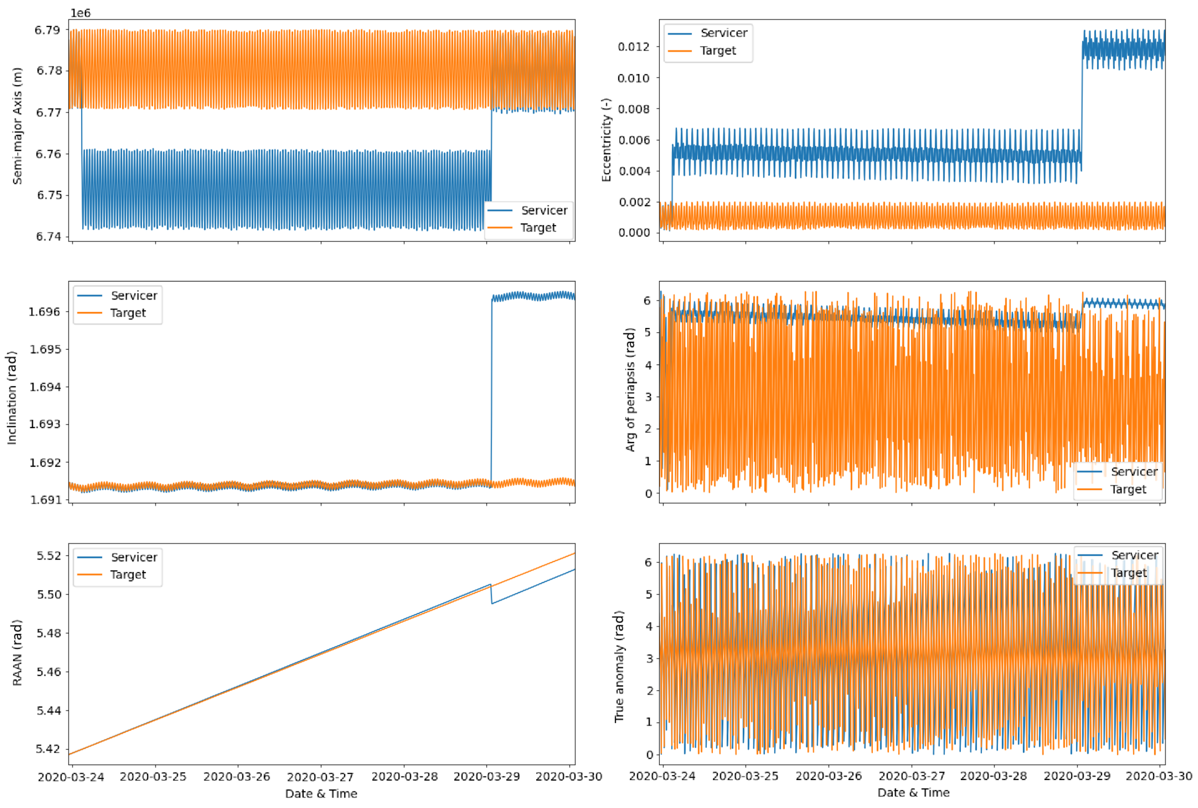


Figure 3.11: Kepler elements of both the servicer and the target as a function of time.

From this figure, it is observed that the semi-major axis is properly matched, but the inclination and the RAAN are not. Unfortunately, what is happening is that the optimization is getting stuck in a local minimum and does not explore the design space sufficiently enough to find the global minimum. After experimenting with many different combinations of cost functions and penalties, it was concluded that the objective space of this transfer optimization is simply too large and irregular for the optimization algorithm to reliably converge to a global minimum. A qualitative comparison of such an irregular objective space is visualized for only two decision variables below in Figure 3.12, which shows the many local minima the problem has.

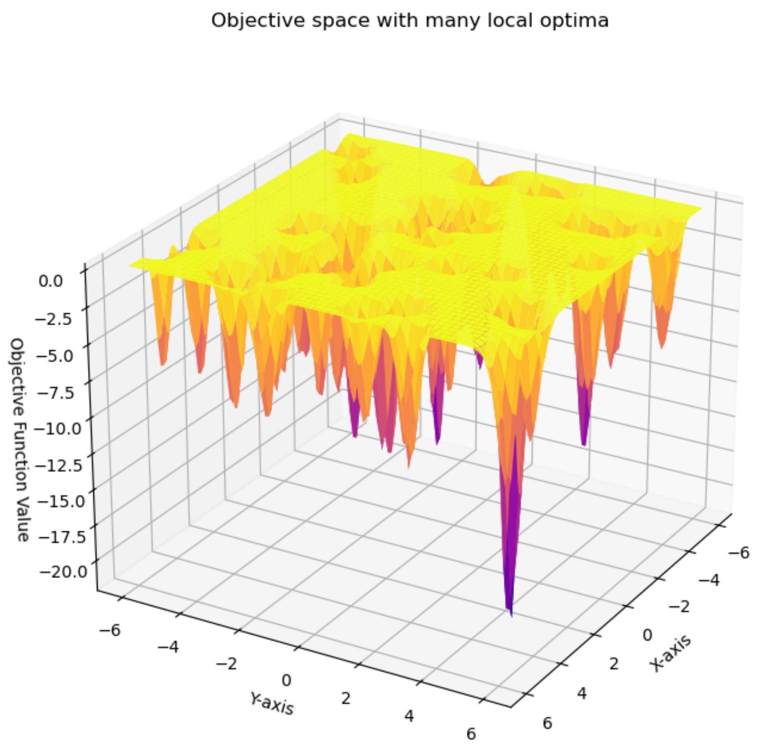


Figure 3.12: Qualitative example of an irregular objective space

To work around this problem of the optimization becoming stuck in a local optimum, use can be made of the underlying physics of the transfer. By doing so, it is realized that the approximate locations of all global minima can be analytically calculated, which is further explained in the next section.

3.2. Local Optimization

In the previous section, the optimization was performed in the entire physically possible decision space, where the only constraints were imposed by the maximum transfer time allowed and the fuel mass of the servicer satellite. In the following calculations, several simplifying assumptions are made to facilitate the analysis of orbital dynamics. The vis-viva equation is employed under the assumption of a Keplerian orbit, which treats the Earth as a point mass and assumes the satellite's motion is solely influenced by Earth's gravity. This idealized approach neglects perturbative forces such as atmospheric drag and higher-order gravitational effects above the J2 term, focusing instead on the primary gravitational interaction between the Earth and the satellite.

Furthermore, orbital maneuvers are modeled as impulsive shots, where velocity changes are considered instantaneous. This assumption simplifies the computation of orbital transfers, though it introduces minor discrepancies compared to the gradual velocity change experienced during real-world propulsion maneuvers.

Finally, since the orbital transfers considered rely on exploiting the Earth's oblateness to passively alter the relative RAAN position, the J2 effect of Earth's gravity field is incorporated into the equations. All other degrees and orders of the spherical harmonics of the Earth's gravity field are omitted. The inclusion of the J2 term provides a more accurate representation of the Earth's gravitational influence by acknowledging the role of its non-spherical shape.

These assumptions provide a foundation for deriving the initial orbital parameters, with the understanding that further refinements may be necessary to achieve the precision required for mission-specific planning, which will be performed by the optimization algorithm described in Section 3.1

3.2.1. Determination of initial guesses

The goal of this section is to obtain a method that structurally finds all possible transfer trajectories between a servicer and a target that could be in any orbit. The most important realization for this is that all satellites move in ellipses around Earth, and when they need to transfer between two different orbits, only the shape and orientation of this ellipse need to be changed. The shape of this ellipse is defined by the following six kepler elements: Semi-major axis a, eccentricity e, inclination i, argument of periapsis ω, right ascension of the ascending node Ω, and the true anomaly θ which are all shown in Figure 3.13 below.

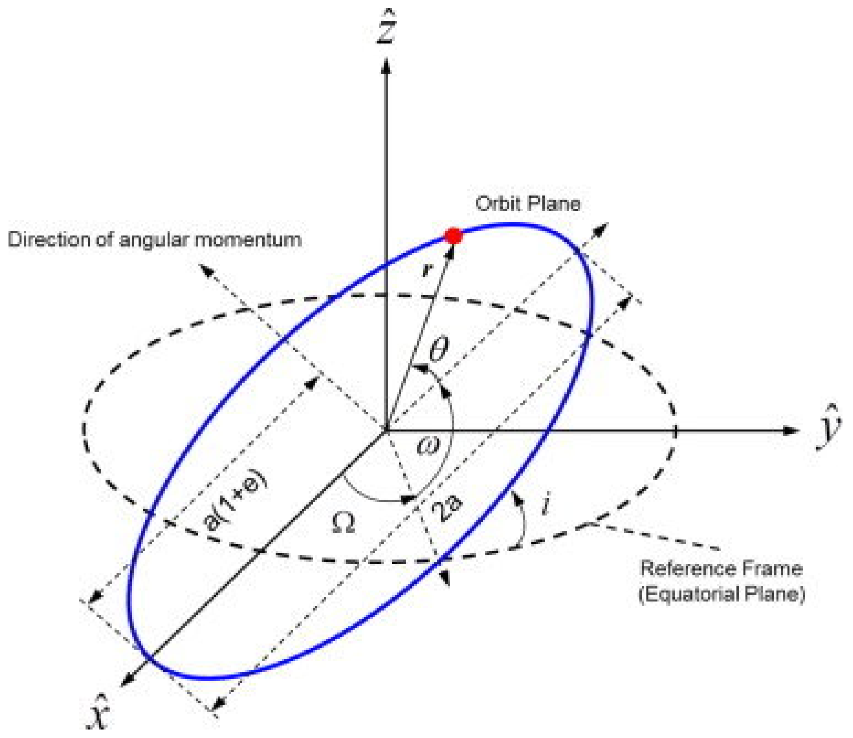


Figure 3.13: Kepler elements [25]

Matching an orbit actually corresponds to changing the Kepler elements of the orbit such that all values are equal to the values from the target orbit. Although all parameters could theoretically be changed continuously by firing the thrusters in any direction, in practice only burns that are along track or out of the orbital plane are performed because these result in the highest efficiency [20]. When all Kepler elements except for the true anomaly are matched, only a change in the so-called phase of the orbit is required. This is depicted in figure 3.14 below.

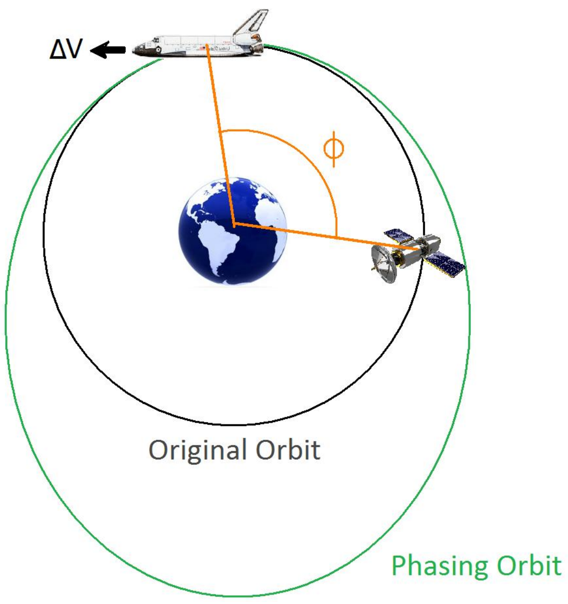


Figure 3.14: Schematic overview of a phase change required for orbital rendezvous [26]

In this figure, if the Shuttle increases its semi-major axis by just the right amount, which in turn increases its orbital period, the phase difference (depicted in this figure as Φ) can be made precisely zero, meaning that the Shuttle is now close to the satellite. The key observation to make here is that this orbital phasing occurs in an integer amount of orbits, which means that within a specified maximum transfer time, there is a limited number of transfers possible. This fact can be exploited to find all possible transfers between the servicer and a target within a certain time frame.

The strategy will go as follows: per number of phasing orbits required for the phase matching, it is calculated what the RAAN drift rate must be in order to arrive at the target RAAN in the same amount of time as the phasing takes. This drift rate can subsequently be directly related to the transfer inclination. It should be taken into account that the final half-orbit of the transfer is either raised or lowered so that the semi-major axes of the orbits can be matched. From these calculations, the required change in velocity, or ΔV, can be calculated. This, in turn, can be directly linked to the amount of fuel that is required for the total transfer. The final calculation that must be performed is to determine what the actual decision variables are.

Phasing

The transfer strategy described above is called a bi-elliptic transfer due to the fact that the spacecraft is put into two elliptical orbits before it reaches its desired final orbit. A schematic of such a transfer is shown in figure 3.15 below.

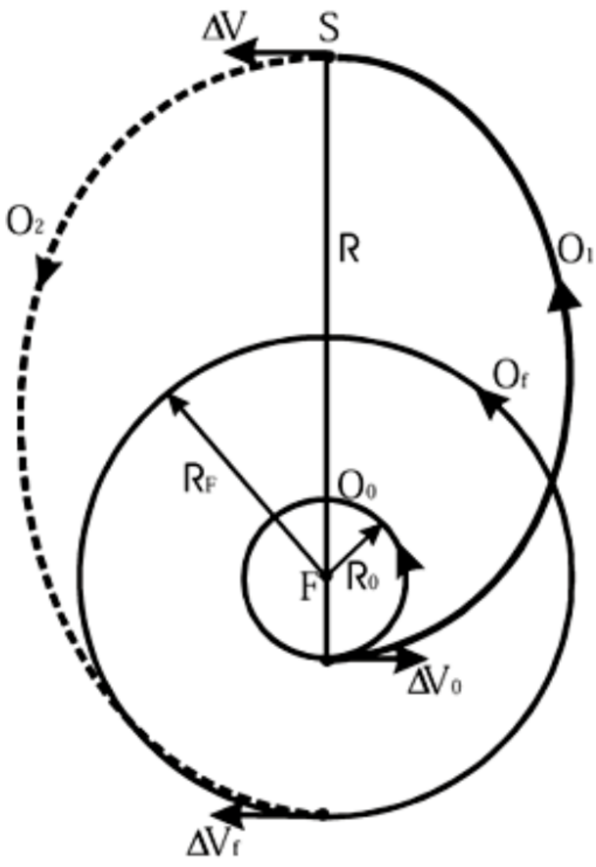


Figure 3.15: Bi-elliptic transfer [27]

For a specific number of phasing orbits O_1, the goal is to find the distance R such that the two spacecraft precisely meet after the set amount of transfer orbits. Starting from Equation 3.1, which gives the mean anomaly M as a function of time for both spacecraft:

$$\begin{aligned} M_S &= M_{S,0} + \frac{dM_S}{dt} \cdot t \\ M_T &= M_{T,0} + \frac{dM_T}{dt} \cdot t \end{aligned} \tag{3.1}$$

Where the subscripts S and T are for the servicer and target, M_0 is the initial mean anomaly of the spacecraft relative to the argument of periapsis, $\frac{dM}{dt} = n = \sqrt{\frac{\mu}{a^3}}$ is the mean angular motion and t is the transfer time and $\mu = 3.986 \cdot 10^{14} [m^3 s^{-2}]$ is Earths gravitational parameter . Since the start of the transfer occurs in the argument of periapsis of the initial servicer orbit, $M_{S,0} = 0$. This means that at t_0, $M_{T,0}$ is equal to the difference in the mean anomaly ΔM at that time. It should be noted that ΔM can take two values: one for each direction in which the target lies. So, for example: If the Target is a quarter phase in front of the servicer, it is also three quarter phases behind the servicer.

In order for the phases of both orbits to align after the transfer, the mean anomaly of both equations in Equation 3.1 must be equal to any integer multiple of 2π, since this means that the satellites are at the argument of periapsis, which is where the two trajectories meet. Continuing with the equation for the target results in Equation 3.2.

$$M_T = 2k_T\pi = \Delta M + n_T t \tag{3.2}$$

All of these values are known, except for the time t. Fortunately, this is simply the transfer time of the servicer for which a formula can be derived as a function of R. Since R is directly related to the semi-major axis of the first transfer orbit by $a_1 = \frac{R+R_0}{2}$, and this variable is directly linked to the transfer time, a_1 will be used in the subsequent derivations.

With the period of any elliptical orbit equal to $T = \frac{2\pi}{n}$, for k transfer orbits the transfer time is thus given by Equation 3.3.

$$T_k = 2k\pi\sqrt{\frac{a^3}{\mu}} \tag{3.3}$$

Since the final phasing orbit includes the orbit raising burn, the two halves are are both from different ellipses. Therefore, the total transfer time of the first part of the transfer is actually the sum of the transfer time of k full orbits, plus the transfer time for another half orbit as given by Equation 3.4.

$$T_{k_1} = 2k_1\pi\sqrt{\frac{a_1^3}{\mu}} + \pi\sqrt{\frac{a_1^3}{\mu}} = (2k_1 + 1)\,\pi\sqrt{\frac{a_1^3}{\mu}} \tag{3.4}$$

Now, the transfer time of the final half ellipse is simply given by Equation 3.5.

$$T_2 = \pi\sqrt{\frac{a_2^3}{\mu}} \tag{3.5}$$

Here, the semi-major axis of this second transfer as a function of the semi-major axis of the first transfer is given by Equation 3.6.

$$a_2 = \frac{a_T + 2a_1 - a_0}{2} \tag{3.6}$$

Combining Equations 3.4, 3.5, and 3.6 gives the final formula for the time t as given by Equation 3.7.

$$t = T_{k_1} + T_2 = (2k_1 + 1)\,\pi\sqrt{\frac{a_1^3}{\mu}} + \pi\sqrt{\frac{\left(\frac{a_T + 2a_1 - a_0}{2}\right)^3}{\mu}} \tag{3.7}$$

The substitution of Equation 3.7 into Equation 3.2 gives the relationship between the semi-major axis of the first transfer orbit a_1 and the difference in the mean anomaly ΔM as given by Equation 3.8.

$$2k_T\pi = \Delta M + \sqrt{\frac{\mu}{a_T^3}}\left((2k_1 + 1)\,\pi\sqrt{\frac{a_1^3}{\mu}} + \pi\sqrt{\frac{\left(\frac{a_T + 2a_1 - a_0}{2}\right)^3}{\mu}}\right) \tag{3.8}$$

Unfortunately, this equation cannot be analytically solved for a_1, so a numerical method must be used to estimate its value.

Right Ascension of the Ascending Node and Inclination

Now that the transfer times of the two transfer phases have been determined, the inclination required for the matching of the RAAN of the target orbit can be determined. The derivation starts in a similar way as the previous derivation for a_1:

$$\begin{gathered}\Omega_S = \Omega_{S,0} + \sum_{i=0}^{z}\left(\frac{d\Omega_S}{dt}\right)_i \cdot t_i \\ \Omega_T = \Omega_{T,0} + \frac{d\Omega_T}{dt} \cdot t_{tot}\end{gathered} \tag{3.9}$$

Equation 3.9 gives the RAAN as a function of time, where Ω_0 is the initial RAAN, $\frac{d\Omega}{dt}$ is the rate at which the RAAN changes, also known as the RAAN drift rate, and z is the number of transfer phases. If the initial RAAN of the servicer is defined as zero, the total difference in RAAN $\Delta\Omega$ can be substituted for $\Omega_{T,0}$. After the transfer, the goal is to have the RAAN values of the servicer and the target equal to each other, or $\Omega_S = \Omega_T$. This results in Equation 3.10.

$$\sum_{i=0}^{z}\left(\frac{d\Omega_S}{dt}\right)_i \cdot t_i = \Delta\Omega + \frac{d\Omega_T}{dt} \cdot t_{tot} \tag{3.10}$$

The equation for the RAAN drift due to the oblateness of the Earth is given by Equation 3.11 [20].

$$\frac{d\Omega}{dt} = -\frac{3}{2}J_2\left(\frac{R_e}{a\left(1-e^2\right)}\right)^2\sqrt{\frac{\mu}{a^3}}\cos i \tag{3.11}$$

Here $R_e = 6.371 \cdot 10^6[m]$ is the mean radius of the Earth. The substitution of Equation 3.11 into Equation 3.10 results in Equation 3.12

$$\begin{aligned}\left(-\frac{3}{2}J_2\left(\frac{R_e}{a_1\left(1-e_1^2\right)}\right)^2\sqrt{\frac{\mu}{a_1^3}}\cos i_1\right)t_1 + \left(-\frac{3}{2}J_2\left(\frac{R_e}{a_2\left(1-e_2^2\right)}\right)^2\sqrt{\frac{\mu}{a_2^3}}\cos i_2\right)t_2 \\ = \Delta\Omega + \left(-\frac{3}{2}J_2\left(\frac{R_e}{a_T\left(1-e_T^2\right)}\right)^2\sqrt{\frac{\mu}{a_T^3}}\cos i_T\right)t_{tot}\end{aligned} \tag{3.12}$$

Since the transition between the first and second transfer phases only increases the semi-major axis and does not change the inclination of the transfer, both inclinations can be equated: $i_1 = i_2 = i$. After some rewriting, the final formula for the inclination that is required for a certain $\Delta\Omega$ is given by Equation 3.13.

$$i = \arccos\left(\frac{\Delta\Omega + \left(-\frac{3}{2}J_2\left(\frac{R_e}{a_T(1-e_T^2)}\right)^2\sqrt{\frac{\mu}{a_T^3}}\cos i_T\right)t_{tot}}{\left(-\frac{3}{2}J_2\left(\frac{R_e}{a_1(1-e_1^2)}\right)^2\sqrt{\frac{\mu}{a_1^3}}\right)t_1 + \left(-\frac{3}{2}J_2\left(\frac{R_e}{a_2(1-e_2^2)}\right)^2\sqrt{\frac{\mu}{a_2^3}}\right)t_2}\right) \tag{3.13}$$

It should be noted that the values of the eccentricities e_1, e_2 and e_T can be easily calculated using the relation between the radii of the periapsis and apoapsis and the semi-major axis as given by Equation 3.14.

$$r_p + r_a = 2a \tag{3.14}$$

Decision variables and estimated fuel mass

Now that all orbital elements have been determined for both transfer phases, the values must be translated to the required decision variables described in Section 3.1. This is done using Figure 3.16.

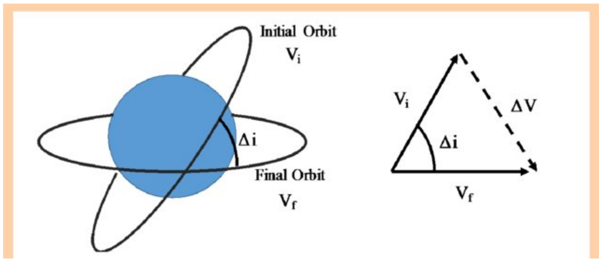


Figure 3.16: Schematic of an impulsive inclination change [28]

As can be seen in this figure, to achieve an inclination change of Δi, a specific magnitude and direction is required for ΔV to change from the initial velocity V_i to the final velocity V_f. Fortunately, all the elements required to calculate these values have already been determined in the previous sections. The only new formula that is required for this velocity calculation is the Vis-Viva equation as given by Equation 3.15 [29].

$$V^2 = GM_E\left(\frac{2}{r} - \frac{1}{a}\right) \tag{3.15}$$

This equation relates the orbital velocity V of the satellite to its current distance from the center of the central body r and the semi-major axis of the orbit. Furthermore, $G = 6.6743 \cdot 10^{-11}[m^3kg^{-1}s^{-2}]$ is the gravitational constant and $M_E = 5.972 \cdot 10^{24}[kg]$ is the mass of Earth, which are commonly grouped in the gravitational parameter of Earth, which is given by $\mu = 3.986 \cdot 10^{14}[m^3 s^{-2}]$, resulting in Equation 3.16 .

$$V = \sqrt{\frac{2\mu}{r} - \frac{\mu}{a}} \tag{3.16}$$

This equation can be used to determine both V_i and V_f for each of the burns. Finally, to determine ΔV, the assumption of an impulsive shot is made. This means that it is assumed that the change in velocity occurs instantaneously. By making this assumption, trigonometry can be used to calculate ΔV. In the triangle on the right of Figure 3.16, the cosine rule given by Equation 3.17 holds.

$$\Delta V^2 = V_i^2 + V_f^2 + 2V_i V_f \cos \Delta i \tag{3.17}$$

Taking the square root of both sides leaves the direct formula for the change in velocity Δv for a given V_i, V_f and Δi.

Moreover, the direction of the burn in the local horizontal plane γ can subsequently be calculated by subtracting the angle between V_i and ΔV, also $\angle(V_i, \Delta V)$, from π, since this thrust angle is measured from the direction of V_i. To determine angle $\angle(V_i, \Delta V)$, the sine rule as given by Equation 3.18 can be used.

$$\frac{\sin \angle(V_i, \Delta V)}{V_f} = \frac{\sin \Delta i}{\Delta V} \tag{3.18}$$

Which after rewriting results in Equation 3.19

$$\gamma = \pi - \arcsin\left(\frac{V_f \sin \Delta i}{\Delta V}\right) \tag{3.19}$$

Special care should be taken when calculating $\angle(V_i, \Delta V)$ with the sine rule, as this always gives the value for $\angle(V_i, \Delta V)$ smaller than $\frac{\pi}{2}$. So, if $V_f > V_i$, then $\angle(V_i, \Delta V) > \frac{\pi}{2}$, and one should subtract the angle obtained from π before proceeding.

The final values that are left to be determined are the durations of the three burns and coasting phases. Starting with the burn times, the analytical equation for the burn time is given by Equation 3.20.

$$t_b = \frac{I_{sp} M_0 g_0}{T} \cdot \left(1 - \frac{1}{\Lambda}\right) \tag{3.20}$$

Where $I_{sp} = 200[s]$ is the specific impulse of the thrusters, M_0 is the total mass of the satellite before firing its thrusters, $g_0 = 9.81[ms^{-2}]$ is the gravitational acceleration, and $\Lambda = \frac{M_0}{M_e}$ is the mass fraction of the burn, with $M_e = M_0 - M_f$ being the final mass of the satellite after the burn when it has burned through M_f kilograms of fuel.

This mass fraction can be directly calculated by rewriting Tsiolkovsky's famous Rocket Equation [20] as given by Equation 3.21 to obtain Equation 3.22.

$$\Delta V = I_{sp} \cdot g_0 \cdot \ln\left(\frac{M_0}{M_f}\right) \tag{3.21}$$

$$\Lambda = \frac{M_0}{M_f} = \exp\left(\frac{\Delta V}{I_{sp} \cdot g_0}\right) \tag{3.22}$$

Finally, since the initial mass of the servicer is known, the fuel mass can also be calculated directly from Equation 3.22. By subtracting the fuel mass from the initial mass, the end mass of the first burn is calculated, which can be used as the initial mass for the second burn. The process can then be simply repeated for the second and third burn.

To determine the coasting durations, the following two methods can be used: For the first coasting phase, one must realize that the inclination change is most optimally performed when the spacecraft is crossing the equator. Therefore, the approximate travel time of the servicer to the nearest equator crossing must be determined. By adding the argument of periapsis and the true anomaly of the servicer at t_0, the angle with respect to the equatorial plane called the argument of latitude $u = \omega + \theta$ is obtained.

Subtracting this from π in case $u < \pi$, or from 2π in case $u > \pi$ results in the angle left to travel to the equatorial plane. Dividing this angle by the mean motion gives the time to the equator. However, one should check if this time is larger than half of the first burn time. If this is the case, the burn will not be properly centered around the equator, and the burn must be initiated at the following equator crossing.

The final step for the calculation of the three coasting times is to subtract half of the burn times that surround the coast phase from the coasting times. This is again to center the burn around the equator crossings.

3.2.2. Results

After having iterated the algorithm presented in Section 3.2.1 until a specified maximum total transfer time is reached, the full population of individuals is obtained. The results of a transfer between two satellites that launched on the Transporter-9 mission are shown in Figure 3.17.

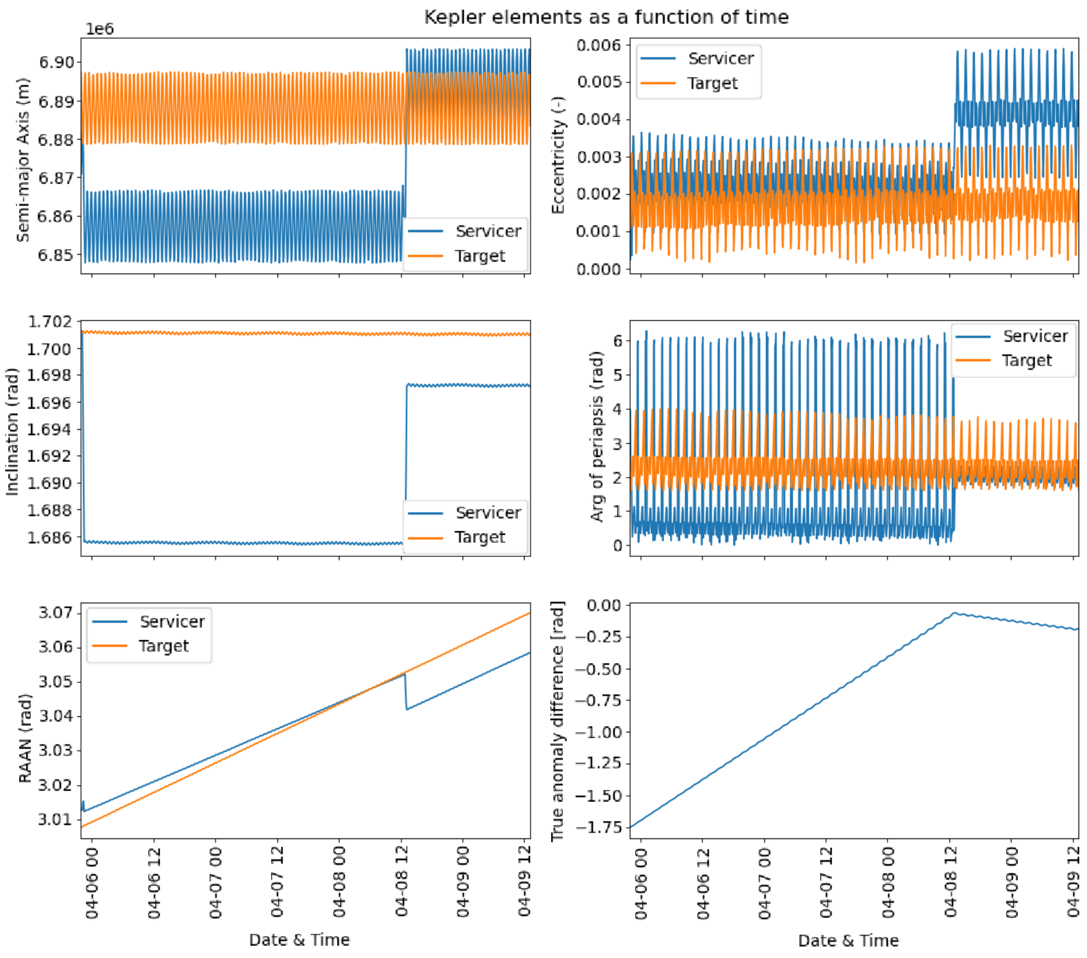


Figure 3.17: Kepler elements as a function of time for both the servicer and the target, where the decision variables have been determined by the analytical method from Subsection 3.2.1

As can be seen from this figure, both the difference in RAAN and the difference in the true anomaly between the two satellites approach zero after the transfer phase. Moreover, the semi-major axis of the servicer is also relatively close to the target value. Finally, the inclination of the servicer also comes close to the target value.

There are, however, some important things to notice in Figure 3.17. The two most obvious observations that are made are the discrepancies between the inclinations and the RAAN between the satellites right after the final burn. Fortunately, this error looks a lot worse than it is, as can be concluded from Figure 3.18.

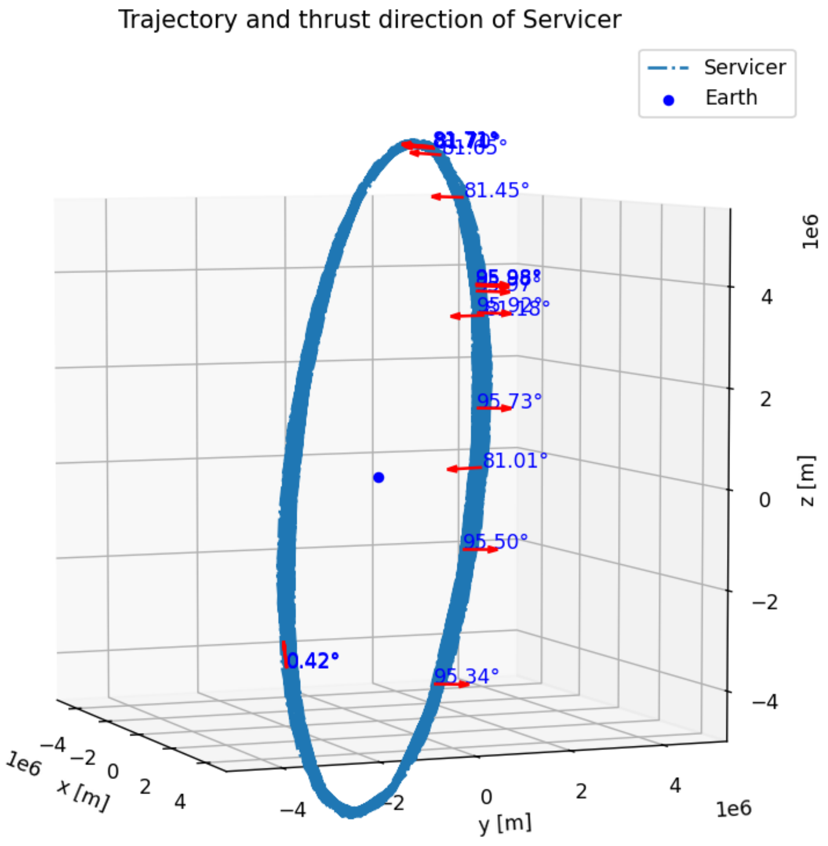


Figure 3.18: Trajectory of the Servicer plotted with the thrust vectors and their angle with respect to the velocity vector of the satellite

In this figure, the trajectory of the servicer is plotted together with the thrust vectors. These include the angle between the velocity vector and the thrust vector. As can be seen, the second short burn that is performed to raise the orbit is performed before the equatorial crossing instead of on top of this crossing. Assuming that the final burn is performed almost exactly half an orbit later, this means that the insertion burn is also not centered correctly around the equatorial crossing. Therefore, the axis around which the orbital plane is rotated does not lie within the equatorial plane, but makes an angle to it. This results in the fact that the RAAN is also shifted during this burn. Additionally, the true anomaly difference also starts to increase slightly after the final burn, which can be attributed to the mismatch in the semi-major axis. By increasing the transfer time such that the burns are correctly centered around the equator, the accuracy of the transfer improves, as can be seen in Figure 3.19.

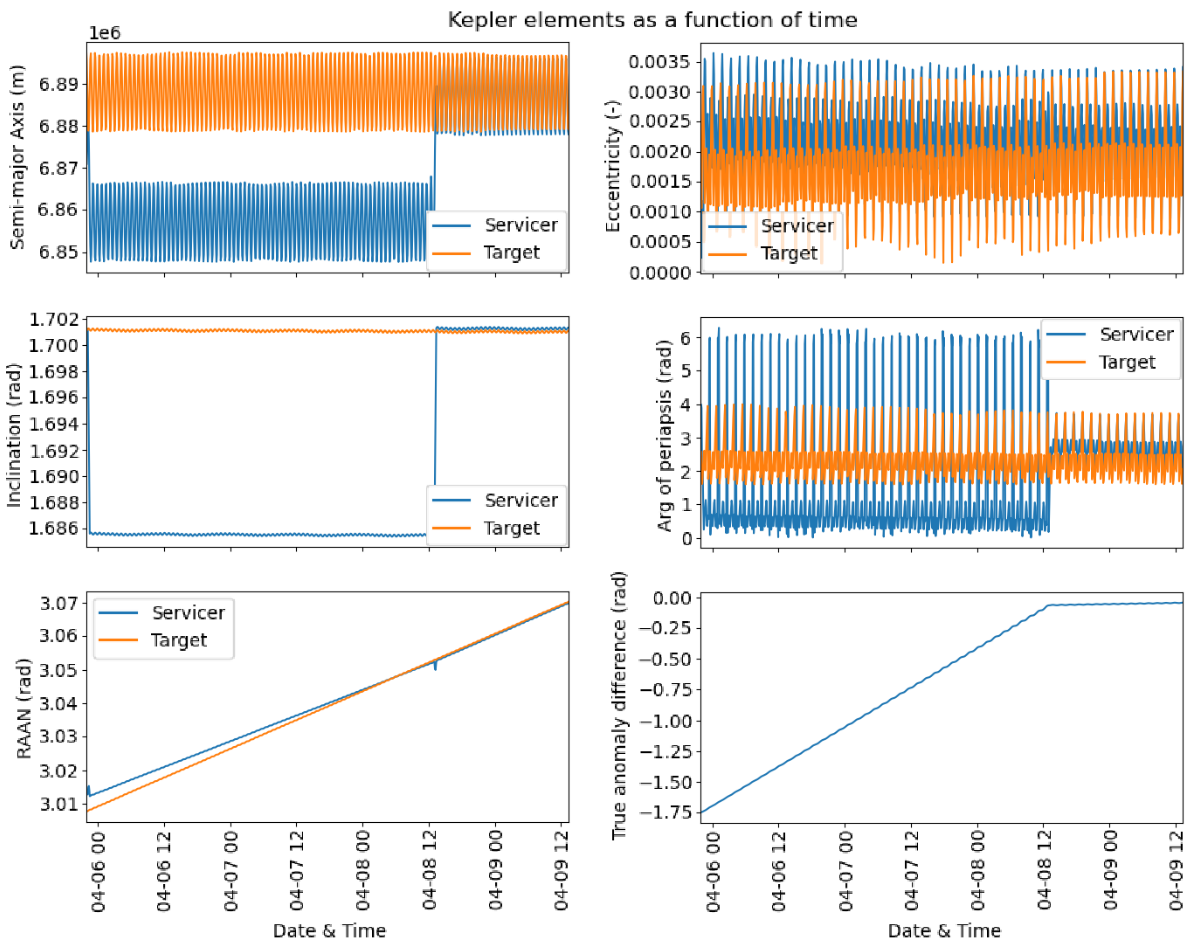


Figure 3.19: Kepler elements as a function of time for both the servicer and the target, where the decision variables have been determined by the analytical method from Subsection 3.2.1, and have been manually modified to improve the orbit matching

Here, the time of the second coasting phase has been increased by 660 seconds, and the direction of thrust γ has been increased by 0.03 radians. This adjustment in decision variables give a large improvement of the accuracy. Instead of manually tuning the decision variables, this adjustment will be made by the optimization algorighm that was presented in Section 3.1, but with some slight modifications that will be presented in the next section.

The example presented above is chosen because it most clearly displays the function and the deficiencies of the analytical model. Looking at figure 3.17, one might wonder why the RAAN is not changed directly instead of using the oblateness of the Earth, since a slight change in burn point clearly results in a RAAN change larger than the initial $\Delta\Omega$. In reality, however, the targets will have drifted further from the servicer than was the case in this example. Therefore, the method of using the oblateness of the Earth to match the satellites RAAN is still the most optimal one.

3.2.3. Local Optimization with Pygmo

Now that the entire population of initial guesses that lie within a specific time window has been determined, the optimization presented in Section 3.1 can be modified such that it converges to the chosen transfer. The simplest way of achieving this is to provide the problem class that is optimized by Pygmo with the initial guess. The boundaries given by the method ”get_bounds” can then be changed so that they form a small box around this initial guess. Moreover, the fitness function that returns the fitness values for a specific set of decision variables is given a penalty in case the number of transfer orbits calculated by the simulation does not match the expected amount of transfer orbits. In this way, it is assured that the obtained optimum is, in fact, the sought optimum.

During testing of this new refined optimization method, it was found that the method works reliably for short transfers up to about a week. However, when the targets are farther apart, causing the transfer time to increase, the reliability of the program to find the optimum decreases. A wide variety of multi-objective, single-objective, and local optimization methods have been deployed in order to reliably home in on the global optimium that is nearest, but unfortunately with no avail. One of the results of this trial and error with single-objective optimizers can be seen in Figures 3.20 and 3.21 below.

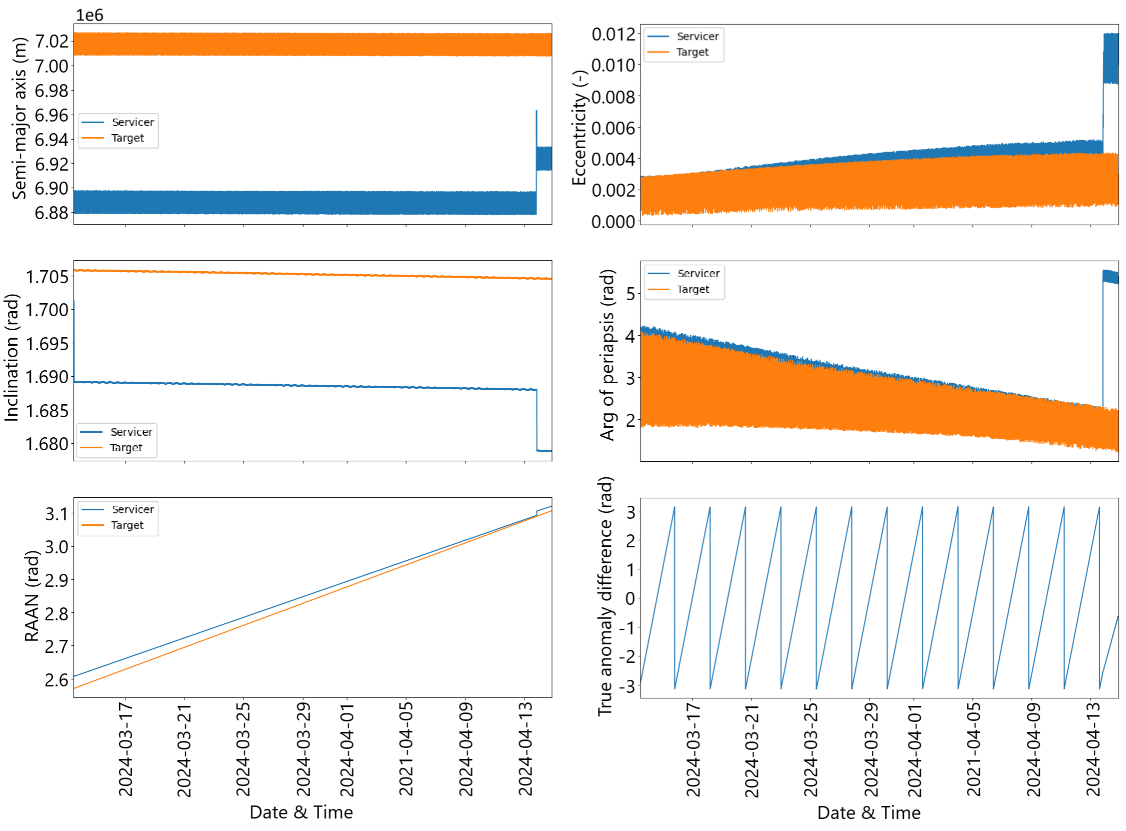


Figure 3.20: Kepler elements of a propagated transfer where the decision variables have been calculated with the algorithm described in Subsection 3.2.1

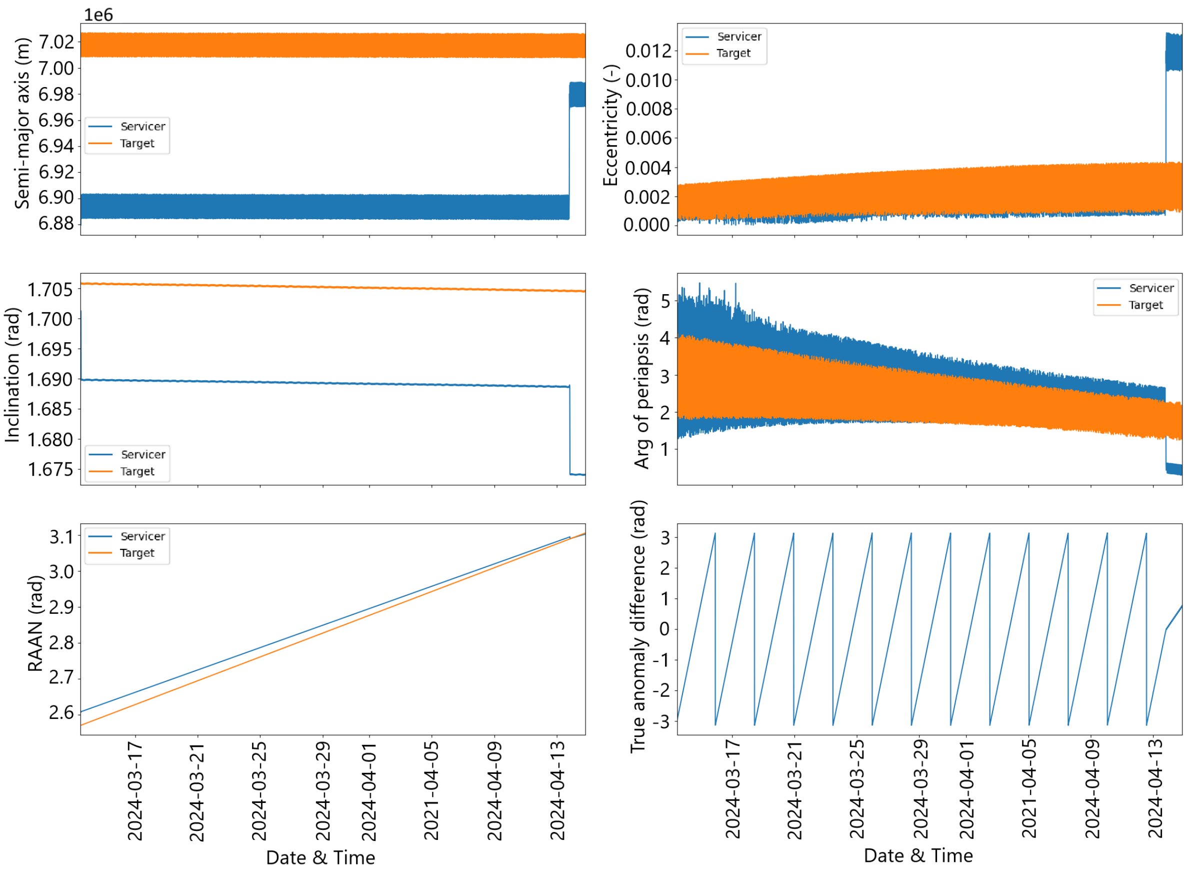


Figure 3.21: Locally "Optimized" transfer, performed with compass-search, with the initial decision variables being the same as used in figure 3.20

Figure 3.20 shows that the analytically calculated guess results in a great RAAN matching. The fact that the inclination moves further away from the target value at the final burn instead of near it indicates that the true anomaly of the propagated value has shifted so far from where the analytical solution estimated it to be, that it is shifted into the other half of the orbit. Therefore, the thrust is applied in the wrong direction, causing the inclination to decrease even further.

After optimizing the initial guess with the local optimizer Compass Search, also provided by Tudat-Py, the Kepler elements of the transfer become as shown in Figure 3.21. Looking at the two graphs below that show the RAAN and the phase difference between the satellites, it is concluded that the optimization has definitely performed well for these two elements, as they both approach zero right after the final burn. However, the inclination performed even worse than for the initial guess. Since the change in inclination during the final burn is approximately the same as during the initial burn, and because the RAAN almost does not change during this final burn, it can be concluded that the burn is performed at the wrong equator crossing. Finally, also the semi-major axis and the eccentricity show poor matching.

This example clearly shows that the optimization, again, becomes stuck in a local optimum, and although it is closer to the true optimum compared to the globally optimized transfer between two different satellites in Figure 3.11, it is still not at the required accuracy.

This problem is solved in the following subsection, where the formulation of an optimization algorithm for this specific transfer model is described.

3.2.4. Custom Developed Optimization Method

The key insight for creating a custom optimization method for this problem is that the decision variables define how a servicer needs to move, at a specific moment in time throughout the transfer. This results in the decoupling of the decision variables in the case that specific intermediate objectives are chosen. This effectively splits the entire problem into multiple smaller optimization problems. An explanation of what this practically means is given in the following.

In Figure 3.22, the initial guess for a transfer between two satellites is shown.

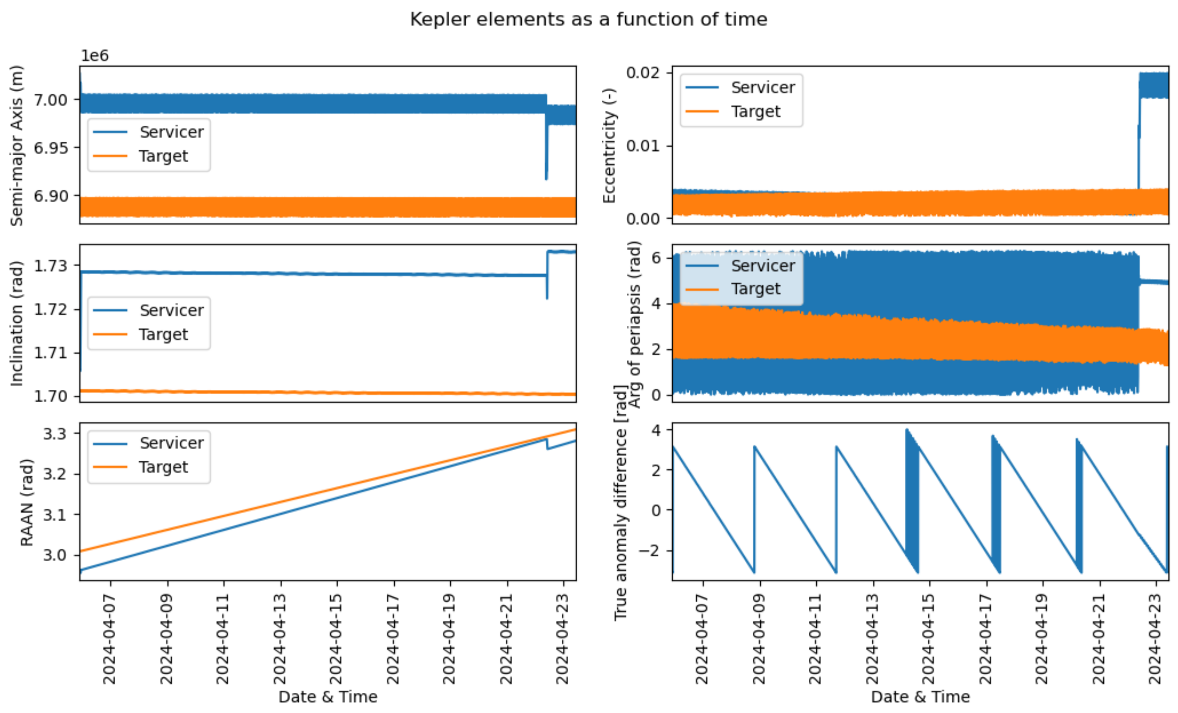


Figure 3.22: Kepler elements over time of a propagated initial estimate of a realistically sized transfer between two satellites in Sun-Synchronous orbit.

Now, say that the decision variables need to be manually adjusted in order to optimize this transfer, where would one start? An attempt might be made like the one presented in Subsection 3.2.2, where the second coast time was extended to force the burning point to be centered around the equator crossing. However, this would need to be repeated in case the duration of one of the phases is adjusted before the second coast phase. Except, this is not the case for the first coast phase, as per definition of the transfer model there is no propagation before it. The start of the simulation is the first coast phase. Therefore, if the first decision variable - which sets the duration of the first coast phase - is adjusted such that the primary objective of this coast phase is reached, there is no possibility that it needs to be changed at a later stage when one of the other decision variables is adjusted.

The same line of reasoning holds for the first burn time and burn direction: Together, they are the only two decision variables that influence the semi-major axis and the inclination during the transfer phase, which in turn, together with the second coast duration, both have a direct effect on the phase and RAAN error at the end of the transfer. If it is somehow managed to alter the first burn direction and duration, as well as the second coast time, such that both the RAAN and phases are exactly matched, together with the second burn point being centered around the equator, they do also not need to be modified when one of the subsequent decision variables is changed.

After this is done, the final coast duration can be modified to center the final burn around the equator. And finally, in a similar fashion as the first burn parameters were estimated, the second burn duration, and the third burn duration and direction can be adjusted to perfectly match the target orbit.

Tuning of the first coast phase As mentioned before, in order to tune the first coast duration in a systematic way, the goal is to determine how off-centered the first burn is is in the first place. This can be determined by making use of Figure 3.23.

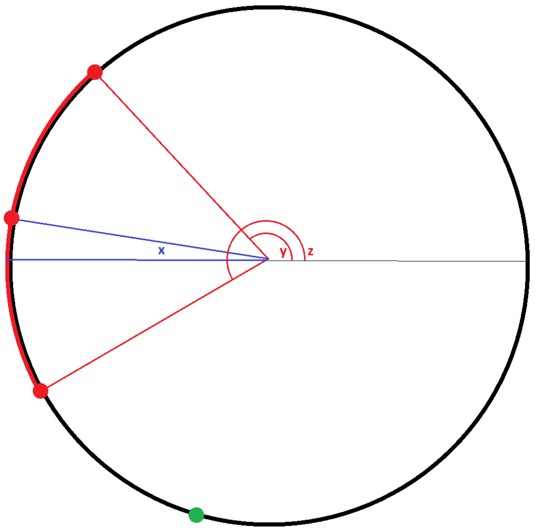


Figure 3.23: Schematic of first coast duration change

Here, the green dot indicates the servicer which moves clockwise along the black circle. The red part of this circle indicates where the thruster is being fired. Here, the red dots indicate the start, middle, and end of the burn, and the horizontal line represents the equator. It is clear that the middle of the burn needs to be moved by x radians to have it centered. Since the values of $y = \omega_e + \theta_e$ and $z = \omega_b + \theta_b$ can be calculated directly from the propagation data, angle x can be found by taking the average of y and z. To convert this to the time that needs to be subtracted from the first coast phase, the angle x must be divided by the mean motion to obtain the time it takes for the servicer to travel that angular distance, or $\Delta t = x\sqrt{\frac{\mu}{a^3}}$.

Tuning of first burn time, angle, and second coast phase In order to ensure that the servicer has matched the target's phase and RAAN after the transfer, the first burn time, first burn direction, and second coast phase all must be changed simultaneously. This is because the parameters have the following effect on the trajectory:

- The first burn time simply increases the magnitude of the change in inclination, semi-major axis, and eccentricity of the orbit change.
- The burn direction in the local horizontal plane defines the ratio between the change in inclination and the change in semi-major axis. If the burn is performed colinear to the velocity, only the semi-major axis will change, whereas if the burn is performed perpendicular to the velocity, only the inclination will change.
- The time of the second coast phase determines the required rate of change of the Phase and the RAAN, both of which are dependent of the semi-major axis and inclination. In addition, the second coast phase also directly determines the centering of the second burn, just like the first coast centered on the first burn. Finally, it must be checked that the second coast duration is precisely as long as was required by the initial guess.

The question now is as follows: How does one determine how these three decision variables must be changed in order to achieve a certain change in desired output values? In other words, a relationship between the in- and output values must be obtained. This is not as easy as it was before, as these three inputs are related in a non-linear way. However, if the function is continuous, its behavior near a specific point can be approximated using a linearization technique based on the first-order Taylor expansion [30].

Let $\mathbf{x} \in \mathbb{R}^n$ represent a vector of input variables and let $\mathbf{y} \in \mathbb{R}^m$ be the corresponding output variables produced by a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. To determine how a small change in the input $\Delta\mathbf{x}$ affects the output $\Delta\mathbf{y}$, the function $\mathbf{f}(\mathbf{x})$ can be approximated around a point $\mathbf{x}_0$ using the first-order Taylor expansion:

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) \tag{3.23}$$

where $\mathbf{J}(\mathbf{x}_0)$ is the Jacobian matrix of $\mathbf{f}$ evaluated at $\mathbf{x}_0$. The Jacobian matrix is given by:

$$\mathbf{J}(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{3.24}$$

For a small change $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$, the change in the output $\Delta\mathbf{y} = \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_0)$ can be approximated as:

$$\Delta\mathbf{y} \approx \mathbf{J}(\mathbf{x}_0) \cdot \Delta\mathbf{x} \tag{3.25}$$

This linear relationship allows for the prediction of how changes in the inputs will affect the outputs.

Given a desired change in the output $\Delta\mathbf{y}_{\text{desired}}$, the required change in the inputs $\Delta\mathbf{x}$ can be solved as:

$$\Delta\mathbf{y}_{\text{desired}} = \mathbf{J}(\mathbf{x}_0) \cdot \Delta\mathbf{x} \tag{3.26}$$

This can be expressed as a system of linear equations:

$$\mathbf{J}(\mathbf{x}_0) \cdot \Delta\mathbf{x} = \Delta\mathbf{y}_{\text{desired}} \tag{3.27}$$

To find $\Delta\mathbf{x}$, the above equation can be solved. If $n = m$ and the Jacobian matrix $\mathbf{J}(\mathbf{x}_0)$ is invertible, the change in inputs can be computed as:

$$\Delta\mathbf{x} = \mathbf{J}(\mathbf{x}_0)^{-1} \cdot \Delta\mathbf{y}_{\text{desired}} \tag{3.28}$$

If $n \neq m$ or if the system is underdetermined or overdetermined, a solution can be found using the least squares approach.

Equation 3.28 can now be used to determine the required changes in the input variables $t_{b,1}$ and γ to achieve the desired changes in the output variables, specifically the RAAN, Ω, and the angle with respect to the equator, $\eta = \omega + \theta$.

The first step involves calculating the partial derivatives of Ω and η with respect to the input variables $t_{b,1}$ and γ using numerical differentiation.

The partial derivatives of RAAN with respect to $t_{b,1}$ and γ are approximated as:

$$\frac{\partial\Omega}{\partial t_{b,1}} \approx \frac{\Omega_{\text{tb}} - \Omega}{\epsilon_{\text{tb}}} \tag{3.29}$$

$$\frac{\partial\Omega}{\partial\gamma} \approx \frac{\Omega_\gamma - \Omega}{\epsilon_\gamma} \tag{3.30}$$

where Ω_{tb} is the RAAN calculated with a small perturbation ϵ_{tb} in $t_{b,1}$, and Ω_γ is the RAAN calculated with a small perturbation ϵ_γ in γ.

Similarly, the partial derivatives of η with respect to $t_{b,1}$ and γ are approximated as:

$$\frac{\partial\eta}{\partial t_{b,1}} \approx \frac{\eta_{\text{tb}} - \eta}{\epsilon_{\text{tb}}} \tag{3.31}$$

$$\frac{\partial\eta}{\partial\gamma} \approx \frac{\eta_\gamma - \eta}{\epsilon_\gamma} \tag{3.32}$$

where η_{tb} is the angle with respect to the equator calculated with a small perturbation ϵ_{tb} in $t_{b,1}$, and η_γ is the angle with respect to the equator calculated with a small perturbation ϵ_γ in γ.

The desired changes in Ω and η are defined as:

$$\Delta\Omega_{\text{desired}} = \Omega_{\text{target}} - \Omega \tag{3.33}$$

$$\Delta\eta_{\text{desired}} = \eta_{\text{target}} - \eta \tag{3.34}$$

where Ω_{target} and η_{target} are the target values for RAAN and η, respectively.

Next, the required changes in $t_{b,1}$ and γ, denoted as $\Delta t_{b,1}$ and $\Delta\gamma$, can be found by solving the following system of linear equations:

$$\begin{bmatrix} \Delta\Omega_{\text{desired}} \\ \Delta\eta_{\text{desired}} \end{bmatrix} = \begin{bmatrix} \frac{\partial\Omega}{\partial t_{b,1}} & \frac{\partial\Omega}{\partial\gamma} \\ \frac{\partial\eta}{\partial t_{b,1}} & \frac{\partial\eta}{\partial\gamma} \end{bmatrix} \begin{bmatrix} \Delta t_{b,1} \\ \Delta\gamma \end{bmatrix} \tag{3.35}$$

To find $\Delta t_{b,1}$ and $\Delta\gamma$, the matrix equation can be solved using:

$$\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b} \tag{3.36}$$

where:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial\Omega}{\partial t_{b,1}} & \frac{\partial\Omega}{\partial\gamma} \\ \frac{\partial\eta}{\partial t_{b,1}} & \frac{\partial\eta}{\partial\gamma} \end{bmatrix} \tag{3.37}$$

$$\mathbf{b} = \begin{bmatrix} \Delta\Omega_{\text{desired}} \\ \Delta\eta_{\text{desired}} \end{bmatrix} \tag{3.38}$$

and

$$\mathbf{x} = \begin{bmatrix} \Delta t_{b,1} \\ \Delta\gamma \end{bmatrix} \tag{3.39}$$

The values of $\Delta t_{b,1}$ and $\Delta\gamma$ obtained by solving this system provide the adjustments required to achieve the desired changes in RAAN and η.

Before proceeding with further iterations to refine the solution, the adjustment required for the second coast phase must be determined using the same method applied to calculate the adjustment for the first coast phase.

The suboptimization required for setting the duration of the second burn is somewhat similar to the method presented above, but then with only one input and output variable. However, the difference between these two methods is the objective. In case of the second burn time, the objective that needs to be minimized is the amount that the radius at the opposite apsis increases. Specifically, this change must be exactly the same as the difference in radii between the initial servicer and target orbits. This is to ensure that the orbit's semi-major axis is properly matched.

Finally, the third coast phase and the third burn duration and direction can be obtained similarly as the corresponding parameters of the first burn.

Figure 3.24 shows the final optimized transfer of the transfer shown in Figure 3.22.

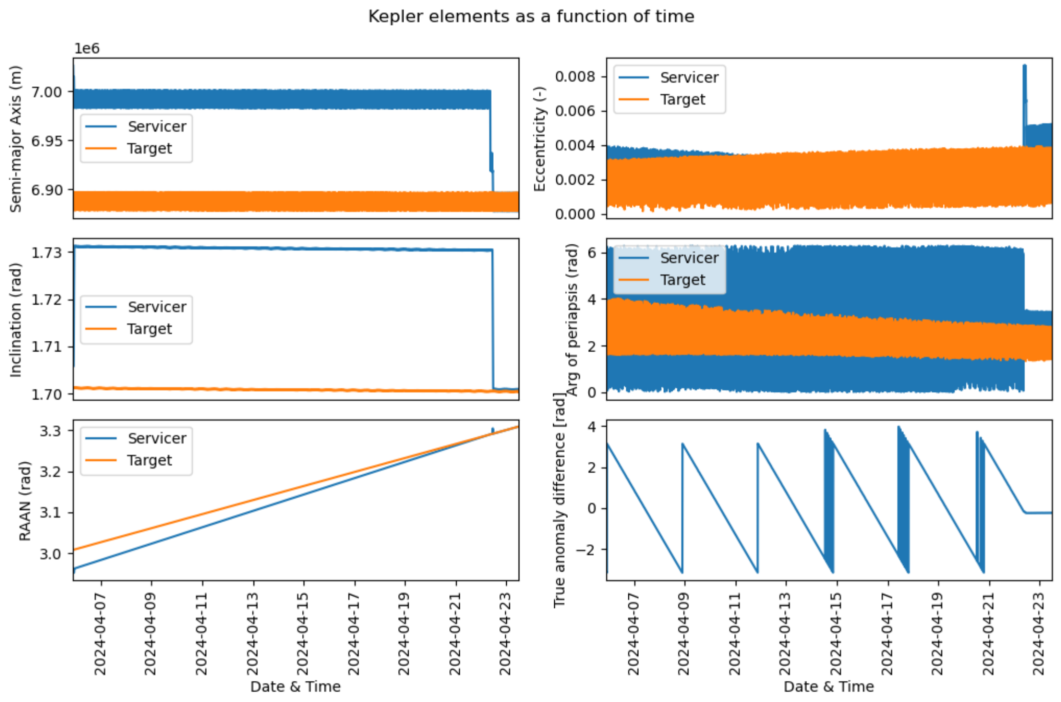


Figure 3.24: Kepler elements as a function of time of a transfer which is optimized by the custom optimization scheme.

Now that the framework has been set up in which it is possible to systematically obtain all possible transfers between any two satellites in Sun-synchronous orbit within a specified time frame, it is possible to set up, optimize and verify the model for the servicer spacecraft that needs to visit multiple moving targets, also known as the Dynamic Traveling Salesman Problem.

4 What sequence of visits to multiple arbitrary Client Satellites optimizes efficiency for the servicing spacecraft, akin to solving the Dynamic Traveling Salesman Problem?

The Dynamic Traveling Salesman Problem (DTSP) represents a complex and evolving challenge in combinatorial optimization, where a salesman must determine the most efficient route to visit a set of cities, which may change over time. This problem becomes particularly intricate and critical in the context of space operations, such as optimizing the path for refueling client satellites in SSO.

Building upon the algorithm previously developed for identifying all possible transfer options between two satellites within a specified timeframe, this chapter addresses the extension of that approach to solve the DTSP for satellite refueling missions. The nature of this problem is dynamic and requires continuous adjustments to the optimal refueling path as new information becomes available. This involves not only determining the most efficient sequence of satellite visits, but also considering the evolving constraints and priorities that affect mission planning in real time.

The dynamic aspect of the problem arises from several factors: The changing positions of satellites in orbit, the variations in required amounts of fuel, the fluctuating mission priorities, and the specific time windows for refueling operations. These elements require an approach capable of adapting to changes and optimizing the refueling schedule on the fly. The goal is to ensure that all CS receive the necessary fuel while minimizing the operational costs, particularly the fuel consumption of the servicer.

In this chapter, the previously developed transfer algorithm will be integrated into the DTSP framework, implementing it to address the multi-objective nature of the problem. This involves not only ensuring optimal refueling sequences, but also considering the impact of each transfer option on the overall mission efficiency. A optimization algorithm will be used to handle the complexities of the dynamic environment, as well as the discrete nature of the DTSP.

This chapter will explore the modeling of these dynamic factors, the integration of the transfer option algorithm, and the optimization methods used to solve the DTSP. The aim is to develop a robust solution that can adapt to the evolving conditions in space, ensuring efficient and effective refueling operations for satellites in SSO.

4.1. Dynamic Traveling Salesman Model

The first step in finding the optimal route between a multitude of CS is to set up the model with the same rules as was done for the single transfer in Section 3.1. To repeat: The goal is to find a set of decision variables that can be tuned, which fully describe the path of the servicer between all of the target satellites. In addition, one or more objectives that need to be minimized must be chosen. Finally, constraints need to be defined to give bounds to give bounds to the optimization.

4.1.1. Decision variables

First, the decision variables must be defined. More attention is required here compared to the single-transfer optimization for several reasons. As outlined in the chapter introduction, the 'dynamic' aspect of the DTSP is what makes it a particularly challenging problem to optimize. The following sections will address solutions for managing the different types of 'dynamic' present in the problem.

Number of satellites to be serviced

The first part of the problem that needs to be variable is the number of satellites that are required to be serviced. The current research that has been done about this problem only considers a fixed number of target satellites that are known before launch. The novelty of this infrastructure lies in the fact that service requests can be done on-demand. However, this requires that the number of satellites in the algorithm to be dynamic. At some moments in time, it could be the case that only one target is in the queue, whereas at other times five or more targets could be in the queue.

The specific problem that needs to be solved is that the decision variables are required to represent precisely the entire path of the servicer. This means both the order of visits and what specific transfer strategy the servicer will take for any specific transfer. This could be a very slow transfer that minimizes fuel consumption or a fast transfer that inadvertently impedes fuel consumption. Ideally, this problem is solved by the model simply adding decision variables as it encounters new targets. Unfortunately, the way that optimization with Pygmo is setup is that the problem class that you provide to the optimization algorithm needs to be an object with a specific format. Precisely, it needs to have at least the following two methods:

- The first method that is required is the method called 'fitness' which only takes the vector of decision variables as an input. Its output should be a vector that contains the fitness values.
- The second method that needs to be in this problem object is the 'get_bounds' method. This method returns two vectors that contain all the lower and upper bounds of the decision variables. Subsequently, Pygmo determines from this method the number of decision variables that it must 'feed' into the fitness function.

The problem, but luckily also the solution, lies in the get_bounds method. The problem is that before the optimization algorithm is started, the number of decision variables must be determined. This prevents the algorithm from being able to take in new CS whilst performing the optimization. The solution is the fact that the boundaries of the decision variables are determined by its own method which has access to the attributes of the problem object, and it does not matter what kind of code is within this method, or within the declaration of the object. Therefore, the number of decision variables and their bounds can be deduced from the number of satellites that request service. The result of building in the dynamic nature of the number of decision variables in this way is that each time a new service request comes in, the entire optimization algorithm must be run again. This creates the physical constraint that the computation time cannot be too large and needs to be kept to a minimum. The reason for this is simple: If it would take a few days to find an optimum each time a new satellite enters the optimization, the relative position of the satellites would change significantly because of their relative drift, resulting in a decrease in confidence that the found optimum is actually the global optimum. Therefore, it is important to find a balance between computation time and confidence in the optimum found.

Servicer Refueling at Fuel Stations

In the event that the problem was to find the optimal route between a certain number of targets only, the question of establishing the length of the decision vector had been solved. Unfortunately, the FS-servicer-CS infrastructure leaves the servicer with the option of refueling at a fuel station before heading to the next client satellite. This means that after each refueling of a CS, the servicer can transfer directly to the next client by using the limited amount of fuel it has left, or it can first transfer to a FS, refuel itself, and continue to the set client with a full tank. Deciding which route to take is a pivotal result of the optimization itself, which is why, per definition, this cannot be determined up front.

Individual Transfer Strategy

The final issue that needs to be solved is finding which transfer strategy a servicer will take for any specific transfer between two satellites. Due to the fact that the satellites move constantly with respect to each other, the set of possible routes depends on time. However, the choice of which route a servicer is going to take must somehow be given by a decision variable, and as was explained earlier in this section, the boundaries of each decision variable must be given beforehand. There are two ways to solve this problem.

The first method is to provide the number of transfer orbits for a specific transfer as a decision variable. This has the benefit of being specific and deterministic, meaning that one specific choice of number of transfer orbits in one part of the whole optimization has about the same effect on the total transfer time as it would have anywhere else in the problem. However, the downside of this method is that the number of possible transfer orbits within a certain timeframe and fuel level is not known beforehand. Therefore, defining fixed limits to this variable is impossible. One way to mitigate this is by choosing the minimum number of transfer orbits to be one, and combining this with setting a large penalty to the fitness value in case the optimizer chooses an infeasible number of transfer orbits. This has the downside that, for a specific leg, this minimum number of orbits varies based on the previous route choices. The variability in when a penalty will be imposed on a decision vector is a property that could be handled poorly by the optimization algorithm, preventing it from finding the global optimum.

The second method is to create the list of all possible transfer routes between two satellites within a specified maximum timeframe, sort this list by transfer time, and finally provide the fraction of where the desired transfer is within this list. This has the benefit of always resulting in a feasible result. The downside is that the problem loses its direct relationship with the chosen transfer. Instead, the fraction indicates whether the transfer will be on the slow side of the spectrum or on the fast side.

Due to the predicted poor behavior of the optimization algorithm caused by the changing location of where the penalty is introduced, the method that will be implemented is that where the fraction is provided.

Final setup of the decision vector

The final decision vector is built up in the following way:

- The first variable dictates what the order of CS to visit will be. Since each CS only needs to be visited once, all possible permutation of the targets can be created at the declaration of the problem and put into a list. The decision variable then basically sets the index of choice from this list. The minimum value is obviously zero, and the maximum value is one less than the total number of permutations, as index counting starts from zero.
- The next part of the decision vector, are n integer values that indicate if after the i-th visited target, the servicer should continue to its next target or visit a fuel station and if so, what fuel station. Here, n is total number of targets to visit. This is implemented by simply providing the index of the fuel station to visit. Choosing to skip a fuel station corresponds to the value -1. Therefore, the bounds of the problem are -1 up to the number of fuel stations minus one.
- The final part of the decision vector are the fractions that indicate how fast a transfer between two satellites should be. As the maximum number of transfers - including those to fuel stations - can be twice the size of the number of targets to visit, this needs to be the amount of decision variables that indicate the transfer speed. It should be noted that this could mean that some of the decision variables that specify the speed of transfer to a fuel stations, could be redundant when the choice is made not to visit a fuel station. One might wonder and think that this would cause the same problems as is encountered when setting a penalty in case of infeasible index values explained above. However, this is not the case since the decision variable simply has no effect on the fitness of the problem. Therefore, the behaviour of the optimization process itself will not be effected by the value of this redundant decision variable. Only when the choice is made to visit the fuel station does the value impact the fitness.

For clarification purposes, an example of a decision variable will be given below, where the simple case of two targets and one fuel station is discussed.

Example of a Decision Variable

When considering finding the optimal path for two client satellites and one fuel station, the decision variable will take the following shape:

- **Choice of path:** As there are only two client satellites, the list of all possible paths is $[(0,1),(1,0)]$, where the numbers indicate the index of the target. If the first path is chosen, this decision variable will simply be zero.
- **Choice of fuel stations:** As there are two client satellites, there is the possibility of visiting two fuel stations as well, and since there is only one fuel station to choose from, the maximum value that the choice of fuel station variables can have is zero. Because a servicer cannot loiter at client satellite, the final fuel station must always be chosen and thus has a minimum value of zero. A possible combination for the choice of fuel station variables could be $[-1,0]$, which indicates that the servicer continues straight onward to the next client after the first client has been serviced, and moves to the fuel station with index zero at the end of the path.
- **Choice of transfer speed:** The final four variables are the choices for the individual transfer speeds. An example could be $[20,50,40,90]$, where the fractions have been converted to percentages. This means that the transfer at 20% of the available transfer list will be chosen for the transfer to the first target, the transfer at 50% of the list will be chosen for the transfer to the first fuel station (which is redundant due to the -1 value for the choice of fuel station), the transfer at 40% of the list of transfers will be chosen for the transfer to the second client, and finally the transfer at 90% of the list of transfers will be chosen for the return transfer to the fuel station.

This results in the final decision vector of $[0,-1,0,20,50,40,90]$

4.1.2. Implementation of the Decision Vector

Now that the format of the decision vector has been set up, the model can be created. However, before proceeding, the final issue of the DTSP must be resolved, that is, the amount of fuel that a client satellite requires and the implementation of some sort of prioritization scheme.

Implementation of Required Fuel and Priority

The issue of how much fuel is required by a client is solved by including this fuel together with the initial satellite state of the client, which should be provided to the model anyways. The reason for a prioritization method needs further explanation.

Why is there a need to divert from a potentially more optimal route by including a priority level to the client satellites? The answer to this question is answered when looking at the problem from the business side. In the end, the goal is to make as much profit as possible from refueling customer satellites. And because this refueling can be pretty sensitive to time, why not sell the fuel with a priority rating? This not only can increase the potential revenue of the program, but it also has the added benefit that it limits the total number of possible routes because high-priority customers need to be served first.

Secondly, the reason for implementing a priority scheme is because it must be prevented that satellites don't get serviced at all due to new client satellites entering in front in the queue. This could happen, for example, when the new target is on the route between the servicer and the other client satellite. If no priority is given to a client at any point in time, it could occur that the new clients end up earlier in the queue, leaving the potential of a client not being serviced at all

Moreover, if only requests from higher priority customers are coming in, the satellite will also not be serviced. Since satellites in SSO have limited life spans due to the slight atmosphere still being present, it must be prevented at all costs that a client is not serviced before it reaches critically low altitudes. To solve this, the date of request and the maximum time the client can wait before requiring to be refueled are provided to the problem together with the client satellites' state and fuel. In this way, if after a certain amount of time after the request has been made the client has still not been refueled, it automatically increases in priority level. This continues until it reaches the highest priority level and the target is first to be serviced.

4.1.3. Objectives

Another essential part of the optimization that needs to be determined before the DTSP model can be implemented is the choice of objectives. As explained in Section 3.1.1, the objectives are the specific values of the model that are to be minimized. In case of the DTSP that is considered in this report, the most suitable choice of optimizers is the total fuel used for the full path of the servicer and the total transfer time of the full path. Because these are opposing goals — meaning that an increase in transfer time results in a decrease in required fuel, and vice versa — a Pareto optimal front can be derived to facilitate a trade-off between the fuel consumption and the transfer duration.

4.1.4. Constraints

Before creating the model, it is necessary to consider the constraints that should be applied to the problem. The first constraint ensures that each transfer leg does not exceed the maximum transfer time allowed per leg. This is inherently managed by the algorithm that generates all possible routes, as it only includes transfers within the specified maximum transfer time. Another constraint pertains to the fuel used during the transfer, the lower limit being zero fuel for the servicer. A less apparent constraint is the minimum altitude that is reached during the transfer. Due to the exponential increase in atmospheric density at lower altitudes, it is advisable to avoid transfers with very low apogees. This is because too much energy is lost at these altitudes, and the drag on the satellite becomes highly unpredictable, making trajectory calculations unreliable. The minimum altitude allowed altitude is, therefore, set to 350 km.

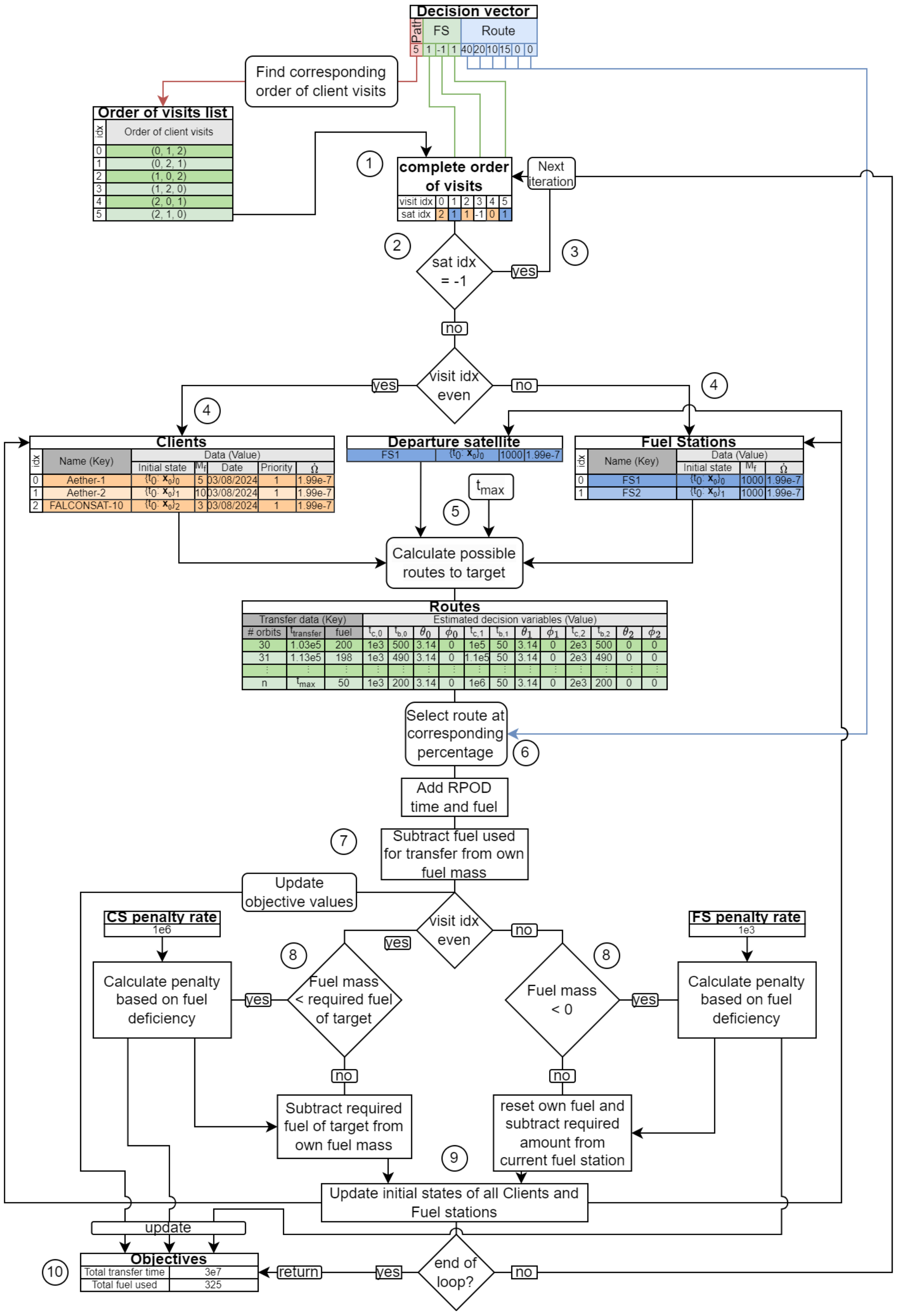
4.2. Chronological Approach to Modeling the DTSP

Now that the method in which all possible paths can be defined uniquely is determined, the actual model that calculates the fitness of any particular decision vector can be created. This is done by evaluating the decision vector in chronological order and properly keeping track of the servicer's used fuel mass and transfer time during each step of the route. In the following, a detailed step-by-step description of what the model entails is given together with a flow diagram of the algorithm itself in Figure 4.1. The numbers of each step are also indicated in the flow chart.

1. The first step of the model is to decipher the first part of the decision vector that contains the path index and the fuel station indices. The goal is to create a single list of satellites indices that alternate between targets and fuel stations. This can simply be done by retrieving the chosen path from the list of paths, after which this is stitched together with the list of fuel stations to visit.
2. After setting up all the required data storage variables, the newly formed list of satellites is processed using a for-loop.
3. In case the current index of the current satellite is equal to -1, the loop is continued to the next satellite, as this means that the servicer is required to skip this fuel station
4. When the index of the current satellite is not equal to -1, the satellite data of the current target and the departure satellite are retrieved.
5. Taking into account the starting conditions of the departure and target satellites, along with the pre-defined maximum transfer time allowed for each segment of the journey, the array of feasible paths to reach the target is determined.
6. Based on the corresponding value of the route fraction that is provided in the decision vector, the route to the target satellite is chosen.
7. Besides the estimate for the decision vector, the chosen individual also provides the estimated transfer time and fuel mass that is required for the chosen transfer. These values are then properly added to the storage variables created earlier. Moreover, a fuel and transfer time margin of 2 kilograms and one day respectively is added as well in order to account for the RPOD to the target satellite.
8. A couple checks are performed in order to verify that the relevant variables do not exceed the constraints. In case they do, they are penalized in the following way: When the Service has run out of fuel, it is given a very large penalty on both of the objectives. This penalty is chosen to be large enough that it will definitely be larger than any of the possible objective values. Since the total transfer time could theoretically be on the scale of years, the penalty value is chosen to be 10^{12}. In addition, some distinction is made in order to 'steer' the optimization algorithm in the right direction. First of all, it is not as bad to arrive at the target with a fuel deficiency of 1 kilogram than it is compared to 100 kilograms. Therefore, an extra small penalty proportional to this fuel deficiency is added. Secondly, it is obviously worse to arrive at a client without fuel as it is to

arrive at a fuel station without fuel, as the servicer still needs to perform an entire transfer after the client visit. Therefore, a final distinction is being made between arriving at a client without fuel versus arriving at a fuel station without fuel. This difference is implemented in the proportionality constant, which for arriving in a fuel deficient state at a fuel station is set to 10^3 and for arriving fuel deficient at a client is set to 10^6.

9. Finally, the initial values required for the next iteration of the loop are calculated. This includes the determination of the states of all the satellites that are still to be visited, at the time when the servicer is ready to leave to the next target. This is done by propagating all the states up to this point, and taking the final states as input values for the next iteration.
10. Now that the total fuel that is used and the total time transfer time of the entire path is calculated, they can be returned as the objectives of the fitness function.


Decision vector
Patr
FS
Route
5 | 1 | -1 | 1 | 40 | 20 | 10 | 15 | 0 | 0
Find corresponding order of client visits
Order of visits list
1
complete order of visits
Next iteration
2
sat idx = -1
yes
3
no
visit idx even
yes
no
4
4
Clients
Departure satellite
Fuel Stations
tmax
5
Calculate possible routes to target
Routes
Select route at corresponding percentage
6
Add RPOD time and fuel
7
Subtract fuel used for transfer from own fuel mass
Update objective values
CS penalty rate
1e6
FS penalty rate
1e3
visit idx even
yes
no
8
8
Calculate penalty based on fuel deficiency
yes
Fuel mass < required fuel of target
no
Subtract required fuel of target from own fuel mass
Fuel mass < 0
yes
no
reset own fuel and subtract required amount from current fuel station
Calculate penalty based on fuel deficiency
9
Update initial states of all Clients and Fuel stations
update
10
Objectives
return
yes
end of loop?
no

Order of visits list

idx	Order of client visits
0	(0, 1, 2)
1	(0, 2, 1)
2	(1, 0, 2)
3	(1, 2, 0)
4	(2, 0, 1)
5	(2, 1, 0)

complete order of visits

visit idx	0	1	2	3	4	5
sat idx	2	1	1	-1	0	1

Clients

idx	Name (Key)	Data (Value)				
		Initial state	M_f	Date	Priority	$\dot{\Omega}$
0	Aether-1	$\{t_0, \mathbf{x}_0\}_0$	5	03/08/2024	1	1.99e-7
1	Aether-2	$\{t_0, \mathbf{x}_0\}_1$	10	03/08/2024	1	1.99e-7
2	FALCONSAT-10	$\{t_0, \mathbf{x}_0\}_2$	3	03/08/2024	1	1.99e-7

Departure satellite

FS1	$\{t_0, \mathbf{x}_0\}_0$	1000	1.99e-7

Fuel Stations

idx	Name (Key)	Data (Value)		
		Initial state	M_f	$\dot{\Omega}$
0	FS1	$\{t_0, \mathbf{x}_0\}_0$	1000	1.99e-7
1	FS2	$\{t_0, \mathbf{x}_0\}_1$	1000	1.99e-7

Routes

Transfer data (Key)			Estimated decision variables (Value)											
# orbits	$t_{transfer}$	fuel	$t_{c,0}$	$t_{b,0}$	θ_0	ϕ_0	$t_{c,1}$	$t_{b,1}$	θ_1	ϕ_1	$t_{c,2}$	$t_{b,2}$	θ_2	ϕ_2
30	1.03e5	200	1e3	500	3.14	0	1e5	50	3.14	0	2e3	500	0	0
31	1.13e5	198	1e3	490	3.14	0	1.1e5	50	3.14	0	2e3	490	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	t_{max}	50	1e3	200	3.14	0	1e6	50	3.14	0	2e3	200	0	0

Objectives

Total transfer time	3e7
Total fuel used	325

Figure 4.1: Flow diagram of the Dynamic Traveling Salesman Problem model

4.3. Simplification of the DTSP model

Although the model described in Section 4.2 works properly and calculates the fitness values correctly, it is rather slow. A single computation of a decision vector was found to be around the order of a couple tens of seconds to a minute for a relatively small problem of only two satellites and one fuel station. As a large number of fitness evaluations are expected to be required for the optimization to converge, combined with the exponentially increasing number of computations for increasing number of client satellites, the model is deemed to be too large to be able to optimize it with only a system with a couple tens of locigal processors. Therefore, the computational load of the model needs to be drastically decreased. Unfortunately, this will inadvertently be paired with loss of precision. In this section, multiple methods for decreasing the computational time whilst keeping a reasonable level of accuracy are described.

4.3.1. Intermediate State Estimation

The part of the current DTSP model that by far takes the most computational effort is updating the state in between each iteration of the main loop by means of propagation. Therefore, one might wonder if there is another way in which the states of all satellites can be estimated at any specific point in time. Recalling Section 3.2.3 where analytical solutions to the model are determined, this does not seem far-fetched at all. In addition, since only satellites in SSO are considered, the only Kepler element that really requires a lot of fuel to change is the RAAN. Because changing the RAAN takes quite some time, there is plenty of time to match the target's phase. Furthermore, changing the semi-major axis also requires far less fuel than initializing an inclination change required for a reasonably fast relative RAAN drift. Therefore, it can be assumed without any loss of accuracy in the final objectives that the only Kepler element that changes over time is the RAAN. Fortunately, Equations 3.9 and 3.11 can be used as an estimate for the RAAN of a satellite after a specific amount of time. Figure 4.2 shows the propagated Kepler elements of a satellite in SSO plotted together with the analytically calculated elements.

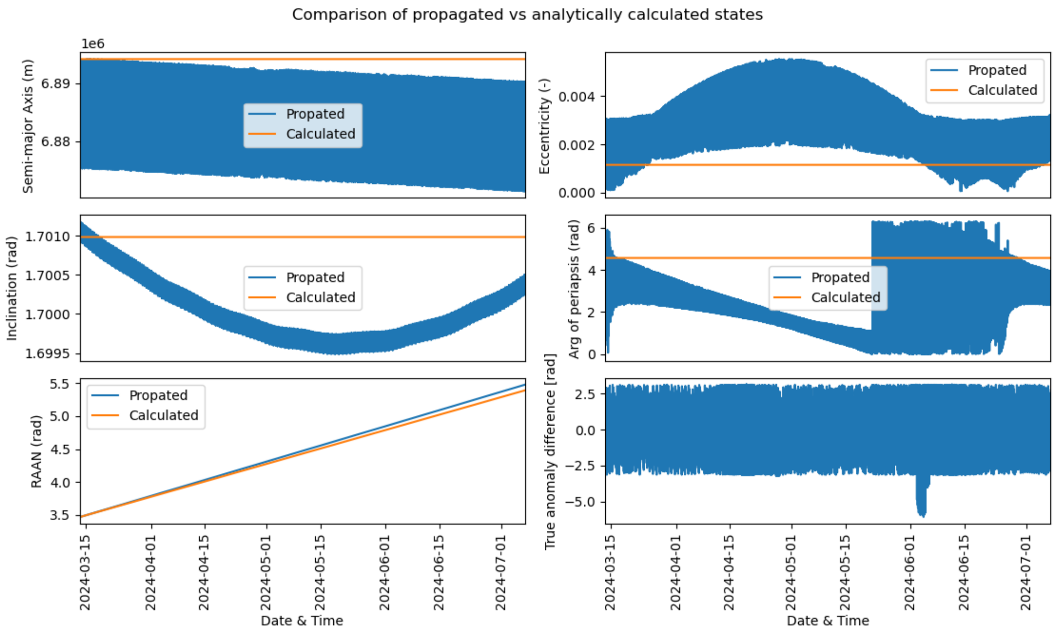


Figure 4.2: Kepler elements over time of a satellite where the states have been propagated both numerically and analytically using only Equation 3.9

As can be seen from the bottom right plot that shows the spacecraft's RAAN, the values align quite well considering that the propagation time is close to four months. After this time, the absolute error in RAAN is only about a tenth of a radian. After one month, which is a far more realistic transfer time between satellites, the difference is a mere 0.02 radians. This shows that the numerical propagation of the states in between iterations of the DTSP model can be substituted for a simple and much faster analytical estimation.

4.3.2. Static Path and Transfer Time Scaling Factor

Despite having cut down quite a chunk of the computation time, there is still a way to go. The second largest computational load of the DTSP model is the calculation of the population of possible transfers between two satellites. When targets are widely spaced, requiring an increase in the maximum permissible transfer time to secure at least one viable path, the computational load can significantly rise. To avoid having to recalculate all the possible routes for each iteration for each individual in the path optimization algorithm, the assumption could be made that the satellites are not moving at all relative to each other. If it can be shown that this static traveling salesman problem (TSP) results in the same optimal path as the dynamic one, the static TSP could then be the problem that is to be optimized, and the values of the actual route can subsequently be calculated by performing the high-accuracy calculation of the DTSP once.

However, there is one caveat: Since the static TSP does not account for the relative drift between target satellites, the distances between them increase over time, causing the model to structurally underestimate the actual fitness of any path. Therefore, what might happen is that if one tries to optimize this TSP, an optimum is found that would be infeasible in case the relative drift is accounted for, which could result in finding no optimum at all. This can be solved by finding a way in which the solutions of dynamic and static problems can be linked to each other. Continuing on the assumption that the only Kepler element that is relevant for the determination of the objectives is the RAAN, this link can be found.

Since Equation 3.11 has no time as a variable in it, it means that the drift rate due to the oblateness of the Earth is constant. Therefore, if the relative drift rate between two satellites is known and constant, the total change in RAAN over time is easily determined by multiplying the two. Finally, because for any specific amount of fuel that is used, the achievable change in inclination is fixed, meaning that the achievable RAAN drift rate is also fixed, there is also a linear relationship between the transfer time and the RAAN difference between the two satellites. This gives all the ingredients that are needed to relate the dynamic and static models to each other.

Given any initial RAAN difference $\Delta\Omega_0$ and a corresponding relative drift rate between the same satellites $\frac{d\Delta\Omega}{dt}$, the change in $\Delta\Omega$ as a function of time is simply given by

$$\Delta(\Delta\Omega) = \frac{d\Delta\Omega}{dt} \cdot t \tag{4.1}$$

Therefore, the relative difference in RAAN $\Delta\Omega$ as a function of time is given as

$$\Delta\Omega = \Delta\Omega_0 + \frac{d\Delta\Omega}{dt} \cdot t \tag{4.2}$$

Finally, since there is a linear relationship between transfer time T_{trans} and $\Delta\Omega$, relation 4.4 is valid.

$$\frac{T_{trans,2}}{T_{trans,1}} = \frac{\Delta\Omega_2}{\Delta\Omega_1} \tag{4.3}$$

This finally results in

$$\frac{T_{trans,2}}{T_{trans,1}} = \frac{\Delta\Omega_0 + \frac{d\Delta\Omega}{dt} \cdot t}{\Delta\Omega_0} = 1 + \frac{\frac{d\Delta\Omega}{dt}}{\Delta\Omega_0} \cdot t \tag{4.4}$$

With Equation 4.4, it is possible to estimate the transfer time between two satellites after some time t, based on the initial condition and the transfer time at t_0.

To verify that this relationship is valid, the actual transfer time factor was plotted as a function of the change in $\Delta\Omega$ for multiple constant fuel values per transfer, as well as for multiple transfers itself. Subsequently, a line was fitted through this data such that an estimate of the initial difference in RAAN could be made.

Figure 4.3 shows the results of one of the transfers, where the values given to each line are the initial transfer times. Furthermore, table 4.1 shows the values of the actual initial RAAN differences and the calculated initial RAAN differences.

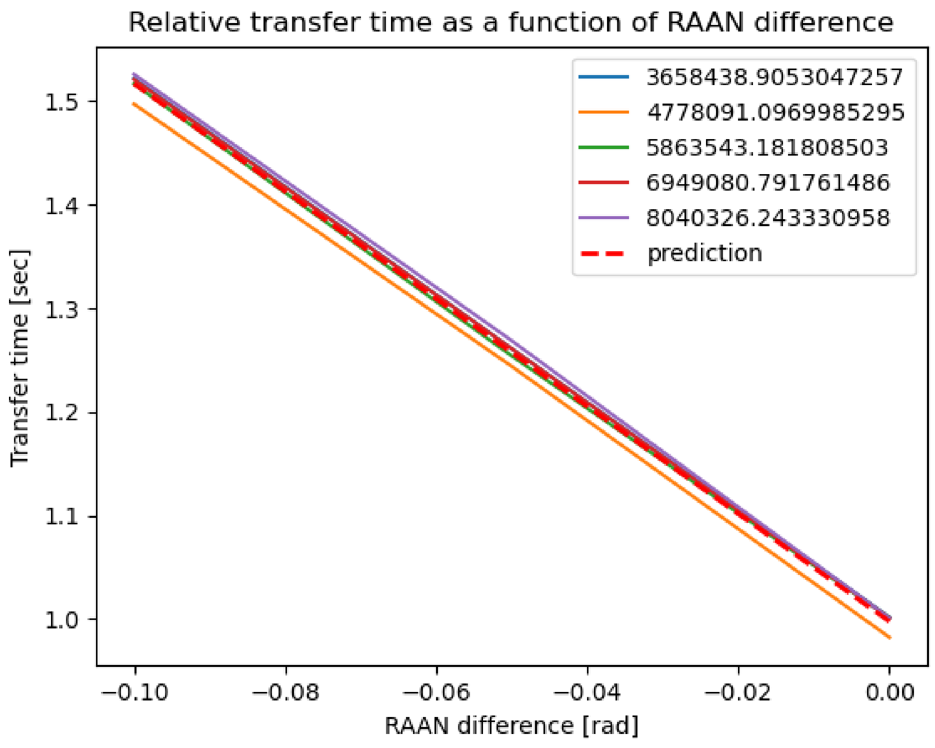


Figure 4.3: Transfer time factor as a function of change in initial RAAN difference for multiple initial RAAN differences

Target	Estimated Coefficients	Fitted Coefficients	Relative Difference
#1	-0.193	-0.193	-0.308%
#2	-0.094	-0.093	-0.652%
#3	-0.244	-0.244	0.195%
#4	-0.148	-0.147	-0.605%
#5	-0.147	-0.145	-1.409%
#6	-0.138	-0.138	-0.619%

Table 4.1: Comparison of Estimated and Fitted Coefficients

Since the relative difference between the estimated coefficients and the fitted coefficients does not exceed 1.5 %, it is concluded that the proposed method of using a transfer time scaling factor on top of the static TSP results in valid solutions.

4.4. Opimization of the DTSP

Now that the DTSP model has been defined, it can be optimized. Whereas Pygmo provided five optimization algorithms for the single transfer problem in Section 3.1.4, for this model it can only provide two: NSGA-II and MACO. This is due to the mixed integer and nonlinear (MINLP) properties of the DTSP. Especially the mixed-integer part is why the optimizers to choose from are constrained. Due to the fact that only discrete integer values can be provided as decision values for the path of choice and fuel station selection, only optimization algorithms that can handle these discrete values can be used. That the to be optimized model contains integer values in its decision vector can be done by implementing a third method next to the ’fitness’ and ’get_bounds’ methods which is called ’get_nix’, which should return the number of discrete decision values present in the decision vector.

4.4.1. Optimizer Selection for the DTSP

To make a proper choice of optimizer, the two algorithms are compared in the same way as was done in Section 3.1.4. Despite having experimented with many optimization settings, it was not achieved to get the MACO optimizer to converge. Therefore, the NSGA-II algorithm is chosen as the most suitable optimization algorithm for the implementation of the DTSP described in this report.

4.4.2. Tuning of the Optimization

In optimizing the model using the NSGA-II algorithm within Pygmo, it is crucial to carefully tune the algorithm parameters to ensure that the evolutionary process efficiently converges towards an optimal set of solutions. NSGA-II operates by evolving a population of potential solutions over several generations. Each generation undergoes processes of selection, crossover, and mutation, which are governed by specific parameters that must be finely adjusted to align with the model's optimization requirements.

One of the main challenges in this tuning process is finding the right balance between exploration and exploitation within the solution space. If the parameters are not tuned correctly, the algorithm might either converge prematurely to suboptimal solutions or fail to converge altogether, leading to a diverse set of solutions that do not adequately approximate the true Pareto front. Therefore, a systematic approach to parameter tuning is essential to achieve the desired optimization results.

Key parameters include population size, which determines how many candidate solutions are considered in each generation. A larger population can explore the search space more comprehensively, potentially avoiding local optima, but this comes with increased computational demands. The crossover probability controls the likelihood of combining parent solutions to produce new offspring, which influences the genetic diversity of the population. Although higher crossover rates can enhance diversity, they can also disrupt well-adapted solutions, potentially slowing convergence.

Similarly, the probability of mutation plays a critical role in introducing random variations into the population, preventing stagnation, and ensuring a broader exploration of the solution space. However, too high a mutation rate can lead to excessive randomness, undermining the algorithm's ability to refine existing solutions. The number of generations directly impacts the total computational time and the quality of the solutions, where more generations allow for more refined results but at the cost of longer processing times.

In summary, the process of tuning the NSGA-II optimizer for this specific model requires a thoughtful and methodical approach. By calibrating the population size, crossover and mutation probabilities, and the number of generations, the optimization process can be steered to efficiently converge on high-quality solutions that both adequately approximate the Pareto front and align with the model's specific objectives.

In the following, the parameters will be varied one by one, assuming that their effect on the quality of the optimization is independent of each other.

Crossover Rate - Cr

The first parameter to be tuned is the crossover rate, which is given the standard value of 0.95 by Pygmo. The crossover rates that will be analyzed in this section are 0.5, 0.75, 0.85, 0.9, and 0.95. The effect of changing these values on the Pareto optimal front is shown below in Figure 4.4

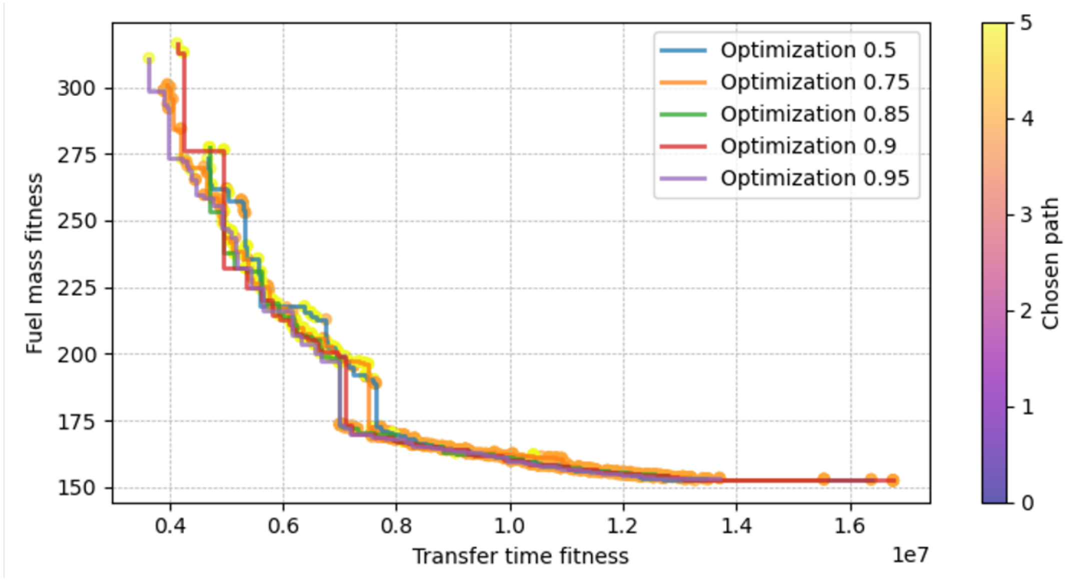


Figure 4.4: Pareto optimal front of the final populations for different values of crossover rate.

As can be seen from this figure, not much difference is observed between the different values of the crossover rate, which means that the model is very robust regarding the crossover rate.

However, examining the convergence rates presented in Figure 4.5 for various values, a notable difference can be observed. Given that a higher convergence rate is preferable, the crossover rate of 0.95 is selected as it yields the optimal convergence.

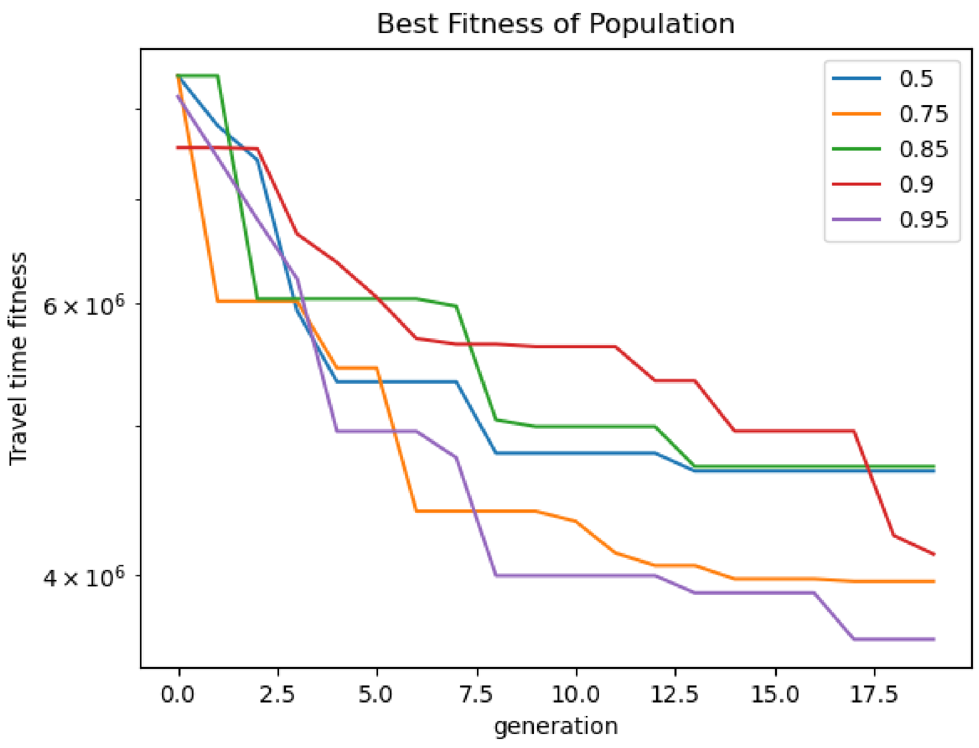


Figure 4.5: Transfer time fitness as a function of number of evolutions for all different values of crossover rate.

Mutation rate - M

A similar approach is used for the mutation rate M. The standard value that is used by Pygmo is 0.01. The values that will be investigated are 0.01. 0.05, 01, 0.2, and 0.5. The results are shown in figure 4.6.

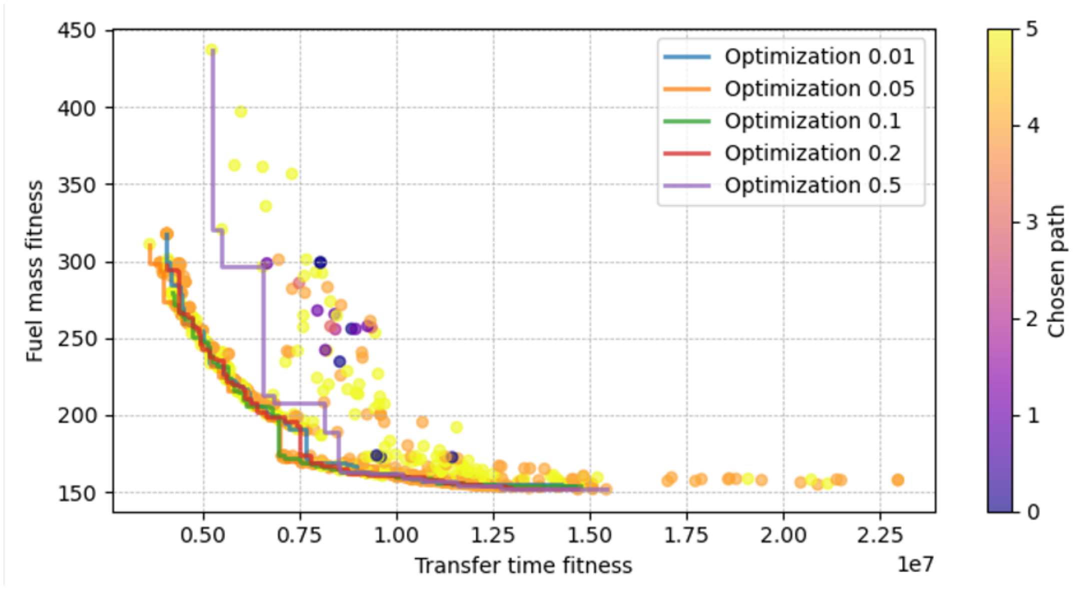


Figure 4.6: Pareto optimal front of the final populations for different values of mutation rate.

In this figure, a more clear distinction can be seen between the Pareto optimal fronts for different mutation rates. Especially the front with a mutation rate of 0.5 is far from the optimal solution. This is due to the fact that the mutation rate is so high that the optimization algorithm struggles to converge to a solution because it is migrating out of it again.

Looking closely at the far left of the front, a slight distinction can be seen between the front with a mutation rate of 0.05 compared to the others. Therefore, this mutation rate is chosen instead of the standard 0.01 provided by Pygmo.

This improvement can be attributed to the fact that the model has a relatively high chance of returning fitness values that have a penalty. Therefore, when the mutation rate increases, the model has a greater chance of mutating to the feasible region.

This choice is backed up by the convergence rate shown in Figure 4.7 below, where it can clearly be seen that the optimization with a mutation rate of 0.05 results in the fastest convergence rate.

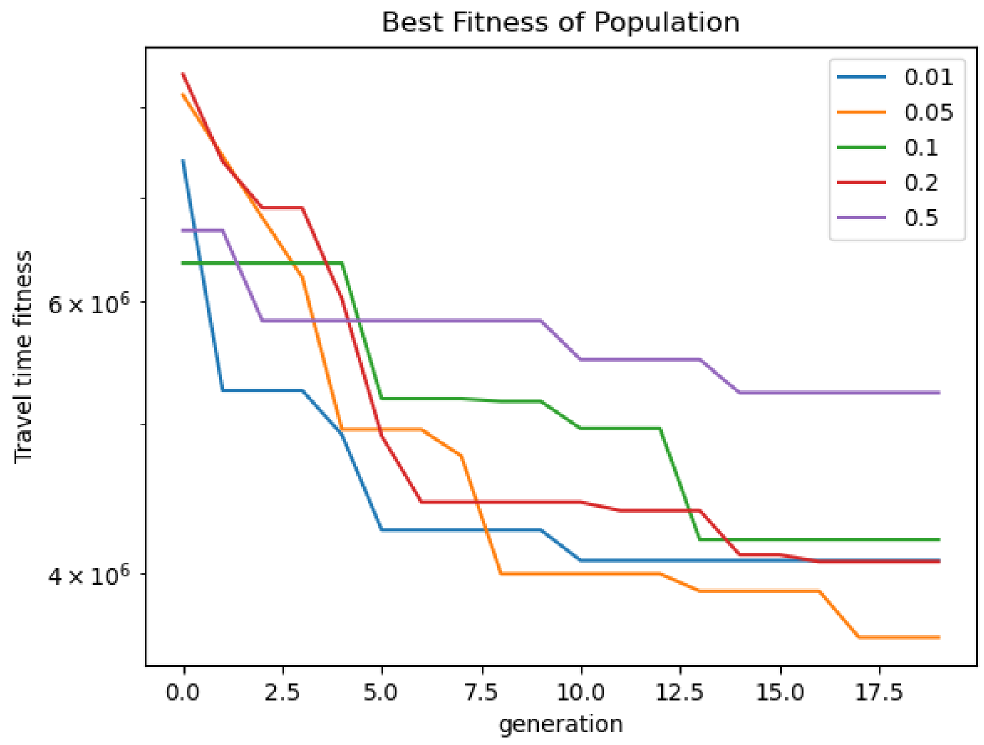


Figure 4.7: Transfer time fitness as a function of number of evolutions for all different values of mutation rate.

Population size
The next value that will be tuned to see which will result in the desired amount of convergence with as minimal computation effort as possible is the population size. The values chosen for this analysis are 40, 80, 200, and 500 individuals. The results are shown in Figure 4.8.

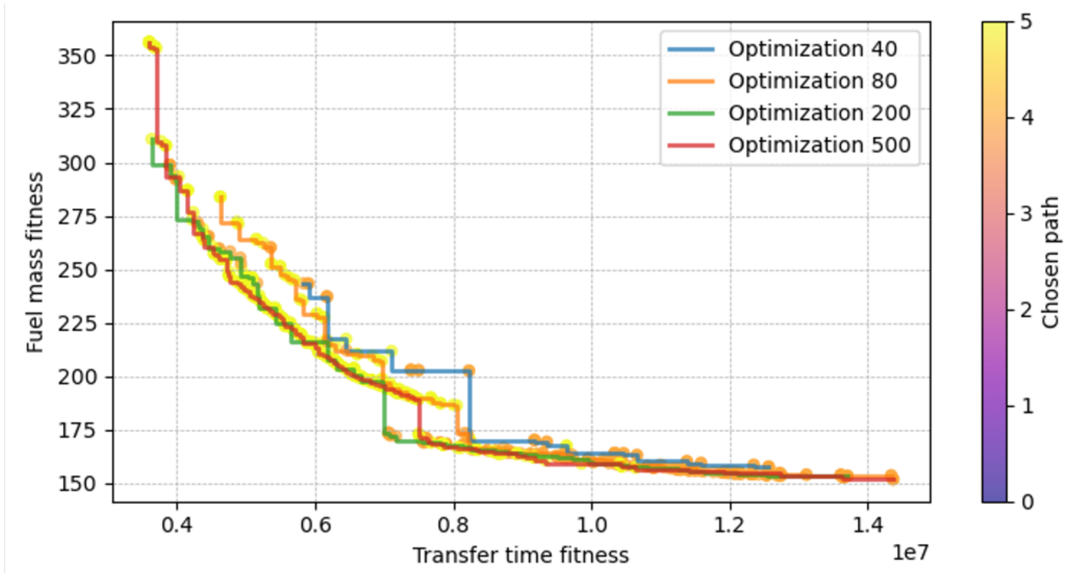


Figure 4.8: Pareto optimal front of the final populations for different population sizes.

Looking at the above figure, it is observed that there is a subtle difference in performance between the four values. The largest population size of 500 performed the best, as expected. However, the Pareto optimal front of a population size of 200 almost exactly resembles the more precise front of 500 individuals. Since this is a difference of a factor 2.5 that has no significant improvements, the optimal number of individual is chosen to be 200. This behavior is backed by the convergence rate plotted below which shows that the optimization that uses 200 individuals converges to the same fitness value as the optimization that uses 500 individuals.

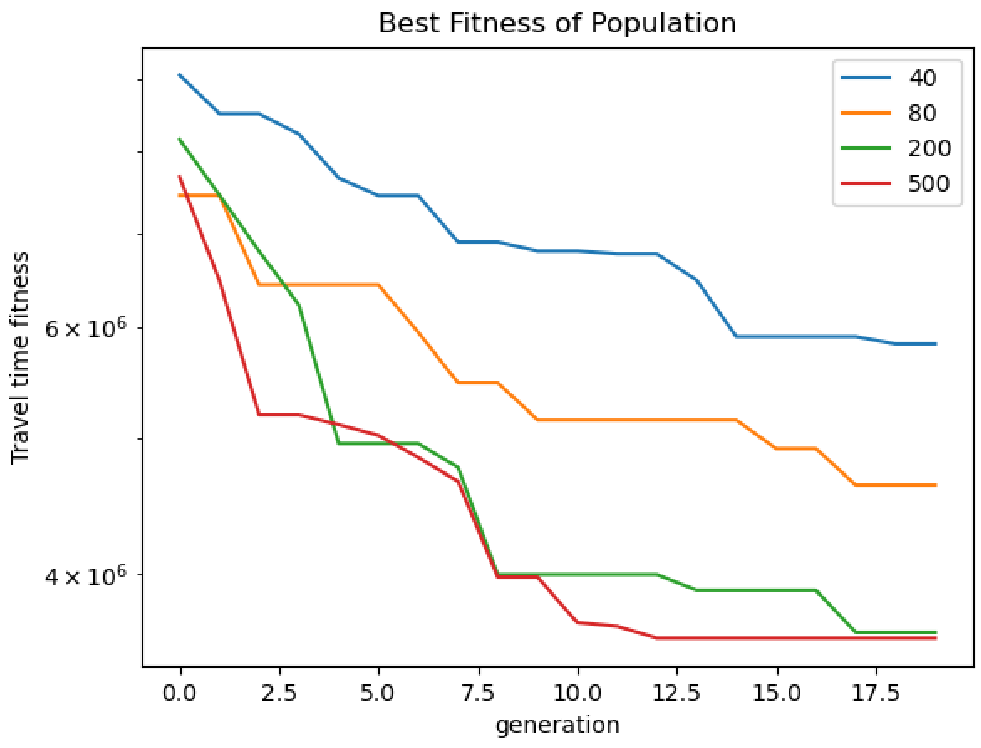


Figure 4.9: Transfer time fitness as a function of number of evolutions for all different values of population sizes.

Number of Generations

The last adjustable parameter is the number of generations for which the optimization process is executed. The numbers chosen to analyze are 20, 40, and 80. Figure 4.10 shows the Pareto fronts for all optimizations.

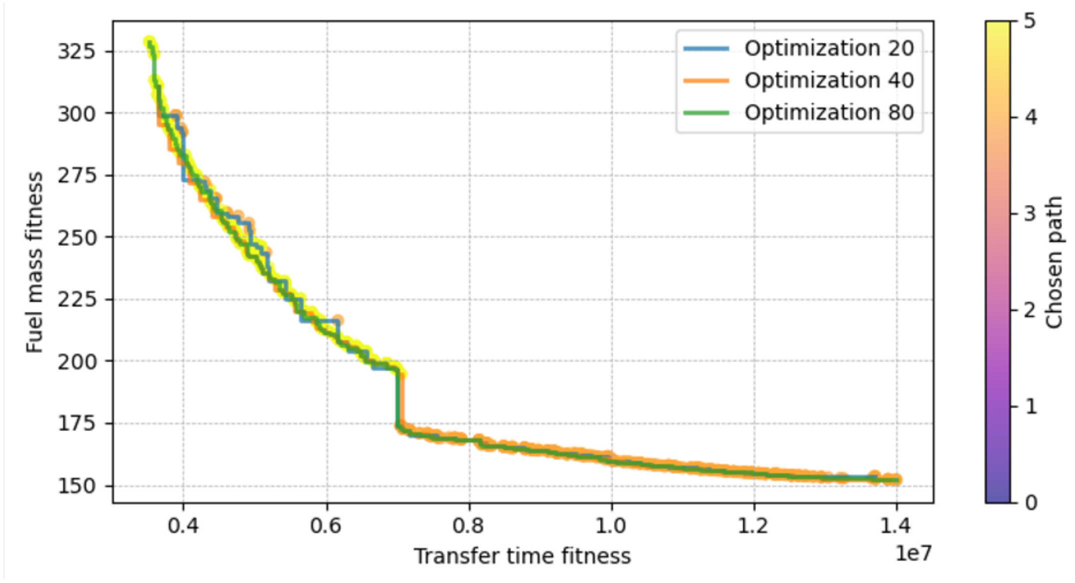


Figure 4.10: Pareto optimal front of the final populations for different number of generations

In this figure, there is virtually no distinction between the three results. Only a small number of fluctuations are visible in the data of 20 generations. However, based on the observed amount of fluctuation in the tuning results of the previous parameters, 20 generations is deemed accurate enough.

Figure 4.11 below confirms this, as no significant improvement in the best value of fitness in travel time after 20 generations can be seen.

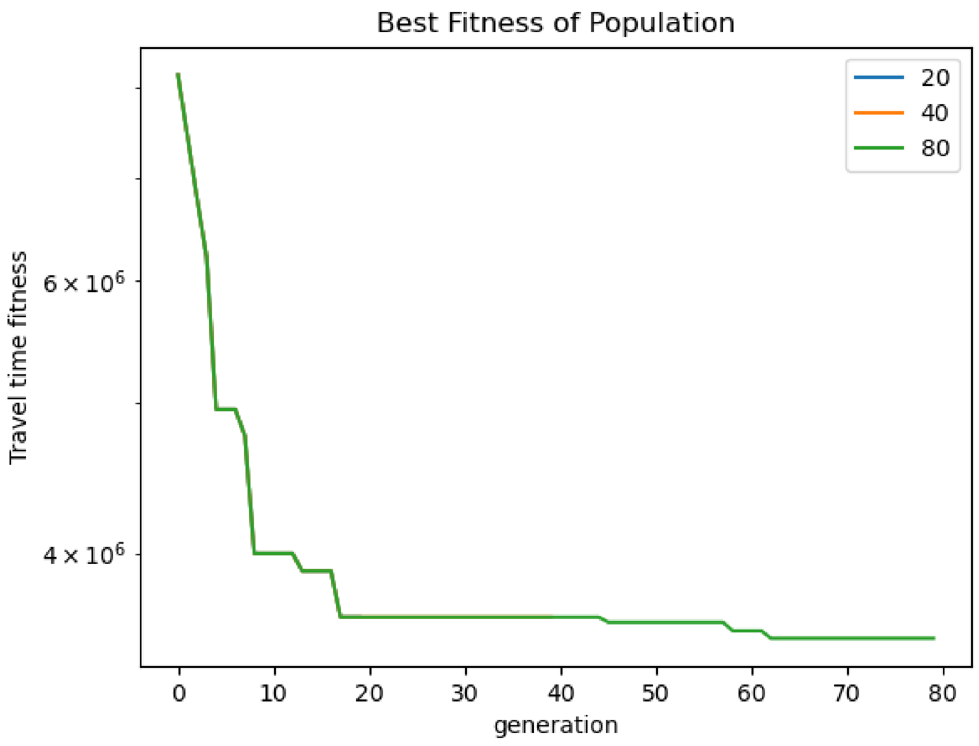


Figure 4.11: Transfer time fitness as a function of number of evolutions for all different values of total number of evolutions.

4.5. Results

The final results of the optimization algorithm are provided in this section. The relevant satellite data that is put into the DTSP optimization algorithm are given in Tables 4.2 and 4.3 below. They show the same structure as the actual data that will be fed into the DTSP optimization, which is also displayed in the flow diagram shown in Figure 4.1. Each table represents a dictionary with each column being a key-value pair. The initial epoch and initial state are the first element of the list that is the value. Per definition of Tudat-Py, the epoch is given in seconds since J2000 and the initial state can be given as Cartesian or Keplerian elements. The initial epoch and state also form a dictionary in order to maintain the same structure in which Tudat-Py represents the state history of a body. The required fuel mass is the fuel mass in kilograms that the client satellite needs. As it is most interesting to see the DTSP optimization work to its fullest potential, no distinction between priorities is made between the client satellites. Therefore, the date of the request for the servicer is set to zero (no need to be serviced within a specific time), and the priority level is set equal to one for each client. Finally, for the determination of the transfer time factor discussed in Section 4.3, the RAAN drift rate of the client satellites is also required. This can be calculated directly from the satellite TLE data.

Together with their initial epoch and state, the fuel stations as given by Table 4.3 have only their total available fuel as input value, as well as their RAAN drift rate.

Parameter	Aether-1	Aether-2	FALCONSAT-10
Initial Epoch	765613345.47 [s]	765609605.84 [s]	765611742.44 [s]
Initial Kepler State	6.889×10^6 [m] 1.085×10^{-3} [-] 1.701 [rad] 2.172 [rad] 3.010 [rad] -2.170 [rad]	6.888×10^6 [m] 1.191×10^{-3} [-] 1.701 [rad] 2.042 [rad] 3.010 [rad] -2.040 [rad]	6.897×10^6 [m] 1.206×10^{-3} [-] 1.701 [rad] 2.064 [rad] 3.005 [rad] -2.062 [rad]
Required Fuel Mass	5 [kg]	10 [kg]	3 [kg]
Date of Service Request	0	0	0
Priority Level	1	1	1
RAAN Drift Rate	1.99×10^{-7} [rad/s]	1.99×10^{-7} [rad/s]	1.99×10^{-7} [rad/s]

Table 4.2: Clients Data

Parameter	FS1	FS2
Initial Epoch	765625932.55	765605717.15
Initial State	7.027×10^6 [m] 1.015×10^{-3} [-] 1.706 [rad] 1.348 [rad] 2.960 [rad] -1.346 [rad]	6.892×10^6 [m] 1.241×10^{-3} [-] 1.701 [rad] 2.096 [rad] 3.007 [rad] -2.094 [rad]
Available Fuel Mass	1000 [kg]	1000 [kg]
RAAN Drift Rate	1.93×10^{-7} [rad/s]	1.99×10^{-7} [rad/s]

Table 4.3: Fuel Stations Data

The Pareto optimal fronts of the final population of the DTSP optimization are shown in Figure 4.12.

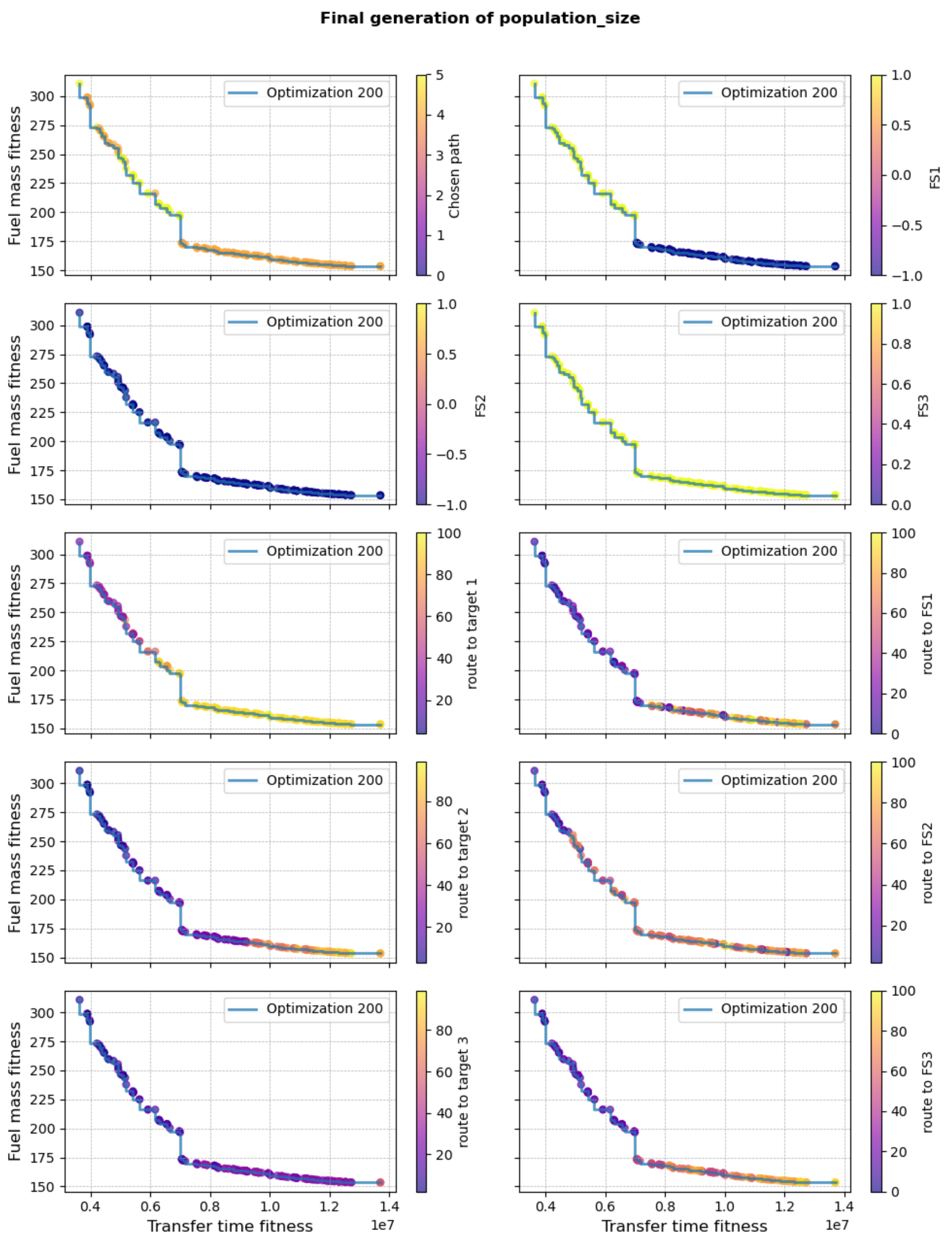


Figure 4.12: Pareto front of the final population of the DTSP optimization

Figure 4.12 shows a total of ten Pareto optimal fronts: one for each decision variable. Each individual has been given a color based on the value of the considered decision variable as indicated by the colorbar on the right side of each subplot. So, for example, which fuel station - if any - the servicer visits after the second client satellite is indicated by the left subplot in the second row. As can be seen, all the individuals have a dark blue color corresponding to the value of -1, indicating that none of the individuals visits a fuel station after their second client.

Three important observations are made from Figure 4.12. The first observation is made in the first subplot that shows the chosen path, where it is clearly visible that in the upper left region of the Pareto front, either path four or five is the optimal path instead of there being a single optimal path as expected. This phenomenon occurs because paths four and five both initially go to the last client. Subsequently, path four proceeds to the first client before heading to the second client, whereas path five visits the second client first and then the first client. Because in the example data, the first two satellites are Aether-1 and Aether-2, which are relatively close to each other, there is only a very slight difference in fitness values between these two paths. Therefore, both paths appear on the Pareto optimal front. In case more generations are performed or a larger population size is taken as in Figures 4.10 and 4.8 respectively, it is found that the optimal path converges to a single value which in this example is path five.

The second observation is made in the second subplot that shows which fuel station, if any, the servicer will visit after the first client satellite. From this figure, the reason for the clear jump in the Pareto front becomes clear: In this example, the fastest option is to first refuel the servicer after the first client, before proceeding to the next client satellite. However, there is a point where the time it takes to visit all three clients directly is the same as first refueling the servicer before heading out to the second client. Because skipping this refueling of the servicer results in no fuel being required for the transfer from and to it, the required fuel mass for the total transfer has a discrete drop at this turnover point.

Finally, because the final six decision variables indicate the speed of the individual legs of the transfer, and the transfer time fitness is directly correlated to these values, it is expected to see a clear trend in the values of these decision variables. Looking at the six bottom subplots of Figure 4.12, this is exactly what is observed: The upper left parts of the Pareto fronts have values that approach zero percent, indicating that these are the fastest individuals, whereas the lower right parts of the Pareto fronts have values that tend to 100 percent, indicating that these are the slowest individuals. Only the subplot showing the 'route to FS2' is showing some erratic behavior as it has mixed individuals along the front. However, this is as expected since all individuals skip the second visit to a fuel station, meaning that the values for this decision variable have no impact on the fitness of the individual.

After running the static optimization, an individual must be chosen from the final population which will represent the chosen transfer strategy. Although from a mission cost perspective it would make sense to choose the strategy that uses the least amount of total fuel, the individual with the fastest total transfer time is taken in this case because it shows the potential of the refueling infrastructure more clearly.

After choosing the transfer strategy, the individual trajectories are refined using the method described in Section 3.2.4. Stitching all the state histories of all the individual sections of the complete transfer to each other results in Figure 4.13, where the differences in the Kepler elements of all the satellites with respect to the servicer have been plotted. It should be noted that the plot showing the argument of periapsis has been omitted because it shows erratic behavior for near-circular orbits.

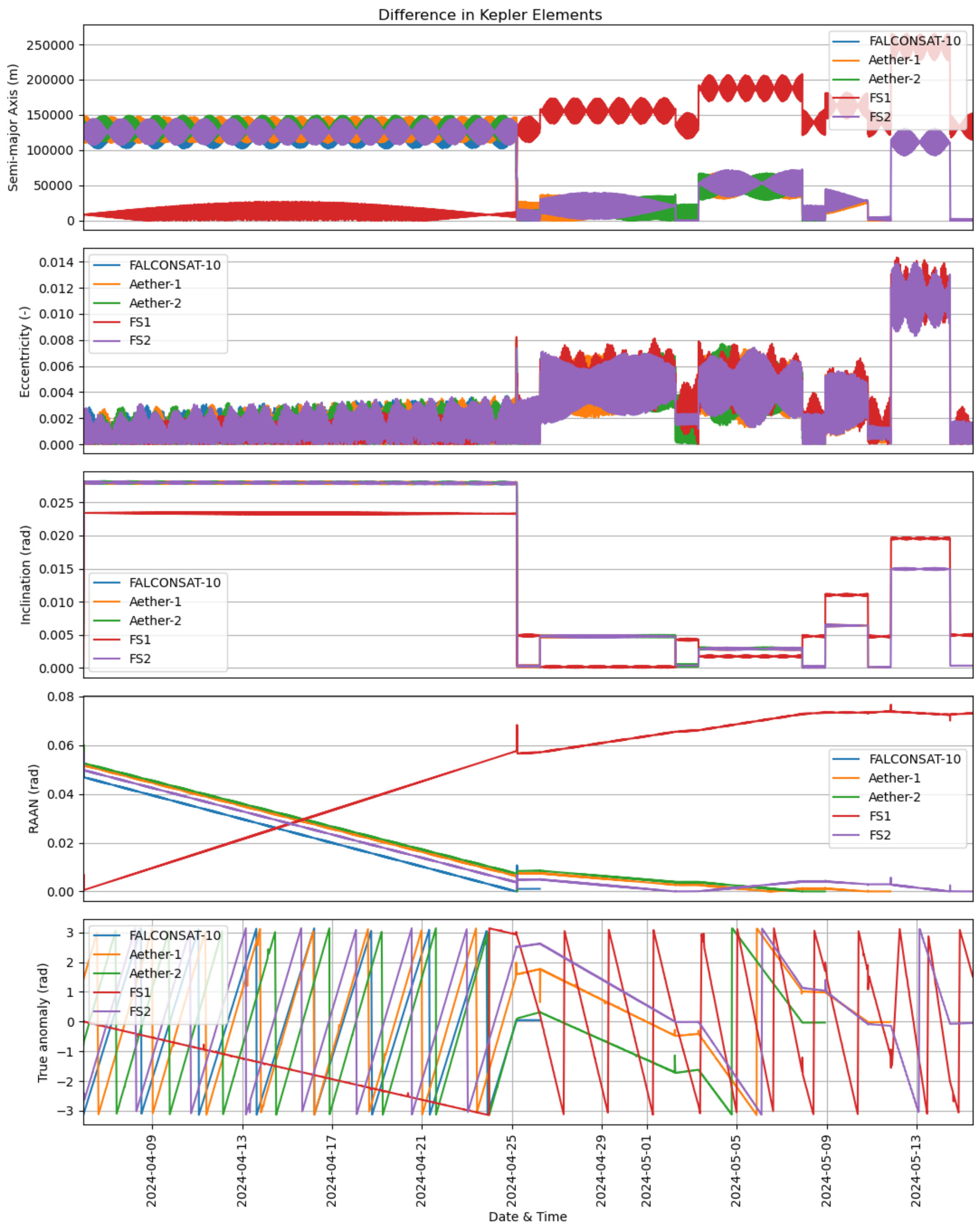


Figure 4.13: Kepler elements as a function of time for the entire refined transfer of the servicer

Figure 4.13 shows the difference in the semi-major axis, eccentricity, inclination, right ascension of the ascending node, and true anomaly between all satellites and the servicer from top to bottom. From the middle plot that shows the difference in inclination, the intervals where the orbits are actually matched can already be observed as the narrow sections of a day length where the difference reaches zero. However, to better observe the matching quality of the orbits over time, the plots are zoomed in around the zero difference line. This results in Figure 4.14.

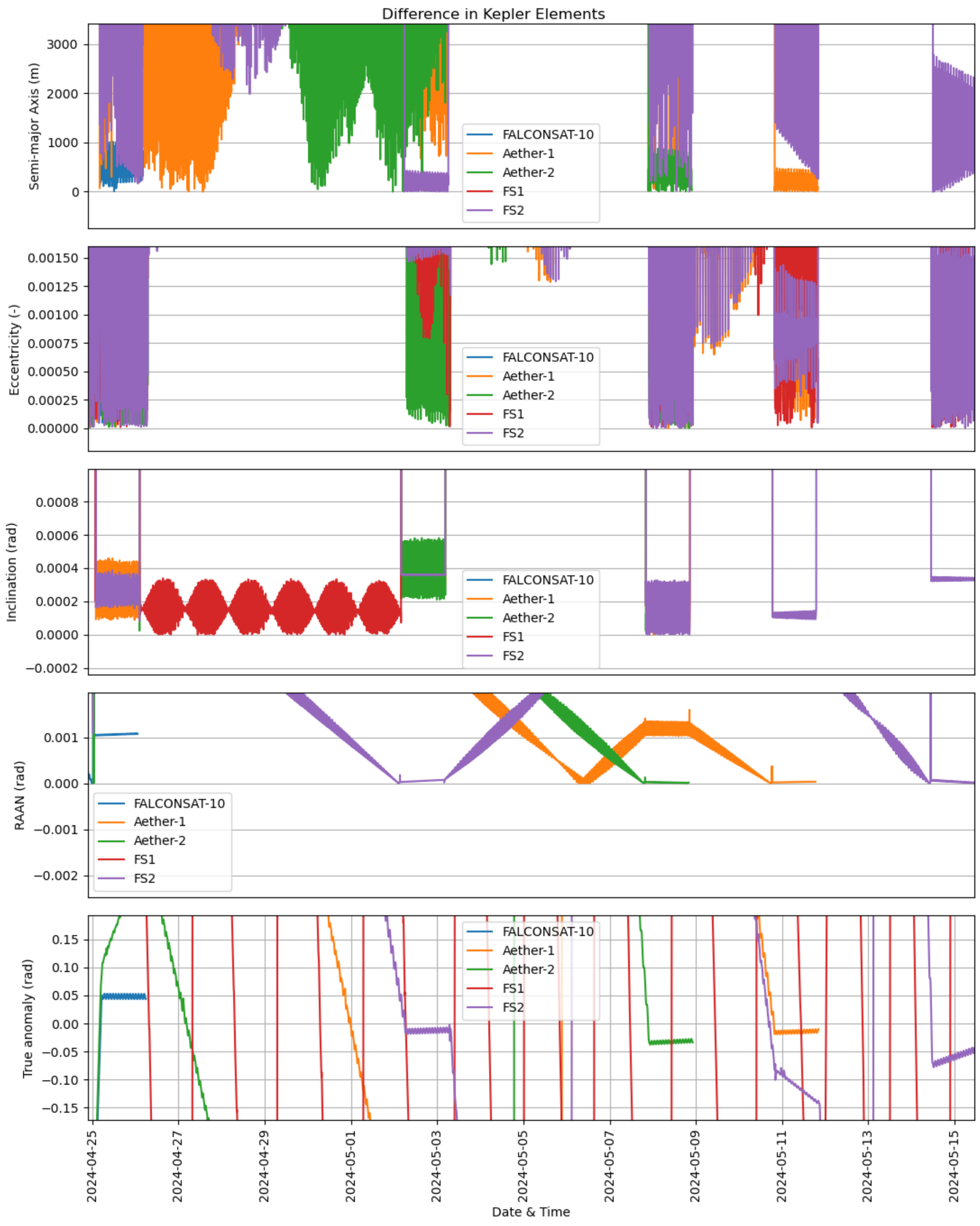


Figure 4.14: Kepler elements as a function of time for the entire refined transfer of the Servicer, zoomed in on the actual orbit matching

From Figure 4.14, it is observed that the servicer first visits FALCONSAT-10. Next, it refuels at fuel station 2 (FS2), after which it goes directly to Aether-2 and Aether-1, respectively. Finally, it makes its way back to FS2 to refuel and wait until another service request arrives. From the two lower subplots, it is observed that the combination of the DTSP and the trajectory refinement results in a continuous path that visits all targets. This conclusion can be drawn from the fact that the difference in these Kepler elements between the servicer and the other satellites all become approximately zero at some point in time. In addition, these differences become zero simultaneously for any specific satellite, indicating that the orbits are correctly matched. The proper matching of the semi-major axis is also observed, as when the true anomaly and RAAN of any satellite are matched, the semi-major axis difference is also at a relatively constant minimum. Finally, the matching quality of the eccentricity and the inclination is less obvious but still present. Since the target orbits are almost circular, the difference between the eccentricities all become approximately zero when the servicer is at a client or fuel station, since it then also has an eccentricity of almost zero. From the subplot showing the difference in eccentricity in Figure 4.13, it is observed that the difference in eccentricity increases significantly between each servicing, indicating that the transfer trajectory of the servicer has become more elliptic in order to achieve correct phasing with the next target. A similar line of reasoning holds for the inclination, as all the satellites considered have the same approximate inclination. From the subplot that indicates the difference in inclination in Figure 4.13, the difference increases significantly between each servicing, again indicating that the inclination of the transfer trajectory of the servicer has changed to achieve a correct RAAN drift rate.

Although this section shows that the DTSP optimization converges and results in a trajectory of the servicer that visits all client satellites and fuel stations as intended, it does not confirm that the found optimum is actually the global optimum. It could very well be the case that the optimization algorithm has converged to a local optimum instead of the global optimum. Therefore, the next chapter focuses on the verification of the DTSP results.

5 Verification

Verification is a critical step in any optimization process, ensuring that the solutions generated by the model are not only theoretically optimal but also practically correct and reliable. In the context of satellite mission planning, where decisions are made based on complex models, it is essential to verify the optimization results to confirm that they accurately represent the best possible solutions within the given design space. Without proper verification, there is a risk that the solutions identified by the optimization algorithm could be suboptimal or, worse, infeasible when implemented in a real-world scenario. The first verification of the results, which is performed in Section 5.1, is applying a brute force method to determine all possible transfers within the design space, after which its Pareto optimal front is compared to the optimized front. Next, in Section 5.2, the key points in the Pareto front are analyzed to verify them. Finally, in Section 5.3, the Pareto front obtained from the static optimization is calculated for the model where no assumptions are made to verify the validity of the assumptions.

5.1. Brute Force Analysis

To verify the optimization model used in this project, a brute-force analysis of the entire design space is performed. This method involves evaluating all possible combinations of the decision variables to generate a complete dataset of potential solutions. While computationally intensive, this approach provides a comprehensive view of the design space, ensuring that no potential solutions are overlooked.

The next step involves comparing the Pareto front obtained from the brute-force analysis with the Pareto front generated by the NSGA-II optimization algorithm. By comparing these two Pareto fronts, it is possible to determine whether the optimization algorithm has correctly identified the true set of optimal solutions. If the optimization model is correct, the Pareto front found through brute-force analysis should match the Pareto front generated by the optimization algorithm. Any discrepancies between the two would indicate potential issues with the optimization process, such as convergence to local optima or insufficient exploration of the design space.

The results of the brute force analysis are shown in Figure 5.1.

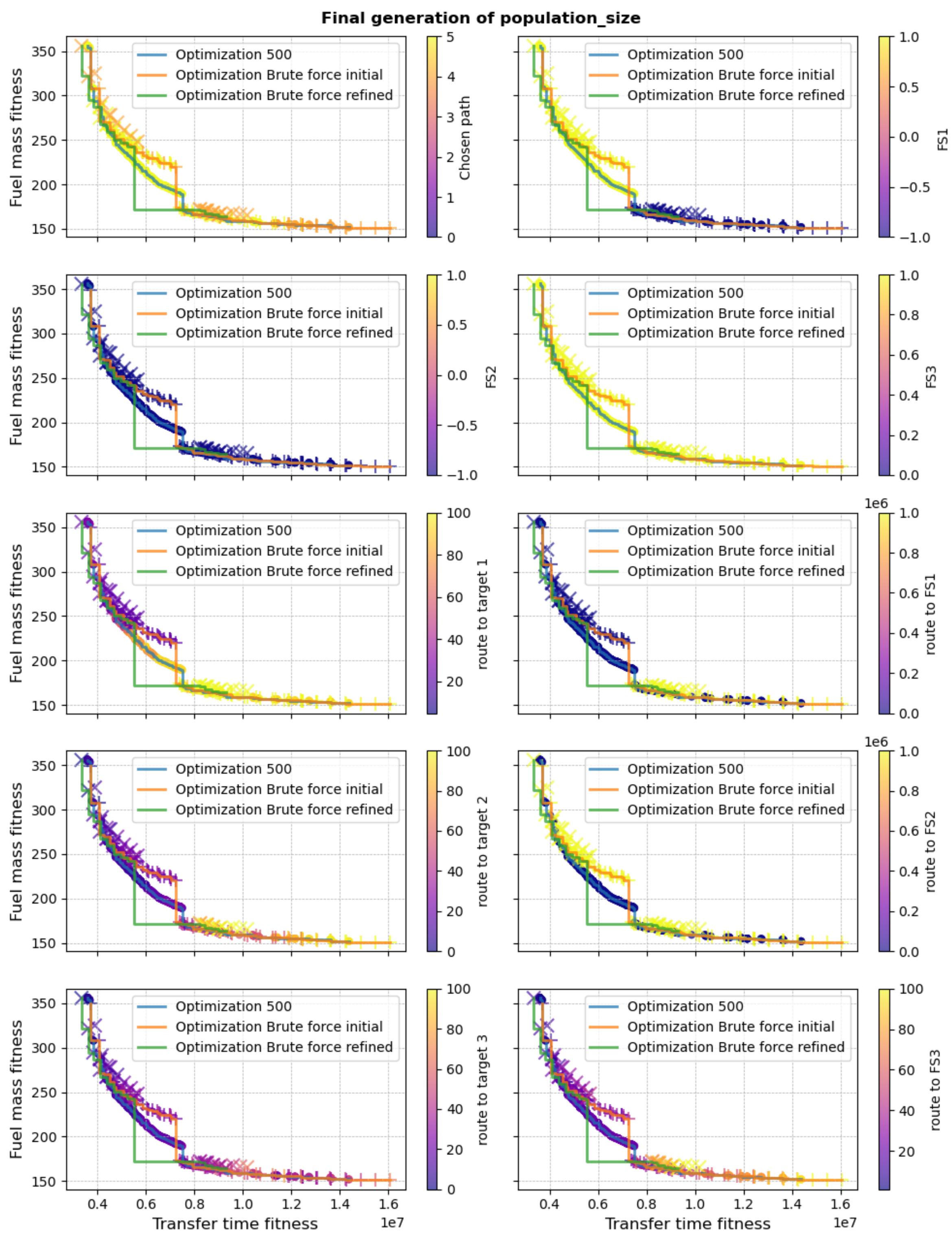


Figure 5.1: Pareto front resultant from the optimization process versus the Pareto front of the brute force calculation

The most immediate observation in Figure 5.1 is the overall alignment between the Pareto front obtained from the NSGA-II optimization and those generated by the brute force method. The Pareto fronts from the optimization and brute force analyses follow a similar curve, indicating that the optimization algorithm successfully identified solutions that are very close to the true Pareto-optimal solutions within the design space.

However, there are slight deviations in the middle regions of the Pareto fronts. The brute-force front remains significantly higher than the optimization front. However, this behavior can be attributed to the fact that not the entire decision space was taken, but for each single transfer route value from the decision vector, ten evenly spaced samples were taken, since the computation would otherwise be too long. Therefore, there will be more optimal solutions available that are not shown in this Pareto front.

5.2. Key Points in the Pareto Front

The next step in verifying the results is to check what the specific intermediate values are of paths taken by a couple of key points in the Pareto front. These points are indicated in the following figure by the crosses.

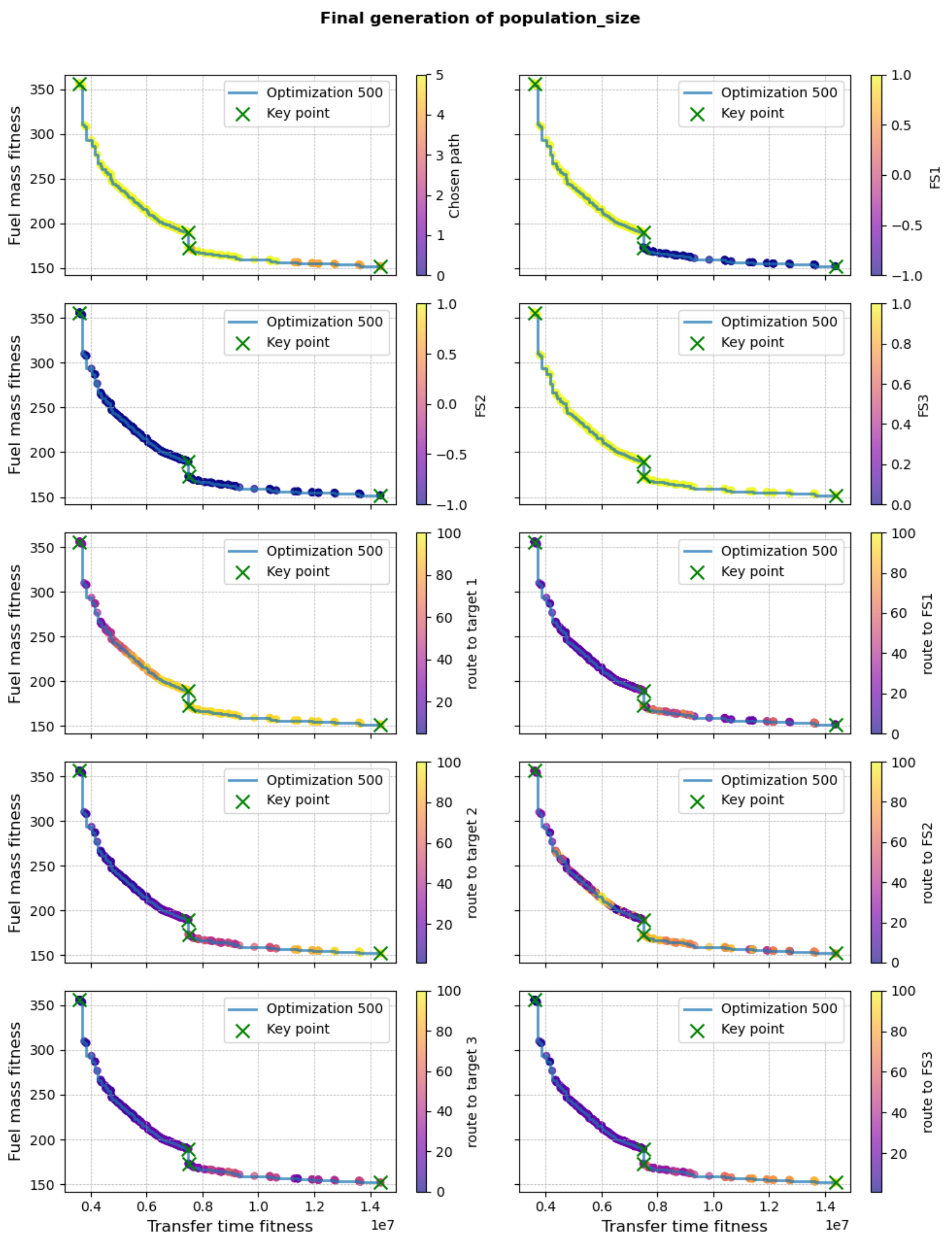


Figure 5.2: Pareto front of optimization with key points highlighted

From left to right, the points indicate the following:

1. The leftmost point indicates the fastest possible transfer. This individual should somewhere along its path be very close to having zero fuel left.
2. The next point is the last point where it is faster to use more fuel by visiting a fuel station for

refueling, before continuing on to the next client. This is very clearly visible from the upper right subplot, where all the points right from the drop off are dark blue, indicating that they skip that fuel station.

3. The point on the bottom of the drop is the point that has just enough fuel to make it past all client satellites faster than it would be to refuel first.
4. The final point is the point that uses the least fuel possible within the specified transfer time limit.

5.3. Hich accuracy results

The final step of the verification process is to recalculate the fitness values for the individuals of the Pareto optimal front that is obtained by optimizing the static TSP, but now without any simplifying conditions imposed. If this results in a similar shape of the Pareto front as the one from the TSP, it can be concluded that the assumptions are valid.

In Figure 5.3, the Pareto fronts of the static optimization are shown, together with a set of 20 evenly sampled points (blue) for which the fitness values of the DTSP have been recalculated (green). There are two things to note from this figure.

First of all, the refined points do not stay in the neighborhood of their original point. This effect becomes more pronounced the further to the right of the Pareto front one moves. Although this is not ideal, it is clear that the shape of the front remains the same and the static Pareto front does not over- or under-estimate the capabilities of the transfer. Furthermore, the static results consistently overestimate the transfer time and only slightly underestimate the required fuel. Since transfer time is considered to be the most important objective, this is not a problem.

Secondly, points that initially resulted in a valid path, but after refinement turned out to be not, are marked by a red plus. In Figure 5.3 it is observed that the points for which this is the case are the points that push the amount of fuel that they use without refueling itself a single time. However, this is not too large of a problem as one simply needs to make sure to verify the individuals found by the static optimization.

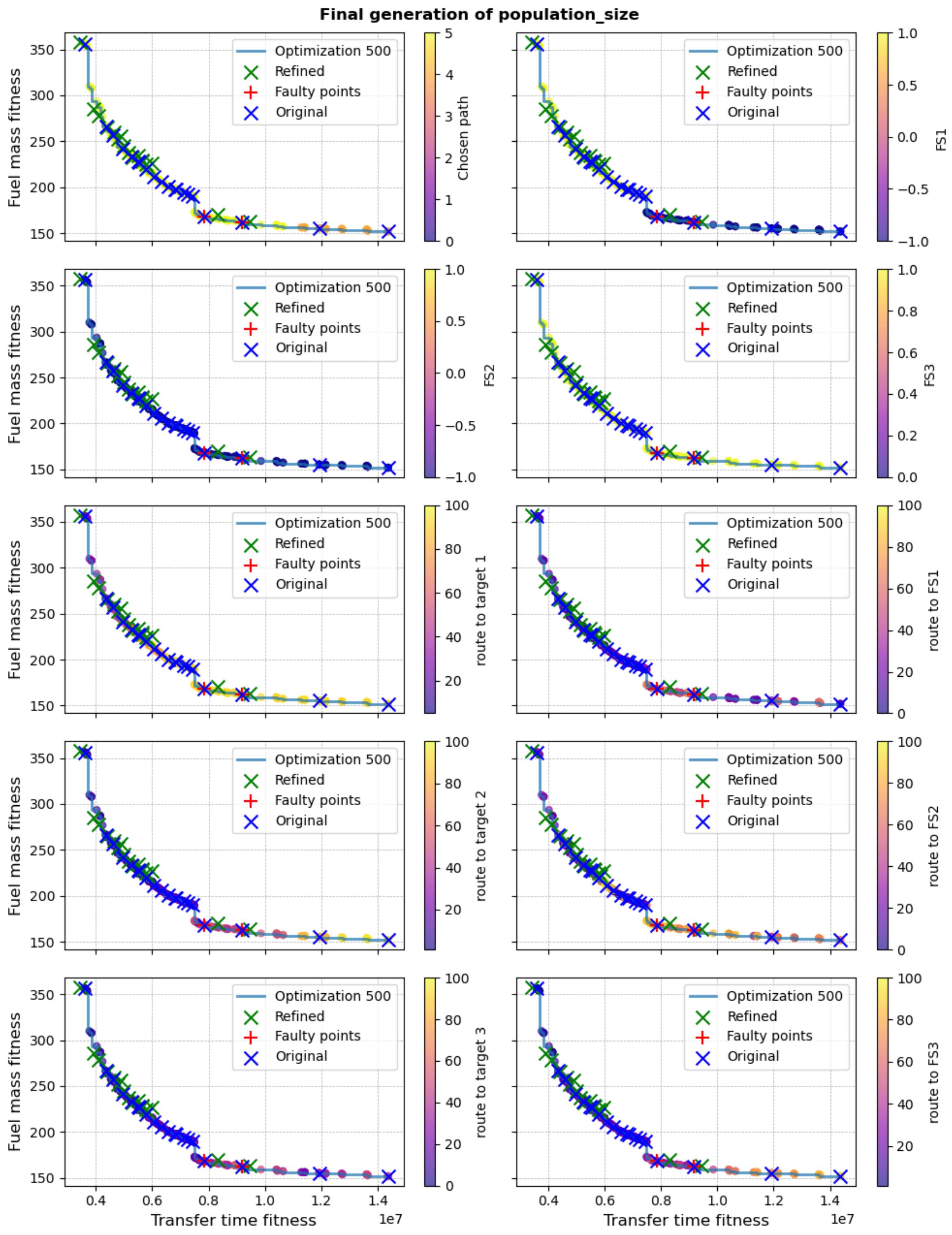


Figure 5.3: Pareto optimal front with the fitness values of some individuals recalculated without any simplifying assumptions.

6 Sensitivity and Robustness Analysis

In the context of optimizing complex models, it is not enough to simply find an optimal set of solutions; understanding how these solutions respond to variations in the model's parameters is equally crucial. Sensitivity and robustness analysis play a pivotal role in this regard, providing insight into how changes in inputs can influence outputs and affect the overall stability of solutions. This type of analysis ensures that the solutions obtained are not only optimal under a specific set of conditions but also resilient to uncertainties and variations that may arise in real-world applications.

The analysis involves systematically varying the model parameters and assessing the effects on both the objective functions and the stability of the solutions. By examining the model's responses to these variations, one can identify which parameters have the most significant impact on performance and which are less influential. This information is important for refining the model and making informed decisions about where to focus efforts to enhance its reliability and adaptability. Parameters that exhibit high sensitivity may require more precise control or further investigation to ensure that the model remains effective across a range of possible conditions.

The goal of sensitivity and robustness analysis is to evaluate the stability and adaptability of the optimal solutions under different perturbations or uncertainties. A solution is considered robust if it maintains its optimality or near-optimality despite changes in input parameters. This process helps determine the degree to which solutions can be trusted to perform well in unpredictable or fluctuating environments. It provides a measure of confidence that the solutions are not overly tailored to a specific scenario, but are instead adaptable and reliable under diverse conditions.

In this chapter, three different analyses will be performed. In Section 6.1, the seed used to generate random numbers within the optimization algorithm is varied. Next, in Section 6.2, the elements of the initial state of a satellite are slightly modified to introduce uncertainty, which will inherently be present in the real-world application of this algorithm. Finally, in Section 6.3, the available amount of fuel is varied to explore whether more optimal regions are nearby. An overview of the specific values used for the three analyzes is given in Table 6.1 below.

Seed (Section 6.1)	**Initial state** (Section 6.2)	**Fuel mass** (Section 6.3)
8	dx = 10 [km]	180 [kg]
42	dy = 10 [km]	200 [kg]
123	dz = 10 [km]	220 [kg]
	dvx = 10 [m/s]	
	dvy = 10 [m/s]	
	dvz = 10 [m/s]	

Table 6.1: Specific values used for sensitivity and robustness analysis

6.1. Variation of the Seed

The seed in optimization algorithms, such as NSGA-II, not only ensures reproducibility by setting a starting point for the random number generator, but it can also be a valuable tool for robustness analysis.

By varying the seed across multiple runs of the optimizer, you can assess the consistency and stability of the algorithm’s performance. This involves observing how the solutions change or remain stable when different seeds introduce slight variations in the optimization process. If the optimizer consistently finds similar high-quality solutions across different seeds, this indicates that the solutions are robust and not overly sensitive to the inherent randomness in the algorithm. This kind of analysis helps validate the reliability of the optimization process, ensuring that the solutions are reliable and not merely the result of a favorable random sequence.

The results of the change of seed on the optimization algorithm are shown in Figure 6.1 below.

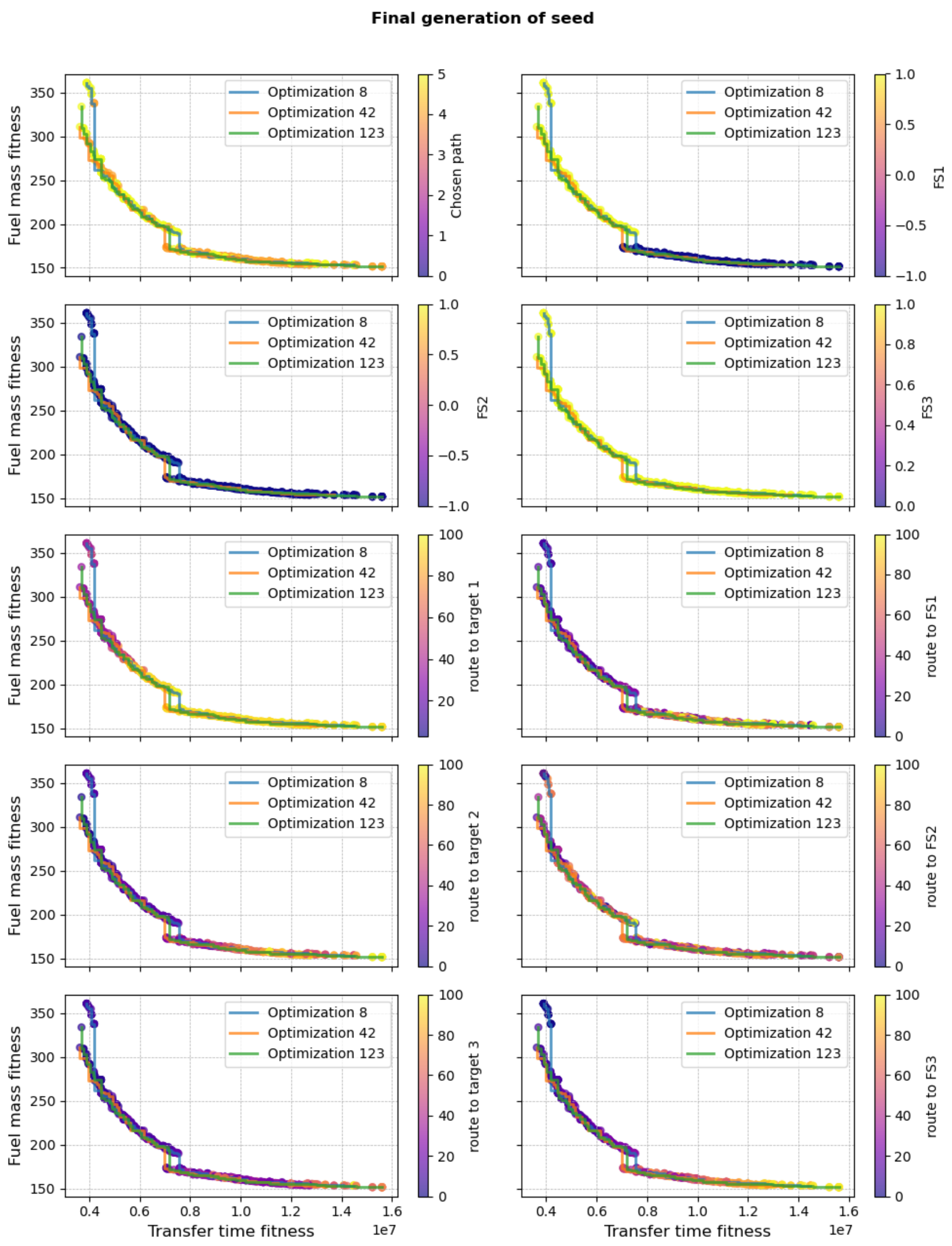


Figure 6.1: Pareto fronts of three optimizations that are all performed with different seeds.

Despite the variations in seeds, the Pareto fronts across the different optimization runs exhibit a high degree of similarity, indicating that the solutions found are robust to changes in the random seed. The solutions consistently form a Pareto optimal curve between the two objectives, with similar shapes and ranges, suggesting that the algorithm reliably converges to a similar set of optimal solutions regardless of the seed used.

6.2. Uncertainty in the initial state

In the field of satellite path optimization, it is crucial to account for uncertainties that may arise due to the dynamic nature of the space environment and the inherent limitations in satellite state measurements. Satellites in orbit are subject to various perturbative forces, including gravitational influences, solar radiation pressure, and atmospheric drag in low Earth orbit. In addition, there are always potential inaccuracies in the measurement and prediction of satellite states, such as position and velocity. These uncertainties can lead to deviations from the predicted paths, which, if not accounted for, could result in suboptimal or even infeasible mission plans.

To ensure that the optimization model is both robust and reliable, it is necessary to perform a thorough robustness and uncertainty analysis. This analysis is designed to test how sensitive the optimization algorithm is to finding solutions when there are small variations in the initial states of the client satellites. By modeling these uncertainties, it is possible to evaluate whether the solutions provided by the optimization algorithm remain valid and effective under different, slightly altered conditions.

In this specific analysis, uncertainties in the initial states of the client satellites are systematically introduced by adding small perturbations to their position and velocity components. Specifically, 10 kilometers are added to one of the position components, and 10 meters per second are added to one of the velocity components. These values represent plausible deviations that could occur due to measurement errors.

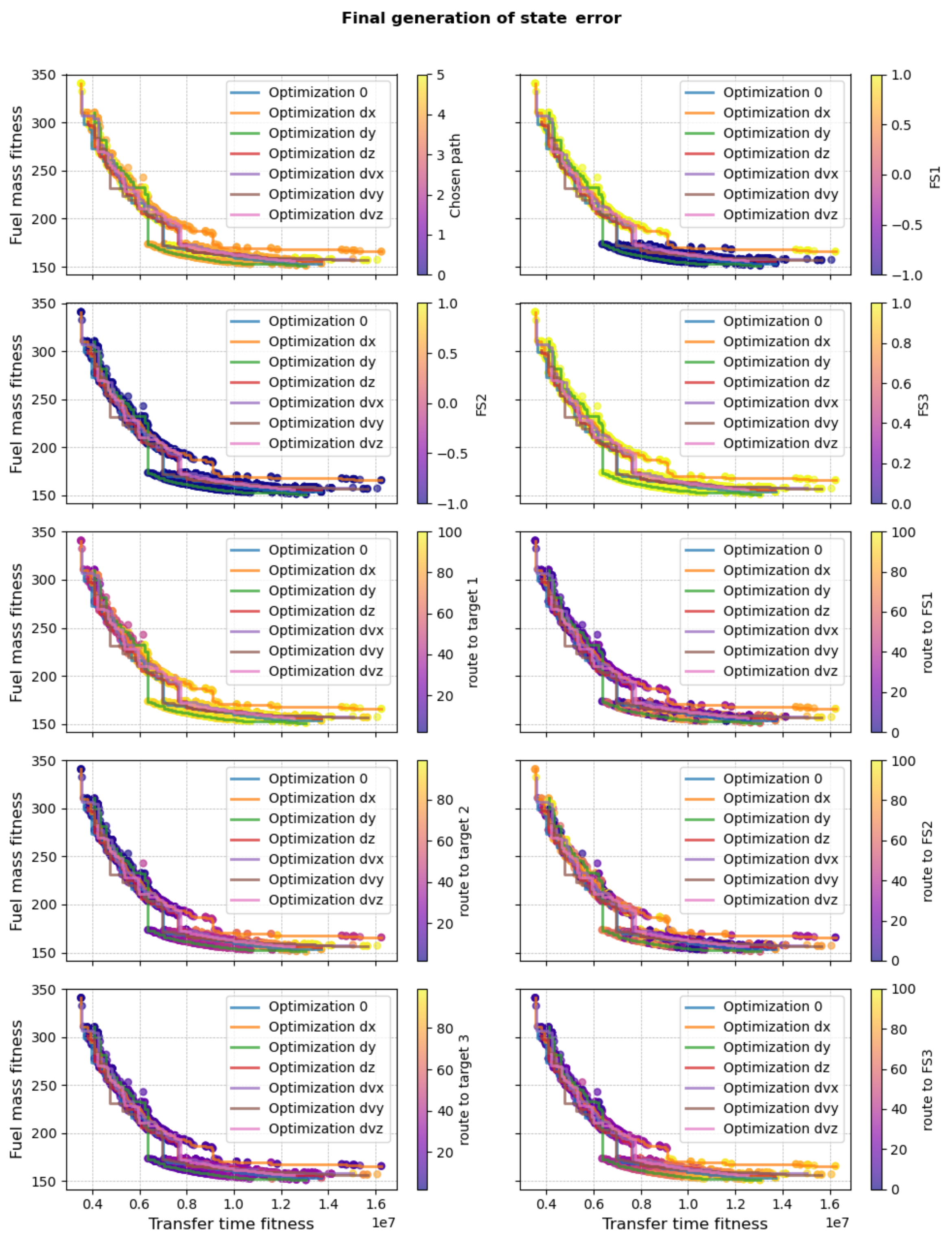


Figure 6.2: Pareto fronts optimizations that are all performed with added uncertainties in the indicated components of the position and velocity

The results shown in Figure 6.2 of this uncertainty analysis demonstrate that the optimization model is robust to small perturbations in the initial states of the client satellites. Although there are minor shifts in the Pareto fronts when position or velocity components are altered, the overall trade-offs between transfer time and fuel mass remain consistent.

The most notable observation from this figure is the sideways shift in the drop-off point in the Pareto fronts. This shift can be explained by the fact that right before this drop, the servicer is just capable of reaching a fuel station to refuel in between client visits. After the drop, the servicer continues directly to the target without refueling, causing the required amount of fuel to suddenly decrease significantly. This phenomenon was further discussed in detail in Chapter 5.

The reason why the location of this drop changes is due to the servicer reaching this client either slightly sooner or slightly later as a result of the perturbations. However, apart from this discrepancy—which should definitely be kept in mind—the rest of the Pareto fronts, especially the most important left portion, remain very similar to each other. This similarity further reinforces the robustness of the model, particularly in the region of the Pareto front where the most optimal solutions are found.

All in all, the figure indicates that the model is capable of producing reliable and effective solutions even in the presence of uncertainties, which is crucial for practical satellite mission planning, where such variations are inevitable. The robustness observed in the different scenarios provides confidence in the validity and applicability of the optimization model in real-world conditions.

6.3. Sensitivity to available fuel

Sensitivity analysis is a critical component in the validation and refinement of optimization models, especially in complex systems like satellite mission planning. By systematically varying key parameters, sensitivity analysis helps to understand how changes in these parameters influence the performance and outcomes of the optimization process. This understanding is essential for identifying the parameters that have the most significant impact on the results and for ensuring that the solutions provided by the optimization model are not overly dependent on specific assumptions or conditions. In essence, sensitivity analysis tests the robustness and flexibility of the model, providing insights into its behavior under different scenarios.

In the context of optimizing the path for a satellite servicer visiting multiple client satellites, one of the most crucial parameters is the amount of fuel available to the servicer. The fuel mass directly influences the feasibility and efficiency of the mission as it determines how many maneuvers the servicer can perform and how much time it can spend in transit between clients. Therefore, understanding how changes in the available fuel mass affect the optimization outcomes is essential to ensure that the mission plan is feasible and optimal under different fuel availability scenarios.

In this analysis, the available fuel mass is systematically varied between three different levels: 180 kilograms, 200 kilograms, and 220 kilograms. These values represent potential variations in the mission planning phase, where fuel restrictions might change due to different mission requirements, safety margins, or unforeseen circumstances. By analyzing the impact of these changes on the optimization results, it is possible to assess the sensitivity of the mission plan to fuel availability, ensuring that the solutions are robust and adaptable to different operational conditions.

The results of this sensitivity analysis will provide valuable insights into the flexibility of the optimization model and help identify the range of fuel availability within which the mission remains viable and efficient. This analysis is crucial for mission planners to make informed decisions about fuel allocations and to ensure that the servicer can achieve its objectives under a variety of possible scenarios.

The results of the available fuel mass sensitivity analysis are shown below in Figure 6.3

From the above figure, a couple things are worth noting:

First of all, similar to the robustness analysis, there is a discernible sideways shift in the drop-off points in the Pareto fronts, particularly in the higher fuel mass scenarios. This drop-off reflects the point at which the servicer can either refuel at a station or proceed directly to the next target, significantly reducing the required fuel mass. The shift is more prominent with higher fuel mass, indicating that increased fuel availability allows the servicer to reach critical decision points (such as refueling or not) at different stages of the mission.

Moreover, the Pareto front for the 180-kilogram scenario generally lies above the other two, indicating that lower available fuel leads to higher fuel mass fitness values (i.e., higher fuel consumption) for similar transfer times. In contrast, the 220-kilogram scenario shows the most favorable outcomes, with lower fuel mass fitness for the same transfer times, highlighting the benefits of increased fuel availability. Therefore, although counter-intuitive, less fuel can generally be used when a servicer with a larger fuel tank is put in orbit.

The sensitivity analysis of varying the available fuel mass from 180 kilograms to 220 kilograms

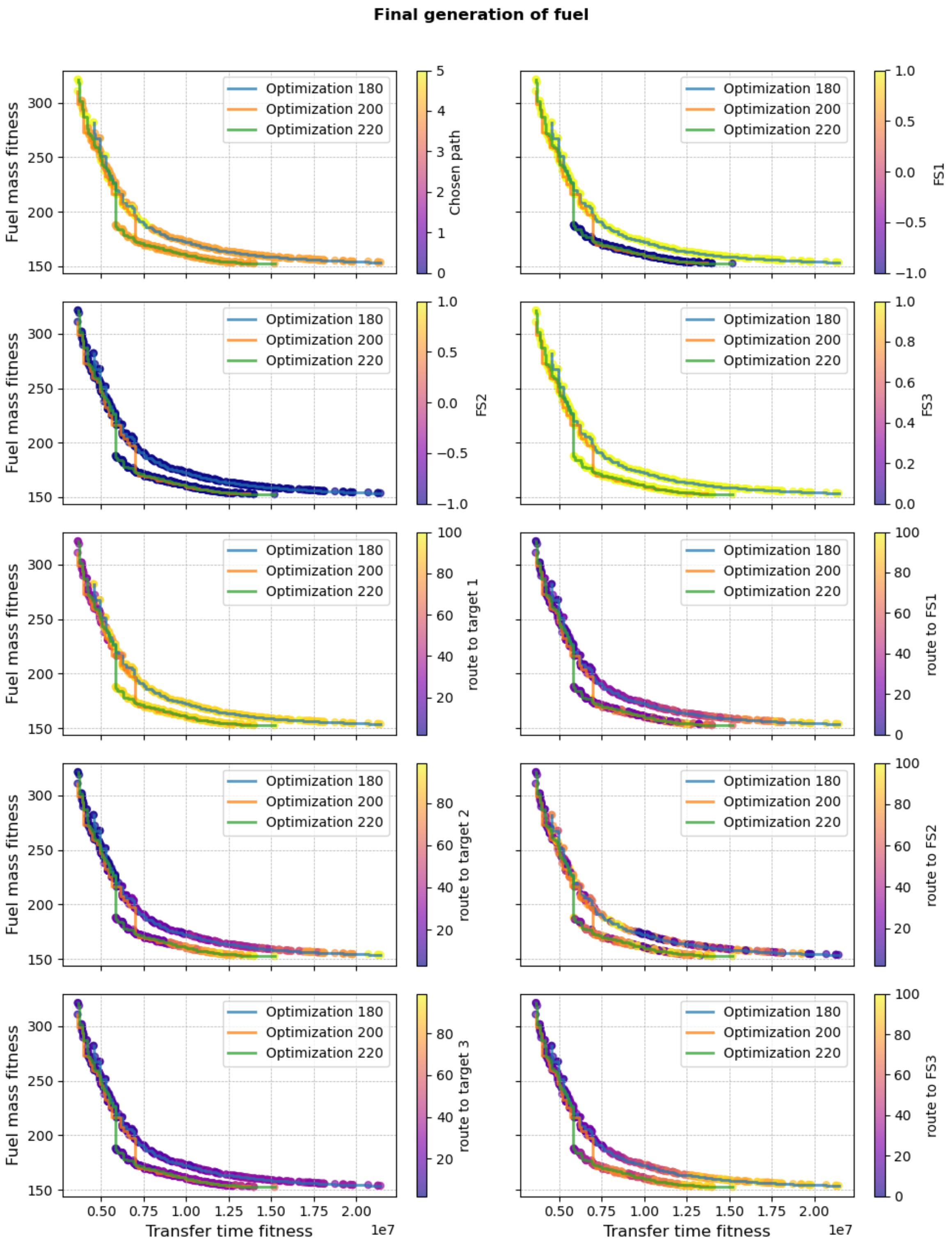


Figure 6.3: Pareto fronts of optimizations performed with different values for the available fuel.

reveals that the optimization model is robust and effectively adapts to changes in resource availability. The overall consistency in the Pareto fronts across different fuel masses demonstrates that the model reliably identifies the trade-offs between transfer time and fuel consumption under varying conditions. The noticeable improvement in fuel efficiency with increased fuel mass highlights the model's capability to utilize additional resources for better optimization. However, the diminishing differences between the 200-kilogram and 220-kilogram scenarios in the most optimal regions suggest that beyond a certain point, additional fuel may offer limited benefits.

7 Conclusions

In this study, an algorithm was developed, optimized, and verified to determine the optimal routing of a servicer satellite in a refueling infrastructure involving passive fuel stations and an active servicer for refueling client satellites in Sun-Synchronous orbits. A key aspect of this algorithm is its on-demand capability, which enables real-time optimization in response to new requests during a mission. The development of the model was structured in two main parts: identifying possible transfers between satellites in arbitrary SSO, and subsequently constructing the transfer model itself. The research findings cover the optimization results, model verification, and sensitivity and robustness analysis.

The optimization of the transfer trajectory from a fuel station to a client satellite identified several key observations. Initially, the study compared analytical propagation methods with numerical approaches. Analytical methods, while simpler, were constrained by idealized assumptions, whereas numerical methods, such as the Runge-Kutta-Fehlberg (RKF-78) integrator, provided greater accuracy and flexibility at a higher computational cost. The Cowell propagator, combined with the RKF-78 integrator, was determined to be the most effective choice, balancing computational efficiency with desired accuracy.

The selection of the Multi-Objective Evolutionary Algorithm with Decomposition Generational (MOEA/D-Gen) as the optimization method was verified by its reliable convergence and robust performance during initial tests. However, the optimization process also revealed challenges due to the complexity of the design space, often resulting in converging into local minima. To address these challenges, a custom optimization method was developed, effectively handling local minima by decoupling decision variables and systematically refining key parameters, such as burn times and directions. This approach enhanced the optimization process, achieving the desired orbital parameters, including matching the right ascension of the ascending node (RAAN), inclination, and semi-major axis between the servicer and the target satellite.

In summary, the analytical algorithm that generates the full list of possible transfers, combined with the custom trajectory refinement algorithm, provides a satisfactory answer to the sub-question of what transfer trajectory offers the highest efficiency when moving from a single fuel station to any given client satellite.

The study also extended the optimization to multi-target scenarios, similar to solving a dynamic traveling salesman problem (DTSP), where the servicer must visit multiple moving targets. The integration of the single-transfer algorithm into the DTSP framework enabled the model to dynamically adapt to changing conditions, such as the varying positions of client satellites and shifting mission priorities. To maintain model accuracy while reducing computational load, the research incorporated intermediate state estimation and static path assumptions by incorporating a transfer time scaling factor in order to correlate the static TSP to the DTSP. These assumptions resulted in an algorithm that can quickly and reliably find an optimal path to multiple client satellites, providing a satisfactory answer to the sub-question of what sequence of visits to multiple arbitrary Client Satellites optimizes efficiency for the servicing spacecraft, akin to solving the Dynamic Traveling Salesman Problem.

Finally, verification of the optimization results against brute-force methods confirmed that the model reliably identifies near-optimal solutions under varying conditions, reinforcing the robustness of the approach. Sensitivity and robustness analyses subsequently demonstrated that the optimization model is resilient against parameter variations and real-world uncertainties, such as initial state perturbations and varying fuel availability.

In conclusion, the optimization models developed in this study for satellite mission planning have proven to be effective and reliable, with the ability to adapt to various mission scenarios and uncertainties. The combined efforts of sensitivity analysis, robustness testing, and model verification ensure that these models provide practical solutions for real-world applications in satellite servicing and complex orbital operations. The findings provide a strong foundation for future developments in dynamic multi-target orbital operations, where adaptability and efficiency are critical.

8 Recommendations

Based on the findings and analyses presented in this report, several future recommendations are proposed to enhance current models and further improve the optimization of satellite servicing missions.

- Extension of the Single Transfer Model: The current single transfer model uses a three-burn strategy, which inherently limits the efficiency of orbital maneuvers. This approach is particularly inefficient for burns that are spread over time, such as high-energy burns required for significant orbital changes. In the current model, the most effective point for changing the orbit's inclination is at the equator, where all the thrust energy is directed into altering the inclination. However, for burns that are not instantaneous, the thrust is applied over an extended period, causing some of the energy to inadvertently change the right ascension of the ascending node (RAAN) instead of focusing solely on the inclination. Although the net RAAN change is zero because the burn is centered around the equator, this energy distribution reduces the overall efficiency of the maneuver. By extending the model to allow for additional burns, the thrust can be applied more precisely and efficiently, concentrating energy where it is most needed and avoiding unnecessary changes to other orbital elements. This would significantly improve the model's effectiveness, particularly for complex orbital transfers.
- Development of a Dynamic Mid-Course Correction Algorithm: The current transfer model, which relies on a fixed three-burn trajectory, is somewhat inflexible in accommodating new client satellites or changing mission demands. Once a transfer is initiated, the servicer follows the predetermined path until it reaches its target, at which point any new instructions or changes can be implemented. However, this approach can lead to inefficiencies, especially when new client satellites are introduced along the servicer's route or if mission priorities change. For example, if a new client satellite is located en route to the original target, it would be more efficient to adjust the servicer's course mid-transfer to refuel the new satellite first, before resuming the original mission plan. This would avoid the need for a full transfer to the original target followed by a separate transfer to the new satellite. To address this, a new algorithm could be developed to incorporate dynamic mid-course corrections. This algorithm would enable the servicer to modify its path during an ongoing transfer, allowing it to efficiently respond to new inputs and optimize the overall mission trajectory. By integrating this flexibility, the model would become more adaptive to real-time changes, ensuring that the most efficient route is always taken.
- Validation of the Full DTSP Model on High-Performance Computing (HPC) Systems: The current DTSP model, while effective for smaller-scale problems, incorporates several simplifying assumptions to reduce computational load. These assumptions, while necessary for running the model on standard computing resources, may limit its accuracy and applicability in more complex scenarios. Running the full DTSP model on a high-performance computing (HPC) system would enable a more detailed and exhaustive exploration of the solution space without these assumptions. This would allow for a more rigorous validation of the optimization algorithm, particularly for larger and more complex problem sizes. By utilizing HPC resources, it would be possible to verify the accuracy of the model and ensure that it can provide truly optimal solutions even under more demanding conditions.

- Expansion to a Multi-Servicer Constellation: The current model is designed for a single servicer operating within a specific orbital region, which limits its applicability to broader satellite networks. Expanding the model to include multiple servicers would allow for a more scalable and flexible approach to satellite servicing. By creating a constellation of servicers and fuel stations, the model could manage a wider range of satellites across different orbital regions. A practical approach would involve generating a list of possible ways to divide client satellites among the available servicers, with each servicer independently solving the DTSP for its assigned satellites. This expansion would require consideration of potential conflicts, such as multiple servicers attempting to refuel at the same station simultaneously, but would ultimately enhance the model's versatility and effectiveness in managing large-scale satellite networks.

These recommendations aim to improve current optimization models by improving their efficiency, flexibility, and scalability. Implementing these suggestions would contribute to the development of more robust and comprehensive solutions for satellite servicing missions, ensuring that these operations can be carried out with greater precision and adaptability in increasingly complex orbital environments.

References

[1] *Cost of space launches to low Earth orbit*. Our World in Data. URL: `https://ourworldindata.org/grapher/cost-space-launches-low-earth-orbit` (visited on 02/05/2024).

[2] Mike Wall. “Wow! SpaceX Lands Orbital Rocket Successfully in Historic First”. In: *Space.com* (Dec. 2015). Published on December 22, 2015. URL: `https://www.space.com/`.

[3] Michael Sheetz. “SpaceX launches a ‘rideshare’ mission carrying 143 spacecraft, a record for a single launch”. In: *CNBC* (Jan. 2021). Published on January 24, 2021. URL: `https://www.cnbc.com/`.

[4] Richard M. Kubicko and Robert Herrick. “Hubble space telescope - First servicing mission “down to earth logistics - From gsfc to ksc and back””. In: 1995, pp. 13–19. DOI: `10.2514/6.1995-904`. URL: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-84983189296&doi=10.2514%2f6.1995-904&partnerID=40&md5=9873790d6953d7d63ce3796275774c7e`.

[5] *Servicing Mission 1 (SM1) - NASA Science*. en. URL: `https://science.nasa.gov/mission/hubble/observatory/missions-to-hubble/servicing-mission-1/` (visited on 02/05/2024).

[6] B.B. Reed et al. “The Restore-L servicing mission”. English. In: *Space Astronaut. Forum Expos.* Journal Abbreviation: Space Astronaut. Forum Expos. American Institute of Aeronautics and Astronautics Inc, AIAA, 2016. ISBN: 978-162410427-5 (ISBN). DOI: `10.2514/6.2016-5478`. URL: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083663011&doi=10.2514%2f6.2016-5478&partnerID=40&md5=e0ce8d1129320f7e06d896510c284956`.

[7] H. Shen and P. Tsiotras. “Optimal two-impulse rendezvous using multiple-revolution lambert solutions”. In: *Journal of Guidance, Control, and Dynamics* 26.1 (2003). Publisher: American Inst. Aeronautics and Astronautics Inc., pp. 50–62. ISSN: 07315090 (ISSN). DOI: `10.2514/2.5014`. URL: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-0037262476&doi=10.2514%2f2.5014&partnerID=40&md5=0afc5406e2e34f27344e7a5422f79232`.

[8] X. Zhu et al. “Orbit determination for fuel station in multiple SSO spacecraft refueling considering the J2 perturbation”. In: *Aerospace Science and Technology* 105 (2020). Publisher: Elsevier Masson SAS. ISSN: 12709638 (ISSN). DOI: `10.1016/j.ast.2020.105994`. URL: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85087105177&doi=10.1016%2fj.ast.2020.105994&partnerID=40&md5=04296d2009e79631ef340ba44e70bd5a`.

[9] P. Han et al. “Optimal Orbit Design and Mission Scheduling for Sun-Synchronous Orbit On-Orbit Refueling System”. In: *IEEE Transactions on Aerospace and Electronic Systems* 59.5 (2023), pp. 4968–4983. DOI: `10.1109/TAES.2023.3247552`. URL: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85151538163&doi=10.1109%2fTAES.2023.3247552&partnerID=40&md5=34e90111fd85de8a5aa9fff4e5657002`.

[10] Dawn Aerospace. *GREEN PROPULSION FOR ANY SATELLITE*. 2024. URL: `https://www.dawnaerospace.com/green-propulsion`.

[11] Dawn Aerospace. *Docking and Fluid Transfer Port*. 2024. URL: `https://www.dawnaerospace.com/dftport`.

[12] Dawn Aerospace. *Dawn Aerospace B20 Thrusters Proven in Space*. Press release. Accessed: 2024-09-12. Sept. 2021. URL: `https://www.dawnaerospace.com/news`.

[13] NASA Goddard Space Flight Center. *General Mission Analysis Tool (GMAT) v.R2016a*. NASA Software Catalog. Version R2016a. `https://software.nasa.gov/software/GSC-17177-1`. 2016.

[14] CS GROUP et al. *Orekit: An Accurate and Efficient Core Layer for Space Flight Dynamics Applications*. Open Source Software Library. Version Apache License 2.0. `https://www.orekit.org/`. 2008.

[15] Tudat Space Team. *Tudat: TU Delft Astrodynamics Toolbox*. `https://tudat-space.readthedocs.io/`. Accessed: 2024-09-12. 2024. URL: `https://tudat-space.readthedocs.io/`.

[16] Hazem Mohamed El-Alfy. “Techniques for Video Surveillance: Automatic Video Editing and Target Tracking”. Doctor of Philosophy. Dissertation. University of Maryland, College Park, 2009. URL: `https://www.researchgate.net/publication/330012457_Techniques_for_Video_Surveillance_Automatic_Video_Editing_and_Target_Tracking`.

[17] Joshi Om Vaibhav, Lee Xun Yong, and Samuel Joo Jian Wen. “In-Orbit Lifetime of Satellites”. In: *Unpublished Manuscript* (2024). Presented at Raffles Institution and Defence Science and Technology Agency.

[18] Vivek Vittaldev, Erwin Mooij, and Marc Naeije. “Unified State Model theory and application in Astrodynamics”. In: *Celestial Mechanics and Dynamical Astronomy* 112 (Mar. 2012). DOI: `10.1007/s10569-011-9396-5`.

[19] Simon P. Shuster. “A Survey and Performance Analysis of Orbit Propagators for LEO, GEO, and Highly Elliptical Orbits”. Thesis. Utah State University, May 2017. DOI: `10.26076/f3c0-4670`. URL: `https://digitalcommons.usu.edu/etd/6510`.

[20] Karel F. Wakker. *Fundamentals of Astrodynamics*. Course AE4874, Faculty of Aerospace Engineering, Artikelnummer 06918290003. Jan. 2015.

[21] Tudat Team. *TudatPy API Reference*. Tudat Team. 2021. URL: `https://py.api.tudat.space/en/latest/integrator.html#`.

[22] William H. Press et al. *Numerical Recipes: The Art of Scientific Computing*. 3rd. Cambridge, UK: Cambridge University Press, 2007. ISBN: 978-0521880688.

[23] Paulo Flores. “Integration Methods in Dynamic Analysis”. In: *Concepts and Formulations for Spatial Multibody Dynamics*. SpringerBriefs in Applied Sciences and Technology. Springer, 2015, pp. 67–74. DOI: `10.1007/978-3-319-16190-7_13`. URL: `https://doi.org/10.1007/978-3-319-16190-7_13`.

[24] Andrea Biscani Iñigo M. Aliaga Dario Izzo. *Pygmo: A Python library for parallel optimization*. Version 2.x. Python package. 2024. URL: `https://esa.github.io/pygmo2/`.

[25] Yuchul Shin et al. “Radiation Effect for a CubeSat in Slow Transition from the Earth to the Moon”. In: *Advances in Space Research* (2015). PII: S0273-1177(15)00054-X. Reference: JASR 12107. Received Date: 21 October 2014; Revised Date: 19 January 2015; Accepted Date: 20 January 2015. DOI: `10.1016/j.asr.2015.01.018`.

[26] *Orbit phasing*. `https://en.wikipedia.org/wiki/Orbit_phasing`. [Accessed: August 19, 2024]. 2024.

[27] Antonio Prado and Helio Koiti Kuga. *Fundamentos de tecnologia espacial*. 1st ed. Brazil: Instituto Nacional de Pesquisas Espaciais (INPE), Dec. 2000.

[28] Nigel Bannister. “Active Learning in Physics, Astronomy and Engineering with NASA’s General Mission Analysis Tool”. In: *University of Leicester* (2024). Journal contribution. URL: `https://hdl.handle.net/2381/25605975.v1`.

[29] John D. Mahony. “Viva ‘Vis-viva’”. In: *The Mathematical Gazette* 108.572 (July 2024). Published online: 23 August 2024, pp. 248–256. DOI: `10.1017/mag.2024.63`.

[30] Panos Y. Papalambros and Douglass J. Wilde. *Principles of Optimal Design: Modeling and Computation*. Includes indexes and bibliographical references (pages 400-407). Cambridge; New York: Cambridge University Press, 1988, pp. xxi + 416. ISBN: 0521306744, 9780521306744.