A Rolling Horizon Dynamic Network Flow Framework for In-Orbit Servicing Mission Planning in Low Earth Orbit

Master's Thesis

Alice Caseiro

Delft University of Technology

TU Delft

A Rolling Horizon Dynamic Network Flow Framework for In-Orbit Servicing Mission Planning in Low Earth Orbit

Master's Thesis

by

Alice Caseiro

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at Delft University of Technology

Cover image credit: 'Canadarm 2 Robotic Arm Grapples SpaceX Dragon' (2016) by NASA, showing the International Space Station's robotic arm capturing a SpaceX Dragon cargo spacecraft during rendezvous and docking operations. Available at: https://www.flickr.com/photos/nasa2explore/26298228022

Student number:	5605903
Faculty:	Faculty of Aerospace Engineering
Supervisor:	Prof. Dr. ir. Prem Sundaramoorthy, Aerospace Structures & Materials
Thesis committee:	Prof. Dr. ir. Coen de Visser, Aerospace Control & Simulation
	Prof. Dr. ir. M. Sevket Uludag, Space Systems Engineering

February 2026

TU Delft

Abstract

This thesis presents an integrated optimization framework for the planning, routing, and scheduling of many-to-many In-Orbit Servicing (IOS) operations in Low Earth Orbit (LEO). Responding to the need for scalable and economically viable servicing infrastructures, the work overcomes major limitations of current planning approaches, including their reliance on deterministic assumptions, the separation of trajectory optimization from task sequencing, and the lack of integrated system-level economic modeling.

The proposed framework models the orbital environment as a fully dynamic, time-expanded logistics network in which customer satellites, servicers, and orbital depots evolve over time. Orbital motion and maneuvering are incorporated through a tailored set of impulsive maneuvers suitable for near-circular Sun-synchronous orbits, including multi-revolution phasing, coasting, and J2-assisted cross-orbital transfers. This dynamic network provides a physically grounded representation of spatiotemporal feasibility while remaining compatible with large-scale optimization.

Building on this network, IOS mission planning is formulated as a Mixed-Integer Linear Programming (MILP) problem that jointly optimizes trajectory selection, task sequencing, and resource management. The formulation explicitly captures key operational features of IOS missions, including service windows, fixed service durations, heterogeneous tools and consumables, depot-based resupply, and collision avoidance. A comprehensive profit-maximizing objective function is incorporated, accounting for service revenues, operating costs, launch costs, purchase, development, and manufacturing costs, as well as delay penalties.

To manage uncertainty arising from unpredictable service needs, such as repairs and active debris removal, the optimization is embedded within a Rolling Horizon framework. Stochastic service requests are modeled as Poisson processes and revealed dynamically, requiring continuous re-optimization as new information becomes available. This approach enables adaptive and computationally tractable planning over extended horizons while preserving temporal and resource consistency across replanning cycles.

The resulting framework supports both operational decision-making and long-term strategic analysis of IOS infrastructures. Verification and case studies demonstrate its ability to generate feasible and efficient mission plans and to evaluate economic and operational trade-offs across different IOS infrastructure architectures under exogenous conditions such as market demand. This capability supports the identification of robust and economically viable IOS configurations under uncertainty. By integrating orbital dynamics, logistics optimization, economic assessment, and adaptive decision-making within a single framework, this work provides a scalable and versatile planning tool for the design and operation of sustainable IOS systems in Low Earth Orbit.

Acknowledgements

This thesis concludes my master's studies at TU Delft, a journey that unfolded across countries, time zones, and life transitions, and that would not have been possible without the support of many remarkable people.

First and foremost, I would like to express my sincere gratitude to my supervisor, Prem, for his constant availability, guidance, and encouragement throughout this work. Despite the physical distance, his presence and engagement were unwavering, and his support played a central role in bringing this thesis to completion.

I am also deeply thankful to the professors and staff at TU Delft who made it possible for me to complete my master's degree partly remotely while already working. Their flexibility, understanding, and commitment were essential in allowing me to finish this chapter of my academic life. I am equally grateful for the opportunity to engage with research topics that genuinely inspire me, and for being able to dedicate my work to questions that align so closely with my academic interests and personal motivation. This intellectual freedom made the research process both meaningful and rewarding.

To my friends from TU Delft, now spread across different countries, thank you for making my time in the Netherlands such an adventurous and formative part of my life. Those years were filled with learning far beyond the classroom. A special thank you to Mafalda, who was always there to lend a hand on the coldest days, and to João and André. I am also very grateful to Bia, Leo, and Simão for the shared moments, laughter, and friendships that continue despite the distance.

To my family, I owe more than words can express. To my mother, Maria João, my father, Carlos, and my brother, João, thank you for your unconditional support, constant encouragement, and belief in me throughout this journey. I am also grateful to Mathilde, whose kindness and inspiration have been a source of warmth and positivity. To my grandparents, Natália and Custódio, who have always been like second parents to me, thank you for your love, care, and steady presence in my life.

To my boyfriend, Pedro, thank you for your patience, warmth, and unwavering support in every possible way. This journey would have been much heavier without you. I am also deeply thankful to his family for their kindness and generosity.

To my friends, who listened, supported, and stood by me in countless ways: especially Ana Rita, my lifelong sister; Patty and Tita, for always reminding me of the strength and importance of female friendships; and Richie, who showed me the beauty of getting lost in unexpected corners of the world. To all the other friends who offered support, conversation, and care along the way, thank you — even if not named here, you are not forgotten.

Finally, I am grateful for all the experiences, places, and people that shaped this journey. This thesis is not only an academic milestone, but also the result of a deeply human and shared path.

Alice Caseiro
Delft, February 2026

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
ADR	Active Debris Removal
AOCS	Attitude and Orbit Control System
CH	Control Horizon
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
EP	Edge Probe
EROSS	European Robotic Orbital Support Services
ESA	European Space Agency
GEO	Geostationary Orbit
GNC	Guidance, Navigation, and Control
IOS	In-Orbit Servicing
ISS	International Space Station
JAXA	Japan Aerospace Exploration Agency
LEO	Low Earth Orbit
MEV	Mission Extension Vehicle
MEO	Medium Earth Orbit
MILP	Mixed-Integer Linear Programming
NASA	National Aeronautics and Space Administration
PDM	Purchase, Development & Manufacturing
PH	Planning Horizon
RAAN	Right Ascension of the Ascending Node
RH	Rolling Horizon
RPO	Rendezvous and Proximity Operations
RSGS	Robotic Servicing of Geosynchronous Satellites
SH	Scheduling Horizon
SRP	Solar Radiation Pressure
SSO	Sun-synchronous Orbit
USD	United States Dollar

1 Introduction

1.1. Background and Motivation

The size, mass, and design of modern spacecraft remain fundamentally constrained by the capabilities of current launch vehicles. Most satellites are still engineered for a finite operational lifespan, after which they are either de-orbited into the atmosphere, transferred to a graveyard orbit, or left in space. Despite the rapid evolution of launch and manufacturing technologies, the core philosophy of spacecraft design and life-cycle management has changed little since the early years of space exploration. The prevailing model continues to treat satellites as expendable assets: launched once, operated for a limited time, and ultimately discarded. This paradigm not only limits mission flexibility and long-term value but also contributes to growing orbital debris and unsustainable practices in space operations (M. A. Luu & Hastings, 2022).

At the same time, the modern world is increasingly dependent on a vast network of satellites that enable essential services ranging from communication and navigation to Earth observation and national security. This reliance highlights the need for a robust and resilient space infrastructure, a need that traditional approaches, centered on replacing satellites at the end of their operational life or after failure, can no longer adequately meet. This paradigm of "static space" (Opromolla et al., 2024, p. 2) has proven both economically inefficient and environmentally unsustainable.

Over the past decades, the space sector has undergone a profound transformation. Once dominated by governmental agencies, it now features a rapidly expanding number of private companies that have introduced market competition and accelerated innovation. However, this expansion has also intensified concerns over orbital congestion and debris proliferation. The exponential growth in satellite launches, driven by mega-constellations and commercial ventures, has significantly increased the risk of collisions and the potential onset of the Kessler Syndrome, where cascading debris events could render certain orbital regions unusable (Froehlich, 2020).

In response, both space agencies and private actors are placing growing emphasis on sustainability to ensure safe and long-term access to orbital environments. This shift has driven interest in technologies that enhance the utilization of existing assets and promote the efficient use of space resources. Within this context, In-Orbit Servicing (IOS) emerges as a key enabler of a new paradigm of "flexible, dynamic, and sustainable space" (Opromolla et al., 2024, p. 2), allowing spacecraft to be maintained, upgraded, and repurposed in orbit, thereby maximizing their operational utility while minimizing their environmental impact (Ellery et al., 2008; Froehlich, 2020; Horsham, 2003).

In-Orbit Servicing refers to a range of operations conducted in space by a dedicated spacecraft (often called a servicer) to interact with another spacecraft (the client or target) already in orbit. These operations, comprehending but not limited to repair, refueling, component upgrades, and even active debris removal, offer a revolutionary approach to extending the operational lifespan of satellites and mitigating the escalating threat posed by orbital debris. As such, IOS allows a crucial step towards a more sustainable and resilient space environment and, thus, represents a paradigm shift from the traditional model of space operations, where satellites are designed for a fixed lifespan with limited capacity for post-launch intervention (Ellery et al., 2008; Hastings et al., 2016; Opromolla et al., 2024).

A useful terrestrial analogy for IOS can be drawn from the evolution of aerial refueling in aviation.

Much like satellites, aircraft are constrained by their onboard fuel capacity, limiting their range and operational flexibility. The introduction of aerial refueling through dedicated tanker aircraft (see Figure 1.1) transformed air operations by allowing missions to be extended without the need to land, thereby enhancing strategic reach and responsiveness. A similar paradigm shift is envisioned for satellites, particularly those in Low Earth Orbit (LEO), where maneuvering capability is restricted by limited propellant reserves and the stronger influence of Earth's gravity (M. A. Luu & Hastings, 2022).

Figure 1.1: Aerial refueling of an SR-71 Blackbird aircraft by a tanker plane (M. A. Luu & Hastings, 2022).

The motivation for the shift towards IOS arises from several converging drivers. First, satellite failures are becoming increasingly frequent as spacecraft grow more complex and operate for longer durations. In-orbit failures have now surpassed launch failures as the leading cause of mission loss, leading to substantial financial and service disruptions (Ellery et al., 2008). IOS offers a means to counter these losses through in-orbit repair, refueling, and maintenance, extending mission lifetimes and reducing the economic impact of failures. Second, society's dependence on uninterrupted satellite services continues to grow with the expansion of mega-constellations, space-based communications, and navigation systems, making orbital resilience a critical concern (Herron, 2023; Horsham, 2003). Third, the escalation of orbital debris poses an urgent threat to operational safety and long-term sustainability, with over 36,500 tracked objects larger than 10 cm and millions of smaller fragments currently in orbit (Cavaciuti et al., 2022; Froehlich, 2020). IOS directly contributes to debris mitigation through active removal and controlled de-orbiting of defunct satellites (Brettle et al., 2019; Chiu, 2019). From an economic and operational standpoint, IOS represents a paradigm shift from the traditional philosophy of maximizing reliability through redundancy toward one of maintainability and adaptability. The conventional approach results in heavier and costlier spacecraft, particularly for high-value platforms designed for long lifetimes. By contrast, IOS offers the potential for significant cost savings through life extension, repair, and upgrades performed directly in orbit, even for satellites that were not initially designed for servicing. Several studies demonstrate that IOS can be more cost-effective than launching replacements, especially for high-value communication satellites in geostationary orbit. Extending operational life not only increases total revenue but also distributes fixed costs over a longer period, improving the economic return of missions (Ellery et al., 2008; Hastings et al., 2016). Beyond these immediate advantages, IOS also serves as a key enabler of future space capabilities. The assembly, maintenance, and operation of large-scale space infrastructure, such as commercial space stations, lunar outposts, and orbital manufacturing facilities, will depend on in-orbit servicing technologies. Moreover, IOS can support astronauts in high-risk tasks, such as repairs on the outer structures of the International Space Station

(ISS), and facilitate deep-space exploration through refueling, maintenance, and assembly capabilities that reduce spacecraft mass, enhance mission efficiency, and enable the construction of large space structures beyond LEO (Ellery et al., 2008; Opromolla et al., 2024). In summary, IOS presents itself as a pivotal solution for ensuring the sustainability, resilience, and economic viability of space operations. By mitigating failures, reducing debris, extending lifetimes, and enabling new mission architectures, IOS lays the foundation for a safer, more efficient, and enduring presence in orbit.

Beyond its conceptual appeal, IOS is increasingly becoming a tangible and achievable paradigm due to a confluence of technological maturity, institutional support, and emerging commercial incentives. In recent years, significant progress in robotics, autonomous navigation, rendezvous and docking, and in-orbit manipulation has laid the foundation for operationally viable servicing missions. These technological developments, supported by sustained governmental investment and policy initiatives, are transforming IOS from a conceptual framework into an implementable reality (Cafolla, 2023; Hastings et al., 2016; Horsham, 2003).

The realization of IOS has been driven by advances across several key technological domains. Progress in robotic manipulation has enabled precise and autonomous operations such as grasping, refueling, and component replacement, with systems like the ISS's *Dextre* (see Figure 1.2) illustrating the maturity of dexterous robotic servicing (Opromolla et al., 2024). Autonomous Rendezvous and Proximity Operations (RPO) technologies, supported by advances in sensing, perception, and guidance, navigation, and control (GNC), enable servicers to safely approach, dock with, or capture target spacecraft, reducing reliance on human supervision. Developments in refueling and fluid transfer systems, including microgravity and cryogenic propellant handling, have demonstrated the feasibility of in-orbit resupply, while ongoing efforts focus on establishing standardized refueling interfaces to ensure interoperability between missions. Finally, modular and standardized spacecraft designs are paving the way for a serviceable and sustainable orbital infrastructure, where spacecraft are built with refueling, repair, and upgrade compatibility in mind. Collectively, these enabling technologies form the foundation for making IOS a routine, scalable, and economically viable element of future space operations (D. Arney et al., 2021; Li et al., 2019; M. Luu & Hastings, 2021; Ma et al., 2023; Nanjangud et al., 2018).

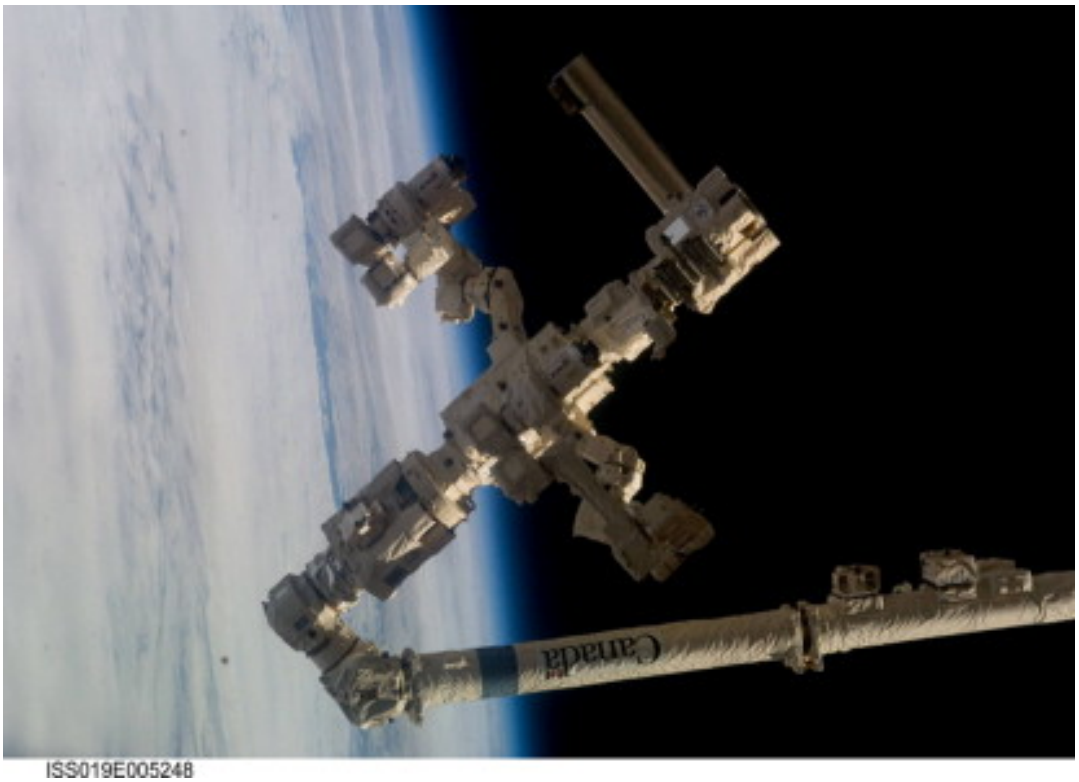


Figure 1.2: The Special Purpose Dexterous Manipulator (Dextre) operating aboard the International Space Station (Aziz, 2013).

The maturation of IOS has been shaped by a growing synergy between governmental and commercial actors. Governments act as catalysts by investing in high-risk technology demonstrations, funding early-stage research, and establishing policy and regulatory frameworks that ensure mission safety, debris management, liability management, and interoperability. Such initiatives have been instrumental in de-risking critical technologies and creating the institutional confidence needed for private-sector engagement. In parallel, private companies have increasingly transformed these advances into operational services and viable business models, expanding IOS from experimental missions to commercial applications such as satellite life extension, refueling, and debris removal. This public-private interplay is now reinforced by emerging standards and market-oriented policies, which are fostering competition, investment, and industrial scalability within the growing IOS sector (Chiu, 2019; Duke, 2021; Hastings et al., 2016; Opromolla et al., 2024).

Governmental agencies such as the National Aeronautics and Space Administration (NASA), the Defense Advanced Research Projects Agency (DARPA), and the European Space Agency (ESA) have long played a central role in advancing key technologies and validating their performance through high-risk demonstration missions. For instance, the origins of Rendezvous and Proximity Operations date back to NASA's efforts to enable crewed lunar missions during the Apollo era. In 1966, the *Gemini VIII* mission successfully demonstrated the first docking of a crewed spacecraft with a target vehicle, under the command of Neil Armstrong, marking a pivotal milestone in orbital operations. Shortly thereafter, the Soviet space program achieved the first fully automated docking in 1967, when the *Kosmos 186* and *Kosmos 188* spacecraft performed a mechanical coupling without human intervention. These pioneering achievements established the foundations of rendezvous and docking technologies that would later be refined during the Space Shuttle era through multiple servicing missions to the *Hubble Space Telescope* (see Figure 1.3). Shortly after its launch in 1990, a defect was discovered in Hubble's primary mirror that rendered its initial observations unusable, prompting one of the first large-scale demonstrations of in-orbit servicing. In December 1993, NASA astronauts successfully repaired and replaced critical components, restoring the telescope's full operational capability. Over the following 16 years, Hubble was serviced on four additional missions, each improving its performance and extending its operational lifetime, thereby demonstrating the technical feasibility, economic value, and scientific importance of in-orbit repair and upgrades (Lallo, 2012; Lillie, 2006). As human missions became increasingly costly and risky, the focus gradually shifted toward robotic and autonomous servicing. The *Engineering Test Satellite VII*, launched by Japan in 1997, marked a pivotal step in this direction. Designed to demonstrate the feasibility of robotic technologies for in-orbit docking and servicing, it successfully carried out three docking operations in both automatic and remote-controlled modes and performed tasks such as remote-controlled refueling and mass transfer. Notably, it was the first mission to feature a free-floating robotic arm in space, enabling the demonstration of several autonomous servicing techniques that provided valuable experience for subsequent developments in IOS. Building on these advances, DARPA's *Orbital Express* mission, launched in 2007, further proved the viability of fully autonomous rendezvous, docking, refueling, and component replacement. The mission, composed of the ASTRO servicer and the NextSat target spacecraft, successfully demonstrated a GNC system capable of non-cooperative RPO and capture, along with autonomous fuel transfer and the in-orbit replacement of key subsystems, representing one of the first integrated demonstrations of robotic servicing in orbit (M. Luu & Hastings, 2021; Ogilvie et al., 2008). More recent programs, such as ESA's *ClearSpace-1* and *European Robotic Orbital Support Services (EROSS)* missions, together with DARPA's *Robotic Servicing of Geosynchronous Satellites (RSGS)*, seek to demonstrate advanced capabilities including autonomous capture, dexterous robotics, and in-orbit assembly. ESA's ClearSpace-1, commissioned to the Swiss company ClearSpace, represents the first mission dedicated to Active Debris Removal (ADR). Scheduled for launch in 2025, it targets the Vespa (Vega Secondary Payload Adapter) upper stage, a remnant of a previous launch, and aims to capture it before performing a controlled atmospheric re-entry, during which both the chaser and the target will burn up. This mission constitutes a landmark step toward sustainable space operations and the development of commercial debris-removal services in Europe. In parallel, EROSS focuses on advancing European expertise in autonomous rendezvous, docking, and robotic manipulation technologies for servicing applications in both low and geostationary Earth orbits (Dubanchet et al., 2020; Froehlich, 2020; Opromolla et al., 2024). DARPA's RSGS program, a public-private partnership between DARPA and Northrop Grumman's SpaceLogistics, aims to demonstrate the feasibility of using a robotic spacecraft to inspect, upgrade, and reposition existing satellites in geostationary orbit, combining government leadership with commercial implementation (Duke, 2021;

M. Luu & Hastings, 2021).

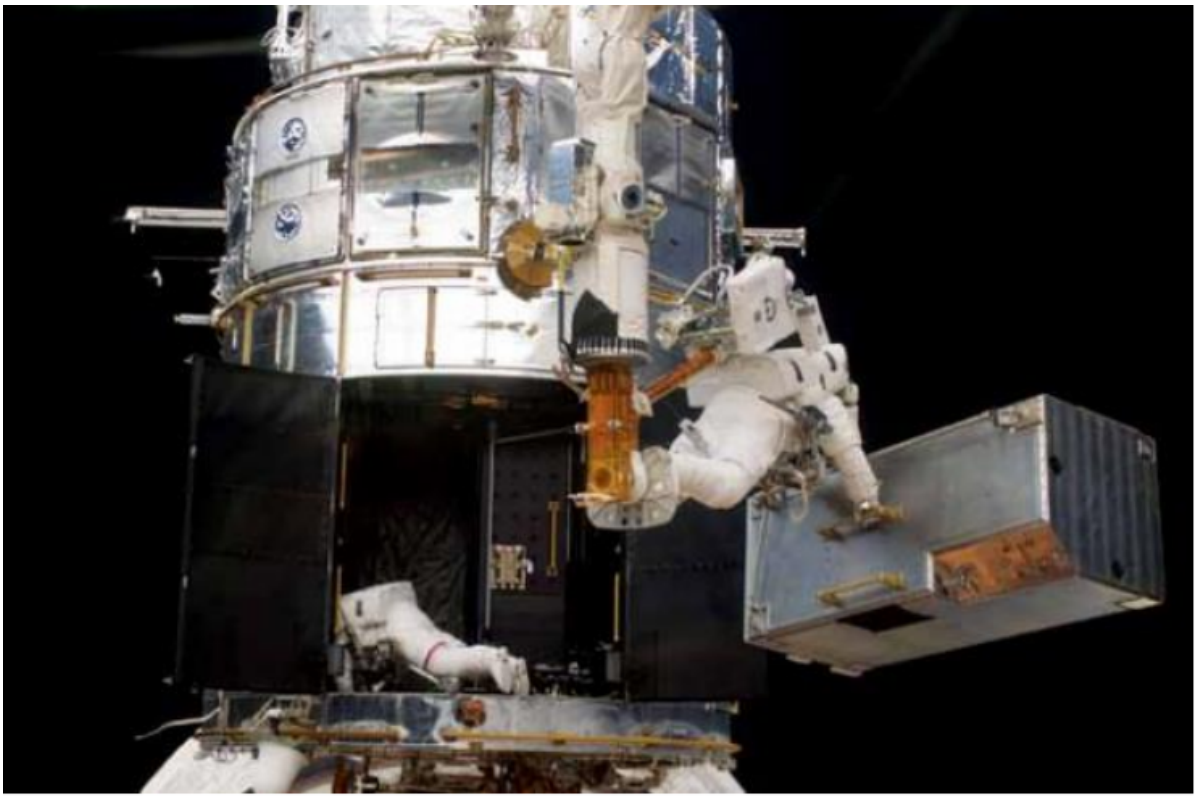


Figure 1.3: Astronaut performing maintenance on the Hubble Space Telescope during an in-orbit servicing mission (M. A. Luu & Hastings, 2022).

Beyond these institutional initiatives, private companies are increasingly taking the lead in developing operational IOS capabilities. Northrop Grumman's *Mission Extension Vehicle* (MEV) missions, for instance, have been the first to successfully provide in-orbit servicing to commercial satellites, demonstrating the technical and economic feasibility of life-extension operations in geostationary orbit. *MEV-1* docked with an Intelsat satellite in 2020, followed by *MEV-2* in 2021, each providing station-keeping services for approximately five years and enabling their clients to extend mission lifetimes without launching replacements. At the end of each service contract, the MEVs are capable of transferring their clients to graveyard orbits before proceeding to new targets. Meanwhile, Astroscale's *ELSA-d* and *ADRAS-J* missions are advancing active debris removal and end-of-life servicing in LEO. ELSA-d, launched in 2021, validated fundamental technologies required for debris capture and removal, such as magnetic docking, capture and release maneuvers, and autonomous GNC operations, while *ADRAS-J*, conducted with Japan Aerospace Exploration Agency (JAXA), is the first mission to approach and inspect an existing rocket upper stage in orbit as a precursor to full debris removal. (M. Luu & Hastings, 2021; Opromolla et al., 2024). Together, these public and private initiatives illustrate a decisive transition from isolated demonstrations toward sustained, operational, and commercially driven servicing.

Growing commercial interest in IOS reflects its increasing economic viability and the convergence of technological maturity with market demand. From a business perspective, IOS offers substantial financial incentives: extending satellite lifetimes through refueling or repair reduces replacement expenses, lowers insurance costs, and enhances the profitability of satellite operations, with market studies projecting up to 150 servicing missions per year and annual revenues of roughly €900 million (Kreisel, 2002). More recent assessments indicate that between 30 and 40 geostationary satellites require servicing each year (Sullivan et al., 2015). A recent report by Analysys Mason forecasts a cumulative revenue opportunity exceeding $6 billion by 2033 for satellite life-extension services (Mason, 2023), as illustrated in Figure 1.4. In parallel, Northern Sky Research projects a $14.3 billion market for all in-orbit services by 2031 (Northern Sky Research, 2022). This sustained market growth underscores the increasing commercial momentum behind operational and scalable in-orbit servicing solutions. The

rapid deployment of mega-constellations further amplifies the demand for flexible maintenance and upgrade capabilities, paving the way for innovative service-based business models. Meanwhile, advancing technologies and evolving policy frameworks are lowering entry barriers and fostering a more predictable commercial environment. Collectively, these developments indicate that IOS is not only technically feasible but also emerging as a sustainable and economically attractive pillar of the future space economy (Chiu, 2019; Ellery et al., 2008; Froehlich, 2020; Hastings et al., 2016; Opromolla et al., 2024).

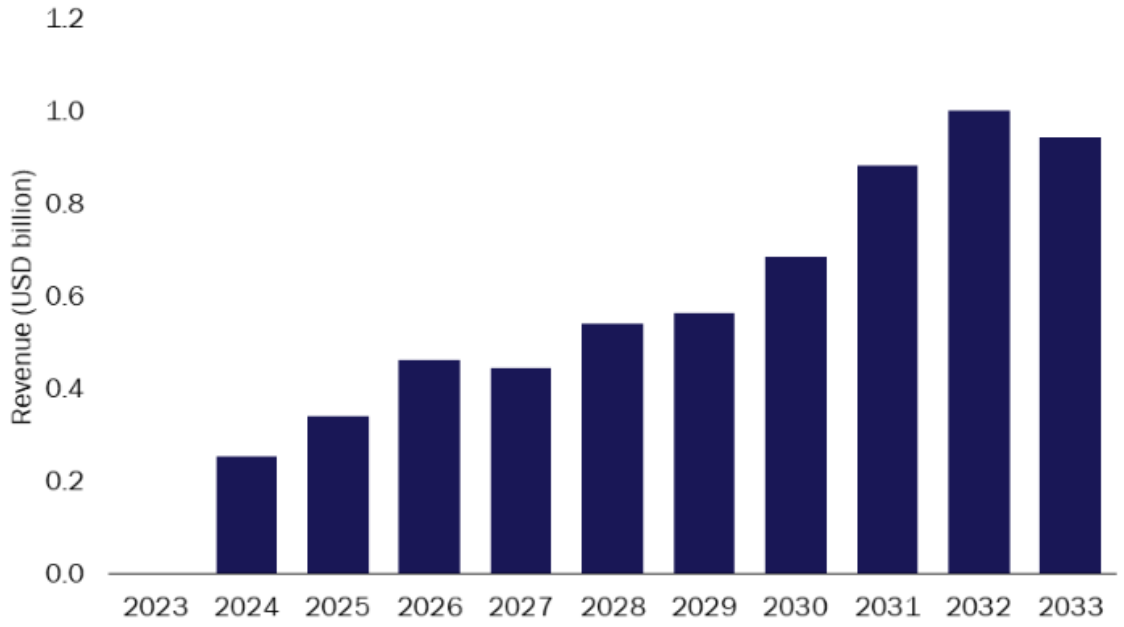


Figure 1.4: Projected global revenue from satellite life-extension services between 2023 and 2033, showing market growth over the next decade (Mason, 2023).

In the near future, corporations are expected to operate fleets of robotic servicers capable of conducting IOS missions across LEO, Medium Earth Orbit (MEO), and Geostationary Orbit (GEO). As the number and complexity of these missions grow, maintaining long-term operations will require in-orbit refueling capabilities that allow servicers to replenish their propellant and continue functioning over extended periods. To support such sustained activity, the establishment of dedicated space-based refueling depots will become essential. Recognizing this emerging need, several development initiatives are already underway. NASA is currently supporting multiple companies in the design and demonstration of space- and lunar-based propellant depots. In parallel, Thales Alenia Space, a company based in the United Kingdom, is developing a chemical refueling station scheduled for launch in 2027. Although the refueling infrastructure remains less mature than servicing technologies, its ongoing development will be a decisive step toward realizing the sustainable and self-sufficient IOS ecosystem envisioned in this work (Sorenson & Nurre Pinkley, 2023).

1.2. The Shift from GEO to LEO

The target market for IOS is shifting from high-value satellites in Geostationary Orbit toward the rapidly growing population of satellites in LEO. Although the highest-value individual servicing missions remain concentrated in GEO, most future demand is expected to arise in LEO (Northern Sky Research, 2022). Traditionally, servicing studies have predominantly focused on GEO satellites, primarily for life-extension missions, due to the high economic value of these satellites and their long operational lifetimes, particularly within the communications sector. However, recent developments have transformed the economic and operational landscape. As of 2021, more than 70% of all operational satellites were located in LEO, between altitudes of approximately 160 and 2,000 km, where the use of reusable launch

vehicles has significantly lowered the cost of space access and enabled the deployment of large-scale constellations composed of hundreds or even thousands of satellites (“Satellite Database”, 2021). Although LEO satellites were once considered unsuitable for servicing due to their lower unit value and shorter lifetimes, the large number of assets now in orbit, combined with their reduced travel distances and costs, introduces economies of scale and creates opportunities for scalable servicing architectures. This characteristic makes LEO constellations an appealing domain for new IOS concepts aimed not only at extending satellite lifetimes but also at enabling novel space system architectures (Hastings et al., 2016; M. A. Luu & Hastings, 2022).

This transition is driven by several converging trends. First, the commercialization of launch services by companies such as SpaceX, Rocket Lab, and Northrop Grumman has drastically reduced the cost of delivering payloads to orbit (see Figure 1.5), opening the space market to new operators and business models. Second, advances in technologies such as phased array antennas, high-throughput satellites, and autonomous spacecraft systems and control have made LEO constellations technically feasible and commercially attractive. In addition, LEO’s reduced communication latency offers a performance advantage compared to GEO. Finally, increased private investment and government initiatives to expand global connectivity have accelerated the proliferation of LEO constellations (M. A. Luu & Hastings, 2022).

Figure 1.5: Launch cost per kilogram versus year of launch in different vehicles (Roberts & Kaplan, 2022).

Future in-orbit services in LEO will also differ from those traditionally pursued in GEO, reflecting both the distinct types of space assets in LEO and the different economic and operational constraints of this orbital regime. Whereas GEO servicing efforts will largely target high-value life-extension missions, such as refueling or attitude and orbit control system (AOCS/ADCS) takeover, the diversity and density of LEO constellations broaden the range of relevant services. In LEO, potential servicing activities include not only life extension but also active debris removal, relocation, repair operations, and close-proximity inspection. Notably, Northern Sky Research identifies active debris removal as the fastest-growing in-orbit service segment, driven by technological advances and the growing need to maintain safe and sustainable orbital environments for satellite constellations (Northern Sky Research, 2022).

Within this evolving context, this thesis focuses on Low Earth Orbit applications of In-Orbit Servicing,

where high satellite density, shorter orbital transfer times, and lower operational costs provide favorable conditions for developing sustainable and scalable servicing architectures capable of supporting large constellations. The insights and models developed herein aim to advance IOS strategies tailored to this orbital regime, enabling not only the extension of satellite lifetimes but also the emergence of more dynamic, flexible, and sustainable space system architectures.

1.3. Problem Definition and Research Gap

Despite the increasing technological maturity and rising commercial interest in In-Orbit Servicing, the field remains in its early stages of development. Building a sustainable In-Orbit Servicing infrastructure capable of operating competitively in this emerging market presents substantial challenges. High initial investment costs, uncertain demand, and the limited flight heritage of key technologies introduce significant financial and technical risks. Overcoming these uncertainties requires strategic, long-term planning to identify robust business models and technology pathways that can ensure the viability, scalability, and resilience of future servicing enterprises (Froehlich, 2020; Hastings et al., 2016; Sarton du Jonchay et al., 2022).

The design space of potential In-Orbit Servicing architectures is vast, encompassing a wide range of orbital regimes, mission types, and servicing technologies. This complexity calls for systematic and multidisciplinary approaches to effectively model, plan, and optimize future servicing infrastructures and missions. To support informed long-term decision-making, the industry requires advanced optimization tools and different types of analyses to be capable of exploring the design trade-space at a system level. Such tools are essential for guiding technology development, investment strategies, and operational planning, ultimately ensuring the resilience and robustness of future In-Orbit Servicing enterprises amid uncertain demand and competitive market dynamics (Sarton du Jonchay et al., 2021, 2022).

Within this context, *mission planning* emerges as a critical enabler for the success and sustainability of In-Orbit Servicing operations. It represents an optimization process that determines how limited servicing resources, such as fuel, time, and servicer availability, are allocated to multiple concurrent tasks. Effective planning directly influences mission efficiency by minimizing transfer time and propellant consumption, while ensuring timely responses to service demands. As multi-target servicing becomes the norm, mission planning must integrate trajectory optimization with task sequencing to support cooperative and continuous operations. Addressing this challenge requires scalable optimization frameworks capable of coordinating complex servicing architectures across multiple orbits and time horizons (Adimurthy et al., 2007).

Previous research has made significant contributions to various aspects of In-Orbit Servicing mission planning. Studies to date examining various servicing architectures can be broadly classified according to the structure of the servicing system. Early investigations addressed *one-to-one architectures*, in which a single servicer is assigned to a single client satellite. Examples include Northrop Grumman's *MEV* missions, which provide life-extension and station-keeping services (Opromolla et al., 2024). Subsequent research explored *one-to-many architectures*, where a single servicer performs sequential missions to multiple client satellites (Du et al., 2015; Shen & Tsiotras, 2012; J. Zhang et al., 2012). The most complex configurations, *many-to-many architectures*, involve multiple servicers operating cooperatively to service multiple clients simultaneously, and can be supported by orbital depots or refueling stations (Bo & Feng, 2011; Jing et al., 2014; T.-J. Zhang et al., 2019). Each of these architectures introduces distinct challenges related to mission design, orbital transfers, logistics, and operational coordination.

The literature can be further categorized into two main research directions. The first focuses on the *analysis and design of orbital trajectories* for multi-transfer servicing missions employing either high- (Zhao et al., 2016) or low-thrust (C. Han et al., 2019) propulsion systems. These studies typically utilize high-fidelity force models to accurately design and optimize transfer trajectories. The second research direction addresses the *operational strategy* of servicing infrastructures, analyzing them from scheduling (Verstraete et al., 2018) and/or design (Sarton du Jonchay & Ho, 2017; Yao et al., 2013) perspectives through simulation, optimization, or hybrid approaches. These works evaluate the performance and value of different servicing architectures by varying operational schedules, fleet configurations, and infrastructure parameters.

Despite steady progress in the field, research on In-Orbit Servicing mission planning remains constrained by several limitations that hinder its translation into scalable, operationally viable frameworks.

A major challenge is the lack of *adaptive decision-making*. Most existing methods rely on deterministic assumptions and therefore cannot respond effectively to real-world contingencies such as spacecraft anomalies, emergent servicing needs, or the need for collision-avoidance maneuvers. This absence of mechanisms for adaptive replanning or rescheduling prevents these approaches from handling the inherently stochastic and time-varying nature of IOS operations. Additionally, many studies employ *hierarchical optimization*, separating target sequencing from trajectory design. This decoupling often produces locally optimal but globally inefficient solutions, underscoring the need for integrated approaches that jointly address both decision layers. From a system-level perspective, ensuring the *economic viability and operational feasibility* of servicing architectures also remains challenging. Existing models frequently omit realistic cost structures, operational resource constraints, and hybrid mission strategies required for practical implementation. Finally, the literature often adopts a *narrow mission scope*, focusing on single services such as refueling or debris removal while neglecting multi-purpose, multi-orbit architectures and their synergies. This limited focus restricts the generalization and scalability of proposed methodologies, whereas real-world scenarios will demand a diverse range of in-orbit services to address the evolving and heterogeneous needs of space operations. These shortcomings are exacerbated by the continued reliance on *simplified physical and operational models*, which, although computationally efficient, fail to capture the full fidelity of orbital dynamics and real mission constraints.

To address these limitations, future research must move beyond deterministic and isolated mission formulations toward frameworks that are adaptive, integrated, and system-oriented. Such approaches should handle operational uncertainty, couple target scheduling with trajectory optimization, and incorporate realistic cost and resource constraints. Moreover, to ensure practical and economic viability, they must support long-term analyses of complex, cooperative servicing architectures.

In this context, the present thesis addresses these gaps by modeling customer orbits in LEO as a dynamic network and formulating IOS operations as a time-expanded, generalized multi-commodity network flow problem solved using Mixed-Integer Linear Programming (MILP). A Rolling Horizon (RH) optimization scheme is also implemented to route and schedule many-to-many servicing operations under uncertain service demands, enabling adaptive and responsive decision-making across multiple time horizons. The resulting framework integrates physical realism, through maneuver modeling tailored to Sun-synchronous orbits and J_2 perturbations, with a comprehensive economic cost model that accounts for fuel use, tools and consumables, operating costs, and infrastructure deployment. Ultimately, the framework supports both the operational routing and scheduling of hybrid IOS missions in a multi-orbit setting, involving multiple servicers and orbital depots, and the strategic evaluation of alternative IOS infrastructure architectures. The justification for focusing on the LEO environment is discussed in Section 1.2.

1.4. Research Objectives and Scope

This thesis addresses the limitations of existing In-Orbit Servicing mission planning methods by developing an integrated optimization framework that:

(i) models IOS architectures, orbital motion, and servicing opportunities through a fully dynamic, time-expanded logistics network;
(ii) formulates IOS operations as a Mixed-Integer Linear Programming model that jointly optimizes trajectory design and target sequencing; and
(iii) embeds this optimization within a Rolling Horizon scheme capable of adapting to uncertain and time-varying service demand.

The overarching objective is to produce a scalable and operationally realistic planning tool that supports both short-term routing and scheduling decisions and long-term strategic assessments of IOS infrastructures in Low Earth Orbit.

This work focuses on impulsive maneuvering, Sun-synchronous orbit operations, and robotic IOS missions involving repair, refueling, inspection, and debris removal. The framework incorporates maneuver sets suitable for near-circular LEO orbits, including multi-revolution phasing and J_2-assisted cross-orbital transfers, and includes economic considerations through a profit-maximizing objective function.

1.5. Research Questions

Building on the research objectives and scope defined above, this thesis is structured around a central research question supported by a set of more specific sub-questions.

The main research question guiding this work is:

> *How can In-Orbit Servicing operations in Low Earth Orbit be modeled and optimized in a dynamically evolving environment so as to support both operational mission planning and strategic infrastructure assessment under uncertain service demand?*

To answer this overarching question, the thesis addresses the following sub-research questions:

(RQ1) *How can orbital motion, servicing opportunities, and IOS logistics be represented within a unified time-expanded network that preserves operational realism while remaining computationally tractable?*

(RQ2) *How can IOS mission planning be formulated as a Mixed-Integer Linear Programming problem that jointly optimizes trajectory design, target sequencing, and resource allocation?*

(RQ3) *How can uncertainty in service demand be incorporated into an IOS mission planning framework through a Rolling Horizon approach, and what limitations does this introduce in terms of optimality, stability, and computational tractability?*

(RQ4) *What operational and strategic insights can be derived from the proposed optimization framework with respect to IOS mission execution, infrastructure design choices, and their interaction with different service demand regimes?*

These research questions collectively structure the development of the logistics network, the optimization formulation, and the case studies presented in this thesis. In the concluding chapter, the findings are synthesized by explicitly revisiting each research question and discussing the extent to which it has been addressed.

1.6. Methodology Overview

This thesis builds on two complementary methodological pillars, space logistics optimization and dynamic Rolling Horizon decision-making, to construct a planning framework tailored to the time-varying and uncertain conditions of IOS operations.

Space logistics have been extensively studied in the context of planning and designing complex human and robotic space exploration missions. Prior research encompasses a broad set of methodological tools, including heuristic approaches (Taylor et al., 2007), simulations (Grogan et al., 2011), graph-theoretic methods (D. C. Arney & Wilhite, 2014), and, notably, network flow formulations (Chen et al., 2019, 2020; Ho et al., 2014). The present work builds on this latter line of research, adapting it to the specific operational characteristics of IOS.

While some authors have introduced dynamic, time-expanded networks by replicating a static network across discrete time intervals, these formulations typically assume fixed supply and demand nodes (i.e., stationary depots and customer locations). To the best of my knowledge, Sorenson and Nurre Pinkley (2023) was the first to model the orbital motion of customers, depots, and servicers directly within a network flow framework, thereby capturing the full spatiotemporal evolution of all agents in orbit. This approach offers a substantially more realistic representation of space logistics. The present thesis adopts and extends this idea, adapting the underlying network construction as detailed in Chapter 3.

A dynamic network allows the integration and selection of different precomputed orbital maneuvers. To support realistic yet computationally tractable rendezvous planning, this thesis employs a maneuver set tailored to near-circular Sun-synchronous orbits (SSOs), the operational regime assumed throughout. In SSOs, the dominant perturbation is the Earth's oblateness, represented by the first zonal harmonic J_2, whose secular nodal precession must be accounted for when modeling cross-plane rendezvous geometries (Wakker, 2015). Coasting 'maneuvers' represent passive motion along each circular orbit. Same-orbit phasing maneuvers follow the classical two-impulse, multi-revolution formulation commonly used in IOS mission planning (P. Han et al., 2022) and verified in standard astrodynamics references (Curtis, 2020), providing discrete time-fuel trade-offs through the choice of the number of phasing revolutions. For cross-orbit transfers, the model adopts the J_2-assisted drift-orbit strategy of Cerf (2013): a servicer performs an initial Hohmann transfer to a drift orbit, exploits differential J_2 nodal

precession to close Right Ascension of the Ascending Node (RAAN) separations, and then returns to the target orbit via an inclined Hohmann transfer with an optimally split plane change, following established methods in the literature (Curtis, 2020; Wakker, 2015). Background on orbital perturbations and the detailed explanation of these maneuvers is provided in Chapter 4.

Multiarcs are also incorporated into the network flow formulation to represent alternative trajectory options, primarily different multi-revolution phasing realizations, allowing the optimizer to select the most advantageous path given fuel constraints, timing requirements, and service commitments (Ishimatsu et al., 2016; Sarton du Jonchay et al., 2022).

A key advantage of network flow models is that they can be formulated as Mixed-Integer Linear Programs, enabling globally optimal solutions under the imposed physical and operational constraints. By jointly optimizing target sequencing and trajectory design, the MILP provides a unified decision-making framework that captures, more efficiently, the tightly coupled operational requirements of IOS missions. Adapting this approach to IOS, however, requires several extensions beyond traditional space logistics MILP formulations. For instance, classical network flow formulations in space logistics typically model supply and demand at stationary or quasi-stationary locations, such as planetary surfaces, parking orbits, or fixed orbital depots, and focus on optimizing vehicle routing and commodity flows between these locations over time, without requiring continuous co-location between vehicles and targets. IOS services, by contrast, require a servicer to rendezvous with a moving client satellite and to remain continuously co-located along its orbital trajectory for a specified service duration within a defined service window, a temporal persistence requirement absent from exploration-oriented formulations. Accordingly, the MILP developed in this thesis extends traditional models by explicitly capturing the defining operational characteristics of IOS, including service assignment, time-windowed and fixed-duration service execution, and the spatiotemporal feasibility of task completion on moving targets. These modeling elements build on and extend existing IOS logistics formulations Sarton du Jonchay et al. (2021, 2022) and Sorenson and Nurre Pinkley (2023) and are fully detailed in Chapter 6.

The goal of the mission planning framework is not only to generate effective short-term routing, scheduling, and resource management decisions, but also to support long-term strategic analysis of IOS architectures. To enable this broader evaluation, the framework adopts a multi-component profit-maximizing formulation inspired by Sarton du Jonchay et al. (2022) but adapted to LEO and the mission concept followed here that incorporates both service revenues and a comprehensive set of cost components. This structure provides a generalized economic representation of IOS operations in LEO, allowing the model to capture the trade-offs between operational feasibility and architectural value over extended time horizons.

On the other hand, Rolling Horizon techniques are widely used in dynamic and stochastic environments where demand evolves over time and future events cannot be predicted with certainty. The central idea is to optimize operational decisions over a limited look-ahead window using short- to medium-term forecasts. Only the first portion of each optimized plan is executed, after which the system state is updated and the optimization is re-solved with newly available information. By decomposing the overall scheduling problem into a sequence of smaller, overlapping subproblems, RH reduces computational complexity while maintaining coherence and near-optimality across the full horizon (Sethi & Sorger, 1991; Silvente et al., 2015). This allows IOS mission planning to adapt continuously to unfolding operational conditions. RH approaches have a record of application in space operations, including Earth observation scheduling (Dishan et al., 2013), and have recently been extended to IOS logistics (Sarton du Jonchay et al., 2021). The present work adopts the modified RH framework introduced by Sarton du Jonchay et al. (2021), with the full structure detailed in Chapter 5.

A key source of uncertainty in IOS mission planning are the unpredictable service needs that arise during a satellite's operational lifetime. Events such as malfunctions or end-of-life failures occur irregularly and independently across IOS customers. To represent this behavior realistically, this thesis models such stochastic service demands as Poisson processes, where event arrivals are characterized by exponentially distributed inter-occurrence times (Florescu, 2014). Under this model, the time between events is sampled from an exponential distribution, and each service request is revealed to the optimizer only at its activation time, mirroring the limited foresight available in real IOS operations. This uncertainty makes the RH framework essential, as it enables continuous re-optimization and preserves feasibility as new information becomes available.

To summarize, this thesis advances the state of the art in IOS mission planning by integrating dynamic decision-making with a realistic, trajectory-aware representation of orbital logistics. First, it

constructs a fully dynamic, time-expanded network that captures the orbital motion of servicers, customers, and depots, and incorporates a set of maneuver options, including multi-revolution phasing and J_2-assisted cross-orbital transfers, represented through user-defined multiarcs. Second, using this dynamic network as input, it formulates the logistics of complex many-to-many IOS infrastructures as a MILP that jointly optimizes target sequencing and trajectory design, extending classical space logistics formulations with IOS-specific constraints and decision variables. Third, the MILP is embedded within a Rolling Horizon framework that continuously re-optimizes routing, scheduling, and resource allocation under stochastic service demand. The framework also integrates a profit-maximizing economic model that accounts for service revenues, as well as different initial and operational costs. Together, these components form a flexible and scalable IOS logistics planning tool capable of generating short-term operational schedules while supporting long-term strategic analyses of sustainable servicing architectures in Low Earth Orbit.

1.7. Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 2 defines the IOS mission scenario and system elements, describing the customer satellites, service demands, servicer fleet, tools, orbital depots, and consumables that together constitute the modeled infrastructure.

Chapter 3 introduces the IOS logistics network, detailing the construction of the static and time-expanded dynamic network that underpins the optimization framework.

Chapter 4 develops the orbital transfer modeling, reviewing the relevant perturbations in Sun-synchronous orbits and presenting the maneuver set and its discrete-time implementation.

Chapter 5 describes the Rolling Horizon algorithm, showing how the network flow model is embedded in a dynamic decision-making framework and how stochastic service demand is incorporated into its Python implementation.

Chapter 6 formulates the Mixed-Integer Linear Programming model, defining the sets, parameters, decision variables, constraints, and profit-maximizing objective function that jointly capture routing, scheduling, resource management, and economic performance.

Chapter 7 verifies the behavior of the integrated framework through a series of controlled test cases, ensuring physical consistency, constraint satisfaction, and numerical robustness.

Chapter 8 applies the framework to representative case studies in Low Earth Orbit, beginning with an operational routing and scheduling analysis and then examining long-term strategic trade-offs, including the influence of depot configurations and fleet architectures across different market scenarios.

Finally, Chapter 9 summarizes the main conclusions and outlines recommendations and directions for future work.

2

IOS Scenario and System Elements

To apply the proposed optimization framework effectively, it is essential to first define the mission planning problem and the operational context in which the framework will be used. Although the framework is designed to be broadly applicable across various In-Orbit Servicing architectures, this thesis adopts a specific mission concept as a representative case study to demonstrate its functionality and relevance. The IOS model developed here draws on the framework of Sarton du Jonchay et al. (2021), adopting a similar mission concept and reusing their parameterization of service demands and servicer tool requirements. Additional parameters for other system elements are adapted to the current context.

In the modeled scenario, an IOS company is responsible for deploying and operating a fleet of servicer satellites and orbital depots. The orbital depots act as resupply hubs, storing critical consumables such as propellant, spare parts and de-orbit modules. The servicers, equipped with impulsive thrust capability and specialized tools, perform a range of services on Low Earth Orbit customer satellites, including refueling, inspection, repair, and Active Debris Removal.

Customer satellites may generate service requests either deterministically (e.g., predictable fuel depletion) or randomly (e.g., unexpected component failures). For each service request, the company must decide whether or not to perform the service. If a service is accepted, exactly one servicer must rendezvous with the target satellite to complete the task before a predefined deadline. Following completion, the servicer is able to transition directly to a new task.

The remainder of this chapter introduces the core modeling constructs that define the IOS infrastructure and service dynamics. These include the representation of customer satellites and their evolving service needs, as well as the modeling of the IOS infrastructure, comprising servicers and their tools, orbital depots, and involved consumables. Together, these elements form the foundation for the network flow-based optimization framework developed in the subsequent chapters.

2.1. Customer Satellites and Service Demands

The optimization framework is designed to operate in a many-to-many servicing architecture, wherein multiple customer satellites require attention from a fleet of servicer spacecraft. All customer satellites are assumed to reside in circular LEOs, although each satellite may occupy a distinct orbit characterized by different values of semi-major axis, inclination and Right Ascension of the Ascending Node. This setup allows for a multi-orbit scenario, which can be further extended to more diverse orbital regimes such as Geostationary Orbits, if needed.

Customer satellites exhibit service demands that may arise either deterministically or stochastically, depending on the nature of the required service. Deterministic services are those that can be planned in advance and generally recur at predictable intervals. These include:

- **Inspection** – The servicer approaches the customer satellite and inspects its condition without performing a docking.
- **Refueling** – The servicer performs a rendezvous and docking maneuver with the customer satellite, and subsequently transfers propellant to replenish the customer's onboard fuel reserves.

While refueling is a life extension service, since it directly prolongs the satellite's operational lifetime, inspection is a diagnostic/enabling service that supports life extension when followed by corrective actions (e.g., repair or refueling) but does not extend lifetime on its own.

Conversely, random services are unplanned and occur unpredictably. Two types of failure-driven services are considered:

- **Repair** – Following rendezvous and docking, the servicer replaces malfunctioning or damaged components on the customer satellite with functional spare parts.
- **Active Debris Removal** – If a satellite experiences a severe anomaly that renders it non-operational, the servicer performs a rendezvous, captures and de-tumbles the customer satellite, then attaches a de-orbit module that enables atmospheric reentry and disposal.

ADR is treated as a random service because it is triggered only when a satellite unexpectedly becomes non-operational, an event that cannot be forecast or scheduled in advance.

Each service type introduces specific logistical and operational requirements that influence scheduling, resource allocation, and maneuver planning. All service types share a common set of core parameters; only their values differ by service. Definitions are provided in Table 2.1. The corresponding entries are stored in the input file `tasks.yaml`, organized by service type and by deterministic vs. random tasks.

While this thesis focuses on the four core services outlined above, the proposed framework is designed to be extensible. Other service types can be integrated, provided they can be described using the same structured set of operational parameters.

Table 2.1: Parameter Definitions for Service Demand Modeling

Parameter	Applicable to	Description
Revenue	Deterministic and random needs	Income earned by the IOS operator for delivering a service in response to a request.
Delay penalty cost	Deterministic and random needs	Cost incurred per unit time by the IOS operator when the service is not initiated at the earliest defined service opportunity.
Service duration	Deterministic and random needs	Time required to complete a service.
Service window	Deterministic and random needs	Permissible time interval during which the service must be initiated, if the operator opts to fulfill it.
Commodity requirements	Deterministic and random needs	Commodities consumed when executing a service.
Inter-occurrence time	Deterministic needs	Fixed time interval used to simulate periodically recurring service needs.
Mean inter-occurrence time	Random needs	Expected time between successive random service needs, assuming a Poisson process.

Modeling each customer satellite additionally requires parameters for its orbit, initial position, and the set of services it requires (or may require in the random case). These are specified in `customers.yaml`. The concrete data used to instantiate all parameters are presented in Chapter 8.

2.2. IOS Architectural Elements

The In-Orbit Servicing infrastructure modeled in this thesis comprises three core components: servicers and their associated tools, orbital depots and consumables. These elements collectively support the planning and execution of service operations within a complex, many-to-many servicing architecture.

2.2.1. Servicers

Servicers are modeled as spacecraft using impulsive thrust, allowing the application of the rocket equation to estimate propellant consumption as a function of the total servicer mass. These vehicles are

responsible for executing orbital maneuvers to travel between customer satellites and provide the requested services. Servicers cannot exchange consumables with one another but are allowed to resupply at orbital depots.

Each servicer must be equipped with specific tools to perform the required services. Tools are treated as non-consumable payload elements that occupy mass and capacity but are not depleted during service execution. Each tool is modeled by two key parameters: its mass and its cost. Because different services require different tools, servicers are not assumed to carry all equipment types and may therefore be limited in the services they can perform. The developed optimization framework is designed to flexibly accommodate any user-defined set of tools; however, for the purposes of this thesis, four distinct tools are considered:

- **T1 – Refueling Apparatus:** Used for transferring propellant during refueling operations.
- **T2 – Observation Sensors:** Used for non-contact inspection and diagnostic operations.
- **T3 – Robotic Arm:** Enables component replacement during repair services and attaches the de-orbit module (also referred to as de-orbit kit) to customer satellites for ADR.
- **T4 – Capture Mechanism:** Used in ADR to secure non-cooperative targets, stabilize and detumble them, and provide a stable interface for de-orbit module attachment.

Servicers are characterized by a set of operational and performance parameters, which are listed and defined in Table 2.2. The corresponding entries are stored in the input file `servicers.yaml`, organized by servicer ID.

Table 2.2: Parameter Definitions for Servicer Modeling

Parameter	Description	Examples
Tools	Types of tools installed on the servicer to perform specific services.	T1, T2, T3, T4
Orbital transfer parameters	Represent the servicer's performance characteristics relevant to orbital maneuvers.	Specific impulse (I_{sp}); propellant tank capacity (F_d^{cap}); dry mass (m_d^{dry})
Cost parameters	Costs required to design, produce, and operate the servicer.	Development and manufacturing costs (c_d^{dm}); operating costs (c_d^{op})
Starting node	Initial location of the servicer at the start of the mission.	s_d
Refueling time	Time required for a full refuel of the servicer's propellant tank at an orbital depot.	RF_d
Payload capacity parameters	Define the servicer's payload capacities used for transporting mission-critical commodities.	Tank capacity for customer propellant; storage capacity for spare parts and de-orbit modules; also includes the tools carried ($Q_{\text{dm}}^{\text{cap}}$)

2.2.2. Orbital Depots

Orbital depots function as resource hubs for servicers, where key consumables are stored, including fuel, customer propellant for refueling services, spare parts for repair operations and de-orbit modules/kits for active debris removal. While the inclusion of depots enhances the responsiveness and flexibility of IOS operations, it also introduces additional deployment and operational costs to the overall infrastructure.

Depot modeling makes use of different parameters: starting position, development/manufacturing cost, operating cost, and dry mass (for launch-cost estimation). All depots are assumed identical, so these values are common to every unit. Station-keeping propellant consumption at depots is neglected in this thesis but could be included as an additional operating-cost term.

Assumptions for orbital depots

The following assumptions are used throughout this thesis:

- **Passive assets:** Depots are incapable of thrust-based orbital maneuvers.
- **Fixed placement:** Depots are deployed at predefined operational orbits, i.e., circular LEO orbits with variability in semi–major axis, inclination and RAAN to represent a realistic multi-orbit environment.
- **Robotic capability:** Orbital depots are assumed to be equipped with robotic systems that enable the autonomous transfer of consumables to servicers.
- **Unlimited consumables (baseline):** Consumables stored at depots are not capacity-limited in the baseline model.
- **Extensibility:** The framework can be extended by constraining depot capacities and by modeling Earth-to-orbit resupply (launch vehicles) that refill them.

2.2.3. Consumables

The IOS architecture models three consumable classes: *propellant*, *spare parts*, and *de-orbit modules or kits*. Spare parts are treated as a single homogeneous commodity, independent of the specific repair task; this simplification can be relaxed in future work to enable the simulation of more detailed and realistic servicing scenarios. Propellant is expended by servicers for orbital maneuvers and transferred to client satellites during refueling. De-orbit modules/kits are consumed in ADR operations.

Each consumable is parameterized by cost and mass: for continuous commodities (propellant), cost is given per unit mass and mass is accounted directly; for discrete commodities (spares, de-orbit modules), cost is per unit and a mass-per-unit parameter is provided to couple logistics to mass and launch constraints.

3

Logistics Network Construction

3.1. Introduction

Network flow models have been widely adopted in space logistics research to support the design and planning of complex space exploration missions. A key advantage of this modeling approach is that it can be formulated as a Mixed-Integer Linear Programming problem, which allows for globally optimal solutions under defined constraints.

Traditional space logistics models focus on transporting commodities from supply to demand nodes, optimizing vehicle assignments in the process. However, IOS missions introduce additional operational complexity: beyond transportation, servicers must remain at customer satellite locations for a certain duration to carry out services. This time-based servicing requirement must be embedded into the network structure, alongside orbital dynamics and maneuver costs to realistically represent operational feasibility.

To meet these requirements, this thesis introduces the IOS logistics network, an extension of conventional dynamic network flow models designed to capture the operational characteristics specific to IOS missions. This network serves as the foundation for the MILP formulation that solves the IOS mission planning problem. It models the complete operational infrastructure, including all elements presented in the previous chapter: customers, servicers, orbital depots and consumables. Additionally, it supports the integration of multiple orbital maneuver types and mission constraints.

The construction of this network proceeds in two stages. First, a static network is defined to capture spatial relationships among nodes representing discrete orbital positions and feasible transitions between them. This static structure supports multi-orbit scenarios and accommodates different maneuver types by assigning arcs between nodes. Then, this network is expanded across discrete time intervals, resulting in a time-extended, or dynamic, network that models the temporal evolution of system states.

To acknowledge prior work, the network construction in this chapter, particularly the time-expanded representation and the procedures used to generate subtasks and resupply arcs, adapts and builds upon the algorithms introduced by Sorenson and Nurre Pinkley (2023) for the Multi-Orbit Routing and Scheduling of Refuellable Space Robots problem. In translating those ideas to the IOS setting considered here, I specialize the maneuver library (coasting, multi-revolution phasing, and a combined maneuver), recast maneuver costs as propellant-to-initial-mass ratios and per step rates consistent with the discrete-time MILP, distinguish rate-limited servicer propellant from payload-limited service consumables, among other ideas that will come up in subsequent chapters. The resulting framework retains the spirit of their time-expanded network while tailoring notation, parameters, and assumptions to the modeling choices and mission constraints developed in this thesis.

In the following sections, the algorithms used to construct the IOS logistics network are presented.

3.2. Static Network

The static network models the spatial and orbital structure of the IOS infrastructure, including feasible trajectories for servicers and other vehicles. It is represented as a connected network composed of a set of nodes N and a set of arcs A. Each orbit is represented by a subset of nodes, with each node

corresponding to a discrete location in space, typically associated with a circular LEO orbit defined by fixed orbital parameters. Although the argument of latitude (assumed equivalent to true anomaly for circular orbits) conventionally describes the time-dependent position of an object along its orbit, here it is assigned a fixed value at each node to represent a static location on the orbit.

A directed arc $a \in A$ connects a node $i \in N$ to a node $j \in N$ if a feasible orbital maneuver exists between them. These arcs represent possible orbital transfers and serve as the pathways through which commodities can move across the IOS network. In the network flow formulation, all relevant entities, vehicles and resources modeled in Chapter 2, represent commodities flowing through this network.

To construct the arc set, an algorithm evaluates all node pairs and generates an arc for each feasible maneuver, based on the orbital parameters of the nodes. The maneuver types supported in the current framework are:

- **Coasting Maneuver:** An arc is added between adjacent nodes along the same orbit in the direction of rotation, representing a passive drift phase without the application of thrust. Nodes are considered adjacent if they are consecutive in the ordered set of nodes discretizing a given orbit.
- **Phasing maneuvers:** For any two non-adjacent nodes on the same orbit (along the direction of motion), a family of candidate phasing arcs indexed by the integer revolution count $n \in \{1, \ldots, N_{\text{max}}\}$ is generated. This yields multiple alternatives between the same pair of nodes, each with a distinct time of flight and Δv (longer duration typically trades for lower Δv), allowing the servicer to align its arrival with the target. The multi-revolution phasing construction is detailed in Chapter 4, and the MILP in Chapter 6 can select the option that best satisfies timing and cost.
- **Combined Maneuver:** To connect nodes on different circular orbits (with distinct altitude, RAAN and/or inclination), the servicer exploits J_2-induced RAAN precession by (i) performing a Hohmann transfer to a drift orbit with radius r_d, (ii) coasting for a chosen drift time t_d so that the relative RAAN rate closes the RAAN gap, and (iii) transferring from the drift orbit to the target via a Hohmann transfer with a split plane change. The inclination change Δi is partitioned between the two impulses by an optimal fraction s that minimizes total Δv. In this implementation, arcs are generated only between nodes located at the ascending or descending node; further details are provided in Chapter 4.

Each arc a is assigned the following parameters:

- τ_a – Time required to traverse the arc from node i to node j.
- ϕ_a – Total propellant fraction associated with the maneuver (i.e. propellant mass divided by the initial mass at the start of the arc).

Note: To compute the actual total propellant required for traversing arc a, the ratio ϕ_a must be multiplied by the servicer's initial mass at the beginning of the maneuver.

The pseudocode for the arc generation process is presented in Algorithm 1.

The maneuver strategies used to compute the transfer time τ_a and fuel cost ϕ_a for each arc are described in detail in Chapter 4. While the current implementation includes only a limited set of orbital maneuvers, specifically, coasting, phasing, and the above mentioned combined maneuver, the framework is designed to be modular and extensible. Additional maneuver types can be incorporated, provided that the corresponding transfer time τ_a and fuel cost ϕ_a for each arc a are available or can be reasonably estimated. Besides, this work focuses solely on the time and fuel costs associated with the phasing portion of the rendezvous process and operations. Other rendezvous phases and the docking process are assumed to have negligible cost, but could be incorporated if desired.

Multiarcs in this framework

To increase flexibility, the static network supports the use of *multiarcs*, multiple arcs between the same ordered node pair. Originally introduced by Ishimatsu et al. (2016) and adopted in space logistics frameworks such as Chen and Ho (2018) and Ho et al. (2016), multiarcs enable the representation of alternative transportation options. These can include variations in vehicle types/designs, propulsion technologies, or maneuver strategies. For example, different servicer configurations or depot designs can be modeled by assigning each to a distinct arc with its own cost and time profile. The multiarc

Algorithm 1 Arc Generation Algorithm

Create arc set $A = \emptyset$.
Input N as the set of nodes with respective orbital parameters.
Group N by orbit ID and sort nodes within each orbit by the argument of latitude.
for each node pair $i, j \in N$, $i \neq j$ **do**
    Compute $\Delta u =$ argument of latitude between nodes i and j.
    if same orbit AND adjacent **then**
        Create arc a representing a Coasting maneuver.
        Set P as the orbital period of node i and j.
        Set $\tau_a = \frac{\Delta u}{360} \cdot P$, $\phi_a = 0$.
    else if same orbit AND NOT adjacent **then**
        for n=1 to N_{max} **do**
            Create arc a representing a Phasing maneuver, indexed by the revolution count.
            Calculate τ_a and ϕ_a as shown in Chapter 4.
        end for
    else if different orbits AND node pair is ascending/descending node **then**
        Create arc a representing a Combined maneuver.
        Calculate τ_a and ϕ_a as shown in Chapter 4.
    end if
end for
return arc set A

structure also accommodates low-thrust transfers or other advanced trajectory options by adding further arc definitions.

In the current implementation, multiarcs represent alternative maneuver realizations rather than vehicle variants: Concretely, multiarcs arise in the multi-revolution phasing case: as described above for phasing maneuvers, for any two non-adjacent nodes on the same circular orbit, a family of candidate phasing arcs indexed by the integer revolution count is generated, producing multiple alternatives between the same (i, j) with distinct time-cost trade-offs.

While this work focuses on impulsive servicers and enables multiarcs only for phasing maneuvers, the framework is readily extensible: the same mechanism can capture advanced vehicle designs, alternative propulsion models, or additional maneuver strategies between the same nodes by introducing extra arc definitions with their corresponding time and cost parameters.

Figure 3.1 presents a conceptual example of the static network structure introduced in this section. The diagram includes a simplified network consisting of six nodes ($N = 1, 2, 3, 4, 5, 6$) and a total of 14 directed arcs. Coasting maneuvers, represented by black arcs, include the subset $A' \subset A = \{(1,2),\ (2,3),\ (3,1),\ (4,5),\ (5,6),\ (6,4)\}$. Phasing maneuvers, shown in green, correspond to the arcs $\{(1,3),\ (2,1),\ (3,2),\ (4,6),\ (5,4),\ (6,5)\}$. Considering the assumption that nodes 2 and 5 lie at $u = 0°$ (ascending node), the only cross-orbit option in this toy network is a single bidirectional pair of combined maneuver arcs (in blue): $(2,5)$ and $(5,2)$.

3.3. Dynamic Network

Once the static structure of the network has been defined, it becomes necessary to introduce a temporal dimension. In reality, customer satellites, orbital depots, and servicers are constantly in motion, following their orbital paths. To account for this dynamic behavior, the static network must be extended into a time-expanded representation that captures the evolution of all elements over both time and space. This temporal dimension is introduced by defining a finite time horizon T, discretized into uniform intervals of length Δt, such that $t \in \{0, \Delta t, 2\Delta t, \ldots, T\}$. Equivalently, in integer time steps $h \in \{0, 1, \ldots, H\}$ with $H = T/\Delta t$.

The choice of time discretization Δt plays a critical role in determining both the fidelity and tractability of the resulting model. A finer discretization improves temporal resolution in capturing orbital servicing operations but increases computational complexity, as the number of MILP variables scales with the number of time steps. Conversely, a coarser discretization reduces computational effort but may in-

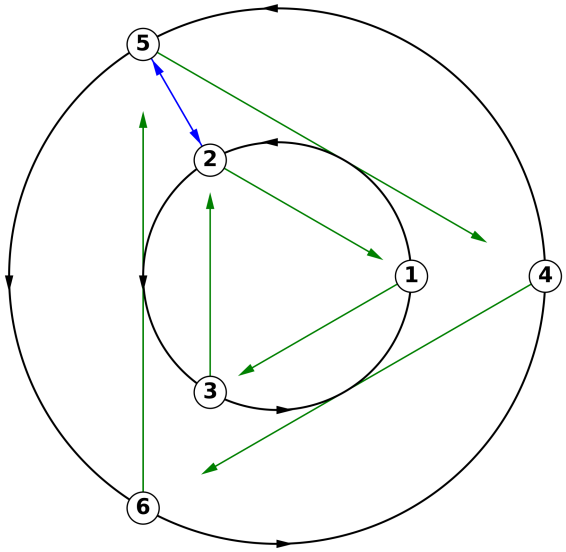


Figure 3.1: Illustrative example of a static network structure with six nodes and 14 arcs. Coasting maneuvers are shown in black, phasing maneuvers in green, and a single bidirectional pair of combined maneuver arcs in blue connecting nodes 5 and 2.

troduce inaccuracies in task timing and feasibility. Therefore, Δt should be chosen to balance model precision and computational efficiency, depending on the application and available computational resources. In Chapter 8, the chosen discretization is adapted to each case study to ensure an appropriate trade-off between accuracy and solvability.

Returning to the arc generation, any arc from Algorithm 1 with traversal time $\tau_a > T$ is discarded. In the discrete-time MILP, each arc's continuous duration is discretized by rounding to the nearest integer number of time steps, $(\tau_a/\Delta t)$, and the total propellant fraction associated with the maneuver, ϕ_a, is correspondingly converted into a per time step propellant fraction, ψ_a, which is applied multiplicatively to the current mass at each timestep of the arc.

Movement of customer satellites

Dynamic modeling begins with the motion of customer satellites, whose trajectories create evolving service opportunities in both space and time. Regardless of whether service demands are deterministic or stochastic, their evolution post-emergence is modeled identically. In what follows, all such demands are referred to as *tasks*.

Let $\mathcal{B}$ denote the set of tasks, where each task $v \in \mathcal{B}$ is characterized by a predefined set of parameters (see Table 2.1). As customer satellites are assumed to follow fixed orbital paths without propulsion, each task $v \in \mathcal{B}$ reappears at different nodes and times along the same orbit as the satellite continues its revolutions. To capture this, each task is decomposed into a set of subtasks $\mathcal{B}_v$. Each subtask $k \in \mathcal{B}_v$ is associated with a specific node-time pair (n_{vk}, t_{vk}), indicating where and when the task can be serviced.

Subtask generation begins at the task's initial orbital location and timestamp, simulating orbital motion using coasting arcs. For each new node reached, the traversal time from the previous node is used to increment the timestamp. This continues iteratively until the full time horizon T is covered. The procedure is formalized in Algorithm 2.

A task $v \in \mathcal{B}$ is considered fully serviced once a sufficient number of its subtasks are completed consecutively, meeting the required service duration. Furthermore, the first completed subtask must fall within the task's designated service window. A subtask (n_{vk}, t_{vk}) is considered complete if a servicer arrives at node n_{vk} precisely at time t_{vk}. This implies that the servicer must have departed from some node i along arc (i, n_{vk}) at time $t_{vk} - \tau_{in_{vk}}$, where $\tau_{in_{vk}}$ denotes the arc traversal time. The servicer must then continue coasting on the same orbit to accumulate sufficient service duration at the task location.

Algorithm 2 SubTask Generation Algorithm

Input time horizon T and network (N, A).
Input task set $\mathcal{B}$, where each task $v \in \mathcal{B}$ has a known starting node and orbit.
Create lookup table τ_a from arc set A, where τ_a is the traversal time steps of coasting arc a.
for each task $v \in \mathcal{B}$ **do**
    Initialize subtask set $\mathcal{B}_v = \emptyset$
    Set *current_node* = starting node of task v
    Set *activation_time* = activation time step of task v
    Set *current_time* $= 0$
    while *current_time* $\leq T$ **do**
        if *current_time* $\geq$ *activation_time* **then**
            Add subtask $(\textit{current\_node}, \textit{current\_time})$ to $\mathcal{B}_v$
        end if
        Set *next_node* = the node following *current_node* in the orbit's direction of motion.
        Set $\tau = \tau_{\textit{current\_node},\textit{next\_node}}$
        Update *current_time* $=$ *current_time* $+ \tau$
        Update *current_node* $=$ *next_node*
    end while
end for
return $\mathcal{B}_v, \forall v \in \mathcal{B}$.

To support the explanation of the subtask generation process, the illustrative static network introduced in Figure 3.1 is used as a foundation. Figure 3.2 provides a graphical representation of a simplified scenario in which two tasks move through this network over time. The tasks, highlighted in red, are shown progressing along their respective orbits, occupying different nodes at the specified time steps. Task movement is restricted to coasting arcs, reflecting the assumption that customer satellites follow passive, non-propelled orbital trajectories. It should be noted that tasks do not necessarily change nodes at every time step, as this depends on the orbital period associated with each orbit and on the choice of time discretization Δt. In the example shown, the distance between nodes in the inner orbit corresponds to one time step, whereas in the outer orbit it corresponds to two time steps. The corresponding subtasks generated from this spatiotemporal evolution are summarized in the Table 3.1, consistent with the visual progression shown in the figure.

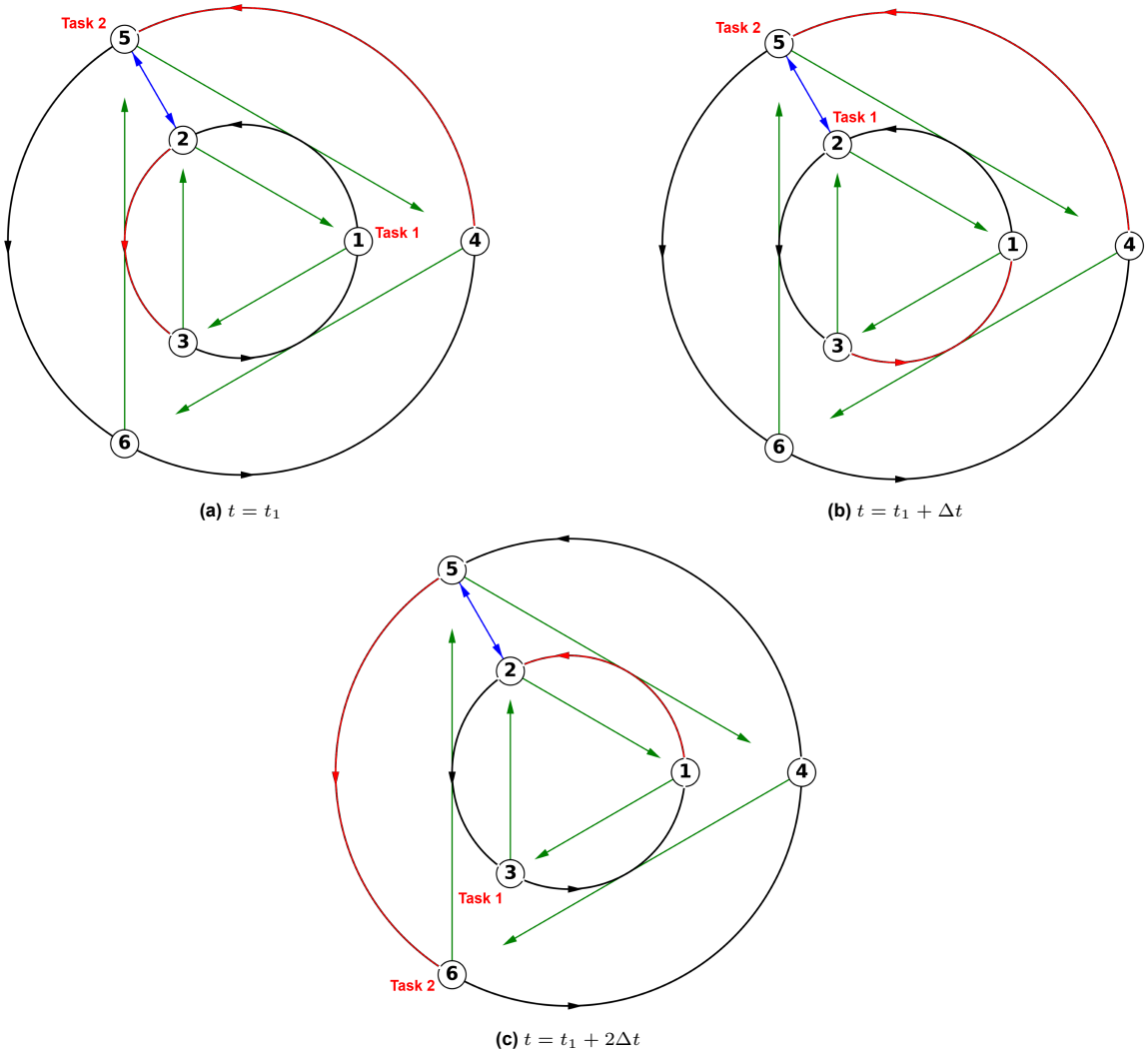
Table 3.1: Subtask Sets for Each Task

Task ID	Subtask #	Node	Time period (t)
Task 1			
	1	1	t_1
	2	2	$t_1 + \Delta t$
	3	3	$t_1 + 2\Delta t$
Task 2			
	1	5	t_1
	2	6	$t_1 + 2\Delta t$

Movement of servicers

The movement of the servicing vehicles is now introduced. The model considers a set $\mathcal{D}$ of identical robotic servicers, each responsible for completing assigned tasks by traversing the network over time. Servicers move along directed arcs $a \in A$, and their trajectories are tracked across the discrete time horizon. Each servicer $d \in \mathcal{D}$ begins at a predefined starting node $s_d \in N$. From there, the servicer may traverse any arc in accordance with the network topology. Flow conservation constraints ensure that a servicer may only proceed from nodes it has previously reached.

The servicer's movement is modeled as a flow through the network, with a binary decision variable $y_{d,a,t}$ indicating whether servicer d initiates traversal of arc a at time t. Because orbital transfers



(a) $t = t_1$

(b) $t = t_1 + \Delta t$

(c) $t = t_1 + 2\Delta t$

Figure 3.2: Time-evolving task and resupply arc positions and transitions over three consecutive time steps.

consume fuel, each servicer's fuel level is tracked using the variable f_{dt}, with a maximum tank capacity F_d^{cap}. A servicer may only begin traversing arc a at time t if it has sufficient fuel onboard, i.e., $f_{dt} \geq \phi_a \cdot m_0$, where ϕ_a is the propellant-to-initial-mass ratio for arc a, and m_0 is the servicer's mass at the start of the traversal.

Movement of orbital depots

The model also includes the movement of orbital depots, which supply fuel, spare parts and de-orbit modules to servicers. Like customer satellites, depots follow fixed orbital paths and are constrained to passive coasting. Consequently, their movement is also limited to coasting arcs in the network.

To capture resupply behavior, resupply events are not tied to fixed node locations but instead associated with specific arcs at specific times. $\mathcal{RS}$ is defined as the set of triplets (t, i, j) indicating that a refuel-eligible (coasting) leg $i \to j$ may depart at time t. Thus, if $(t, i, j) \in \mathcal{RS}$ and a servicer departs at time t on any arc a with $I(a) = i$ and $J(a) = j$, it can receive fuel, parts and/or de-orbit modules during the transfer.

The propellant a servicer can load while traversing a resupplying arc depends on its tank capacity, the refueling rate, and the arc duration. Depot capacity is neglected in the baseline (unlimited supply). Let F_d^{cap} be servicer d's propellant capacity and let RF_d denote the number of time steps required to refill from empty to F_d^{cap}. For an arc a with traversal time τ_a, the per arc refuel cap is defined as $F_{d,a} = \frac{\tau_a}{RF_d} F_d^{cap}$, which is further bounded in the MILP by the remaining tank space $F_d^{cap} - f_{dt}$ at the arc entry time.

Servicer propellant (used by the servicer for maneuvers) is distinguished from *service consumables* carried as payload for customers (e.g., spare parts, de-orbit modules, and *customer propellant* used in refueling services). Servicer propellant is *rate-limited* as described above. By contrast, for simplicity in this work service consumables are *not rate-limited*: on any resupplying arc the servicer may replenish up to the relevant payload capacities $Q_{d,m}^{cap}$ (for commodity m), i.e., replenishment depends only on payload limits, not on arc duration. These modeling choices are revisited in the MILP formulation, and can be relaxed to include depot capacity and commodity-specific transfer rates.

The identification of resupplying arcs for each depot begins at its initial node and orbit. The process simulates motion along coasting arcs, updating time based on τ_a, and generating triplets (t, i, j) until the time horizon T is reached. The logic is detailed in Algorithm 3.

Algorithm 3 Resupply Arc Generation Algorithm

Input time horizon T and network (N, A).
Input orbital depot set $\mathcal{R}$, where each depot $r \in \mathcal{R}$ has a known starting node and orbit.
Initialize resupply arc sets $A_t^R = \emptyset$ for all $t \in T$.
Create lookup table τ_a from arc set A, where τ_a is the traversal time steps of coasting arc a.
for each depot $r \in \mathcal{R}$ **do**
    Set *current_node* $=$ starting node of depot r.
    Set *current_time* $= 0$.
    for each $t \in T$ **do**
        if $t <$ *current_time* **then** ▷ A resupplying arc spans multiple time steps.
            Reuse last arc and add it to A_t^R
            continue
        end if
        Set *next_node* = the node following *current_node* in the orbit's direction of motion.
        Set arc $a = ($*current_node*, *next_node*$)$.
        Add arc a to A_t^R as a resupplying arc.
        Store arc a as last arc used by depot r.
        Set $\tau = \tau_{\text{current\_node,next\_node}}$.
        Update *current_time* $=$ *current_time* $+ \tau$.
        Update *current_node* $=$ *next_node*.
    end for
end for
return $A_t^R, \forall t \in T$.

To further support the explanation of the resupply arc generation process, the illustrative static network from Figure 3.1 is again used. In the same figure where the subtask creation was previously illustrated (c.f. Figure 3.2), an example of resupply arcs over multiple time steps is shown for two orbital depots. The depots, beginning in different orbits at nodes 2 and 4 respectively, follow passive coasting maneuvers and at each time step are associated with resupply arcs (highlighted in red) that can be used by robotic servicers. Table 3.2 lists the specific arcs and corresponding time steps at which each depot is available. Servicers requiring resupply must initiate traversal of one of these arcs at the specified times, defining the valid triplets $(t, i, j) \in RS$.

Table 3.2: Resupplying Arcs and Associated Time Periods for Each Depot

(a) Depot 1

From i	**To** j	**Time Periods** (t)
1	2	$t_1 + 2\Delta t$
2	3	t_1
3	1	$t_1 + \Delta t$

(b) Depot 2

From i	**To** j	**Time Periods** (t)
4	5	$t_1, t_1 + \Delta t$
5	6	$t_1 + 2\Delta t$

4

Rendezvous Maneuver Modeling and Discrete Transfer Representation

4.1. Introduction

The logistics network in Chapter 3 defines where and when transfers may occur; this chapter details *the orbital transfer module*, which specifies *how* those transfers are executed at the maneuver level. Each admissible arc a in the static network corresponds to an orbital maneuver option. The orbital transfer module assigns to every such arc a transfer time τ_a and a propellant fraction ϕ_a (with per time step propellant fraction ψ_a), enabling the MILP in Chapter 6 to evaluate time-fuel trade-offs on a discrete horizon and plan end-to-end rendezvous operations. The design goal is fidelity sufficient for scheduling decisions while remaining lightweight, vehicle-independent, and compatible with Rolling Horizon re-optimization.

Operationally, a servicer spacecraft must first perform an *out-of-plane orbital transfer* to reach the orbital plane of each target, followed by a phasing maneuver to align the along-track position, and finally rendezvous and, if applicable, docking operations required to execute the service. In this chapter, only the out-of-plane transfer and phasing maneuvers are explicitly modeled. The Δv cost associated with the final rendezvous and docking phase is *neglected*, as it is assumed to be small compared to the other transfer costs. Hereafter, the term *rendezvous* is used in a broader sense to denote the act of meeting the target in its orbital position, while only the costs of the modeled transfer and phasing components are considered. The general rendezvous condition is defined by the alignment of key orbital elements: inclination i, right ascension of the ascending node Ω, semi-major axis (altitude), and along-track phase (for near-circular orbits, the argument of latitude).

The orbital transfer module therefore employs three types of impulsive maneuvers designed to match the connectivity of the logistics network and to be combined as needed to achieve target rendezvous: (i) *coasting* along the orbit (zero propellant consumption, deterministic time of flight); (ii) *same-orbit phasing* between non-adjacent nodes using multi-revolution transfers that offer discrete time-fuel trade-offs; and (iii) a *combined cross-orbit transfer* that leverages J_2-induced RAAN precession by inserting into a drift orbit using a Hohmann transfer, coasting to close the RAAN gap, and executing a two-impulse return Hohmann transfer with a split plane change, where the inclination change Δi is divided between the impulses by an optimal fraction s. Coasting is straightforward and will not be discussed further. This maneuver set is consistent with the thesis scope, impulsive servicers operating in circular LEO orbits, and supports fast, fuel-aware planning for rendezvous with customer satellites.

Before detailing the maneuver strategies used for rendezvous, the chapter first establishes the perturbation environment and modeling assumptions that justify the transfer designs (Section 4.2). It then motivates and defines the maneuver set used in this work (Section 4.3) and finally it describes the algorithms that parameterize each transfer (Section 4.4), followed by brief notes on verification.

4.2. Orbital Perturbations

In astrodynamics, *orbital perturbations* are deviations from ideal two-body (Keplerian) motion caused by forces not captured by the point-mass model and by the non-spherical component of the primary's gravity field. Accounting for these effects is essential to describe actual spacecraft motion. Representative contributors for Earth satellites include the Earth's non-spherical gravity (zonal and tesseral harmonics), atmospheric drag, Solar Radiation Pressure (SRP), and third-body gravitational attractions (e.g. Sun and Moon). The set of perturbations included in any orbit computation depends on the accuracy required; in practice, only effects whose magnitude exceeds a prescribed threshold are retained (Wakker, 2015).

The perturbed motion, i.e., the satellite's acceleration, can be written as the sum of the central Newtonian term and a net perturbing acceleration (Cowell's formulation, (Wakker, 2015)):

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{a}_{\text{pert}}, \tag{4.1}$$

4.2.1. Orders of Magnitude

Across orbital altitudes, the two-body acceleration dominates while the relative importance of perturbations varies systematically with radius. Very near Earth, the largest effects are (i) the J_2 zonal harmonic due to Earth's equatorial bulge and (ii) atmospheric drag; with increasing altitude, drag decays rapidly, whereas J_2 remains significant through LEO and into Medium Earth Orbit. At GEO, third-body attractions and SRP become comparable or dominant (Wakker, 2015). These trends are illustrated in Figure 4.1, which compares the order of magnitude of major perturbing accelerations with orbital radius.

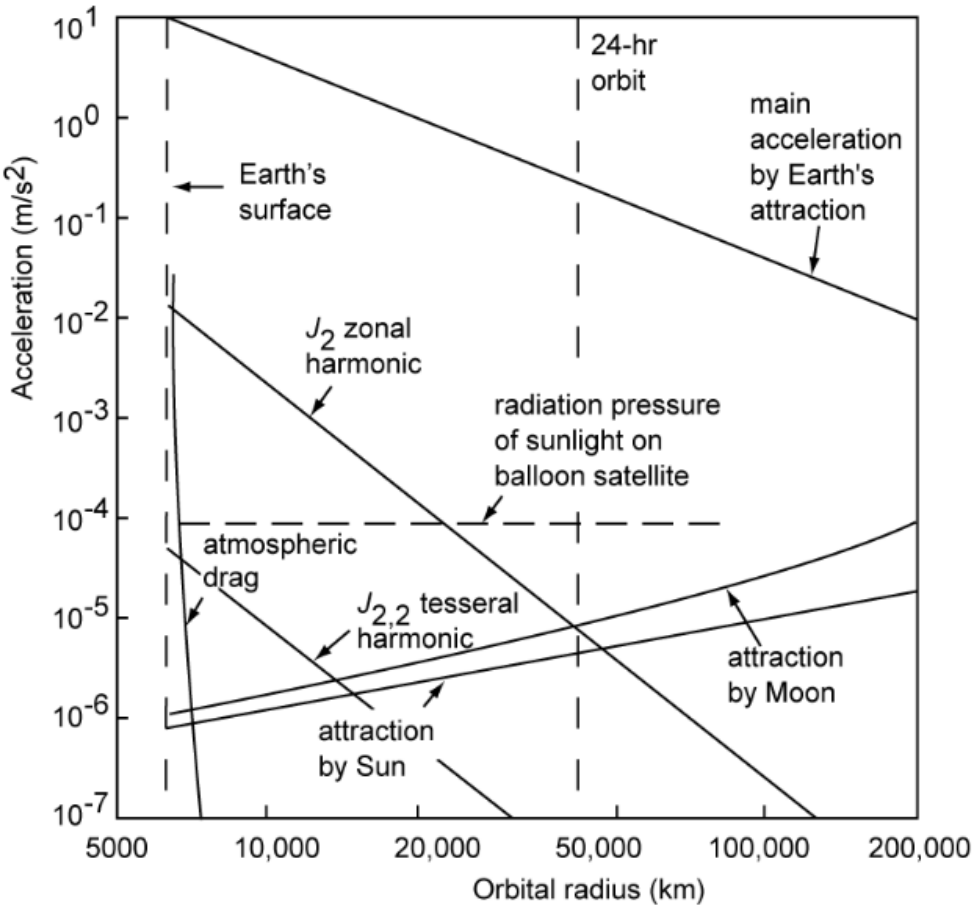


Figure 4.1: Order of magnitude of perturbing accelerations versus orbital radius (Wakker, 2015).

4.2.2. Why Differential Perturbations Matter Most

For IOS rendezvous scheduling in LEO, the governing quantity is *relative* motion, not the absolute perturbation on any single body. Two layers of "difference" matter. First, in a geocentric formulation the satellite's equation of motion is obtained by subtracting the Earth's acceleration; any perturbation that acts nearly equally on Earth and satellite is *common-mode* and largely cancels in the relative dynamics. Second, the chaser-target separation evolves according to the *difference* between the perturbations acting on each vehicle. Because gravitational accelerations are independent of spacecraft mass, third-body gravity from the Sun and Moon accelerates Earth and an Earth satellite almost equally; what remains after subtraction is a small differential term. By contrast, non-gravitational effects (e.g., atmospheric drag, solar radiation pressure) scale with spacecraft properties such as area-to-mass ratio, so design differences between chaser and target can produce genuine differential drifts that accumulate over time. In practice, the planning model therefore needs to emphasize perturbations that create meaningful differentials over the horizon, while treating effects that are common-mode or actively controlled as negligible (Wakker, 2015).

4.2.3. Modeled Perturbations

This work restricts case studies to near-circular Sun-synchronous orbits. Among LEO perturbations, the dominant effects are the Earth's oblateness (J_2) and atmospheric drag as illustrated in Figure 4.1. Because rendezvous planning is driven by *relative* (differential) motion rather than absolute perturbation magnitudes, the modeling retains only those effects that generate meaningful differentials over a given time horizon.

Atmospheric drag

Drag primarily induces a secular decay of the semi-major axis a (and damps the eccentricity), thereby changing the orbital period as well. In a standard free-molecular-flow model,

$$a_D \;=\; -\tfrac{1}{2}\,\rho(h)\,\frac{C_D A}{M}\,|\bar{v}|\,\bar{v}. \tag{4.2}$$

Consequently, drag accelerations scale with the spacecraft's area-to-mass ratio A/M and can, in principle, create *differential* drifts between vehicles with dissimilar designs (Wakker, 2015).

In the present SSO setting, atmospheric drag is treated as *negligible in relative terms* based on the operational assumption that routine stationkeeping is actively performed by client satellites: Customers are assumed to maintain their mean altitude (and thus their mean motion) through periodic stationkeeping. This suppresses long-term *differential* decay in a between targets and eliminates the primary channel by which drag would affect rendezvous timing.

Scope note: The study covers servicing of *active* clients and ADR on satellites that have *recently* failed. Up to the failure time, targets are assumed to have held their orbits via routine stationkeeping; after failure, the response time to ADR is taken to be short enough that drag-driven changes in a (and thus mean motion) between failure and rendezvous are negligible. Long-drifting debris, where differential drag effects become dominant, are excluded from the baseline model and left to future work, which could explicitly propagate post-failure drag evolution and failure-to-service timing to refine rendezvous feasibility and costs.

J_2 zonal harmonic

Earth's gravity field is modeled by a spherical-harmonic expansion of the geopotential, in which zonal and tesseral/sectorial terms capture departures from a perfect sphere. The leading correction coefficient is the first zonal term (J_2), which represents the equatorial bulge, and is orders of magnitude larger than higher-degree terms (e.g., $J_2 \approx 1.083 \times 10^{-3}$ while $|J_{n\geq 3}| \lesssim 2.6 \times 10^{-6}$). Consequently, first-order secular effects in LEO, are well described by retaining J_2 alone, while most higher harmonics average out except near resonances (Wakker, 2015).

The Earth's equatorial bulge (J_2) exerts a torque on the orbit's angular momentum vector, producing a steady precession of the orbital plane about the Earth's spin axis and a secular drift of the nodes along the equator (nodal regression, see Figure 4.2a). In addition, J_2 causes a secular rotation of the line of apsides within the plane (apsidal rotation, see Figure 4.2b). For first-order, averaged motion these effects are captured by the rates below (Cerf, 2013; Vallado, 2001):

$$\dot{\Omega} \;=\; -\frac{3}{2}\, n\, J_2 \left(\frac{R_E}{p}\right)^2 \cos i, \tag{4.3}$$

$$\dot{\omega} \;=\; \frac{3}{2}\, n\, J_2 \left(\frac{R_E}{p}\right)^2 \left(2 - \frac{5}{2}\sin^2 i\right). \tag{4.4}$$

The symbols in (4.3) - (4.4) denote:

$\mu_E \;=\; 398{,}600.4415$ km^3/s^2 $\;=\; 3.986004415 \times 10^{14}$ m^3/s^2	Earth gravitational parameter,
$R_E \;=\; 6378.1363$ km	mean equatorial radius,
$J_2 \;=\; 1.0826357 \times 10^{-3}$	first zonal harmonic; dimensionless,
a [km]	semi-major axis,
e [-]	eccentricity,
i [rad or deg]	inclination,
Ω [rad or deg]	right ascension of the ascending node,
ω [rad or deg]	argument of perigee,
$n \;=\; \sqrt{\mu_E/a^3}$ [rad/s]	Keplerian mean motion,
$p \;=\; a(1-e^2)$ [km]	semilatus rectum.

In a standard perturbation treatment, element variations are decomposed into *secular*, *long-period*, and *short-period* parts. Over the time horizons considered in this thesis and for near-circular SSOs, the periodic terms are small compared with the dominant secular effects; accordingly, *mean* (averaged) elements are used for propagation and scheduling. For first order modeling in J_2, the only nonzero secular rates are $\dot{\Omega}$, $\dot{\omega}$, and the mean-anomaly (or mean-longitude) rate, while $\dot{a} = \dot{e} = \dot{i} \approx 0$. Treating (a, e, i) as constant is therefore consistent with first-order theory and with LEO magnitudes (Cerf, 2013; Wakker, 2015).

Furthermore, for circular (or nearly circular) SSOs the perigee is undefined, so the argument of perigee ω is indeterminate. In this regime the natural in-plane phase angle is the *argument of latitude* $u = \omega + \nu$, where ν is the true anomaly. The argument of latitude is the angle measured in the orbital plane from the ascending node to the satellite, and it uniquely specifies the satellite's location within the plane; it thus replaces the pair (ω, ν) with a single parameter. Using *mean* elements, u advances uniformly with the mean motion (i.e., $u(t) = u_0 + n\,t$) and periodic oscillations are filtered (Cerf, 2013). First-order J_2 effects do not contradict this choice: *nodal regression* ($\dot{\Omega}$) rotates the orbital plane in inertial space but leaves the node direction fixed in the orbit-plane frame, while *apsidal rotation* ($\dot{\omega}$) is operationally irrelevant as $e \approx 0$ (the line of apsides is undefined). Hence each node in the time-expanded network can be assigned a fixed (mean) argument of latitude, with along-track evolution handled by the uniform advance of u; J_2 is accounted for separately through the slow inertial rotation of $\Omega(t)$ when cross-plane geometry matters.

With a, e, i taken constant under first-order J_2, $\dot{\Omega}$ may be treated as constant as well, yielding a linear secular evolution:

$$\Omega(t) = \Omega_0 + \dot{\Omega}\, t. \tag{4.5}$$

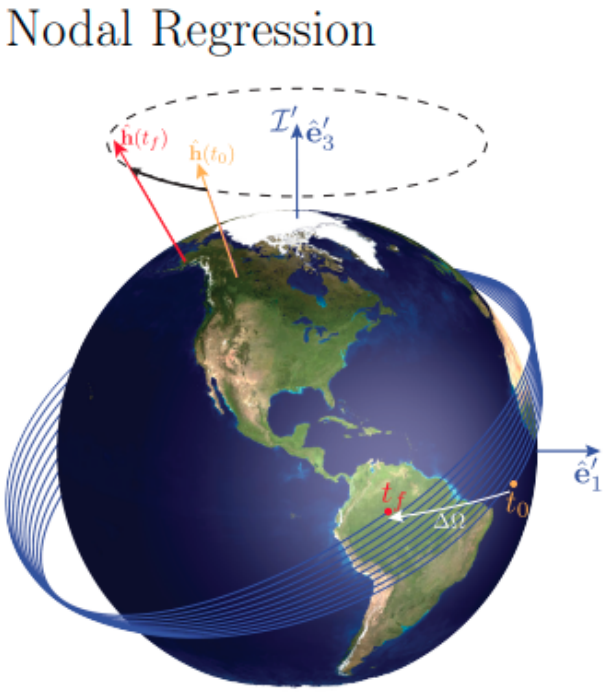
Satellites with similar (a, e, i) therefore exhibit nearly identical nodal precession; the differential rate

$$\Delta\dot{\Omega} \;=\; \dot{\Omega}(a_1, e_1, i_1) \;-\; \dot{\Omega}(a_2, e_2, i_2) \tag{4.6}$$

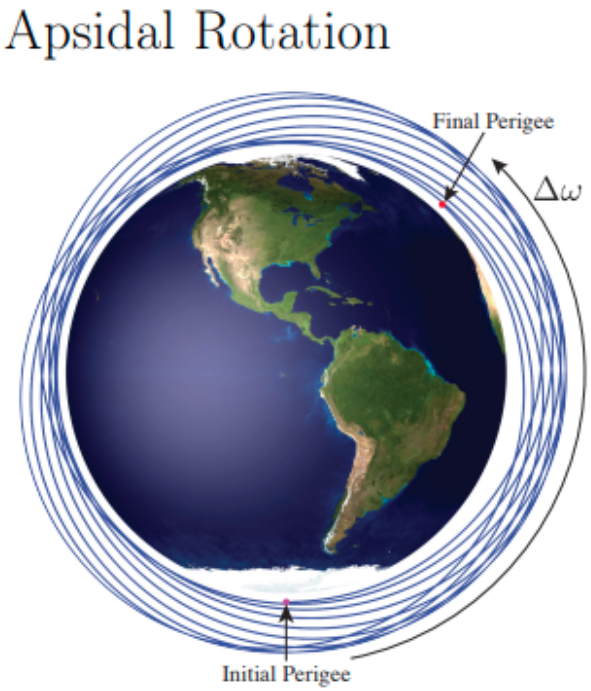
is close to zero, and their RAAN separation remains approximately constant over long horizons. In SSOs, $\dot{\Omega}$ is designed to match the Sun's apparent motion, i.e., the Earth's revolution rate around the Sun ($\approx 0.9856^\circ$/day), fixing a relationship among a, e, and i; consequently, the differential nodal rate between two satellites in SSOs is typically negligible. However, the differential J_2 effect can be *intentionally* exploited by inducing distinct nodal rates, for example, inserting the servicer into a drift orbit so that $\Delta\dot{\Omega} \neq 0$. The RAAN offset between two satellites then evolves linearly,

$$\Delta\Omega(t) \;=\; \Delta\Omega(0) \;+\; \Delta\dot{\Omega}\, t, \tag{4.7}$$

and can be closed with substantially less Δv than a direct plane change (Cerf, 2013).



(a) Nodal regression: the orbital plane precesses about Earth's spin axis; the line of nodes rotates by $\Delta\Omega$ while i remains (to first order) constant.



(b) Apsidal rotation: within the fixed plane, the ellipse rotates and perigee advances/regresses by $\Delta\omega$.

Figure 4.2: Two secular effects of Earth's oblateness (J_2) (Vallado, 2001). Left: $\dot{\Omega}$ from (4.3). Right: $\dot{\omega}$ from (4.4).

4.2.4. Implications for Orbital Maneuvers

- *Phasing maneuvers:* Phasing arcs are short compared to the nodal-precession timescale ($\dot{\Omega} \sim 1^\circ/\text{day}$ in SSO). Over one or a few revolutions, the change in RAAN during the transfer is negligible, so J_2 is neglected *within* the arc without loss of relative accuracy. Time of flight and Δv are therefore computed Keplerianly.
- *Combined maneuvers:* When an initial RAAN offset $\Delta\Omega$ exists, the developed combined strategy exploits J_2 by inserting the chaser into a *drift orbit* with nodal rate $\dot{\Omega}_d$ different from the target's, coasting until the gap is closed and then transferring back to the target's plane. The bracket transfers are modeled as impulsive Hohmann legs (the last one with a combined split plane change); their durations (tens of minutes) are negligible relative to the time on the drift orbit (days-weeks), so J_2 is again ignored *within* the short transfers, and retained only to set the drift-rate difference driving RAAN closure.

4.2.5. Why Sun-synchronous Orbits

SSOs are near-polar LEOs tuned so that the J_2-driven nodal precession rate matches the Sun's apparent motion ($\sim 0.9856^\circ/\text{day}$), keeping the orbital plane at a nearly fixed angle to the Sun and yielding passes at almost constant local solar time. SSOs are adopted here because they are both operationally relevant, providing stable illumination for Earth-observation and environmental-monitoring payloads, and modeling-friendly: satellites with similar (a, e, i) share nearly identical $\dot{\Omega}$, so RAAN separations remain essentially constant over the horizons considered. This near-uniform drift enables precomputation of transfer times and fuel costs for network arcs without time-indexed RAAN divergence terms. When cross-plane alignment is required, J_2 can be exploited via temporary drift orbits to close RAAN offsets with substantially less Δv than direct plane changes.

4.3. Maneuver Selection

In this work, servicers are assumed to use chemical propulsion and all burns are modeled as impulsive (negligible burn time). The perturbation environment retains only the first zonal harmonic J_2 to capture secular nodal precession; higher-order gravity terms, atmospheric drag, solar radiation pressure, and third-body effects are neglected over the time horizons considered (see Section 4.2). All customer spacecraft and the initial servicer configuration are placed in near-circular SSOs with nearly identical nodal precession, so inter-vehicle RAAN separations remain effectively constant. Within this regime, the maneuver set is selected to minimize propellant expenditure (Δv) while keeping the model lightweight and fully precomputable.

Direct out-of-plane corrections (RAAN or inclination) dominate Δv (and fuel costs) and scale steeply (e.g. on the order of 130 m/s per degree for RAAN in SSO), so the strategy deliberately avoids pure RAAN impulses. Instead, RAAN alignment is achieved by inserting into a drift orbit that modifies $\dot{\Omega}$ via J_2, coasting until the RAAN gap closes linearly in time, and then returning to the target plane; this J_2-assisted approach trades modest bracket Hohmann costs for substantial savings in out-of-plane Δv and is preferred when sufficient coast time is available ("weak" duration constraint), whereas direct-change or Lambert-based rendezvous become attractive only under "strong" time pressure (Cerf, 2013).

Following the drift, the model executes a two-impulse Hohmann transfer with the required plane change *embedded* in the bracket burns. Standard results show that combining the speed change with the plane change at the low-speed point (apogee) lowers the total Δv relative to sequential maneuvers. In addition, a small but consistent further saving is achieved by assigning a nonzero portion of the inclination change to the perigee burn rather than concentrating all of it at apogee. Accordingly, the model *splits* the required inclination change Δi between perigee and apogee and selects the fraction s that minimizes the total cost of the two combined burns (Curtis, 2020).

Because cross-orbit transfers require inclination changes that can only be performed at orbital nodes, combined maneuver arcs are restricted to transfers between ascending and descending nodes of different orbits.

For along-track phase closure on a common orbit, the model adopts classical two-impulse, multi-revolution phasing maneuvers, i.e., transfer to an elliptical phasing orbit and return to the original circular orbit after N revolutions. With purely in-plane tangential burns adjusting the orbital period, the specified phase closure together with an integer number of revolutions N maps in closed form to the phasing semi-major axis, yielding deterministic times of flight and simple Δv estimates. These are fuel-efficient and

inexpensive to compute and thus well suited as a discrete "menu" within the MILP scheduler. Increasing N reduces impulse cost with diminishing returns, so an upper bound N_{max} is imposed at the point where marginal Δv savings saturate; exposing $N = 1, \ldots, N_{\text{max}}$ as choices gives the optimizer a clear time-fuel trade without overfitting the trajectory layer. Compared with alternatives that require solving multi-revolution Lambert or primer-vector problems, this approach keeps the trajectory module lightweight and avoids nonlinearities, continuous optimization or iterative trajectory solves at the maneuver level (Curtis, 2020; Luo et al., 2007).

Overall, the selected maneuver types prioritize low-Δv rendezvous for high-thrust SSO operations while remaining lightweight and MILP-friendly for Rolling Horizon scheduling. If mission needs shift (e.g., tighter time bounds, different orbits, or low-thrust propulsion), the module is extensible: alternative transfer templates can be plugged in via the same interface by supplying arc attributes (τ_a, ϕ_a, ψ_a), without altering the network topology, MILP formulation, or Rolling Horizon logic.

4.4. Maneuver Implementation

4.4.1. Combined Maneuver

To connect nodes on different circular orbits in the base network, i.e., transfers between circular orbits with distinct altitude, RAAN, and/or inclination, $(r_1, i_1, \Omega_1) \to (r_2, i_2, \Omega_2)$, the servicer performs a *combined maneuver* comprising three phases and four impulsive burns:

1. **Hohmann transfer to the drift orbit:** Starting at t_1, raise/lower from the current circular orbit of radius r_1 to an intermediate circular drift orbit of radius r_d using two in-plane apsidal burns. No plane change is performed in this phase.
2. **Waiting phase for RAAN closure:** Coast on the drift orbit for a chosen drift time t_d at fixed inclination $i_d = i_1$ until the RAAN difference closes under J_2-driven nodal precession. No burns occur during this phase.
3. **Inclined Hohmann transfer to the target:** Starting at $t_2 = t_1 + t_d$, transfer from the drift orbit back to the target orbit with two apsidal burns that *combine* the in-plane impulses with an optimally split plane change to minimize total Δv. Let $\Delta i = |i_2 - i_1|$ and $s \in (0, 1)$ be the optimal split factor; then the plane-change components are $s\,\Delta i$ at the first burn and $(1 - s)\,\Delta i$ at the second.

All combined maneuvers are assumed to start and end at orbital nodes (ascending or descending), ensuring that inclination changes are applied at physically feasible plane-intersection points.

Phase 1

Given (r_1, i_1, Ω_1) and a candidate drift radius r_d, the insertion leg is a two-impulse Hohmann transfer between coplanar circles. The Hohmann transfer is the minimum-Δv two-impulse maneuver between *coplanar, concentric circular* orbits; bi-elliptic transfers only outperform it at very large radius ratios, so in LEO/SSO regimes the Hohmann remains the standard. It consists of a first tangential impulse at the initial circle of radius r_1 that injects the spacecraft onto an elliptical transfer orbit tangent to both circles, a coast for half of that ellipse, and a second tangential impulse at the final circle of radius r_d to circularize. The maneuver works in either direction, raising ($r_d > r_1$) or lowering ($r_d < r_1$) the orbit, by aligning each impulse with (raise) or opposite to (lower) the velocity vector to adjust the orbital energy appropriately (Figure 4.3).

The transfer ellipse has perigee $r_{p,H} = \min\{r_1, r_d\}$, apogee $r_{a,H} = \max\{r_1, r_d\}$, and semi-major axis:

$$a_H = \frac{r_1 + r_d}{2}. \tag{4.8}$$

From vis-viva, the circular speeds are $v_{c1} = \sqrt{\mu/r_1}$ and $v_{c,d} = \sqrt{\mu/r_d}$, while the speeds on the transfer ellipse at perigee and apogee are:

$$v_{t,r_1} = \sqrt{2\mu\left(\frac{1}{r_1} - \frac{1}{2a_H}\right)}, \qquad v_{t,r_d} = \sqrt{2\mu\left(\frac{1}{r_d} - \frac{1}{2a_H}\right)}. \tag{4.9}$$

The impulse magnitudes are then:

$$\Delta v_1 = \left| v_{t,r_1} - v_{c1} \right|, \qquad \Delta v_2 = \left| v_{c,d} - v_{t,r_d} \right|, \qquad \Delta v_H = \Delta v_1 + \Delta v_2 \tag{4.10}$$

, and the time of flight equals half the transfer-orbit period:

$$t_H = \pi\sqrt{\frac{a_H^3}{\mu}}. \tag{4.11}$$

In practice the same formulas apply whether the transfer is a raise ($r_d > r_1$) or a lower ($r_d < r_1$) because they use the *magnitudes* of the velocity changes at each burn (Curtis, 2020).

Since a Hohmann leg lasts only minutes to hours versus days to weeks on the drift orbit, any J_2-driven RAAN evolution is neglected *during* this leg and t_H is omitted from the RAAN-closure budget; only its impulse cost Δv_H is counted here.

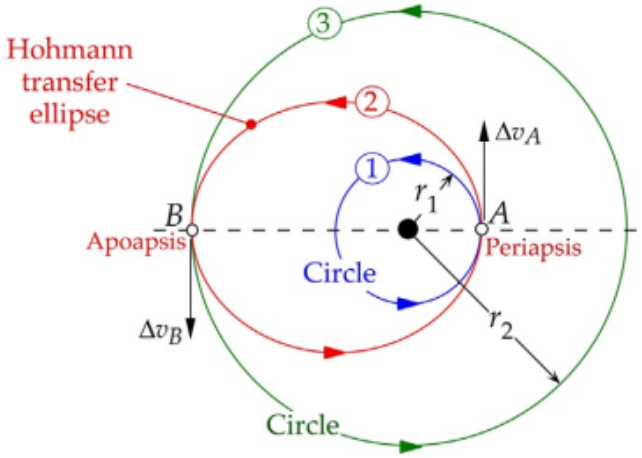


Figure 4.3: Generic two-impulse Hohmann transfer between coplanar circular orbits. At point A on the circle of radius r_1, the first tangential impulse Δv_A injects onto the transfer ellipse; after half an orbital period, a second tangential impulse Δv_B at point B circularizes on the circle of radius r_2. The same construction applies to orbit lowering with the roles of A and B interchanged (Curtis, 2020).

Phase 2

On the drift circle (r_d, i_d, Ω_d) with $i_d = i_1$ and $\Omega_d(t_1) = \Omega_1(t_1)$, the secular nodal rate for near-circular orbits is:

$$\dot{\Omega}(a,i) = -\tfrac{3}{2}J_2\left(\tfrac{R_E}{a}\right)^2\sqrt{\tfrac{\mu}{a^3}}\cos i \tag{4.12}$$

, thus, with i held fixed, $|\dot{\Omega}| \propto a^{-7/2}$. Let the initial raw RAAN difference be $\Delta\Omega_{\text{raw}} = \Omega_2(t_1) - \Omega_d(t_1)$. To ensure closure proceeds along the shortest arc, define the RAAN gap magnitude as $|\Delta\Omega_0| = |\Delta\Omega_{\text{raw}}|$, if $|\Delta\Omega_{\text{raw}}| \le 180°$; otherwise set $|\Delta\Omega_0| = 360° - |\Delta\Omega_{\text{raw}}|$. For the sign, $\Delta\Omega_0 > 0$ when the chaser's RAAN must *increase* toward the target's and $\Delta\Omega_0 < 0$ when it must *decrease*: concretely,

$$\Delta\Omega_0 = \begin{cases} +|\Delta\Omega_0|, & \Omega_d(t_1) < \Omega_2(t_1) \text{ and } |\Delta\Omega_{\text{raw}}| \le 180°, \\ -|\Delta\Omega_0|, & \Omega_d(t_1) < \Omega_2(t_1) \text{ and } |\Delta\Omega_{\text{raw}}| > 180°, \\ -|\Delta\Omega_0|, & \Omega_d(t_1) > \Omega_2(t_1) \text{ and } |\Delta\Omega_{\text{raw}}| \le 180°, \\ +|\Delta\Omega_0|, & \Omega_d(t_1) > \Omega_2(t_1) \text{ and } |\Delta\Omega_{\text{raw}}| > 180°. \end{cases}$$

During the coast the gap evolves linearly as:

$$\Delta\Omega(t) = \Delta\Omega_0 + \left[\dot{\Omega}_2(r_2, i_2) - \dot{\Omega}_d(r_d, i_1)\right](t - t_1) \tag{4.13}$$

, and closure over a chosen drift time t_d requires:

$$\left[\dot{\Omega}_d(r_d, i_1) - \dot{\Omega}_2(r_2, i_2)\right] t_d \;=\; \Delta\Omega_0. \tag{4.14}$$

Equation (4.14) fixes the required rate offset $(\dot{\Omega}_d - \dot{\Omega}_2) = \Delta\Omega_0/t_d$. In *retrograde SSO* ($i_1 > 90°$, $\cos i_1 < 0$), $\dot{\Omega} > 0$ and decreases with altitude; therefore:

$$\Delta\Omega_0 > 0 \Rightarrow \dot{\Omega}_d > \dot{\Omega}_2 \Rightarrow \text{lower } r_d, \qquad \Delta\Omega_0 < 0 \Rightarrow \dot{\Omega}_d < \dot{\Omega}_2 \Rightarrow \text{raise } r_d.$$

For *prograde* orbits ($i_1 < 90°$, $\cos i_1 > 0$), the raise/lower actions are reversed.

Two illustrative cases are shown in Fig. 4.4. *Left* ($|\Delta\Omega_{\text{raw}}| < 180°$): the chaser is behind, so set $\Delta\Omega_0 = +|\Delta\Omega_{\text{raw}}|$ and enforce $\dot{\Omega}_d > \dot{\Omega}_2$ (retrograde SSO: lower r_d). *Right* ($|\Delta\Omega_{\text{raw}}| > 180°$): the chaser is ahead, so wrap to $\Delta\Omega_0 = -(360° - |\Delta\Omega_{\text{raw}}|)$ and enforce $\dot{\Omega}_d < \dot{\Omega}_2$ (retrograde SSO: raise r_d). Choosing the opposite direction leads to very long closure times (Cerf, 2013).

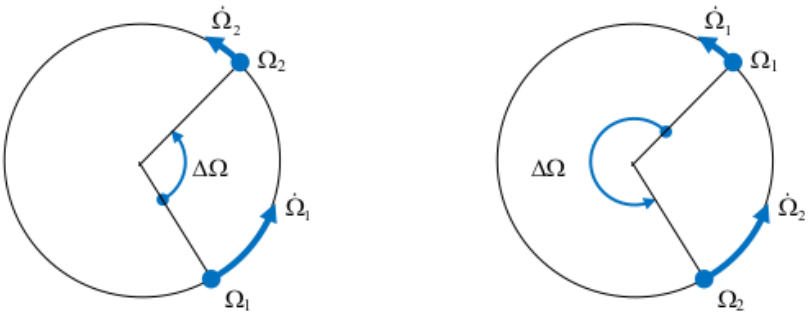


Figure 4.4: Forwards or backwards RAAN correction (examples for retrograde SSOs adapted from Cerf (2013)).

To avoid excessive waiting for small RAAN gaps, the drift time is chosen by a simple threshold rule:

$$t_d = \begin{cases} T_{\text{max}}, & |\Delta\Omega_0| > \Delta\Omega_{\text{TH}}, \\ \dfrac{|\Delta\Omega_0|}{\Delta\Omega_{\text{TH}}}\, T_{\text{max}}, & \text{otherwise} \end{cases}$$

, where T_{max} and $\Delta\Omega_{\text{TH}}$ are tunable parameters (see Section 8.1).

Given t_d, the implemented algorithm scans a bounded LEO altitude grid $h \in [400, 2000]$ km, sets $r_d = R_E + h$, evaluates $\dot{\Omega}_d(r_d, i_1)$ keeping all orbital parameters constant but the altitude, and accepts the first altitude satisfying the closure condition:

$$\left| \left(\dot{\Omega}_d(r_d, i_1) - \dot{\Omega}_2(r_2, i_2)\right) t_d - \Delta\Omega_0 \right| < \varepsilon_\Omega \tag{4.15}$$

, with a small tolerance ε_Ω (e.g., 10^{-2} deg) to balance runtime and accuracy. Once r_d is fixed, the Phase 1 Hohmann transfer cost Δv_H is computed; Phase 2 itself is a propellant-free coast of duration t_d (its contribution is time only, not Δv).

Phase 3

At the end of Phase 2, the RAANs are aligned, so the servicer transfers from the drift circle (r_d, i_1, Ω_2) at t_2 to the target circle (r_2, i_2, Ω_2) using a two-impulse Hohmann transfer with a *split plane change* (Figure 4.5). Let $\Delta i = |i_2 - i_1|$ and let $s \in [0, 1]$ denote the fraction of Δi applied at the first burn (the remainder $1 - s$ is applied at the second). Combining each tangential speed change with part of the plane rotation is more efficient than separating the plane change from the Hohmann legs.

The circular speeds at the endpoints are:

$$v_{c,d} = \sqrt{\mu/r_d}, \qquad v_{c2} = \sqrt{\mu/r_2} \tag{4.16}$$

, and the (coplanar) Hohmann transfer ellipse has semi-major axis:

$$a_H = \tfrac{1}{2}(r_d + r_2), \qquad v_{\text{t},r_d} = \sqrt{2\mu\left(\frac{1}{r_d} - \frac{1}{2a_H}\right)}, \quad v_{\text{t},r_2} = \sqrt{2\mu\left(\frac{1}{r_2} - \frac{1}{2a_H}\right)}. \tag{4.17}$$

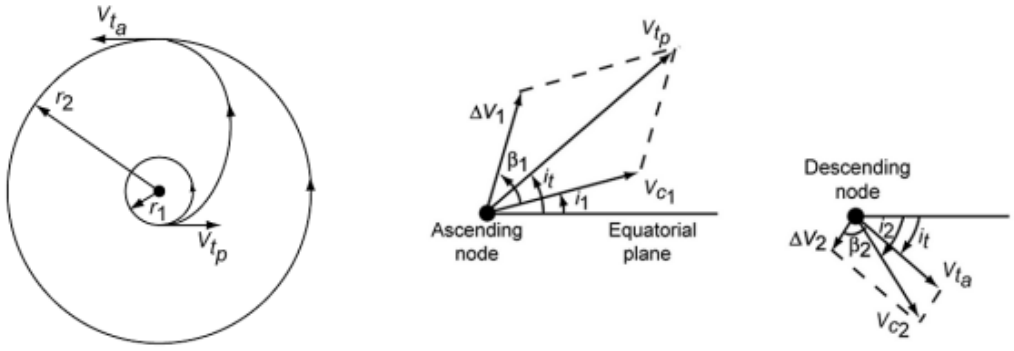


Figure 4.5: Illustrative Hohmann transfer with *split inclination change* (after Wakker (2015)). *Left:* geometry of the circular-to-circular transfer from radius r_1 to r_2 via a tangent ellipse (periapsis speed v_{tp}, apoapsis speed v_{ta}). *Middle:* vector diagram at the ascending node: the first impulse Δv_1 combines the tangential speed change with part of the plane change, taking the velocity from the circular value v_{c1} to the transfer value v_{tp}. *Right:* vector diagram at the descending node: the second impulse Δv_2 completes the plane change and circularizes, matching v_{ta} to the circular speed v_{c2}.

From the velocity-triangle geometry for a speed change accompanied by a plane change, the burn magnitudes are:

$$\Delta v_1(s) = \sqrt{v_{c,d}^2 + v_{\mathrm{t},r_d}^2 - 2\, v_{c,d}\, v_{\mathrm{t},r_d} \cos\big(s\,\Delta i\big)}, \tag{4.18}$$

$$\Delta v_2(s) = \sqrt{v_{c2}^2 + v_{\mathrm{t},r_2}^2 - 2\, v_{c2}\, v_{\mathrm{t},r_2} \cos\big((1-s)\,\Delta i\big)} \tag{4.19}$$

, and the return Hohmann transfer cost is:

$$\Delta v_{\mathrm{IH}}(s) = \Delta v_1(s) + \Delta v_2(s). \tag{4.20}$$

As standard results show, performing plane change where the speed is smaller reduces cost; hence the optimal split assigns more of Δi to the higher-radius (lower-speed) burn. However, concentrating the entire plane change at that burn is generally *not* optimal. Shifting a small portion of the angle to the other burn lowers $\Delta v_1 + \Delta v_2$ because each burn's cost is a vector difference, not a simple sum. For non-degenerate cases ($\Delta i > 0$, $r_d \neq r_2$), the minimizer therefore lies strictly in the interior $0 < s < 1$.

The optimal s minimizes $\Delta v_{\mathrm{IH}}(s)$. Differentiating (4.20) and setting $\mathrm{d}\Delta v_{\mathrm{IH}}/\mathrm{d}s = 0$ yields the condition:

$$\sin\big(s\Delta i\big) = \frac{\Delta v_1(s)\, v_{c2}\, v_{t,r_2}\, \sin\big((1-s)\Delta i\big)}{\Delta v_2(s)\, v_{c,d}\, v_{t,r_d}}, \tag{4.21}$$

, which is solved iteratively using the Newton-Raphson method. The Lisowski closed-form heuristic provides the starting guess:

$$s_0 = \frac{1}{\Delta i}\, \arctan\!\left(\frac{\sin\Delta i}{\sqrt{\frac{r_2^3}{r_d^3}} + \cos\Delta i}\right), \tag{4.22}$$

Obtaining the optimal s, the minimized cost $\Delta v_{\mathrm{IH}}(s)$ can be calculated (Curtis, 2020).

As in Phase 1, any J_2-driven RAAN evolution *during* this short leg is neglected, as well as the transfer's duration.

Total maneuver

The total Δv is the sum of the Phase 1 insertion and the Phase 3 inclined return. For timing in this work, only the Phase 2 drift coast is counted; the short Hohmann legs are neglected:

$$\Delta v_{\mathrm{tot}} = \Delta v_{\mathrm{H}} + \Delta v_{\mathrm{IH}}(s), \qquad T_{\mathrm{tot}} = t_d. \tag{4.23}$$

The corresponding propellant fraction (propellant-to-initial mass ratio) is obtained from the ideal Tsiolkovsky rocket equation (Curtis, 2020):

$$\phi_a \;\equiv\; \frac{m_{\mathrm{prop}}}{m_0} \;=\; 1 - \exp\!\left(-\frac{\Delta v_{\mathrm{tot}}}{g_0\, I_{sp}}\right) \tag{4.24}$$

, where g_0 is the standard gravity, I_{sp} is the servicer's engine specific impulse, m_0 is the servicer's pre-maneuver mass, and m_{prop} is the propellant consumed.

To represent the same maneuver in the discrete-time network, the maneuver time or arc traversal time, τ_a = T_{tot}, is discretized into N timesteps,

$$N \;=\; \Big(\,\text{round}\Big(\frac{\tau_a}{\Delta t}\Big)\Big)\,,$$

where Δt is the time step length of the Rolling Horizon algorithm. The total velocity increment (Δv_{tot}) is distributed evenly over the N time steps so that the per step velocity increment is

$$\Delta v_{\text{step}} \;=\; \frac{\Delta v_{\text{tot}}}{N}.$$

Applying the Tsiolkovsky relation to a single step gives the per step multiplicative propellant fraction:

$$\psi_a \;\equiv\; 1 - \exp\Big(-\frac{\Delta v_{\text{step}}}{g_0 I_{sp}}\Big), \tag{4.25}$$

i.e. the fraction of the *current* mass consumed as propellant mass in a time step. The per step multiplicative mass update is therefore:

$$m_{k+1} \;=\; m_k(1-\psi_a),$$

and after N identical steps the total propellant fraction ϕ_a satisfies:

$$\phi_a \;=\; 1-(1-\psi_a)^N,$$

which is algebraically equivalent to the Tsiolkovsky form $\phi_a = 1 - \exp\big(-\frac{\Delta v_{\text{tot}}}{g_0 I_{sp}}\big)$.

Verification
The Phase 1/2 implementation was verified against the drift-strategy analysis of Cerf (2013). Two aspects were checked:

- **Magnitude and sign of $\Delta\Omega_0$:** The RAAN gap is wrapped to the shortest arc $(-180^\circ, 180^\circ]$ and signed so that a positive value denotes that the chaser must *increase* its RAAN toward the target. This convention ensures closure along the minimal path (hence minimal drift time and cost) and matches the "forwards/backwards" rule illustrated in Figure 4.4.
- **RAAN-closure rate vs. drift altitude:** The closure rate produced by the algorithm was compared with Cerf (2013), Fig. 11. From that chart, for a *circular* drift orbit at $h_d \approx 690$ km starting from an SSO at $h \approx 800$ km and $i \approx 98.6^\circ$, the expected drift duration is ~ 20 days per degree of RAAN. The same figure (left panel) also provides the impulsive cost for the Hohmann insertion to, and return from, the drift altitude; for example, at $h_d \approx 750$ km the indicated cost is ~ 52 m/s. The implemented computations reproduce these magnitudes and trends.

The Phase 3 implementation is verified against Sec. 13.4 of Wakker (2015), which analyzes a Hohmann transfer between two circular orbits of different inclination. In that treatment the total impulse $\Delta V_{\text{tot}} = \Delta V_1 + \Delta V_2$ (here equivalent to $\Delta v_{\text{IH}}(s)$) is parameterized by the radius ratio $n = r_2/r_1$, the total inclination change Δi, and the *relative* inclination of the transfer ellipse Δi^*. In the present formulation the split factor s maps directly to that last parameter via $\Delta i^* = s\,\Delta i$, with the second burn rotating by $(1-s)\Delta i$.

The implemented algorithm computes $\Delta v_1(s)$ and $\Delta v_2(s)$ using Equations 4.18 and 4.19 (speed change with simultaneous plane rotation), forms $\Delta v_{\text{IH}}(s) = \Delta v_1(s) + \Delta v_2(s)$, and locates the minimizer s by Newton-Raphson iteration initialized by the Lisowski closed-form guess. The resulting optimal $\Delta i^* = s\,\Delta i$ and the normalized total impulse

$$\widehat{\Delta V} \;\equiv\; \frac{\Delta v_{\text{IH}}(s)}{v_{c,d}}, \qquad v_{c,d} = \sqrt{\mu/r_d},$$

are compared against the reference curves in Figures 13.5 - 13.6 of Wakker (2015) for representative $(n, \Delta i)$ pairs (with $v_{c,d}$ playing the role of V_{c1} in the textbook).

The observed behavior matches the cited results: (i) the optimal transfer-orbit inclination Δi^* is *much* smaller than Δi for $r_2 > r_1$, reflecting the preference to perform most of the plane change where the speed is lowest (e.g., for $n \approx 1.75$ and $\Delta i \simeq 30^\circ$, one finds $\Delta i^* \approx 5.2^\circ$); (ii) the dimensionless minimum $\widehat{\Delta V}$ follows the trends of Fig. 13.6 across n and Δi (for $\Delta i \approx 30^\circ$, $\Delta V_{\text{tot}} \approx 0.5\, v_{c,d}$ nearly independent of n); and (iii) enforcing $\Delta i^* = 0$ (no split; all rotation in one burn) reproduces the percentage penalties in the right panel of Fig. 13.6, which are small for large n (e.g., LEO→GEO) and larger for modest n and Δi.

These checks confirm that the optimizer assigns most, but not all, of the plane rotation to the higher-radius burn, and that the numerical minima agree with Equations (13.20) - (13.21) and the associated figures in Wakker (2015).

4.4.2. Phasing Maneuver

After the plane-change maneuver, once coplanarity is achieved, the servicer and target share the same circular SSO. The remaining task is to remove the *in-plane* phase offset, which is achieved with a phasing maneuver. A phasing maneuver is a two-impulse, tangent-ellipse trajectory between two different positions on the *same* circular orbit: the servicer (i) performs an impulsive burn to enter a phasing ellipse tangent to the circle at the maneuver point, (ii) completes an integer number $n \geq 1$ of revolutions on the ellipse, and (iii) performs a second impulse at the same tangent point to re-circularize, arriving there simultaneously with the target (P. Han et al., 2022). In the base network, this maneuver is represented by arcs linking two non-adjacent nodes on the same circular orbit.

Let r be the common circular radius, μ the gravitational parameter, and let u_s, u_t denote the arguments of latitude of servicer and target at a common epoch. The phasing angle to be closed is taken along the shortest in-plane arc (P. Han et al., 2022):

$$\Delta u = u_s - u_t, \qquad \Delta\phi = \begin{cases} \Delta u, & |\Delta u| \leq \pi, \\ -2\pi + |\Delta u|, & \Delta u > \pi, \\ 2\pi - |\Delta u|, & \Delta u < -\pi \end{cases}$$

, so that $\Delta\phi \in (-\pi, \pi]$. Intuitively, $\Delta\phi$ is the angle the *target* advances on the initial orbit while the servicer completes its phasing loop and returns to the tangent point.

Depending on the initial relative phase, the phasing transfer is executed either *forward* or *backward*. If the target *leads* the servicer at the maneuver point (the servicer is behind), it is advantageous to enter a *lower-energy* phasing ellipse with a shorter period than the circular orbit: the first impulse is opposite the velocity (deceleration) and the second is along the velocity to re-circularize when returning to the tangent point. Conversely, if the servicer *leads* the target (the target is behind), the phasing ellipse should have a *longer* period: the first impulse is along the velocity (acceleration) to raise the orbit and the second is opposite the velocity to return to the circular orbit. We refer to phasing with $a_{\text{phase}} < r$ as *forward* and with $a_{\text{phase}} > r$ as *backward*. Figure 4.6 illustrates both cases.

For a chosen revolution count n (assuming the cost-minimal phasing uses equal counts of full revolutions for target and servicer), the closure condition is:

$$\frac{2\pi n + \Delta\phi}{T_{\text{phase}}} = \frac{2\pi n}{T_{\text{circ}}}$$

, which yields the required phasing semi-major axis (i.e., the ellipse is chosen so that the chaser returns to the tangent point exactly when the target also arrives there):

$$a_{\text{phase}} = r \left(\frac{2\pi n + \Delta\phi}{2\pi n} \right)^{2/3}. \tag{4.26}$$

With this a_{phase}, the time of flight equals n periods of the phasing ellipse:

$$t_{\text{phase}} = 2\pi n \sqrt{\frac{a_{\text{phase}}^3}{\mu}}. \tag{4.27}$$

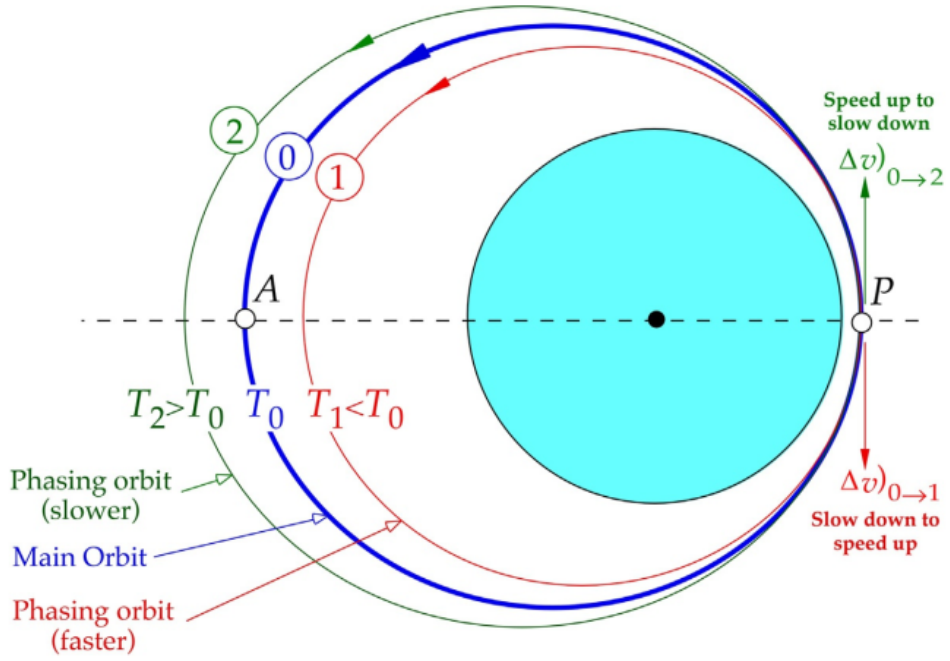


Figure 4.6: Phasing directionality on a common circular orbit. At the maneuver point P, a retrograde impulse ($\Delta v_{0\to 1} < 0$) drops the servicer to a faster, lower-energy phasing ellipse (red) with period $T_1 < T_0$ - this is *forward* phasing when the target is ahead. A prograde impulse ($\Delta v_{0\to 2} > 0$) raises to a slower, higher-energy phasing ellipse (green) with period $T_2 > T_0$ - *backward* phasing when the servicer is ahead.

Because the phasing ellipse is symmetric and tangent to the circular orbit at r, the in-plane impulses at entry and exit have equal magnitude, so the total cost is:

$$\Delta v_{\text{phase}} = 2\left|\sqrt{\mu}\left(\sqrt{\frac{2}{r}-\frac{1}{a_{\text{phase}}}}-\sqrt{\frac{1}{r}}\right)\right|. \tag{4.28}$$

Finally, the corresponding total propellant fraction follows from the rocket Equation 4.24 and the per time step propellant fraction is derived in the same way as in Section 4.4.1.

Multi-revolution phasing arcs

To expose a clear time-fuel trade to the optimizer, the network includes *multi-revolution* phasing arcs on a common circular orbit. The multi-arc strategy constructs one candidate arc per chosen revolution count, allowing the MILP to select the option that best satisfies servicing constraints (e.g., time windows, propellant).

For each ordered node pair a on the same circular orbit for which a phasing maneuver is admissible, the algorithm generates arcs for $n = 1, \ldots, N_{\text{max}}$ using Eqs. (4.26) - (4.28). Each n defines a distinct directed arc linking the same node pair, with its own phasing semi–major axis and duration.

As the number of revolutions n increases, the phasing maneuver cost Δv_{phase} decreases with diminishing returns, while the transfer time t_{phase} grows approximately linearly. Consequently, an upper bound N_{max} is calibrated in Chapter 8 and imposed at the point where additional revolutions yield negligible Δv savings. Defining the discrete set $n = 1, \ldots, N_{\text{max}}$ thus provides the MILP with a compact "menu" of time-fuel trade-offs without resorting to continuous trajectory optimization.

For each node pair and each n, the output comprises the arc attributes required by the network; in particular, the maneuver duration t_{phase} is mapped onto the network's discrete time grid used for the time expansion. The same discretization is applied to the combined maneuver arc durations.

Verification

The phasing implementation is verified by reproducing Example 6.5 of Curtis (2020), which performs a GEO longitude change in a fixed number of phasing revolutions. The prescribed longitude shift is

treated as the in-plane phase to be closed ($\Delta\phi = \Delta\Lambda = 12^\circ$ at GEO). The resulting phasing period, semi-major axis, and total Δv match the published solution, confirming the correctness of Eqs. (4.26) - (4.28) and the handling of multi-revolution phasing arcs.

5

Rolling Horizon Algorithm

5.1. Introduction

To optimize IOS operations, this thesis integrates dynamic Rolling Horizon decision-making with methods from space logistics, particularly the network flow model developed in the previous chapters. This network is formulated as a MILP and embedded within a RH framework that periodically updates input data, particularly service demand, to reflect the evolving system state and newly available information. This integration supports adaptive and computationally efficient planning, enabling responsive and robust scheduling in the face of uncertain and time-varying servicing needs. This is particularly important in IOS, where a portion of the service demand cannot be forecast deterministically and must be accommodated as it arises. While the detailed MILP formulation is presented in Chapter 6, the focus of this chapter is on the structure, design, and role of the RH framework.

To enable proactive, long-term, and sustainable decision-making in IOS, operators must account for the inherent uncertainty in service demand (Sarton du Jonchay et al., 2021). In this thesis, such uncertainty arises from randomly occurring service needs, specifically repair operations and active debris removal, which are modeled as stochastically generated tasks. These events are assumed to occur at unpredictable times, making strategic planning over extended time-frames particularly challenging. The RH optimization approach provides an effective method for addressing this uncertainty.

Rolling Horizon decision-making is a well-established approach for handling optimization problems in dynamic stochastic environments, where demand evolves over time and future conditions are inherently uncertain (Sethi & Sorger, 1991). This approach has been widely applied in scheduling problems under uncertainty, including in space-related contexts such as the optimal scheduling of Earth observation satellites (Dishan et al., 2013) and, more recently, in the operational planning of IOS infrastructures (Sarton du Jonchay et al., 2021).

The core idea is that instead of solving a single, large-scale optimization problem across the entire mission duration, RH divides the overall timeline, referred to here as the Scheduling Horizon (SH), into a sequence of smaller, overlapping segments known as Planning Horizons (PHs). Within each PH, operational decisions are optimized based on an assumed accurate forecast of service demands for that interval. Only the decisions corresponding to the initial segment of each PH, termed the Control Horizon (CH), are implemented and executed. While the PH specifies how far ahead the model *looks*, the CH specifies how much of that plan it *commits* to before re-solving. After each CH, the system state is updated, new information is incorporated (i.e., new tasks/service demands), and the optimization is re-run for the next PH. This iterative process continues at regular intervals, with the Planning Horizon continuously ”rolling forward” in time (Sarton du Jonchay et al., 2021; Sethi & Sorger, 1991).

Figure 5.1 illustrates the RH concept, highlighting three consecutive PHs and CHs within a traditional RH cycle.

The RH framework allows the system to adapt dynamically to new service demands by re-optimizing operations as new information becomes available (Silvente et al., 2015). While the schedule may change with each iteration due to updated forecasts and tasks, this flexibility is a core strength of the approach, ensuring feasible and efficient decision-making under uncertainty. As illustrated in Figure 5.2, certain tasks may initially be omitted simply because no information about their occurrence

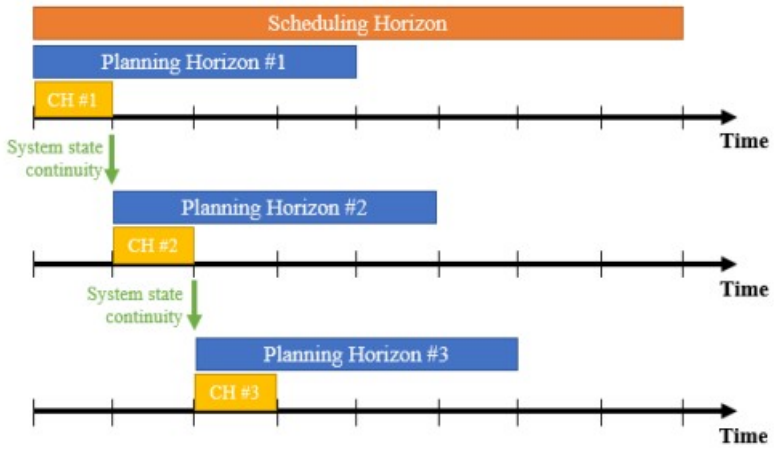


Figure 5.1: Illustrative example of traditional Rolling Horizon approach (Sarton du Jonchay et al., 2021).

is yet available. As the system progresses and additional data becomes accessible, these tasks are incorporated into subsequent Planning Horizons.

In the RH framework, each PH is optimized independently, without accounting for events beyond its time window. As a result, the aggregated solution over the full Scheduling Horizon may be suboptimal, particularly in dynamic environments where future conditions are uncertain. This is a well-known limitation of RH approaches. Therefore, the selection of an appropriate PH length is crucial to strike a balance between short-term responsiveness and long-term solution quality (Silvente et al., 2015).

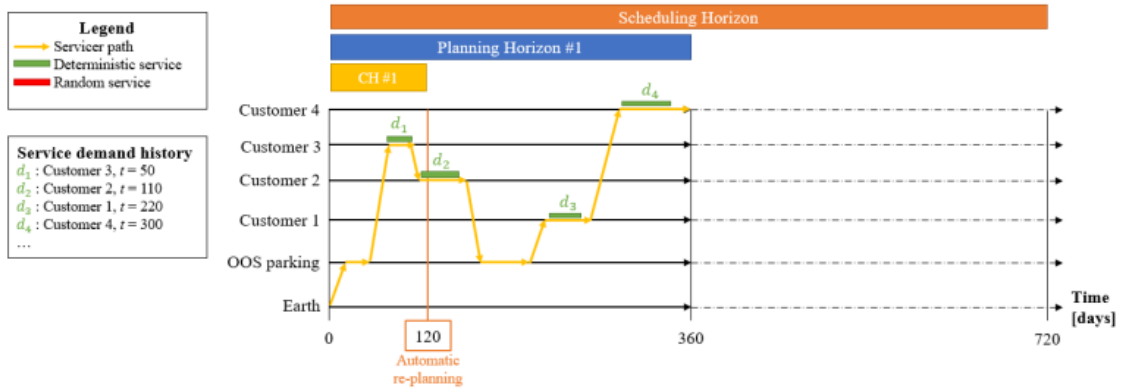
The optimal length of the PH depends on several factors, including the responsiveness of the system and the accuracy of demand forecasts. In systems with limited flexibility that require decisions to be made well in advance, longer PHs are generally more suitable, provided that future demand can be forecast with sufficient accuracy. In contrast, for highly dynamic and unpredictable environments, where reliable long-term forecasting is not feasible, shorter horizons are often more effective. They enhance the system's ability to react promptly to new information and reduce the risk of committing to infeasible or suboptimal long-term plans (Kopanos & Pistikopoulos, 2014).

The study by Kopanos and Pistikopoulos (2014) demonstrates a clear relationship between the length of the PH and solution quality. In their case study, extending the PH from one to three time intervals resulted in a significant improvement in the total objective value, by up to 47%. This demonstrates the advantage of broader planning scopes: longer PHs generally yield better solutions, provided that demand forecasts are sufficiently accurate. However, even with a three-period PH, the solution remained approximately 7% worse than the benchmark case with perfect information (i.e., full-horizon optimization with exact demand knowledge). This underscores a fundamental limitation of Rolling Horizon methods, their performance is bounded by the quality and scope of the available forecast data.

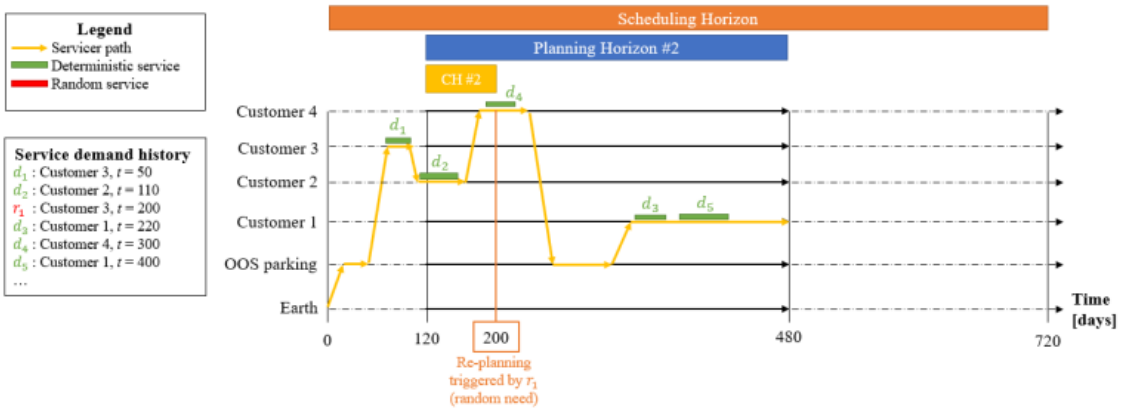
Ultimately, the choice of PH reflects a fundamental trade-off between computational effort and decision quality. Longer horizons typically lead to improved planning outcomes by capturing more future context, but they also increase model complexity and computational burden, particularly when solving large-scale MILP formulations. In contrast, shorter horizons reduce computational demands and enable faster responsiveness, but often at the expense of overall solution optimality (Kopanos & Pistikopoulos, 2014). Accordingly, Chapter 8 specifies and justifies the PH length adopted for the case studies.

Furthermore, to ensure feasibility, a critical requirement of the RH framework is the consistency of the system state across successive iterations. Specifically, the terminal state at the end of each CH must be accurately propagated as the initial state for the subsequent PH. This continuity is essential for maintaining the feasibility of the solution over time and enables IOS operators to implement responsive and efficient servicing strategies under uncertainty (Sarton du Jonchay et al., 2021; Silvente et al., 2015).

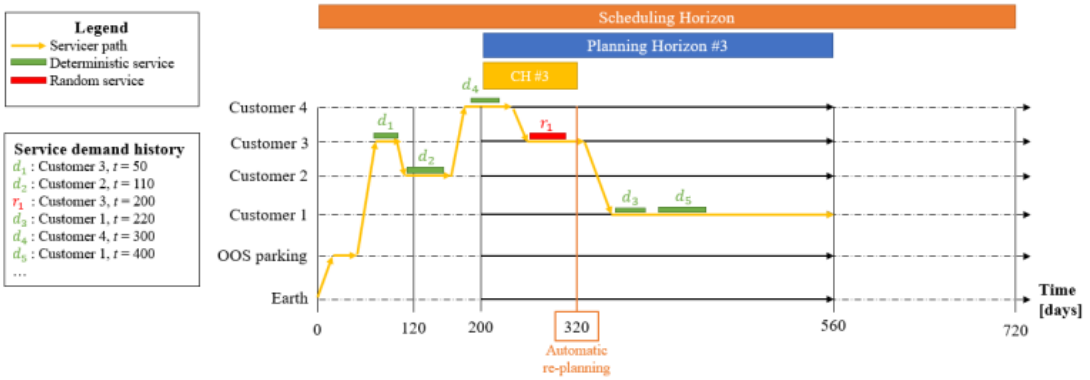
The continuity requirement, along with the reactive and iterative nature of the RH approach, is illustrated in Figure 5.3.



(a) Planning Horizon #1 with Control Horizon #1; initial execution and first automatic replanning trigger.



(b) Planning Horizon #2 with Control Horizon #2; replanning triggered by a new random service request.



(c) Planning Horizon #3 with Control Horizon #3; continued execution and subsequent automatic replanning.

Figure 5.2: Rolling Horizon execution of IOS operations. Each panel shows a servicer's path (yellow), deterministic services (green), and random services (red) over successive planning and Control Horizons within the overall Scheduling Horizon (Sarton du Jonchay et al., 2021).

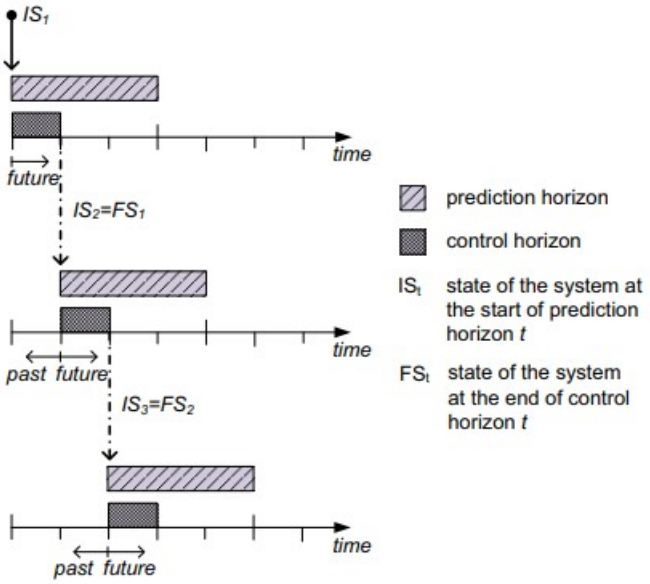


Figure 5.3: Illustration of the continuity requirement and the iterative, reactive nature of the Rolling Horizon framework. Adapted from Kopanos and Pistikopoulos (2014).

5.2. Application to IOS

As shown in Figure 5.3, the standard RH method updates the system's operational plan at fixed intervals, each corresponding to the length of the CH. This approach assumes that service demand can be reasonably forecast over the duration of each PH. While this assumption may hold in certain logistical contexts, it becomes problematic in the case of IOS, where events such as satellite malfunctions and end-of-life failures are inherently uncertain and cannot be predicted with accuracy in advance.

To address this, a modification to the traditional RH strategy is adopted from Sarton du Jonchay et al. (2021) and applied here. In this approach, re-optimization is triggered not only at fixed intervals but also upon the arrival of random service demands, which cannot be forecast in advance and are revealed in real time. These event-driven updates improve responsiveness by re-solving exactly when the realized system state changes. If no such events occur within a predefined time frame, periodic replanning is still performed, controlled by a maximum allowable idle time, to ensure continued responsiveness to gradual system evolution. As a result, the Control Horizon length varies.

Under this approach, the demand forecast within each PH includes two components: (i) the newly realized random task that triggered replanning (when applicable), and (ii) all previously known tasks, deterministic or stochastic, whose service windows intersect the PH and remain unserved. As in the traditional RH framework, it is essential to maintain state continuity, that is, the state of the IOS system at the end of each CH must be propagated as the initial condition for the following PH. Here, the system state at any time t includes the positions and quantities of all relevant commodities distributed across the nodes of the static network.

Figures 5.2a–5.2c illustrate the RH execution of IOS operations on an illustrative logistics network taken from Sarton du Jonchay et al. (2021). The servicer's itinerary is represented in yellow, the deterministic services in green and random services in red, together with a log of request arrivals presented on the left.

Figure 5.2a shows the plan over the first Planning Horizon (PH#1) and Control Horizon (CH#1). Because no random request appears during the execution of the initial window, an automatic replanning is triggered at $t = 120$ days. After rolling the PH forward (Figure 5.2b, i.e., to $t \in [120, 480]$ days), the portion already executed ($t \in [0, 120]$) is frozen while the remaining schedule is re-optimized; a new deterministic request d_5 alters the plan beyond $t = 120$, and the realization of a random request r_1 at $t = 200$ prompts another replanning step. The updated itinerary after serving r_1 is shown in Figure 5.2c; a subsequent automatic replanning occurs at $t = 320$. Note that the CH is event-driven in this implementation and therefore does not have a fixed length.

For readability, only one servicer is depicted, but the RH procedure applies to any fleet size. At each

PH, the IOS logistics problem is formulated and solved as a MILP over the current window, yielding a sequence of MILP solves across the overall Scheduling Horizon.

5.3. Algorithm Implementation

Now that the theoretical foundations have been presented, this section describes the practical implementation of the RH algorithm and its integration with the broader IOS operational optimization framework.

To provide an overview of the software architecture, Figure 5.4 summarizes the structure of input files, supporting modules, and their interactions within the Rolling Horizon framework.

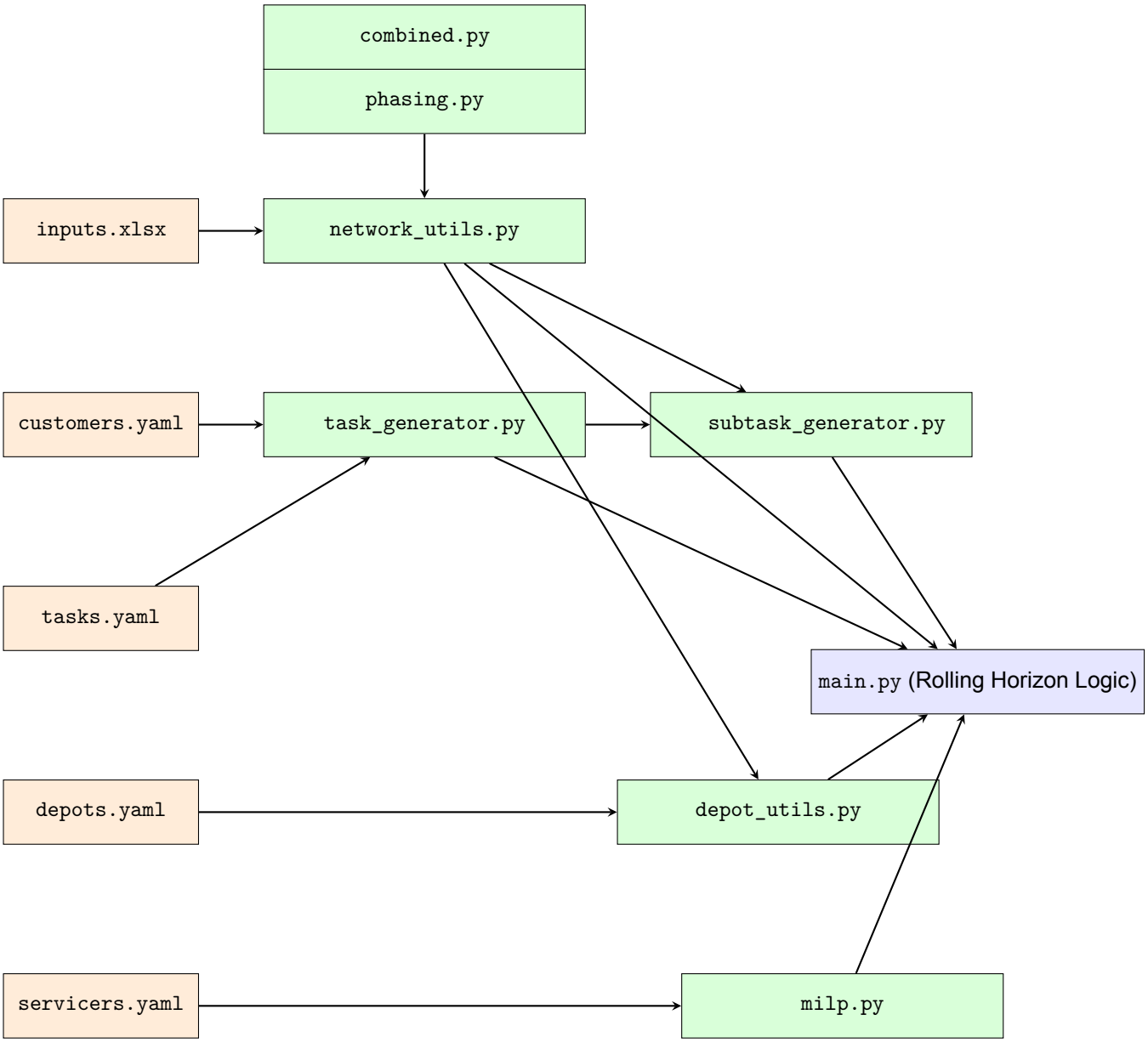


Figure 5.4: Flowchart illustrating the input files and module dependencies that structure the Rolling Horizon algorithm implementation.

The implementation is handled via a central script, `main.py`, which coordinates the Rolling Horizon logic. The structure of the algorithm is organized into two main phases: the initial setup and the iterative RH cycle.

5.3.1. Setup Phase

The setup phase prepares all data inputs and static structures required throughout the RH procedure:

- **Simulation parameter setup:** Key simulation parameters, including time step size (Δt), scheduling and Planning Horizon lengths, and maximum allowed idle time, are initialized at the beginning of the script.
- **Static orbital network construction:** Orbital node data are loaded from `inputs.xlsx` and passed to the arc creation functions implemented in `network_utils.py`. These functions, based on the arc generation algorithm 1 described in Chapter 3.2, construct the static network by enumerating admissible node pairs and interconnecting arcs and delegating the computation of transfer attributes to the orbital transfer module: `phasing.py` supplies the closed-form maps for same-orbit phasing arcs, and `combined.py` supplies the maps for cross-orbit transfers with J_2-assisted RAAN closure. In both cases the formulas for $(\tau_a,\, \phi_a,\, \psi_a)$ come from Chapter 4; `network_utils.py` simply queries these modules and attaches the results to each arc.
- **Service demand generation:** All deterministic and stochastic (random) service needs over the full Scheduling Horizon are precomputed in a separate module, `task_generator.py`, as further detailed below (cf. Section 5.3.1, Service Demand Generation). Deterministic tasks are generated at fixed intervals based on a specified inter-occurrence time, while random tasks follow a Poisson process characterized by a defined mean inter-occurrence time. The corresponding subtasks are then generated dynamically within each Rolling Horizon iteration using `subtask_generator.py`, ensuring that only the subtasks relevant to the current planning window are computed.
- **Orbital depot integration:** The depot configuration specified in `depots.yaml`, consistent with the modeling framework described in Section 2.2.2, is loaded and parsed for integration into the optimization environment.
- **System state initialization:** Initialize the system conditions and set the starting simulation time.

Service Demand Generation

To support the simulation and planning of service demands, all tasks are precomputed for the entire Scheduling Horizon prior to the execution of the Rolling Horizon framework. This pre-computation step generates two types of service tasks, deterministic and stochastic, based on configuration data defined in external YAML files. The demand generation procedure begins by loading structured input data for task definitions and customer profiles from `tasks.yaml` and `customers.yaml`, respectively. These inputs define the parameters of each task type, e.g., (mean) inter-occurrence time, and provide customer-specific information such as orbital assignment, starting location, and assigned service needs, in accordance with the modeling approach outlined in Section 2.1. The algorithm then iterates over all customers and generates task instances based on the corresponding definitions. Deterministic tasks are scheduled at fixed intervals, specified by their associated inter-occurrence time parameter, and are instantiated repeatedly from their first occurrence up to the mission's end, i.e., the end of the Scheduling Horizon.

For the generation of stochastic service demands, the algorithm assumes that such events (e.g., satellite malfunctions or end-of-life) occur randomly and independently over time. This behavior is modeled using a Poisson process, a widely used stochastic model for random event occurrences. Specifically, inter-occurrence times between events are treated as independent and identically distributed (i.i.d.) random variables following an exponential distribution (Florescu, 2014).

Let μ denote the mean inter-occurrence time defined in the task configuration file. The exponential distribution is then parameterized by the rate $\lambda = \frac{1}{\mu}$, yielding a probability density function $f(t) = \lambda e^{-\lambda t}$ for $t \geq 0$. The key property of this distribution, the memoryless property, ensures that the probability of an event occurring in the next interval is independent of how much time has already passed. This makes it particularly suited for modeling unpredictable and uncorrelated service needs in space environments (Florescu, 2014).

In practice, inter-occurrence times are sampled from the exponential distribution using inverse transform sampling, implemented via the `random.expovariate` function in Python. These samples are accumulated until the total time exceeds the Scheduling Horizon, producing a time series of randomly timed task occurrences. Although all stochastic tasks are generated at the start of the simulation to ensure reproducibility across experimental runs, they are only revealed to the RH optimizer in real time, as their

occurrence time falls within the current simulation window. Each time a new random task becomes active, it triggers a re-optimization of the schedule and is incorporated into the upcoming Planning Horizon. This approach emulates the limited foresight available in real-world operations, while still allowing for controlled and repeatable experimentation.

5.3.2. Rolling Horizon Loop

The core of the algorithm is an iterative loop that advances through successive Planning Horizons, dynamically updating the schedule based on current system state and newly available information. Each iteration performs a series of structured steps, as outlined below and illustrated in the flowchart in Figure 5.5.

- **Deterministic task filtering:** Identifies all uncompleted (*task flag* `done` = `False`) deterministic tasks whose service windows intersect the current PH and includes them in the demand forecast for scheduling.
- **Random task inclusion:** Adds to the PH forecast any random task that triggered the current re-optimization, along with any previously triggered but uncompleted (*task flag* `done` = `False`) random tasks whose service windows still overlap with the current PH.
- **Subtask computation:** The subtasks corresponding to the tasks within the current planning horizon are computed in `subtask_generator.py`. This process is based on the orbital dynamics and spatial distribution of customer satellites, following the subtask generation algorithm presented in Algorithm 2 of Chapter 3.3.
- **Resupply integration:** The depot configuration loaded from `depots.yaml` is used to dynamically generate time-dependent resupply arcs for each orbital depot, simulating their orbital motion. This procedure, implemented in `depot_utils.py`, follows the resupply arc generation algorithm presented in Algorithm 3 and described in detail in Chapter 3.3.
- **MILP optimization:** Formulates and solves the IOS planning problem over the current PH using the MILP in Chapter 6, returning the schedule and resource flows to execute over the CH.
- **Random task triggering:** Detects the activation of new random service demands in real time and triggers re-optimization when such events occur.
- **Replanning trigger decision:** Determines the start of the next iteration based on either the arrival of a new random task or the maximum allowed idle time.
- **State update:** Implements the scheduling decisions corresponding to the current Control Horizon (bounded by the computed replanning trigger time), updates the system state accordingly, and advances the current simulation time to the next planning step; further detailed below in Section 5.3.3.
- **Termination:** The loop terminates once the current time reaches or exceeds the total Scheduling Horizon.

5.3.3. Control Horizon Execution and State Propagation

This subsection details the execution of the Control Horizon within the Rolling Horizon loop and the propagation of the system state across replanning boundaries. Specifically, it commits the plan for the current CH and prepares consistent initial conditions for the next re-optimization. Given (i) the optimized model solution over the active PH, (ii) the current infrastructure state, and (iii) the replanning boundary (determined by the event or periodic trigger), the routine applies the executed portion of the schedule, updates all system state variables at the replanning boundary, and records any in-progress activities (e.g., mid-arc transfers and mid-service executions) that must continue in the subsequent MILP. By explicitly carrying *in transit* and *in service* flags across the replanning boundary, this method guarantees temporal continuity and feasibility when the MILP is re-solved under updated information. Without these continuity constraints, re-optimization could artificially interrupt transfers or services at replanning boundaries, leading to infeasible or non-implementable schedules. The resulting state at the replanning boundary is passed as the initial condition for the next PH solve. Lastly, the simulation clock advances to the replanning time.

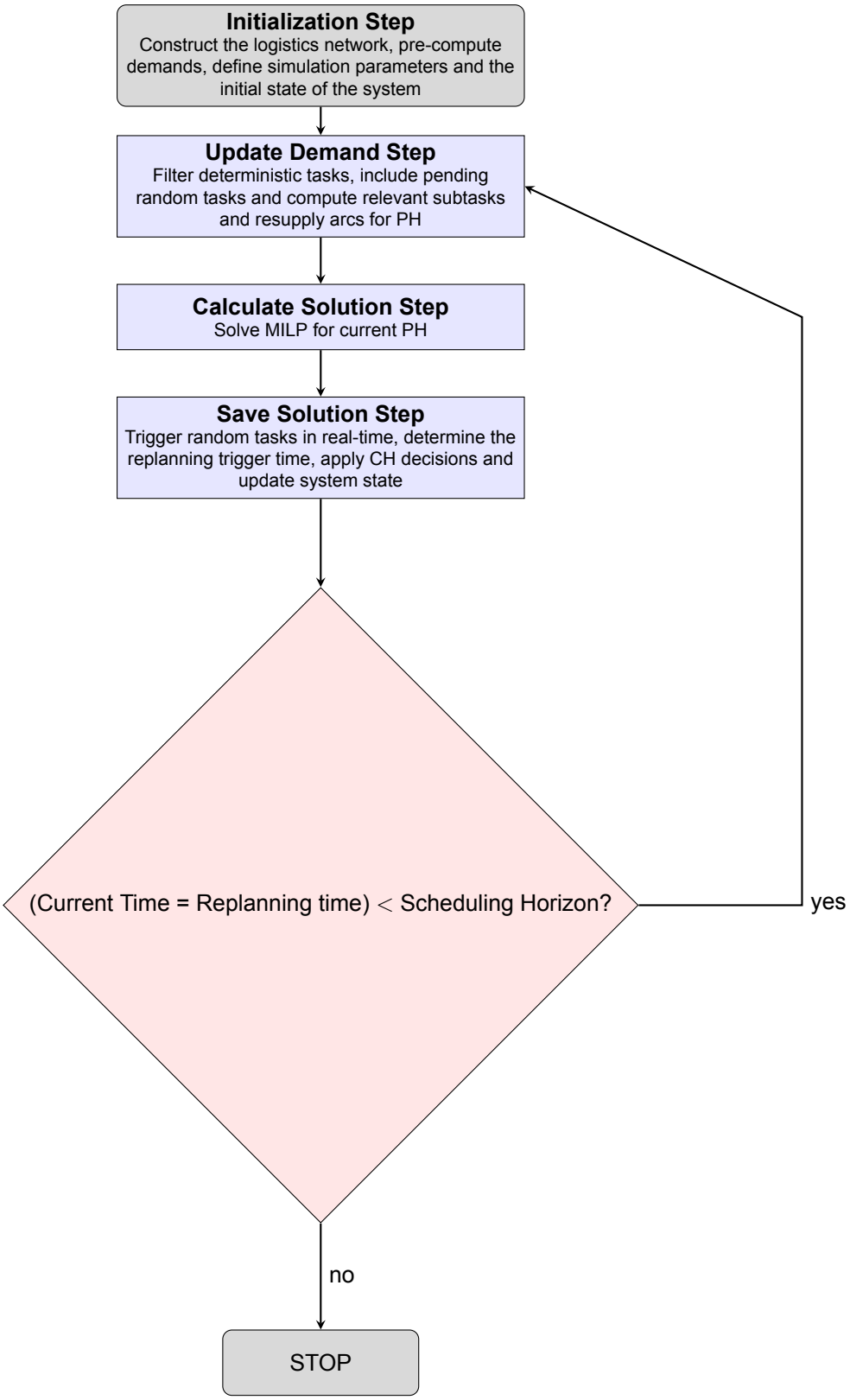


Figure 5.5: Flowchart of the Rolling Horizon algorithm.

Inputs and outputs
Inputs: the optimized MILP model solution, the mutable `infrastructure_state` dictionary, the next replanning time, and the list of service tasks.
Outputs: `infrastructure_state` updated with (a) per-servicer position, fuel, and payload quantities at the replanning time; (b) path and resource histories up to the replanning time; (c) flags for *mid-arc* and *mid-service* conditions; and (d) task completion marks for services that finished by the replanning time.

Procedure
The implementation follows these stages:

1. **Initialize `infrastructure_state`:** If `infrastructure_state` is empty (as in the first iteration), initialize per-servicer records. For each $d \in \mathcal{D}$, create:

   (i) *boundary-state* fields $\{s_d, F_d^0, Q_{dm}^0, \texttt{in\_transit}, \texttt{in\_service}\}$ to carry the state passed to the next PH

   (ii) *logging* fields $\{\texttt{path\_history}, \texttt{services\_done}, \texttt{fuel\_history}, \texttt{cargo\_history}\}$ to record the executed trajectory, completed services, and time series of fuel and commodity inventories up to the next replanning time

   Together, these structures preserve the executed CH and provide the boundary conditions (node, fuel, payload, interrupted activities) for the subsequent PH.
2. **Save servicer trajectories up to the replanning time:** Parse the model's solution moves and extend each servicer's `path_history` with arcs that *arrive* on or before the replanning time. Special handling merges any trailing "dummy" arc in `path_history` ending on dummy node X_d (a placeholder for an unfinished arc) with the current optimization window's first move to keep the history continuous.
3. **Append resource histories:** Copy time series for fuel and each commodity up to $t^{\text{replanning}}$ into `fuel_history` and `cargo_history` respectively for logging purposes.
4. **Set discrete starting position:** Set s_d to the node arrived at by $t^{\text{replanning}}$. If mid-arc at the boundary, set $s_d = X_d$ (dummy) and rely on the forced remainder arc $X_d \to j$ in the next MILP.
5. **Read fuel and payload at the boundary:** At the replanning time, query the model variables f_{dt} and q_{dmt}, i.e., propellant and commodity inventories respectively, and store them in `infrastructure_state` as the next PH initial conditions F_d^0 and Q_{dm}^0.
6. **Mark completed services:** Any service performed with $t_{\text{end}} \leq t^{\text{replanning}}$ is logged into `services_done` and its task flagged as `done`.
7. **Flag in-progress activities:**

   - *Mid-arc:* If a move on an arc satisfies $t_{\text{start}} < t^{\text{replanning}} < t_{\text{arrival}}$, mark the servicer as in transit by setting $\texttt{in\_transit} = \{I(a), J(a), t_{\text{start}}, t_{\text{arrival}}, \psi_a\}$. Append to `path_history` a truncated segment of the original arc $= \{I(a), X_d, t_{\text{start}}, t^{\text{replanning}}\}$, with X_d a dummy node indicating an unfinished arc. In the next MILP, continuity is enforced by applying a constraint to the next MILP to complete the arc from X_d to $J(a)$.
   - *Mid-service:* If a service interval satisfies $t_{\text{start}} < t^{\text{replanning}} < t_{\text{end}}$, mark the servicer as actively servicing by setting $\texttt{in\_service} = \{v, t_{\text{start}}, t_{\text{end}}\}$. In the next MILP, this flag allows the enforcement of the continuity of the service across the boundary: the servicer must remain at the customer location and continue the operation for the remaining service duration before any subsequent movement is allowed.
   - *MILP enforcement at the boundary:* In the next PH, the MILP treats these flags as hard initial condition constraints. If `in_transit` is set, a continuation arc $X_d \to j$ with residual travel time $t_{\text{arrival}} - t^{\text{replanning}}$ (and corresponding fuel use) is forced, blocking any alternate departure until completion. If `in_service` is set, the servicer is forced to execute the unfinished service right away for the remaining service time $t_{\text{end}} - t^{\text{replanning}}$ before any reassignment. These constraints ensure path and service continuity across the replanning boundary; see Chapter 6 for further implementation details.

6

Mathematical Formulation of the MILP Optimization Model

This chapter formalizes the MILP model that enables the IOS scheduling framework developed in this thesis. The formulation operates on the discrete, time-expanded network introduced in Chapter 3, where nodes represent orbital locations at discrete time steps and directed arcs represent feasible orbital transfers. It supports many-to-many servicing with multiple servicers, heterogeneous tools and consumables, multiple customers requiring distinct in-orbit services, resupply opportunities at orbital depots, and RH replanning under demand uncertainty. At each RH iteration, the MILP is instantiated over the current Planning Horizon using updated boundary conditions and demand forecasts, and its solution provides the executable Control Horizon decisions.

The MILP formulation builds on established optimization models for space logistics and in-orbit servicing that rely on time-expanded network representations and mixed-integer programming. In particular, the representation of spacecraft motion as unit flows over a discrete time-expanded network, together with explicit onboard capacity limits, resource balance constraints, and commodity flow transformation constraints such as propellant burn equations, follows the integrated space logistics modeling paradigm introduced in Chen and Ho (2018). This paradigm provides the foundational structure for coupling trajectory and sequencing decisions with mass and resource accounting.

Building on this foundation, the formulation incorporates modeling constructs that are specific to in-orbit servicing operations. These include explicit service assignment decisions, discretized service windows, fixed-duration service execution, and the coupling between service start decisions and enforced service activity over consecutive time steps. Together with the profit-maximizing objective function, these constructs are inspired by the MILP-based IOS logistics frameworks by Sarton du Jonchay et al. (2021, 2022). They ensure that servicers remain co-located with customer spacecraft throughout service execution and that services are performed consistently with their temporal availability constraints.

The formulation further aligns with the recent extensions of IOS logistics models to multi-orbit settings with moving customers and refueling opportunities by Sorenson and Nurre Pinkley (2023). These developments motivate the explicit representation of time-varying task presence along orbital trajectories and the inclusion of refueling-enabled arcs within the network, enabling depot-supported operations to be integrated directly into the routing and scheduling problem.

The extensions introduced to accommodate the Rolling Horizon framework, specifically the explicit carry-over of mid-arc transfers and mid-service executions, together with boundary-state injection mechanisms, are designed to ensure temporal continuity and feasibility across successive replanning iterations. These elements are therefore treated as methodological adaptations necessary for practical Rolling Horizon deployment, rather than as direct transcriptions of existing formulations.

The model interfaces with the rest of the framework as follows: The static network and its time expansion come from Chapter 3; arc attributes (time of flight and propellant cost) are populated by the orbital transfer module in Chapter 4; and the RH algorithm in Chapter 5 supplies boundary data at each replanning time t_0, i.e., the current system state (servicer start nodes, fuel levels, commodity inventories) and flags for any *in-transit* or *in-service* activities. To guarantee temporal continuity across

RH iterations, the MILP uses these flags to enforce the remainder of any active transfer and to resume ongoing services at t_0 (see Sections 6.1.2 and 6.3.5).

The MILP maximizes net mission profit over the planning horizon, i.e., service revenues minus launch; Purchase, Development and Manufacturing (PDM); operating; and delay costs. Solving it once yields an optimal short-term schedule; embedding it in the RH loop produces an adaptive long-horizon plan that updates as new information arrives.

The remainder of the chapter is organized as follows: It first specifies the sets and parameters, then explains how each problem instance is constructed at every RH iteration, including initial and boundary inputs. It then defines the decision variables for movement, service management, and onboard resources, develops the constraint blocks, and concludes with the profit-maximizing objective function and its components.

6.1. Sets and Parameters

$\mathcal{T}$: Set of discrete time steps in the planning window.

Δt: Time step length.

t_0: Current step (Rolling Horizon "now"); $t_0 = \min \mathcal{T}$.

t_F: Final step (end of the Planning Horizon); $t_F = \max \mathcal{T}$.

$\mathcal{N}$: Set of nodes; $i, j \in \mathcal{N}$.

$\mathcal{A}$: Set of directed *arcs (arc IDs)*; $a \in \mathcal{A}$. Multiple arcs may share the same endpoints (i, j) (e.g., different phasing options n).

$\mathcal{I}(a)$: Origin of arc a.

$\mathcal{J}(a)$: Destination of arc a.

$\mathcal{RS} \subseteq \mathcal{T} \times \mathcal{N} \times \mathcal{N}$: set of resupply triplets (t, i, j) (departure time, origin, destination), meaning any arc a with $\mathcal{I}(a) = i,\ \mathcal{J}(a) = j$ is resupply-enabled for a departure at t.

$\mathcal{D}$: Set of servicers; $d \in \mathcal{D}$.

$s_d \in \mathcal{N}$: Starting node of servicer $d \in \mathcal{D}$.

$\mathcal{B}$: Set of service tasks; $v \in \mathcal{B}$.

$\mathcal{B}_v$: Set of subtasks for task $v \in \mathcal{B}$; $k \in \mathcal{B}_v$.

$n_{vk} \in \mathcal{N}$: Node of subtask $k \in \mathcal{B}_v$ of task $v \in \mathcal{B}$.

$t_{vk} \in \mathcal{T}$: Time step of subtask $k \in \mathcal{B}_v$ of task $v \in \mathcal{B}$.

τ_a: Traversal time to traverse arc $a \in \mathcal{A}$.

ψ_a: Propellant fraction per time step to traverse arc $a \in \mathcal{A}$.

ϕ_a: Total propellant fraction to traverse arc $a \in \mathcal{A}$.

$\mathcal{W}_v \subset \mathcal{T}$: Feasible start times (representing the service window) for task $v \in \mathcal{B}$.

$\mathcal{M}$: Set of onboard commodities (e.g., spares, tools); $m \in \mathcal{M}$.

$\mathcal{K} \subset \mathcal{M}$: Subset of commodities that are *tools* carried by servicers required for operations.

F_d^{cap}: Maximum fuel tank capacity of servicer $d \in \mathcal{D}$.

Q_{dm}^{cap}: Maximum inventory capacity of commodity $m \in \mathcal{M}$ on servicer $d \in \mathcal{D}$.

$F_d^0 \in [0, F_d^{\text{cap}}]$: Initial fuel at t_0 on servicer $d \in \mathcal{D}$.

$Q_{dm}^0 \in [0, Q_{dm}^{\text{cap}}]$: Initial inventory at t_0 of commodity $m \in \mathcal{M}$ on servicer $d \in \mathcal{D}$.

RF_d: Number of time steps required to refuel servicer $d \in \mathcal{D}$ from empty to its fuel capacity F_d^{cap} at a depot.

$F_{d,a} = \frac{\tau_a}{RF_d} F_d^{\text{cap}}$: Maximum fuel that servicer $d \in \mathcal{D}$ can take while traversing a refueling arc a; applicable when $(t - \tau_a,\ \mathcal{I}(a),\ \mathcal{J}(a)) \in \mathcal{RS}$.

$\gamma_{v,\text{tool}} \in \{0, 1\}$: Service-tool mapping; $\gamma_{v,\text{tool}} = 1$ if task $v \in \mathcal{B}$ requires tool $tool \in \mathcal{K}$.

δ_{vm}: Amount of commodity $m \in \mathcal{M}$ consumed when completing task $v \in \mathcal{B}$; for tools $m \in \mathcal{K}$, $\delta_{vm} = 0$.

duration_v: Service duration of task $v \in \mathcal{B}$.

N_{dep}: Number of depots deployed.

mass_m: Mass per unit of commodity $m \in \mathcal{M}$.

m_d^{dry}: Dry mass of servicer $d \in \mathcal{D}$.

$m_{\text{dep}}^{\text{dry}}$: Dry mass of a depot.

revenue_v: Revenue for completing task $v \in \mathcal{B}$.

c_v^{delay}: Penalty rate per unit time applied when task $v \in \mathcal{B}$ is initiated later than its earliest admissible start time $\min \mathcal{W}_v$.

c_m^{purchase}: Purchase cost per unit of commodity $m \in \mathcal{M}$.

$c_{\text{fuel}}^{\text{purchase}}$: Purchase cost per unit mass of fuel.

$c_{\text{kg}}^{\text{launch}}$: Launch cost per unit mass.

c_d^{dm}: Development and manufacturing cost per servicer $d \in \mathcal{D}$.

$c_{\text{dep}}^{\text{dm}}$: Development and manufacturing cost per depot.

c_d^{op}: Operating cost per unit time per servicer $d \in \mathcal{D}$.

$c_{\text{dep}}^{\text{op}}$: Operating cost per unit time per depot.

$\bar{f}$: Minimum desirable terminal fuel reserve for a servicer at the end of the Planning Horizon.

λ_f: Fuel safeguard penalty weight applied per unit of terminal fuel shortfall.

M: A sufficiently large positive constant used for big-M linearizations.

The constant M is chosen as a valid upper bound on the corresponding physical quantities involved in each linearization (e.g., total servicer mass, fuel tank capacity, or maximum inventory levels). This ensures correctness of the logical implications enforced by the big-M constraints while avoiding unnecessary relaxation that could degrade numerical performance.

6.1.1. Instance Generation and RH Data

This subsection explains how the sets and parameters used by the MILP are generated at each Rolling Horizon iteration.

Network and arc attributes: The node and arc sets $(\mathcal{N}, \mathcal{A})$ are taken from the static logistics network built in Chapter 3. Arc attributes are populated by the orbital transfer module (Chapter 4), which provides the traversal time τ_a, the per time step propellant fraction ψ_a, and the total arc propellant fraction ϕ_a for each $a \in \mathcal{A}$.

Time discretization and planning window: The start time of each Rolling Horizon iteration, and thus the start of each planning window $t_0 = \min \mathcal{T}$, is supplied by the Rolling Horizon module, consistent with the replanning time policy in Chapter 5. The planning horizon length $|\mathcal{T}|$ and the time step size Δt are tunable design parameters; the values adopted in this thesis are selected and justified in Chapter 8.

Demand forecast and tasks: At each iteration, the candidate task set $\mathcal{B}$ is formed from the demand forecast (pre-computed in `task_generator.py`; see Subchapter 5.3.1) restricted to the current Planning Horizon $\mathcal{T}$, following the logic in Chapter 5: it includes (i) any newly realized random task that triggered replanning (when applicable) and (ii) all previously known tasks, deterministic or random, whose service windows intersect $\mathcal{T}$ and remain unserved. Task types and their economic/operational parameters are specified in `tasks.yaml`, aligned with Table 2.1: revenue (revenue_v), delay penalty cost (c_v^{delay}), service duration (duration_v), and commodity requirements (δ_{vm}). Additionally, the service window, i.e., the interval within which a service must start after its activation, is discretized into a feasible start-time set $\mathcal{W}_v \subset \mathcal{T}$. From the commodity requirements δ_{vm}, the consumables $\mathcal{C} = \mathcal{M} \setminus \mathcal{K}$ are defined; together with the tool set $\mathcal{K}$, these form the overall commodity set, $\mathcal{M} = \mathcal{K} \cup \mathcal{C}$ with $\mathcal{K} \cap \mathcal{C} = \varnothing$. Because tools are non-consumable, $\delta_{v,tool} = 0$ for all $v \in \mathcal{B}$ and $tool \in \mathcal{K}$.

Subtasks and presence along the orbit: For each task $v \in \mathcal{B}$, the subtask index set $\mathcal{B}_v$ and the associated node/time pairs $\{(n_{vk}, t_{vk})\}_{k \in \mathcal{B}_v}$ are precomputed via Algorithm 2. In each iteration, for each considered task, only subtasks whose times lie within the current planning window are included, i.e., $t_{vk} \in \mathcal{T}$.

Resupply opportunities: Resupply triplets $\mathcal{RS}$ indicate where resupply is available at each time t. This set is precomputed by Algorithm 3. For each servicer d and resupply arc a with $(t-\tau_a,\, \mathcal{I}(a),\, \mathcal{J}(a)) \in \mathcal{RS}$, the per-traversal refueling capacity is:

$$F_{d,a} \;=\; \frac{\tau_a}{RF_d}\, F_d^{\text{cap}}, \qquad \forall (t-\tau_a,\, \mathcal{I}(a),\, \mathcal{J}(a)) \in \mathcal{RS},\; d \in \mathcal{D}. \tag{6.1}$$

Servicers and tools: Servicer modeling is detailed in Chapter 2.2.1. The tools a servicer can carry (T1–T4) define the set $\mathcal{K} \subset \mathcal{M}$. For each task $v \in \mathcal{B}$, the required tools identified in that chapter yield the binary requirement parameters $\gamma_{v,\text{tool}} \in \{0,1\}$, where $\gamma_{v,\text{tool}} = 1$ if task v requires $tool \in \mathcal{K}$. The parameters in Table 2.2 enter the MILP as follows.

For each servicer $d \in \mathcal{D}$:

- *Tool parameters:* initialize tool inventories and capacities, $Q^0_{d,tool}$ and $Q^{\text{cap}}_{d,tool}$ for $tool \in \mathcal{K}$.
- *Orbital-transfer parameters:* fuel-tank capacity F_d^{cap} and dry mass m_d^{dry} enter the mass and fuel-balance equations.
- *Cost parameters:* development and manufacturing cost c_d^{dm}, and operating cost c_d^{op} appear in the objective.
- *Starting node:* discrete initial position s_d.
- *Refueling rate:* parameter RF_d governing depot refueling dynamics.
- *Payload capacity:* capacities for non-tool commodities $m \in \mathcal{M} \setminus \mathcal{K}$ define Q^{cap}_{dm} (e.g., spare parts, de-orbit modules, customer propellant).

Additional parameters: Additional parameters are introduced for the objective function and for a uniform depot model. For simplicity, all depots share common values (this assumption can be relaxed in future work).

Depots:

- Dry mass: $m^{\text{dry}}_{\text{dep}}$.
- Development & manufacturing cost: $c^{\text{dm}}_{\text{dep}}$.
- Operating cost: $c^{\text{op}}_{\text{dep}}$.

Consumables:

- Servicer propellant purchase price (per unit mass): $c^{\text{purchase}}_{\text{fuel}}$.
- Service-commodity purchase price for $m \in \mathcal{M}$: c^{purchase}_m.
  - *Continuous commodities* (e.g., customer propellant): c^{purchase}_m is per unit mass. Set $\text{mass}_m = 1$ and inventory is tracked in mass.
  - *Discrete commodities* (e.g., spare parts, de-orbit modules): c^{purchase}_m is per unit item. Let mass_m be the mass per item and inventory is tracked in items.

Launch:

- Launch cost per unit mass: $c^{\text{launch}}_{\text{kg}}$.

6.1.2. RH Boundary State and Flags

At every iteration after the first, the replanning time t_0 (end of the previous Control Horizon and start of the current planning window) comes with updated boundary data provided by the Rolling Horizon module. These data override the default initial conditions for the MILP and are passed via the propagated `infrastructure_state` (see Subsection 5.3.3). For each servicer $d \in \mathcal{D}$ the RH module supplies:

- **Start location:** the node s_d at time t_0;
- **Propellant:** the initial fuel $F_d^0 \in [0, F_d^{\text{cap}}]$;
- **Inventories:** the initial onboard quantities $Q_{dm}^0 \in [0, Q_{dm}^{\text{cap}}]$ for all $m \in \mathcal{M}$;
- **(Optional) mid-arc flag:** $\texttt{in\_transit} = \{\mathcal{I}(a), \mathcal{J}(a), t_{\text{start}}, t_{\text{arrival}}, \psi_a\}$ indicating that $d \in \mathcal{D}$ is in transit on arc a at t_0;
- **(Optional) mid-service flag:** $\texttt{in\_service} = \{v, t_{\text{start}}, t_{\text{end}}\}$ indicating that $d \in \mathcal{D}$ is executing task $v \in \mathcal{B}$ at t_0 and has a remaining service time $t_{\text{end}} - t_0 > 0$;

These data are treated as *parameters* of the current MILP. Specifically, s_d, F_d^0, and Q_{dm}^0 overwrite the first-iteration defaults from the inputs, while the optional flags supply the minimal information required to enforce the continuation of any in-transit arc or in-progress service at t_0.

When a mid-arc flag is present for servicer $d \in \mathcal{D}$, the residual travel time $r_d = t_{\text{arrival}} - t_0 > 0$ is defined. A dummy start node $X_d \notin \mathcal{N}$ and a synthetic *remainder arc* with a new arc index $a_{\text{rem}} = \max\{a : a \in \mathcal{A}\} + 1$ are introduced such that $\mathcal{I}(a_{\text{rem}}) = X_d$, $\mathcal{J}(a_{\text{rem}}) = \mathcal{J}(a)$, $\tau_{a_{\text{rem}}} = r_d$, and $\psi_{a_{\text{rem}}} = \psi_a$. The start location is set to the dummy node, i.e., $s_d = X_d$. For subsequent constraint enforcement, the triple (d, a_{rem}, t_0) is recorded in the list `arc_fix`.

If a mid-service flag is present for $d \in \mathcal{D}$ on task $v \in \mathcal{B}$, the residual service time $r_v = t_{\text{end}} - t_0 > 0$ is computed and the task data is updated as $\text{duration}_v = r_v$ and $\mathcal{W}_v = \{t_0\}$. Let i_{hold} be the node where d is executing v at t_0. If there exists a subtask $k \in \mathcal{B}_v$ with $(n_{vk}, t_{vk}) = (i_{\text{hold}}, t_0)$, then (d, v, k) is added to the *arrival-credit* set $\mathcal{C}^{arr} \subseteq \{(d, v, k) : d \in \mathcal{D},\, v \in \mathcal{B},\, k \in \mathcal{B}_v\}$. For all $(d, v, k) \in \mathcal{C}^{arr}$, the parameter $\xi_{dvk} = 1$ is set in (6.12), thereby permitting activation at t_0 without an explicit arrival arc. For subsequent constraint enforcement, the pair (v, d) is recorded in the list `service_fix`.

The mathematical enforcement of these boundary conditions, i.e., the forced continuation of mid-arcs and mid-services, is presented in Section 6.3.5.

6.2. Decision Variables

The following decision variables are defined over indices $d \in \mathcal{D}$, $v \in \mathcal{B}$, $k \in \mathcal{B}_v$, $i \in \mathcal{N}$, $a \in \mathcal{A}$, $m \in \mathcal{M}$, $t \in \mathcal{T}$, and $\tau \in \mathcal{W}_v$:

Movement and subtask completion

$y_{d,a,t} \in \{0,1\}$	1 if servicer d *starts* moving on arc a at time t; 0 otherwise,
$\beta_{dvk} \in \{0,1\}$	1 if subtask k of task v is *completed* by servicer d; 0 otherwise.

Terminal occupancy at horizon end

$x_{di}^{\text{end}} \in \{0,1\}$	1 if servicer d is located at node i at t_F; 0 otherwise.

Service assignment and ongoing activity

$h_{vd\tau} \in \{0,1\}$	1 if task v is *started* by servicer d at time τ; 0 otherwise,
$b_{vdt} \in \{0,1\}$	1 if task v is *being served* by d at time t; 0 otherwise.

Onboard resources

$f_{dt} \geq 0$	fuel onboard servicer d at time t,
$q_{dmt} \geq 0$	inventory of commodity m on servicer d at time t,
$m_{dt}^{\text{tot}} \geq 0$	total mass of servicer d at time t (dry mass + fuel + commodities).

Terminal fuel safeguard

$$s_d^{\text{low}} \;\geq\; 0 \quad \text{terminal fuel reserve slack variable for servicer } d.$$

Fuel burn linearization

$z_{d,a,t} \geq 0$ pre-burn mass used in the fuel burn term when servicer d is in flight on arc a during $(t-1,t]$.

Meaning: The variable $z_{d,a,t}$ is the linearization proxy for the bilinear product $m^{\text{tot}}_{d,t-1} \times \alpha_{d,a,t}$, where the in-flight indicator

$$\alpha_{d,a,t} \;=\; \sum_{s=\max\{t_0,\, t-\tau_a\}}^{t-1} y_{d,a,s} \in \{0,1\}$$

equals 1 iff servicer d is traversing arc a during $(t-1,t]$. Thus, $z_{d,a,t} = m^{\text{tot}}_{d,t-1}$ when the transfer on a is active in $(t-1,t]$, and $z_{d,a,t} = 0$ otherwise. This construction allows the per-step propellant burn to be written linearly as

$$\text{burn}_{dt} \;=\; \sum_{a\in\mathcal{A}} \psi_a \, z_{d,a,t}, \qquad \forall d \in \mathcal{D},\; t \in \mathcal{T} \setminus \{t_0\}.$$

Refueling on resupply arcs

$r_{d,a,t} \geq 0$	fuel mass refueled by servicer d on arc a and *credited* at time t,
$\kappa_{d,a,t} \geq 0$	free fuel tank capacity on servicer d at time t (pre-refuel),
$r^{\max}_{d,a,t} \geq 0$	effective refuel capacity upon arrival on arc a at time t.

Note: The above resupplying variables are defined only for resupply (coasting) arcs. $r^{\max}_{dat} = \min\{F_{d,a},\, \kappa_{d,a,t}\}$, where $F_{d,a}$ is the maximum fuel that servicer d can take on arc a.

6.3. Constraints

6.3.1. Initial Conditions

Fuel and inventories are fixed at t_0, and the total mass is initialized accordingly:

$$f_{dt_0} = F_d^0, \qquad \forall d \in \mathcal{D}, \tag{6.2}$$

$$q_{dmt_0} = Q_{dm}^0, \qquad \forall d \in \mathcal{D},\; m \in \mathcal{M}, \tag{6.3}$$

$$m^{\text{tot}}_{dt_0} = m_d^{\text{dry}} + f_{dt_0} + \sum_{m\in\mathcal{M}} q_{dmt_0}\, \text{mass}_m, \qquad \forall d \in \mathcal{D}. \tag{6.4}$$

Together, (6.2) - (6.4) ensure state continuity across Rolling Horizon solves and guarantee that the initial total mass, comprising dry mass, propellant, and payload is physically consistent.

6.3.2. Flow Conservation and Horizon Closure

Let $t_0 = \min \mathcal{T}$ and $t_F = \max \mathcal{T}$. For each $d \in \mathcal{D}$, node $i \in \mathcal{N}$, and time $t \in \mathcal{T}$, define the *inflow* to and *outflow* from node i as:

$$\text{In}_{dit} = \sum_{\substack{a\in\mathcal{A}:\; \mathcal{J}(a)=i \\ t-\tau_a \geq t_0}} y_{d,a,\,t-\tau_a}, \qquad \text{Out}_{dit} = \sum_{a\in\mathcal{A}:\; \mathcal{I}(a)=i} y_{d,a,t}.$$

Flow is conserved at every node-time pair except at the unique source and at the terminal time:

$$\text{In}_{dit} - \text{Out}_{dit} \;=\; \begin{cases} -1, & \text{if } i = s_d \text{ and } t = t_0, \\ x^{\text{end}}_{di}, & \text{if } t = t_F, \\ 0, & \text{otherwise}, \end{cases} \qquad \forall d \in \mathcal{D},\; i \in \mathcal{N},\; t \in \mathcal{T}. \tag{6.5}$$

Equation (6.5) constructs, for each servicer, a unit path in the time-expanded network: one unit departs s_d at t_0. At the final step t_F, the terminal-balance term x^{end}_{di} assigns that unit to exactly one node, thereby closing the path within $[t_0, t_F]$ (see also (6.7)). If s_d is a dummy node X_d due to a mid-arc continuation, the forced remainder arc at t_0 (Section 6.3.5) keeps (6.5) satisfied.

6.3.3. Horizon-Respecting Departures

Departures that would arrive after the end of the Planning Horizon are forbidden. This ensures that arrivals, capacity checks, refueling credits, and costs are all enforced within the modeled window, avoiding over-the-horizon spillover (decisions whose consequences occur after t_F and would otherwise be unpriced or unconstrained in the current solve).

$$y_{d,a,t} \;=\; 0, \quad \forall\, d \in \mathcal{D},\; a \in \mathcal{A},\; t \in \mathcal{T} \text{ s.t. } t + \tau_a > t_F. \tag{6.6}$$

6.3.4. Unique Terminal Location

At the end of the Planning Horizon, each servicer must occupy exactly one node. This closes the unit path within $[t_0, t_F]$ and enforces flow feasibility at the boundary.

$$\sum_{i \in \mathcal{N}} x_{di}^{\text{end}} \;=\; 1, \qquad \forall\, d \in \mathcal{D}. \tag{6.7}$$

This condition also applies in the final Planning Horizon, meaning that at the end of the finite Scheduling Horizon, each servicer may terminate at any node in the network. While it could be assumed that servicers return to an orbital depot thereafter, this is not enforced in the model, as such a requirement would not contribute to the servicing objective.

6.3.5. Boundary Continuation Constraints

Forced mid-arc continuation:

Let arc_fix denote the list of mid-arc continuation triples (d, a_{rem}, t_0) assembled at the replanning boundary t_0. Here a_{rem} is a synthetic remainder arc index created so that $I(a_{\text{rem}}) = X_d$, $J(a_{\text{rem}}) = j$, $\tau_{a_{\text{rem}}} = r_d = t_{\text{arrival}} - t_0$, and $\psi_{a_{\text{rem}}}$ equals the carried-over per step propellant fraction of the interrupted arc (see Section 6.1.2). For every $(d, a_{\text{rem}}, t_0) \in$ `arc_fix`, the remainder arc must be taken at the replanning boundary:

$$y_{d,a_{\text{rem}},t_0} = 1. \tag{6.8}$$

Because $s_d = X_d$ and the remainder arc has $\tau_{a_{\text{rem}}} = r_d > 0$ and $\psi_{a_{rem}} = \psi_a$, the start-node flow balance at t_0 enforces exactly one departure by d from X_d; (6.8) selects that unique departure to be the required remainder arc with the correct residual time and fuel usage.

Forced mid-service continuation:

Let $service\_fix$ be the list of mid-service continuation pairs (v, d) recorded at t_0 (each element stores the ongoing task and its assigned servicer; see Section 6.1.2 for mid-service flags explanation). For each $(v, d) \in$ `service_fix`, the shortened service must resume immediately at t_0:

$$h_{vdt_0} = 1. \tag{6.9}$$

This constraint works simultaneously with the updates defined in Section 6.1.2, namely $\text{duration}_v = r_v = t_{\text{end}} - t_0$, the restricted start window $\mathcal{W}_v = \{t_0\}$ and the defined 'arrival credit'.

6.3.6. Unique Assignment per Task

This constraint enforces that a task $v \in \mathcal{B}$ may be started at most once across all servicers and all admissible start times $\tau \in \mathcal{W}_v$. Formally,

$$\sum_{d \in \mathcal{D}} \sum_{\tau \in \mathcal{W}_v} h_{vd\tau} \;\le\; 1, \qquad \forall v \in \mathcal{B}. \tag{6.10}$$

Modeling note: if some tasks are *mandatory*, one can tighten (6.10) to equality $\sum_{d,\tau} h_{vd\tau} = 1$ for those v.

6.3.7. Linking Assignment to Time-Activity

Let $h_{vd\tau} \in \{0, 1\}$ indicate that task $v \in \mathcal{B}$ is assigned to servicer $d \in \mathcal{D}$ with start time $\tau \in \mathcal{W}_v$, and let $b_{vdt} \in \{0, 1\}$ indicate that v is *actively* being served by d at time $t \in \mathcal{T}$. The binary indicator

$$\text{act}_{v\tau t} \;=\; \begin{cases} 1, & \text{iff } \tau \le t < \tau + \text{duration}_v, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall\, v \in \mathcal{B},\; \tau \in \mathcal{W}_v,\; t \in \mathcal{T}.$$

, where duration_v is the (known) service duration of task v needs to be pre-computed. Thus $\text{act}_{v\tau t}$ simply marks the duration_v consecutive time steps during which a start at τ would keep the task active.

The activity variable b_{vdt} is then the (assignment $\times$ activity-profile) convolution:

$$b_{vdt} \;=\; \sum_{\tau \in \mathcal{W}_v} \text{act}_{v\tau t}\, h_{vd\tau}, \qquad \forall v \in \mathcal{B},\; d \in \mathcal{D},\; t \in \mathcal{T}. \tag{6.11}$$

The equality in (6.11) is valid (and numerically tight) because each task is assigned at most once, i.e., $\sum_{d\in\mathcal{D}} \sum_{\tau\in\mathcal{W}_v} h_{vd\tau} \le 1$, so the right-hand side cannot exceed 1.

6.3.8. Arrival Requirement for Subtask Completion

Each subtask $k \in \mathcal{B}_v$ is defined by a node-time pair (n_{vk}, t_{vk}). Completion of k by servicer $d \in \mathcal{D}$ is only permissible if d *reaches* node n_{vk} exactly at time t_{vk}. Let

$$A_{dvk} \;=\; \sum_{\substack{a\in\mathcal{A}:\; \mathcal{J}(a)=n_{vk} \\ t_{vk}-\tau_a \ge t_0}} y_{d,\,a,\,t_{vk}-\tau_a}$$

, count such arrivals (i.e., departures that, after τ_a steps, arrive at (n_{vk}, t_{vk}) within the current planning window). To cover boundary cases where the servicer is already at (n_{vk}, t_{vk}) due to the RH state injection when there is a service ongoing at t_0, a binary parameter

$$\xi_{dvk} \in \{0,1\}, \qquad \xi_{dvk} \;=\; 1 \text{ iff } (d,v,k) \in \mathcal{C}^{arr}$$

, where $\mathcal{C}^{arr}$ is the *arrival-credit* set provided by the RH module (Section 6.1.2) is introduced.

The subtask completion variable β_{dvk} must then satisfy:

$$\beta_{dvk} \;\le\; A_{dvk} + \xi_{dvk}, \qquad \forall d \in \mathcal{D},\; v \in \mathcal{B},\; k \in \mathcal{B}_v. \tag{6.12}$$

Constraint (6.12) prevents "teleportation": without credit ($\xi_{dvk} = 0$), a subtask can only be activated if there exists at least one incoming movement decision that *arrives* at the correct node and time. With credit ($\xi_{dvk} = 1$), completion is also allowed when the servicer is already present at (n_{vk}, t_{vk}) at the boundary t_0, eliminating the need to simulate an artificial arrival arc.

6.3.9. Linking Assignment to Subtask Completion

If servicer $d \in \mathcal{D}$ starts task $v \in \mathcal{B}$ at time $\tau \in \mathcal{W}_v$ ($h_{vd\tau} = 1$), then every subtask $k \in \mathcal{B}_v$ whose scheduled time t_{vk} falls within that service window $[\tau,\, \tau + \text{duration}_v]$ must be completed by the same servicer ($\beta_{dvk} = 1$). Formally,

$$h_{vd\tau} \;\le\; \beta_{dvk}, \qquad \forall\, v \in \mathcal{B},\; d \in \mathcal{D},\; \tau \in \mathcal{W}_v,\; k \in \mathcal{B}_v \text{ s.t. } t_{vk} \in [\tau,\, \tau + \text{duration}_v]. \tag{6.13}$$

, where duration_v is the service duration of task v. This ties the start decision to executing all associated subtasks; together with movement/availability constraints that ensure reachability, it prevents assigning a service when the servicer cannot be present to perform those subtasks at their scheduled times.

6.3.10. Single-Service Capacity per Node and Time

At most one service operation can be active at a given node and time, meaning a customer satellite can receive *at most one* service per time step. This prevents overlapping operations, whether it's two servicers working on the same customer simultaneously, or a single servicer attempting two services at once on that customer. Formally:

$$\sum_{v\in\mathcal{B}} \sum_{d\in\mathcal{D}} \sum_{\substack{k\in\mathcal{B}_v: \\ n_{vk}=i,\; t_{vk}=t}} b_{vdt} \;\le\; 1, \qquad \forall\, i \in \mathcal{N},\; t \in \mathcal{T}. \tag{6.14}$$

Here, $b_{vdt} = 1$ flags that task $v \in \mathcal{B}$ is active on servicer $d \in \mathcal{D}$ at time $t \in \mathcal{T}$; summing over all tasks and servicers whose subtask (v,k) occurs at node i and time t imposes a unit capacity at that node-time, ensuring exclusivity without possibility of double booking.

6.3.11. Node-Time Exclusivity (Operational Collision Avoidance)

To avoid collisions, at most one servicer may *arrive* at a node at any given time. The left-hand side counts all arrivals to node $i \in \mathcal{N}$ at time $t \in \mathcal{T}$; the right-hand side sets a unit capacity:

$$\sum_{d \in \mathcal{D}} \sum_{\substack{a \in \mathcal{A}:\ \mathcal{J}(a)=i \\ t-\tau_a \geq t_0}} y_{d,a,\,t-\tau_a} \ \leq\ 1, \qquad \forall\, i \in \mathcal{N},\ t \in \mathcal{T}. \tag{6.15}$$

(Here, $y_{d,a,\,t-\tau_a} = 1$ indicates a departure at $t - \tau_a$ that arrives at node i at time t.)

Constraint (6.15) complements the single-service capacity constraint (6.14): while (6.14) forbids concurrent *service activity* at a node-time, (6.15) forbids concurrent *arrivals*, preventing two servicers from reaching the same node at the same instant even if neither is serving.

6.3.12. Commodity Capacity Bounds

Each onboard commodity is confined to its feasible storage range:

$$0 \ \leq\ q_{dmt} \ \leq\ Q^{\text{cap}}_{dm}, \qquad \forall\, d \in \mathcal{D},\ m \in \mathcal{M},\ t \in \mathcal{T}. \tag{6.16}$$

The lower bound forbids negative inventories, while the upper bound enforces the servicer- and commodity-specific onboard capacity limit (e.g., tool slots; payload bay for spare parts and de-orbit modules; tank capacity for client propellant). This keeps transfers within physical limits and ensures consistency with mass accounting, since m^{tot}_{dt} depends on q_{dmt}.

6.3.13. Commodity Balance with Resupply Logic

At each step $t \in \mathcal{T} \setminus \{t_0\}$, the onboard amount of commodity $m \in \mathcal{M}$ for servicer $d \in \mathcal{D}$ either (i) resets all commodities to full capacity if a resupply arrival occurs at t, or (ii) decreases by the amount consumed by tasks that *finish* at t.

End-of-service consumption: Let $\delta_{vm} \geq 0$ be the units of commodity $m \in \mathcal{M}$ required by task $v \in \mathcal{B}$, duration_v its service duration, and $h_{vd\tau} \in \{0,1\}$ the assignment indicator at time $\tau \in \mathcal{W}_v$. Consumption is deducted at completion and defined as:

$$c_{dmt} \ =\ \sum_{v \in \mathcal{B}} \sum_{\tau \in \mathcal{W}_v:\, \tau + \text{duration}_v = t} \delta_{vm}\, h_{vd\tau}, \qquad \forall d \in \mathcal{D},\ m \in \mathcal{M},\ t \in \mathcal{T} \setminus \{t_0\}. \tag{6.17}$$

Resupply indicator: Let τ_a denote the flight time on arc $a \in \mathcal{A}$, and let $\mathcal{RS}$ be the set of *departure–stamped* resupply triplets (t, i, j), i.e., times t at which a resupply-eligible (coasting) leg $i \to j$ may depart.

Define:

$$\rho_{dt} \ =\ \sum_{\substack{a \in \mathcal{A} \\ (t-\tau_a,\, \mathcal{I}(a),\, \mathcal{J}(a)) \in \mathcal{RS} \\ t-\tau_a \geq t_0}} y_{d,a,\,t-\tau_a}, \qquad \forall d \in \mathcal{D},\ t \in \mathcal{T} \setminus \{t_0\}. \tag{6.18}$$

This equals 1 iff servicer $d \in \mathcal{D}$ departs at $t - \tau_a$ on a resupply arc and thus *arrives* at time t; otherwise it is 0. Hence $\rho_{dt} \in \{0,1\}$ acts as the resupply indicator at time t.

Case distinction via big-M: Two pairs of bounds are added.

Resupply pair (resets to maximum capacity when $\rho_{dt} = 1$):

$$q_{dmt} \ \leq\ Q^{\text{cap}}_{dm} + (1 - \rho_{dt})M, \tag{6.19}$$

$$q_{dmt} \ \geq\ Q^{\text{cap}}_{dm} - (1 - \rho_{dt})M, \tag{6.20}$$

$$\forall d \in \mathcal{D},\ m \in \mathcal{M},\ t \in \mathcal{T} \setminus \{t_0\}.$$

If $\rho_{dt} = 1$, the M-terms vanish and we get $q_{dmt} = Q^{\text{cap}}_{dm}$. If $\rho_{dt} = 0$, these bounds relax to $q_{dmt} \in [Q^{\text{cap}}_{dm} - M,\ Q^{\text{cap}}_{dm} + M]$ and become nonbinding (with a large M).

No-resupply pair ("prior level minus consumption" when $\rho_{dt} = 0$):

$$q_{dmt} \;\le\; q_{dm,t-1} - \mathsf{c}_{dmt} + \rho_{dt} M, \tag{6.21}$$

$$q_{dmt} \;\ge\; q_{dm,t-1} - \mathsf{c}_{dmt} - \rho_{dt} M, \tag{6.22}$$

$$\forall d \in \mathcal{D},\ m \in \mathcal{M},\ t \in \mathcal{T} \setminus \{t_0\}.$$

If $\rho_{dt} = 0$, the M-terms vanish and we get $q_{dmt} = q_{dm,t-1} - \mathsf{c}_{dmt}$. If $\rho_{dt} = 1$, these bounds relax to a wide interval and do not bind.

Thus, when a resupply arrival occurs ($\rho_{dt} = 1$), q_{dmt} is *forced* to capacity; otherwise ($\rho_{dt} = 0$) it is *forced* to the prior level minus the completed task consumption. Choosing M larger than any feasible deviation ensures the inactive case is truly nonbinding.

6.3.14. Tool Feasibility

A servicer may only execute tasks for which it carries the required tools:

$$q_{d,tool,t} \;\ge\; \sum_{v \in \mathcal{B}} \gamma_{v,\mathsf{tool}}\, b_{vdt}, \qquad \forall\, d \in \mathcal{D},\ tool \in \mathcal{K},\ t \in \mathcal{T}. \tag{6.23}$$

Here, $q_{d,tool,t}$ is the onboard quantity of $tool \in \mathcal{K}$, $b_{vdt} = 1$ flags that task $v \in \mathcal{B}$ is being served by $d \in \mathcal{D}$ at time $t \in \mathcal{T}$, and $\gamma_{v,\mathsf{tool}}$ is the task-tool requirement indicator ($\gamma_{v,\mathsf{tool}} = 1$ iff task v requires tool k). The constraint aggregates the tool demand of the task currently active on d and enforces that the available tools on board cover that demand.

6.3.15. Fuel Capacity Bounds

The fuel level is restricted to the physically feasible range:

$$0 \;\le\; f_{dt} \;\le\; F_d^{\mathsf{cap}}, \qquad \forall d \in \mathcal{D},\ t \in \mathcal{T}. \tag{6.24}$$

The lower bound enforces non-negativity, while the upper bound prevents overfilling and keeps refueling within the fuel tank capacity.

6.3.16. Fuel Balance with Refueling Logic

For each servicer $d \in \mathcal{D}$ and time step $t \in \mathcal{T} \setminus \{t_0\}$, the onboard propellant follows an inventory balance: propellant burned on transfer arcs between $(t-1, t]$ is subtracted, and any refueling at designated depots that completes at time t is added. Refueling is credited only upon completion. Propellant burn scales with the *pre-burn* mass because maneuver cost depends on the servicer's mass at the start of the step. In Chapter 4, ψ_a for each arc $a \in \mathcal{A}$ is defined as the per time step propellant fraction obtained from the rocket equation,

$$\psi_a \;=\; 1 - \mathsf{exp}\Big(- \frac{\Delta V_{a,step}}{g_0 I_{sp}} \Big),$$

where $\Delta V_{a,step}$ is the velocity increment assigned to one timestep of arc a. Accordingly, the propellant burned during each interval $(t-1, t]$ equals ψ_a multiplied by the servicer's mass in flight at that timestep.

Let $y_{d,a,s} \in \{0,1\}$ be the departure indicator of servicer d on arc a at time s, and let τ_a be the arc flight time in steps. Define the *in-flight indicator*

$$\alpha_{d,a,t} \;=\; \sum_{s=\mathsf{max}\{t_0,\, t-\tau_a\}}^{t-1} y_{d,a,s} \;\in\; \{0,1\},$$

which equals 1 iff the servicer is traversing a during $(t-1, t]$. Because directly multiplying the pre-burn mass $m^{\mathsf{tot}}_{d,t-1}$ by $\alpha_{d,a,t}$ would be bilinear, an auxiliary variable $z_{d,a,t}$ is introduced to represent that product and enforce a big-M linearization:

$$\begin{aligned}
0 \;\le\; z_{d,a,t} \;&\le\; m^{\mathsf{tot}}_{d,t-1}, && \forall d \in \mathcal{D},\ a \in \mathcal{A},\ t \in \mathcal{T} \setminus \{t_0\},\\
z_{d,a,t} \;&\le\; M\, \alpha_{d,a,t}, && \forall d \in \mathcal{D},\ a \in \mathcal{A},\ t \in \mathcal{T} \setminus \{t_0\},\\
z_{d,a,t} \;\ge\; m^{\mathsf{tot}}_{d,t-1} &- M\,(1 - \alpha_{d,a,t}), && \forall d \in \mathcal{D},\ a \in \mathcal{A},\ t \in \mathcal{T} \setminus \{t_0\},
\end{aligned}$$

With M chosen as a valid upper bound on $m^{\text{tot}}_{d,t-1}$, these constraints yield $z_{d,a,t} = m^{\text{tot}}_{d,t-1}$ whenever the servicer is in flight on a during $(t-1,t]$, and $z_{d,a,t} = 0$ otherwise. The per step propellant burn then becomes:

$$\text{burn}_{dt} \;=\; \sum_{a\in\mathcal{A}} \psi_a \, z_{d,a,t}, \qquad \forall d \in \mathcal{D},\; t \in \mathcal{T} \setminus \{t_0\}.$$

Refueling: At time t, refueling is credited only on those transfers $a \in \mathcal{A}$ that *arrive* at t and are eligible for propellant transfer. Eligibility holds iff: (i) there was a departure at $t-\tau_a$ that yields an arrival at t; (ii) the arc is flagged as refuel-enabled for that departure instant, i.e., $(t-\tau_a, \mathcal{I}(a), \mathcal{J}(a)) \in \mathcal{RS}$; and (iii) an arc-servicer transfer bound exists, $F_{d,a}$, specifying the maximum propellant that servicer d may take while traversing a.

Tank capacity after burn: Because propellant burn over $(t-1,t]$ is considered to be deducted before any refuel is credited, the free tank capacity at t is:

$$\kappa_{d,a,t} \;=\; F^{\text{cap}}_{d} \;-\; f_{d,t-1} \;+\; \text{burn}_{dt}, \qquad \forall d \in \mathcal{D},\; a \in \mathcal{A},\; t \in \mathcal{T} \setminus \{t_0\},$$

, where F^{cap}_{d} is the total tank capacity. In this formulation, refueling is permitted only on depot/coast arcs where no thrust is applied during $(t-1,t]$; hence $\text{burn}_{dt} = 0$ whenever refueling is credited, and the tank capacity at such arrivals simplifies to:

$$\kappa_{d,a,t} \;=\; F^{\text{cap}}_{d} \;-\; f_{d,t-1}.$$

Refuel-capacity bound (min of arc limit and tank capacity): The amount of fuel that can be transferred upon arrival on a is limited by both the arc-servicer transfer bound ($F_{d,a}$) and the available tank capacity:

$$r^{\text{max}}_{d,a,t} \;\le\; F_{d,a}, \qquad r^{\text{max}}_{d,a,t} \;\le\; \kappa_{d,a,t}, \qquad \forall d \in \mathcal{D},\; a \in \mathcal{A},\; t \in \mathcal{T} \setminus \{t_0\}.$$

(Operationally this is enforced with a $\min\{\cdot,\cdot\}$ constraint).

Activation by the flown arc: Let $y_{d,a,\,t-\tau_a} \in \{0,1\}$ indicate that d departed on a at $t-\tau_a$ and thus arrives at t. The credited refuel $r_{d,a,t}$ equals the bound $r^{\text{max}}_{d,a,t}$ *iff* that arrival occurs; otherwise it must be zero. A standard big-M activation enforces

$$r_{d,a,t} \ge r^{\text{max}}_{d,a,t} - M\big(1 - y_{d,a,\,t-\tau_a}\big), \qquad r_{d,a,t} \le r^{\text{max}}_{d,a,t} + M\big(1 - y_{d,a,\,t-\tau_a}\big), \tag{6.25}$$

$$0 \;\le\; r_{d,a,t} \le M\, y_{d,a,\,t-\tau_a}, \qquad \forall d \in \mathcal{D},\; a \in \mathcal{A},\; t \in \mathcal{T} \setminus \{t_0\}. \tag{6.26}$$

Thus, if the arc was flown ($y_{d,a,\,t-\tau_a} = 1$), (6.25) forces $r_{d,a,t} = r^{\text{max}}_{d,a,t}$; if not, (6.26) gives $r_{d,a,t} = 0$.

Fuel stock update: Summing the refueling credit over all arcs, the fuel state evolves as:

$$f_{dt} \;=\; f_{d,t-1} \;-\; \text{burn}_{dt} \;+\; \sum_{a\in\mathcal{A}} r_{d,a,t}, \qquad \forall\, d \in \mathcal{D},\; t \in \mathcal{T} \setminus \{t_0\} \tag{6.27}$$

, considering both fuel burn and refueling.

6.3.17. Mass Accounting

For $t > t_0$, the total mass for each servicer $d \in \mathcal{D}$ is the sum of its dry structure, onboard fuel, and the mass of all carried commodities (tools and consumables) at time $t \in \mathcal{T}$:

$$m^{\text{tot}}_{dt} \;=\; \underbrace{m^{\text{dry}}_{d}}_{\text{servicer dry mass}} \;+\; \underbrace{f_{dt}}_{\text{onboard fuel at } t} \;+\; \underbrace{\sum_{m\in\mathcal{M}} q_{dmt}\,\text{mass}_m}_{\text{mass of tools \& consumables at } t}, \qquad \forall d \in \mathcal{D},\; t \in \mathcal{T} \setminus \{t_0\}. \tag{6.28}$$

6.4. Objective Function

At each MILP iteration, the objective is to maximize net profit over the current Planning Horizon: revenues from executed services minus all modeled costs. The cost components considered are launch costs, PDM costs, operating costs, and delay penalties.

6.4.1. Revenues

Each task $v \in \mathcal{B}$ pays a fixed fee (revenue_v) when its service is *assigned* (i.e., started) to some servicer $d \in \mathcal{D}$ at an admissible time $\tau \in \mathcal{W}_v$. Let $h_{vd\tau} \in \{0, 1\}$ indicate that assignment. The total revenue over the planning horizon is the sum of the revenue associated with all service assignments over all servicers:

$$J_{\text{rev}} = \sum_{v \in \mathcal{B}} \sum_{d \in \mathcal{D}} \sum_{\tau \in \mathcal{W}_v} \text{revenue}_v \, h_{vd\tau}. \tag{6.29}$$

By (6.11), (6.12) and (6.13), any valid assignment $h_{vd\tau} = 1$ induces the activity profile b_{vdt} over $[\tau, \tau + \text{duration}_v)$ and forces completion of the required subtasks at their scheduled node-time pairs. Thus, although revenue is recognized at assignment, the constraints ensure that the service is subsequently completed.

6.4.2. Launch Costs

Launch costs are modeled as a cost per kilogram of mass placed in orbit and are conceptually divided into two components: launch costs associated with servicing spacecraft and launch costs associated with orbital depots.

Servicer launch cost: The launch cost of servicers is computed from their initial mass state at the beginning of each planning horizon, including dry mass, onboard propellant, and service commodities. Because these quantities are fixed parameters within a given MILP instance and do not depend on any optimization variables, the corresponding launch cost terms enter the objective function as constant offsets and therefore do not influence the optimization outcome. The economic impact of these initial launch costs is accounted for in the Rolling Horizon economic ledger, as discussed in Section 6.4.8. The corresponding launch cost is given by:

$$J_{\text{L,srv}} = \sum_{d \in \mathcal{D}} \underbrace{c_{\text{kg}}^{\text{launch}}}_{\text{launch cost per kg}} \left(\underbrace{m_d^{\text{dry}}}_{\text{servicer dry mass}} + \underbrace{f_{dt_0}}_{\text{initial fuel at } t_0} + \underbrace{\sum_{m \in \mathcal{M}} \text{mass}_m \, q_{dmt_0}}_{\text{initial commodities mass}} \right). \tag{6.30}$$

Depot launch cost: In contrast, orbital depots are modeled as operational infrastructure that supplies fuel and service commodities without explicit capacity constraints in the MILP. Consequently, launch costs associated with depot-supplied propellant and commodities are not internalized in the MILP objective and do not constrain resupplying decisions. Instead, the economic impact of all fuel and commodities provided by orbital depots is captured in the Rolling Horizon economic ledger, where the corresponding launch and PDM costs are assigned based on realized mass flows. This separation preserves tractability in the MILP while ensuring that the cumulative economic evaluation accurately reflects the total mass effectively placed in orbit.

Within each MILP instance, the launch cost associated with orbital depots accounts only for the deployment of the depot infrastructure itself and is therefore computed based solely on the depot dry mass:

$$J_{\text{L,dep}} = \underbrace{c_{\text{kg}}^{\text{launch}}}_{\text{launch cost per kg}} \left(\underbrace{N_{\text{dep}} \, m_{\text{dep}}^{\text{dry}}}_{\text{depot dry mass}} \right). \tag{6.31}$$

Total launch cost:

$$J_{\text{launch}} = J_{\text{L,srv}} + J_{\text{L,dep}}. \tag{6.32}$$

6.4.3. Purchase, Development, and Manufacturing Costs

PDM costs capture both the fixed costs associated with the infrastructure and the variable costs associated with propellant and service commodities. Similar to launch costs, PDM costs are conceptually divided into servicer-side and depot-side components.

Servicers: Each servicing spacecraft incurs a fixed development and manufacturing cost, as well as the purchase cost of its initial onboard resources. These include the initial propellant load and any service commodities carried at the beginning of the planning horizon. As with launch costs, these

quantities are fixed parameters within a given MILP instance and therefore enter the objective function as constant offsets that do not influence the optimization outcome. The economic impact of servicer-side PDM costs is accounted for in the Rolling Horizon economic ledger as discussed in 6.4.8, where they are applied once based on the realized initial configuration. The corresponding PDM cost is given by:

$$J_{\text{pdm,srv}} = \sum_{d \in \mathcal{D}} \left(\underbrace{c_d^{\text{dm}}}_{\text{servicer PDM cost}} + \underbrace{c_{\text{fuel}}^{\text{purchase}} f_{dt_0}}_{\text{initial fuel purchase}} + \underbrace{\sum_{m \in \mathcal{M}} c_m^{\text{purchase}} q_{dmt_0}}_{\text{initial commodity purchases}} \right) \tag{6.33}$$

Depots: Orbital depots incur a fixed development and manufacturing cost associated with their deployment. Because depot storage and resupply capacities are not explicitly modeled in the MILP, the purchase of propellant and service commodities used during operations is not priced within the MILP objective and therefore does not constrain resupplying decisions within a planning horizon. Instead, the quantities of fuel and commodities supplied by depots are tracked over each Rolling Horizon iteration and evaluated cumulatively in the Rolling Horizon economic ledger, where the corresponding purchase and manufacturing costs are assigned based on realized mass flows. Accordingly,

$$J_{\text{pdm,dep}} = \underbrace{N_{\text{dep}}\, c_{\text{dep}}^{\text{dm}}}_{\text{depot PDM cost}} . \tag{6.34}$$

Total PDM cost:

$$J_{\text{pdm}} = J_{\text{pdm,dep}} + J_{\text{pdm,srv}}. \tag{6.35}$$

6.4.4. Operating Costs

Operating costs accumulate over the entire Planning Horizon, under the assumption that all servicers and depots remain active throughout and that their per time unit operating rates are constant and independent of activity type. Let $|\mathcal{T}|$ denote the number of time units in the horizon, and let $c_{\text{dep}}^{\text{op}}$ and c_d^{op} be the per time unit operating costs of depots and servicers, respectively. Then:

$$\text{Depots:} \quad J_{\text{ops,dep}} = N_{\text{dep}}\, |\mathcal{T}|\, c_{\text{dep}}^{\text{op}}, \tag{6.36a}$$

$$\text{Servicers:} \quad J_{\text{ops,srv}} = |\mathcal{T}| \sum_{d \in \mathcal{D}} c_d^{\text{op}}. \tag{6.36b}$$

and the total operating cost is:

$$J_{\text{ops}} = J_{\text{ops,dep}} + J_{\text{ops,srv}}. \tag{6.37}$$

6.4.5. Delay Costs

No penalty is applied when the optimizer decides not to address a service need. However, if a service is selected for execution and initiated later than the earliest feasible time within its service window, a time-proportional penalty is incurred. The delay is defined as the difference between the time step at which a service need is triggered and the time step at which the service begins.

For each task $v \in \mathcal{B}$, servicer $d \in \mathcal{D}$, and feasible start time $\tau \in \mathcal{W}_v$,

$$J_{\text{delay}} = \sum_{v \in \mathcal{B}} \sum_{d \in \mathcal{D}} \sum_{\tau \in \mathcal{W}_v} c_v^{\text{delay}} \left(\tau - \min \mathcal{W}_v\right) h_{vd\tau}. \tag{6.38}$$

The penalty is zero if the service starts at the earliest feasible time ($\tau = \min \mathcal{W}_v$) and increases linearly with the delay $\tau - \min \mathcal{W}_v$.

6.4.6. Net profit

The MILP maximizes net profit over the current planning window, i.e., revenues from executed services minus launch, PDM, operating, and delay costs:

$$J_{\text{net}} \;=\; J_{\text{rev}} \;-\; J_{\text{launch}} \;-\; J_{\text{pdm}} \;-\; J_{\text{ops}} \;-\; J_{\text{delay}}. \tag{6.39}$$

The objective defined in (6.39) represents the purely economic objective; an additional safeguard penalty is introduced in Section 6.4.7.

6.4.7. Fuel Safeguard Penalty

In addition to the explicit economic terms introduced above, the objective function includes a soft penalty that discourages plans which would result in critically low fuel levels at the end of the Planning Horizon. This mechanism is not intended to represent a physical or contractual cost, but rather to mitigate the limited look-ahead inherent to the Rolling Horizon formulation. The parameters $\bar{f}$ and λ_f, together with the slack variables s_d^{low}, are introduced solely for this safeguard mechanism and have no physical or economic interpretation outside the optimization process. Their numerical tuning is discussed in Section 8.1.

Motivation: At each iteration, the MILP optimizes over a finite Planning Horizon, while future service demands beyond this window are unknown to the solver. In the absence of a safeguard, the optimizer may rationally deplete most of the onboard fuel to maximize short-term profit, even if doing so leaves the servicer unable to respond to likely future tasks revealed in subsequent iterations. This behavior is a well-known artifact of receding-horizon optimization and can lead to myopic decisions that degrade long-term operational performance.

Modeling: To counteract this effect, a minimum desirable fuel reserve $\bar{f}$ is defined. For each servicer $d \in \mathcal{D}$, a non-negative slack variable s_d^{low} is introduced at the end of the Planning Horizon t_F:

$$s_d^{\text{low}} \;\geq\; \bar{f} - f_{dt_F}, \qquad s_d^{\text{low}} \;\geq\; 0. \tag{6.40}$$

Whenever the terminal fuel level f_{dt_F} exceeds the threshold $\bar{f}$, the constraint is inactive and $s_d^{\text{low}} = 0$. If the fuel level falls below $\bar{f}$, the slack variable becomes positive and measures the shortfall.

A linear penalty proportional to this shortfall is subtracted from the objective:

$$J_{\text{fuel}} \;=\; \lambda_f \sum_{d \in \mathcal{D}} s_d^{\text{low}}, \tag{6.41}$$

where $\lambda_f > 0$ is a tuning parameter controlling the strength of the incentive.

Interpretation: This term acts as a *soft terminal constraint* on fuel availability. It does not reward high fuel levels, nor does it encourage unnecessary refueling: once $f_{dt_F} \geq \bar{f}$, the penalty is identically zero and additional fuel has no marginal benefit in the objective. Consequently, there is no incentive to waste fuel or to refuel beyond operational needs.

Instead, the penalty biases the optimizer toward scheduling depot visits or conserving propellant *only when* the alternative would leave the servicer under-resourced at the end of the horizon. In this sense, it approximates the opportunity cost of future, unseen service opportunities without explicitly forecasting them.

Role in the Rolling Horizon framework: In the Rolling Horizon framework, optimization is carried out over a Planning Horizon, while only decisions within the Control Horizon are executed and propagated to the next iteration. Although the terminal state at t_F is not directly enforced, it influences which actions are selected within the Control Horizon. Penalizing excessive fuel depletion at the end of the Planning Horizon discourages short-sighted decisions near the horizon boundary and biases the optimizer toward solutions that preserve operational flexibility for future planning windows, without imposing hard terminal constraints.

Final objective: Including the fuel safeguard, the complete objective maximized at each iteration is:

$$J_{\text{net}} = J_{\text{rev}} - J_{\text{launch}} - J_{\text{pdm}} - J_{\text{ops}} - J_{\text{delay}} - J_{\text{fuel}}. \tag{6.42}$$

6.4.8. Rolling Horizon Accounting

Although the MILP formulation for the objective function described in the previous sections is solved independently at each Rolling Horizon iteration, the economic performance of the mission is evaluated cumulatively over the executed control horizons. To this end, the framework maintains an explicit Rolling Horizon accounting ledger that records the *realized* revenues and costs associated with decisions that are actually executed over successive Control Horizons to avoid double-counting across Planning Horizons.

At each Rolling Horizon iteration, only decisions implemented within the applied control horizon are monetized. Revenues are recognized at service start times occurring within the interval, in accordance with Equation 6.29, and delay penalties are charged accordingly following Equation 6.38. Operating costs for servicers and depots accrue proportionally over the same interval and are computed using Equation 6.37.

One-off costs, namely servicer PDM costs (6.33), depot development and manufacturing costs as modeled in the MILP (6.34), as well as the corresponding launch costs (6.30 and 6.31), are applied only in the first Rolling Horizon iteration. This prevents double counting as the horizon advances and ensures that subsequent iterations account exclusively for incremental operational effects.

Costs associated with depot-supplied resources are recorded at the time they are physically realized. Propellant purchase and the corresponding launch costs are charged upon completion of depot refueling events within the control horizon, whereas commodity purchase and launch costs are inferred from net inventory changes and service-induced consumption over the same interval.

Any artificial penalty terms introduced in the MILP objective for numerical stability or operational safeguarding, such as the fuel-level penalty described in Section 6.4.7, are not recorded in the Rolling Horizon accounting ledger and do not contribute to the reported economic performance.

By aggregating the per-iteration ledger entries over all executed control horizons, the framework reconstructs the total realized profit over the full scheduling horizon. This separation between per-iteration optimization and cumulative accounting ensures temporal consistency and enables a faithful post-analysis of the economic performance of the mission.

Fuel mass supplied by depots: Let $\mathcal{RS}$ be the set of *departure-stamped* resupply triplets (t, i, j), i.e., times t at which a refuel-eligible (coasting) leg $i \to j$ may depart. The total fuel credited upon arrival over the horizon is:

$$M_{\text{sup}}^{\text{fuel}} = \sum_{d \in \mathcal{D}} \sum_{a \in \mathcal{A}} \sum_{\substack{t \in \mathcal{T}: \\ (t-\tau_a, \mathcal{I}(a), \mathcal{J}(a)) \in \mathcal{RS} \\ t-\tau_a \geq t_0}} r_{d,a,t}. \tag{6.43}$$

This quantity is used in the Rolling Horizon accounting ledger to assign fuel purchase and launch costs based on the refueling events that are actually executed.

Commodity mass supplied by depots: Using inventory conservation (final minus initial plus total task consumption), the total commodity mass supplied by depots during the full scheduling horizon is computed as:

$$M_{\text{sup}}^{\text{comm}} = \sum_{d \in \mathcal{D}} \sum_{m \in \mathcal{M}} \underbrace{\text{mass}_m}_{\text{mass per unit of } m} \left[\underbrace{q_{dm,\,t_F} - q_{dm,\,t_0}}_{\text{final - initial inventory}} + \underbrace{\sum_{v \in \mathcal{B}} \sum_{\tau \in \mathcal{W}_v} \delta_{vm}\, h_{vd\tau}}_{\text{task consumption of commodity } m} \right]. \tag{6.44}$$

The corresponding commodity purchase and launch costs are assigned in the Rolling Horizon accounting ledger based on this realized mass flow.

7

Framework Verification

Prior to addressing operational or strategic analyses, it is necessary to verify that the framework behaves as intended in a controlled environment. The verification stage consists of a series of preliminary tests designed to ensure that all model constraints, orbital maneuvers, and resource and economic balances are correctly enforced. These tests confirm that the framework produces physically consistent and numerically stable results, establishing confidence in its reliability before extending it to full-scale analyses. Verification here focuses on correctness of implementation and internal consistency with the mathematical formulation, rather than validation against real mission data or operational realism.

The verification employs the same YAML-based infrastructure definitions for servicers, customers, tasks, and depots used later in the case studies in Chapter 8 and introduced in Chapter 2. However, it operates on a deliberately simplified configuration consisting of a single servicer, one or no depot, a reduced set of customers and tasks, and fixed random seeds for stochastic task generation to guarantee reproducibility. This setup allows the framework's internal logic and constraint interactions to be isolated and evaluated without the complexity of large-scale scenarios.

The purpose of this stage is not to derive operational decisions or architectural insights, but to establish confidence in the framework's feasibility, internal consistency, and numerical reliability before proceeding to the case studies presented in the following chapter.

The verification is structured into the following steps:

- Baseline feasibility verification
- Maneuver and orbital transfer consistency
- Task execution and temporal logic verification
- Resource and mass conservation checks
- Economic consistency verification
- Edge-case and boundary-condition probes

Setup

Both the verification and case study setups begin by defining the nodes and corresponding orbital parameters that serve as inputs to the network. Nodes are placed along each orbit where either an orbital depot or a customer satellite operates, the latter marking potential task locations. For verification, three SSOs with similar orbital elements were selected to ensure relatively short maneuver durations. Orbital data were obtained from Space-Track (U.S. Space Command, 2025). Two of the selected orbits represent customer satellites, while the third corresponds to the depot's operational orbit, from which the servicer departs.

All orbits are approximated as circular, with eccentricity set to zero. For each orbit, six equidistant nodes are defined, spaced at 60° intervals. Given that the average orbital period of the considered orbits is approximately 96 minutes, the simulation time unit Δt is set to 16 minutes, such that motion between two adjacent nodes within the same orbit corresponds to a single simulation time step. This discretization level is chosen to support verification of feasibility and temporal consistency within MILP, rather than high-fidelity trajectory timing. Table 7.1 summarizes the orbital parameters of the selected

orbits and their corresponding 18 nodes considered in the preliminary verification.

Table 7.1: Orbital parameters for the nodes used in the verification.

Orbit	Nodes	a (km)	i (deg)	RAAN (deg)	Based on	Period (min)	Separation (min/deg)
1	1–6	6956.651	97.706	308.753	SATELIOT_1	96.2	16.0 / 60.0
2	7–12	6960.291	97.721	308.843	HAWK-10A	96.3	16.1 / 60.0
3	13–18	6873.499	97.387	308.848	HORUS 2	94.5	15.8 / 60.0

The remaining parameters used in this preliminary verification are presented below without extensive explanation or contextual discussion. Their detailed justification, including literature references and parameter tuning procedures, is provided in the subsequent chapter, in Section 8.1 on the case study setup. To constrain the problem dimensionality and maintain computational tractability, the maximum number of revolutions, N_{max}, considered for phasing maneuvers is limited to four. Given the close similarity between the orbital parameters of the selected orbits, the parameters for the combined maneuver type are defined with a maximum drift time of 1.0044 days and a $\Delta\Omega_{\text{TH}}$ of 0.0925°. Based on these maneuver parameters, the resulting network consists of 252 arcs in total, of which 18 correspond to coasting maneuvers, 24 to combined maneuvers, and 210 to phasing maneuvers (cf. Table 7.2):

Table 7.2: Arc summary by start/end orbit (IDs) and maneuver type.

Maneuver	Start orbit	End orbit	Count	Time (h)			ΔV (km/s)			ΔV per hour (km/s/h)		
				Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
Coasting	1	1	6	0.27	0.27	0.27	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	2	2	6	0.27	0.27	0.27	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	3	3	6	0.26	0.26	0.26	0.00	0.00	0.00	0.00	0.00	0.00
Combined Maneuver	1	2	4	23.45	23.45	23.45	0.20	0.20	0.20	0.01	0.01	0.01
Combined Maneuver	1	3	4	24.10	24.10	24.10	0.17	0.17	0.17	0.01	0.01	0.01
Combined Maneuver	2	1	4	23.45	23.45	23.45	0.19	0.19	0.19	0.01	0.01	0.01
Combined Maneuver	2	3	4	1.31	1.31	1.31	0.17	0.17	0.17	0.13	0.13	0.13
Combined Maneuver	3	1	4	24.10	24.10	24.10	0.16	0.16	0.16	0.01	0.01	0.01
Combined Maneuver	3	2	4	1.31	1.31	1.31	0.16	0.16	0.16	0.12	0.12	0.12
Phasing	1	1	72	1.87	4.59	7.22	0.20	0.63	1.70	0.03	0.19	0.71
Phasing	2	2	72	1.87	4.59	7.22	0.20	0.63	1.70	0.03	0.19	0.70
Phasing	3	3	66	1.84	4.54	7.09	0.20	0.63	1.71	0.03	0.20	0.72

For the verification setup, a single servicer is modeled and considered. It is equipped with the complete set of tools described in Chapter 2, enabling it to perform all defined service types, namely inspection, refueling, repair, and active debris removal. The corresponding servicer parameters are summarized below in Table 7.3.

Table 7.3: Servicer parameters.

Tools	Dry mass [kg]	Fuel cap. [kg]	Impulse [s]	PDM cost [M$]	Op. cost [$/day]	Start node	Cap. spares	Cap. prop. [kg]	Cap. d. modules
T1/2/3/4	1500	500	316	76.05	13K	1	4	400	4

Additionally, in the baseline verification scenario, no depot is included in order to isolate and assess the framework's baseline behavior. In subsequent verification scenarios, a single depot can be introduced to verify the resupply functionality and its integration within the overall optimization process. The modeled depot is assumed to be deployed and operating in orbit 1, which also serves as the departure orbit for all servicers. The parameters associated with the depot are summarized in Table 7.4.

For verification purposes, only a limited set of customers and tasks is required. Consequently, a single customer satellite is placed in each of the two customer orbits (Orbits 2 and 3) listed in Table 7.1. In the baseline scenario, where depot resupply is not enabled, one customer is assigned deterministic refueling tasks, while the other is assigned random repair tasks. In the subsequent verification scenarios, the second customer may also receive inspection tasks. The overall demand is varied by adjusting the inter-occurrence times of tasks. It is important to emphasize that this configuration is purely designed for functional testing and does not reflect the realistic demand modeling adopted in the later case stud-

Table 7.4: Depot parameters.

Parameter	Value	Units
Depot operating cost	$13K	USD/day
Depot development & manufacturing cost	$200M	USD
Depot dry mass	3K	kg

ies. The customer assignment data are summarized in Table 7.5, while the parameters associated with each task type are detailed in Table 7.6.

Table 7.5: Customer assignments.

Customer ID	Orbit ID	Start node	Tasks
1	2	9	refueling
2	3	13	repair/inspection

Table 7.6: Task parameters. Service duration and service window are in time steps (Δt) and inter-occurence time in minutes.

Type	Task	Revenue	Delay penalty	Duration	Service window	(Mean) Inter-occurence time	Commodities
Deterministic	Inspection	$1.67M	$833.3	5	300	3500	—
Deterministic	Refueling	$2.5M	$16.67K	5	300	3500	prop.: 100 kg
Random	Repair	$5M	$16.67K	5	300	6000	spares: 1 unit

Finally, the commodities stored in depots, transported by servicers, and consumed during services have their associated parameters summarized in Table 7.7. Additionally, the launch cost is set to 1,410 USD per kilogram. Although a distinction is made between the fuel stored for customer refueling and that used for the servicers' orbital maneuvers, the unit fuel cost is assumed to be identical for both cases.

Table 7.7: Commodity purchase costs and masses.

Commodity	Purchase cost	Unit	Mass per unit [kg]
Spares	$86K	per unit	86
Propellant	$230	per kg	1
De-orbit kit	$11K	per unit	1.4
T1	$100K	per unit	100
T2	$100K	per unit	100
T3	$100K	per unit	100
T4	$100K	per unit	100

Baseline Scenario

The baseline verification scenario, conducted without any modeled depot or resupply opportunities, employs the parameters listed in Table 7.8 to support the Rolling Horizon optimization framework.

Under these settings, the three service tasks precomputed for the scheduling horizon are:

- Deterministic `refueling` task for Customer 1 at 3500 min = 219 steps, requiring 100 kg of propellant,
- Random `repair` task for Customer 2 at 6120 min = 383 steps, requiring one spare unit,
- Random `repair` task for Customer 2 at 6272 min = 392 steps, requiring one spare unit.

Solver diagnostics: All MILP problems were solved using Gurobi 12.0.3 (Linux 64-bit, Ubuntu 22.04) on an Intel® Core™ i7-10510U CPU (4 physical cores, 8 threads). The solver was configured with a relative optimality gap tolerance of 10^{-4}, and all instances were solved to optimality within this bound.

Across the Rolling Horizon iterations, model sizes ranged from approximately $19{,}000$ to $160{,}000$ variables and up to $145{,}000$ constraints, with 6-32% of variables being binary. Pre-solve routines con-

Table 7.8: Simulation parameters used in the baseline verification scenario.

Parameter	Value	Units
Time step size (Δt)	16	min
Scheduling horizon	6,500	min
Planning horizon	2,000	min
Maximum idle time	2,000	min

sistently removed 70-90% of rows and columns, indicating strong structural redundancy and constraint tightness from network-flow formulations.

Solver logs report that the largest instance (about 160k variables, 145k constraints) required 2,123 simplex iterations and one explored branch-and-bound node, reaching optimality in 20.4 s. Smaller instances converged in under 1.3 s with negligible search effort (0-71 simplex iterations). Root relaxations typically solved within 1 s, and all MILP instances achieved a final optimality gap of 0.0000%.

These results confirm that the model is numerically stable and computationally efficient in the context of verification-scale instances. The solver found feasible integer solutions early in the pre-solve stage and required only minimal branching to reach optimality.

Baseline verification results

Figure 7.2 and Table 7.9 should be read jointly to interpret the results of the baseline verification run. The figure illustrates the optimized mission timeline of Servicer d1 together with the time evolution of onboard fuel and service commodities across the full Scheduling Horizon, while the table reports the same sequence of executed service tasks and depot resupply events shown in the figure in a structured and quantitative form, ordered according to their occurrence along the timeline and annotated with their corresponding start and end times. For each service, the corresponding activation time, introduced in the scenario description above, is also reported, enabling the reader to directly relate task availability, execution timing, and the resulting delays. The economic evolution over the Rolling Horizon iterations, expressed as cumulative net profit, is shown in Figure 7.1.

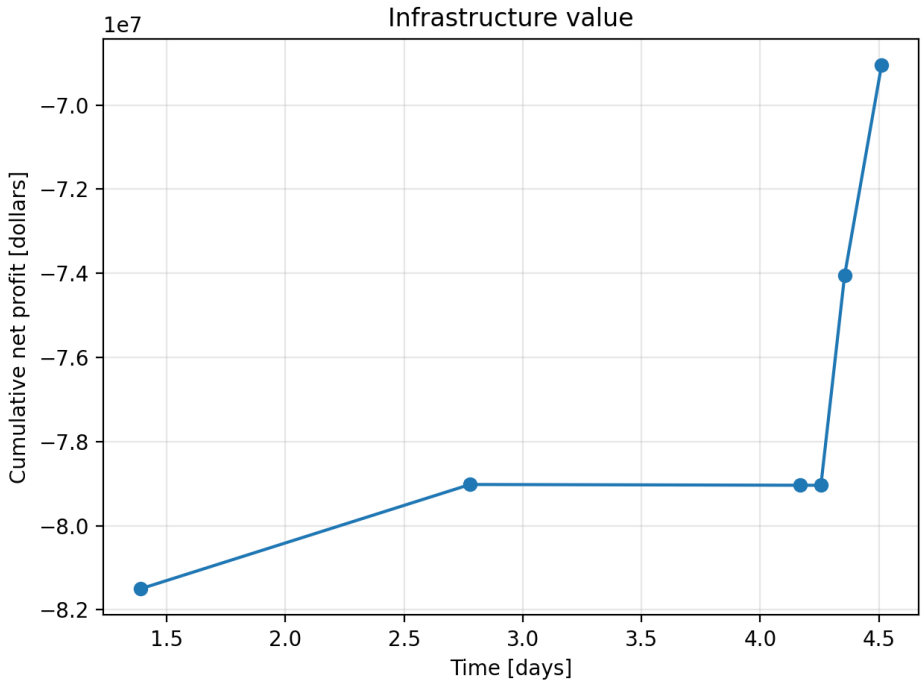


Figure 7.1: Cumulative net profit evolution over the Rolling Horizon iterations.

Table 7.9: Executed service tasks for Servicer d1 in the baseline verification run, ordered by execution start time. Activation times refer to task availability, while execution times correspond to the scheduled service start.

Service	Activation [min]	Start [min]	End [min]	Delay [min]	Duration [min]
Refueling of Customer 1	3500	3504	3584	≈0	80
First repair of Customer 2	6120	6240	6320	120	80
Second repair of Customer 2	6272	6320	6400	48	80

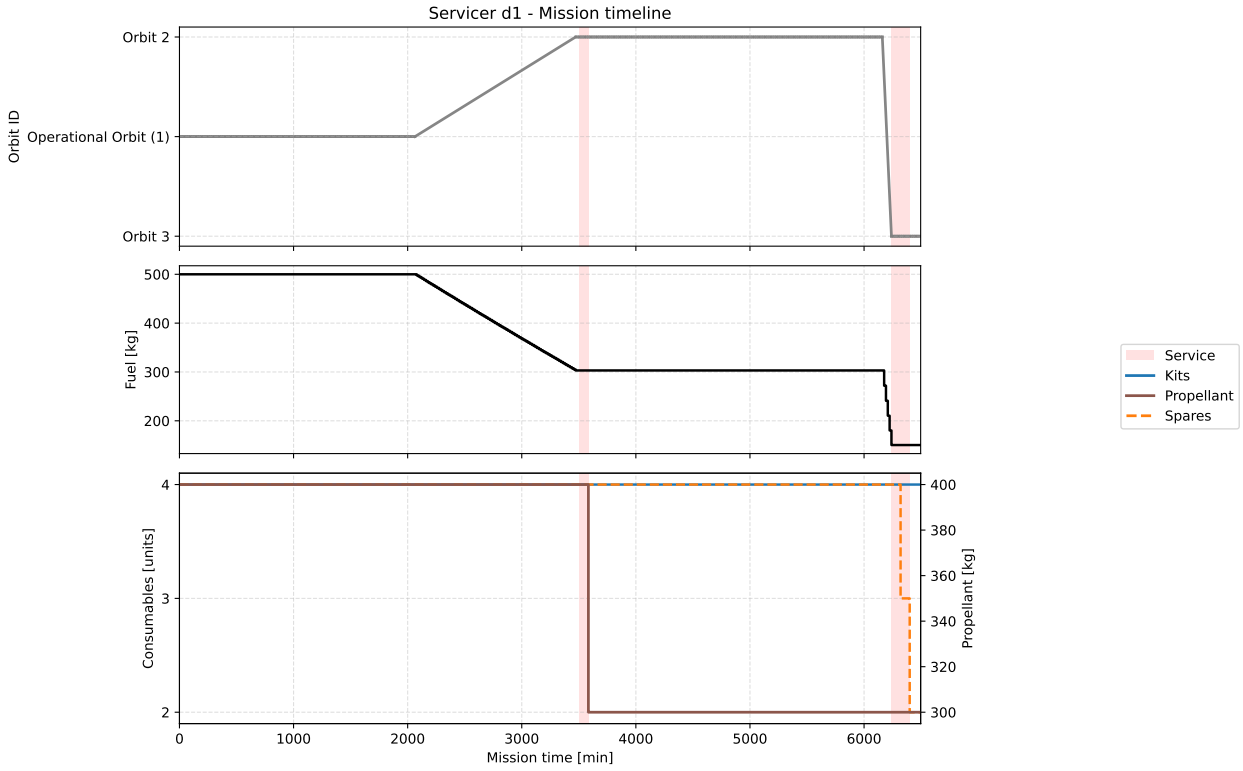


Figure 7.2: Mission timeline and resource evolution for Servicer d1. The upper panel shows the servicer's trajectory across orbital planes, ordered by increasing altitude, where diagonal segments represent combined maneuvers between orbits. The lower panels depict the time evolution of onboard resources, including propellant reserved for the servicer's own maneuvers (middle panel) and service commodities carried for customer operations (refueling propellant, de-orbit kits, and spare parts).

Feasibility

The optimized trajectory obtained for this depot-free verification run, illustrated in the top panel of Figure 7.2, satisfies all feasibility-imposing constraints defined in the MILP formulation. It is important to note that the orbits on the y-axis, labeled by their IDs, are displayed in order of increasing altitude; this ordering is purely a visualization choice. The resulting schedule forms a single, continuous trajectory from $t = 0$ to $t = 406$ steps, with no temporal discontinuities between successive segments, thereby fulfilling the flow conservation and terminal closure requirements of Eqs. (6.5) and (6.7).

Each task is assigned exactly once, as confirmed by the unique appearance of every task in the optimized service list shown in Table 7.9, in accordance with the task assignment constraint (Eq. (6.10)). No overlapping services occur at the same customer, satisfying the exclusivity condition imposed by Eq. (6.14). This constraint ensures that, at any given time step, each customer node can be serviced by only one operation, thereby preserving logical feasibility. In this case, the two repair tasks at Customer 2 are executed sequentially, with one ending at $t = 395$ and the next beginning immediately thereafter.

Finally, all executed services require tools that are available on the modeled servicer, which in this verification setup is equipped with the full toolset (T1-T4). This satisfies the tool-availability constraint defined in Eq. (6.23), confirming that each service is feasible given the servicer's onboard equipment configuration.

Maneuver realism

The maneuvers represented in this study are drawn from the precomputed network arcs introduced in Chapter 3. Arc attributes (transfer time and cost) are generated by the orbital transfer module described in Chapter 4 and have been cross-checked against standard orbital mechanics references.

Accordingly, the maneuvers used in the MILP formulation are not solved dynamically but are drawn from this precomputed set of feasible arcs, ensuring consistency between the optimization layer and the underlying orbital mechanics model. This approach allows the scheduling problem to retain a discrete and computationally tractable structure while preserving realistic orbital transfer costs.

For illustration, consider node $i = 3$ in Orbit 1. This node has a true anomaly of $120°$, a radius of $6{,}956.651$ km, an inclination of $97.7062°$, a RAAN of $308.7533°$, and an orbital period of 96.241 min. From node $i = 3$, the feasible maneuvers comprise a *coasting* arc to the adjacent node on the same orbit ($j = 4$) and a set of *phasing* arcs to non-adjacent same-orbit nodes ($j \in \{1, 2, 5, 6\}$) with different k values, as summarized in Table 7.10. Because node $i = 3$ is not located at an ascending or descending node, no combined maneuvers originate or terminate at this node. Each arc is characterized by its transfer time, total ΔV, and the associated fuel ratios ϕ (total) and ψ (per time step), which link orbital transfer cost to discrete fuel usage in the MILP.

Table 7.10: Precomputed arcs originating from node $i = 3$ in Orbit 1. ϕ denotes the total propellant fraction (mass fraction of fuel consumed relative to initial mass) and ψ the per-time step propellant fraction.

Start node	End node	Maneuver Type	Time [min]	ΔV [km/s]	ϕ_a	ψ_a
3	4	Coasting	16.04	0.000	0.0000	0.0000
3	1	Phasing (k=1)	128.32	1.268	0.3358	0.0499
3	1	Phasing (k=2)	224.56	0.722	0.2078	0.0165
3	1	Phasing (k=3)	320.80	0.505	0.1504	0.0081
3	1	Phasing (k=4)	417.04	0.388	0.1178	0.0048
3	2	Phasing (k=1)	112.28	0.722	0.2078	0.0327
3	2	Phasing (k=2)	208.52	0.388	0.1178	0.0096
3	2	Phasing (k=3)	304.76	0.266	0.0822	0.0045
3	2	Phasing (k=4)	401.00	0.202	0.0631	0.0026
3	5	Phasing (k=3)	256.64	0.632	0.1844	0.0127
3	5	Phasing (k=4)	352.88	0.459	0.1377	0.0067
3	6	Phasing (k=1)	144.36	1.698	0.4218	0.0591
3	6	Phasing (k=2)	240.60	1.012	0.2787	0.0215
3	6	Phasing (k=3)	336.84	0.722	0.2078	0.0110
3	6	Phasing (k=4)	433.08	0.561	0.1657	0.0067

Task execution logic

The temporal execution of the serviced tasks further confirms the internal consistency of the solution. Each service precisely spans its prescribed duration of 80 minutes (five time steps), demonstrating the correct linkage between assignment variables and activity indicators as defined in Eq. (6.11).

All services are initiated within their respective service windows. The `refueling` task at Customer 1 begins at $t = 219$ steps, exactly matching the scheduled activation time (3500 min$/16$ min ≈ 219). The two `repair` tasks begin at $t = 390$ and $t = 395$ steps, corresponding to delays of 7 and 3 steps, respectively, after their activation notices ($t = 383$ and $t = 392$), yet still well within the 300-step service windows imposed by Eqs. (6.11) and (6.13).

These short delays naturally arise from the Rolling Horizon structure of the algorithm, which captures a more realistic operational behavior: replanning is initiated only upon task activation for stochastic tasks, requiring the servicer to complete the necessary orbital transfer before service execution can begin. For the first repair task, the observed 7-step delay corresponds to the time required to reach an orbital node (i.e. the ascending or descending node) and perform a 5-step transfer to the customer's orbit, whereas the second repair task can only begin once the first service has been completed. This sequential and time-consistent behavior verifies that the temporal coordination between activation, travel, and service execution is correctly implemented within the MILP formulation.

The spatial conditions governing task execution, ensuring that the servicer successfully rendezvouses

with each customer and remains co-located throughout the operation, are enforced by Eqs. (6.12) - (6.13). This behavior is clearly reflected in the optimized trajectory. The servicer departs from node 1 at $t = 0$ and at $t = 126$ begins a combined transfer maneuver from Orbit 1 (node 1) to Orbit 2 (node 7), lasting approximately 88 time steps (1408 minutes), consistent with the arc transfer time derived in Chapter 4. It arrives at node 7 at $t = 214$ steps (3424 minutes). Meanwhile, Customer 1, initially located at node 9, continues coasting along its orbit and reaches the same node simultaneously. The refueling task is executed from $t = 219$ to $t = 224$ steps (3504–3584 minutes), during which both vehicles remain co-located, effectively coasting together for the entire 80-minute service duration, consistent with the spatial and temporal coupling enforced by Eqs. (6.12) - (6.13).

After completing the refueling, the servicer performs a second combined transfer from Orbit 2 (node 10) to Orbit 3 (node 13), where Customer 2 operates. This maneuver runs from $t = 385$ to $t = 390$ steps (6160-6240 minutes), concluding with rendezvous at node 13. The first repair task then begins immediately at $t = 390$ steps and concludes at $t = 395$ steps (6240-6320 minutes), followed by a second repair from $t = 395$ to $t = 400$ steps (6320-6400 minutes). As before, the servicer remains co-located with the customer throughout both operations, satisfying the spatial and temporal conditions prescribed by the above mentioned constraints. Overall, the trajectory exhibits a coherent and physically consistent sequence of transfers and in-orbit services, confirming that the implemented constraints successfully preserve both positional and temporal feasibility throughout the mission timeline.

Conservation of mass and resources

Initial conditions: At the start of the scheduling horizon ($t_0 = 0$), fuel, inventories, and total mass are fixed by Eqs. (6.2) - (6.4). In this verification instance, the single servicer is equipped with the full toolset and initialized to its *full* capacity: $F_d^0 = F_d^{\text{cap}} = 500$ kg, $Q_{d,\text{spares}}^0 = 4$ units, $Q_{d,\text{prop}}^0 = 400$ kg, $Q_{d,\text{modules}}^0 = 4$ units, and one unit of each tool T1-T4 (each 100 kg). With $m_d^{\text{dry}} = 1500$ kg and $\text{mass}_{\{\text{spares,prop,modules,T1-T4}\}} = \{86, 1, 1.4, 100\}$ kg per unit, this yields

$$m_{d,t_0}^{\text{tot}} \;=\; 1500 + 500 + (4 \cdot 86 + 400 \cdot 1 + 4 \cdot 1.4 + 4 \cdot 100) \;=\; 3{,}149.6 \text{ kg}.$$

These initial values define the reference state for the subsequent conservation checks, and their consistency with the model initialization can be visually confirmed at t_0 in the middle and bottom panels of Figure 7.2, which depict the evolution of fuel and service commodities over the mission timeline. It is important to note the distinction between the fuel allocated for the servicer's orbital maneuvers and the propellant reserved for customer refueling. This separation is clearly observable in Figure 7.2, where the middle plot represents the servicer's own fuel consumption, while the lower plot includes the customer propellant inventory.

Fuel capacity bounds and balance: The servicer's propellant evolution follows the fuel balance equation (Eq. (6.27)), constrained by the capacity limits $0 \le f_{dt} \le F_d^{\text{cap}}$ in Eq. (6.24). At each time step, the fuel consumption (burn_{dt}) is computed as a linearized product of the vehicle's pre-burn mass and the per time step propellant fraction ψ_a, derived from the rocket equation (Chapter 4) and precomputed for each arc during network construction. In the depot-free verification run, refueling is disabled $(r_{d,a,t} \equiv 0)$, and the balance reduces to

$$f_{dt} = f_{d,t-1} - \text{burn}_{dt}.$$

During coasting or service intervals, thrusting is absent and thus $\text{burn}_{dt} = 0$ by construction.

In the resulting trajectory, fuel is consumed exclusively on the two combined transfers:

$$(Nodes\ 1 \to 7):\ t = 126 \to 214 \text{ (88 steps)}, \qquad (Nodes\ 10 \to 13):\ t = 385 \to 390 \text{ (5 steps)}.$$

All other motion between services corresponds to orbital coasting with zero propellant burn. The fuel trajectory (Figure 7.2, middle) satisfies the balance equation throughout: (i) it remains within the tank capacity $F_d^{\text{cap}} = 500$ kg; and (ii) it decreases monotonically during thrusting periods, consistent with the multiplicative update $m_{t+1} = m_t(1 - \psi_a)$, where ψ_a is the per time step propellant fraction derived from the rocket equation. This behaviour aligns with the precomputed arc values generated by the `orbital transfer` module (Chapter 4). The fuel level remains constant at 500 kg from $t = 0$ to $t = 126$, then decreases during the first combined transfer $(1 \to 7)$ from 500.000 to 303.101 kg over 88 steps ($1{,}408$ min), consistent with $\Delta v_{1\to7} = 0.200$ km/s and $\phi_{1\to7} = 0.06252$. It drops again in the second

transfer $(10 \to 13)$, from 303.101 to 150.381 kg over 5 steps (80 min), matching $\Delta v_{10\to13} = 0.171$ km/s and $\phi_{10\to13} = 0.054$. Outside these thrusting intervals ($t = 214-385$ and during the service window $t = 219-224$), the curve remains constant, as expected.

To illustrate the discrete-time update, consider the first thrusting interval between $t = 126$ and $t = 127$. The total mass (dry mass plus onboard commodities and propellant) decreases from $m_{126} = 3{,}149.600$ kg to $m_{127} = 3{,}147.290$ kg, corresponding to a propellant mass loss of 2.3096 kg. The observed per step propellant fraction, defined with respect to the current total mass, is therefore

$$\psi_{\text{obs}} = \frac{m_{126} - m_{127}}{m_{126}} = \frac{2.3096}{3{,}149.6} \approx 7.33 \times 10^{-4} \quad (\text{or } 0.0733\%).$$

Using $I_{sp} = 316$ s and $g_0 = 9.80665$ m/s^2, the rocket equation gives the per-step velocity increment

$$\Delta v_{\text{step}} = -\, g_0 I_{sp} \ln(1 - \psi_{\text{obs}}) \approx 2.273 \text{ m/s},$$

which is consistent with the total arc $\Delta v_{1\to7} = 0.200$ km/s distributed over the 88 time steps of the transfer (average $\approx 200\,\text{m/s}/88 \approx 2.27\,\text{m/s}$ per step) and with the discretized ψ_a applied along that arc. At each time step, an identical propellant fraction is applied to the current total mass, leading to a fuel decline whose accumulated consumption equals the total propellant fraction for the transfer,

$$\phi_a \;=\; 1 - \exp\Big(-\frac{\Delta v_{\text{tot}}}{g_0 I_{sp}}\Big),$$

as expected from the continuous rocket-equation formulation.

Commodity capacity bounds and balance: Onboard commodities are tracked by discrete-time inventory variables $q_{d,m,t}$ for servicer $d \in \mathcal{D}$ and commodity $m \in \mathcal{M}$, constrained by payload capacity limits

$$0 \le q_{d,m,t} \le Q^{\text{cap}}_{d,m},$$

as defined in Eq. (6.16). The evolution of onboard inventories follows the balance equations in Eqs. (6.19)-(6.22) of Chapter 6, which implement a case distinction via big-M to model either consumption or replenishment depending on the resupply indicator r_{dt} defined in Eq. (6.18).

At each time step, inventories decrease when a service finishes and consumes the required amount $\delta_{v,m}$ for task $v \in \mathcal{B}$, and reset to full capacity when a resupply (depot) arc is completed ($r_{dt} = 1$). Accordingly, the model enforces two regimes:

(i) when a task occurs, the inventory evolves as

$$q_{d,m,t} = q_{d,m,t-1} - c_{d,m,t},$$

where $c_{d,m,t}$ denotes the end-of-service consumption defined in Eq. (6.17); or

(ii) when a resupply arc is completed, the inventory is forced to its maximum payload capacity,

$$q_{d,m,t} = Q^{\text{cap}}_{d,m}.$$

In the depot-free verification run, resupply is disabled ($r_{dt} \equiv 0$), reducing the balance to

$$q_{d,m,t} = q_{d,m,t-1} - c_{d,m,t}.$$

During transfers, coasting or service intervals in which no tasks finish, consumption is zero ($c_{d,m,t} = 0$), and inventories remain constant by construction.

Unlike propellant, commodities are not consumed continuously over transfer arcs but only upon task completion or resupply. Throughout the mission, three commodity types are tracked onboard: spares, propellant (used for refueling operations), and de-orbit modules. According to the task parameters in Table 7.6, each refueling task consumes 100 kg of *propellant*, each repair task consumes one *spare unit*, and inspection tasks do not consume any commodities.

At $t = 224$, one refueling operation is completed, consuming 100 kg of onboard propellant and reducing $q_{d,\text{prop},t}$ from 400 to 300 kg. Later, at $t = 395$, a repair task finishes, consuming one spare and

reducing $q_{d,\text{spares},t}$ from 4 to 3 units, followed by a second repair task at $t = 400$, which further decreases $q_{d,\text{spares},t}$ from 3 to 2 units. Between service events, all inventories remain constant, confirming that no unintended consumption occurs outside task completions.

The discrete-time profiles (Figure 7.2, bottom) confirm this stepwise behavior: commodity levels remain flat between services and drop instantaneously upon task completion. This evolution fully satisfies the balance equations, demonstrating consistency between the MILP formulation and the recorded state transitions in the simulation output.

Mass balance: The total mass of the servicer evolves consistently with the sum of its dry mass, onboard fuel, and onboard commodities, as enforced by Eq. (6.28). At initialization, this condition matches Eq. (6.4), ensuring that the total launch mass equals the dry mass plus the full inventories and fuel capacity, yielding $m_{d,0}^{\text{tot}} = 3{,}149.6$ kg in this verification case.

Over the mission timeline, $m_{d,t}^{\text{tot}}$ decreases solely as a result of modeled fuel and commodity consumption. During the first combined transfer $(1 \to 7)$, the servicer expends 196.9 kg of fuel, reducing its total mass from $3{,}149.6$ to $2{,}952.7$ kg. A further decrease occurs during the refueling service, which consumes 100 kg of onboard propellant, lowering the total mass to $2{,}852.7$ kg. Subsequently, the transfer $(10 \to 13)$ consumes an additional 152.7 kg of fuel, further reducing the mass to $2{,}700.0$ kg. Finally, two repair tasks completed at $t = 395$ and $t = 400$ remove one spare each (86 kg per unit), yielding a final mass of $2{,}528.0$ kg. Between these events, both fuel and commodity inventories remain constant, resulting in flat segments in the total-mass trajectory as implied by the constant sections in the bottom two panels of Figure 7.2.

Although $m_{d,t}^{\text{tot}}$ is not plotted explicitly, its evolution is fully consistent with the fuel and commodity balance equations and their corresponding trajectories shown in Figure 7.2. The conservation of mass is therefore satisfied at every discrete time step: any decrease in total mass results exclusively from physical consumption events, with no artificial gains or losses, confirming the internal coherence of the MILP formulation.

Economic consistency

The overall economic outcome of this verification scenario is evaluated according to the profit-maximizing objective function defined in Eq. (6.39), which aggregates all revenue and cost components over the mission horizon. The cumulative results obtained from the Rolling Horizon optimization framework are presented below, while the analytical formulations introduced in Chapter 6 are used to verify the consistency of each corresponding objective function term.

Revenues (J_{rev}): Revenues arise from completed service tasks according to Eq. (6.29):

$$J_{\text{rev}} = \underbrace{2.5\ \text{M}}_{\text{refueling revenue}} + 2 \times \underbrace{5\ \text{M}}_{\text{repair revenue}} = 12.5\ \text{M USD}.$$

This value corresponds to the three executed services listed in Table 7.9, with individual task revenues specified in Table 7.6.

Launch costs (J_{launch}): The total launch cost is computed according to Eq. (6.30) and accounts for both the servicer dry mass and its initial onboard load:

$$J_{\text{launch}} = \underbrace{1410}_{\text{USD/kg (launch cost per kg)}} \times \underbrace{3149.6}_{\text{kg (total initial servicer mass)}} = 4{,}440{,}936\ \text{USD}.$$

Purchase, Development, and Manufacturing Costs (J_{pdm}): The PDM component is computed according to Eq. (6.33), combining the servicer manufacturing cost with the purchase of its initial fuel and onboard commodities:

$$J_{\text{pdm}} = \underbrace{76.05\ \text{M}}_{\text{servicer manufacturing}} + \underbrace{500 \times 230 + 4 \times 86\ \text{k} + 400 \times 230 + 4 \times 11\ \text{k} + 4 \times 100\ \text{k}}_{\text{initial fuel and commodity purchases}} = 77{,}045{,}000\ \text{USD}.$$

The servicer manufacturing cost is taken from Table 7.3, while unit prices for fuel and commodities are drawn from Table 7.7.

Operating costs (J_{ops}): Operating costs increase proportionally to mission duration, following Eq. (6.36b):

$$J_{\text{ops}} = \underbrace{13\text{ k}}_{\text{servicer daily operating cost}} \times \underbrace{4.51}_{\text{mission duration [days]}} = 58.6\text{ k USD}.$$

The servicer's daily operating cost is taken from Table 7.3, and the total mission duration from Table 7.8.

Delay costs (J_{delay}): Delay penalties are applied at the *start* of each service and are proportional to the time by which the service begins after its earliest admissible start (activation time step). While deterministic tasks can be scheduled in advance to begin exactly at their activation time and thus can incur no delay, random tasks are only revealed at their notice time step, which by definition coincides with their activation. Since the servicer usually requires a nonzero transfer time to reach the corresponding customer after notification, such tasks inherently accumulate a finite delay. The total delay cost accumulated over all services is given by Eq. (6.38).

In the baseline verification run:

- The deterministic refueling task at Customer 1 starts at $t = 219$, exactly matching its activation step, and therefore incurs no delay.
- The two repair tasks at Customer 2 start at $t = 390$ and $t = 395$, while their respective notice steps are $t = 6120\ min = 383\ steps$ and $t = 6272\ min = 392\ steps$. The total postponement amounts to $7 + 3 = 10$ time steps, corresponding to

$$10 \times \frac{16}{1440} = 0.11 \text{ days}.$$

Using the daily delay penalty rate for repair tasks from Table 7.6, the total delay cost is:

$$J_{\text{delay}} = \underbrace{16.67\text{ k}}_{\text{daily delay cost}} \times 0.11 = \$1{,}851.85,$$

which matches exactly the realized economic output for this verification scenario.

Net profit (J_{net}): Finally, the overall profit is computed by aggregating all components according to Eq. (6.39):

$$\begin{aligned} J_{\text{net}} &= 12{,}500{,}000 - 4{,}440{,}936 - 77{,}045{,}000 - 58{,}644 - 1{,}851.85 \\ &= -69{,}046{,}432 \text{ USD}. \end{aligned}$$

This strongly negative outcome confirms that the one-off PDM costs dominate over the limited revenues of this short verification case. The result is consistent with expectations, as the scenario is designed for model verification rather than profitability assessment. Figure 7.1 illustrates the cumulative net profit evolution over the Rolling Horizon iterations, where the infrastructure value increases discretely with each completed service.

Edge Probes (EP)

Following the baseline verification run, a series of additional edge probes is conducted to validate the robustness and logical consistency of the framework under extreme or boundary conditions. Each probe isolates a specific modeling feature, such as timing, fuel capacity, or resupply logic, and slightly perturbs one or more parameters while keeping the remaining setup identical to the baseline configuration. The purpose of these tests is not to derive operational insights but to confirm that the optimization framework reacts predictably and in full accordance with the underlying constraints when confronted with limiting scenarios.

EP1 - Tight Service Windows $\Rightarrow$ Task Dropping

Setup: To verify the correct enforcement of temporal feasibility and service window constraints, the repair tasks assigned to Customer 2 are modified to have a substantially reduced service window of only 2 time steps (32 minutes), instead of the baseline 300 steps (80 hours). All other parameters remain identical to the baseline scenario. Since a non-deterministic task can only be addressed after

its notification and requires a nonzero transfer time before execution, a delay relative to its notification time is unavoidable. This configuration therefore intentionally renders both repair tasks unreachable within their service windows given the servicer's travel time and the required orbital transfers.

Expectation: If no feasible trajectory can reach a task's location within its defined service window, the solver should *drop* the corresponding task rather than violating the timing constraints or forcing infeasible arcs. Eq. (6.13) ensures that each task can only be assigned if its start time lies within the admissible service window. Consequently, any unreachable task should appear as unassigned ($h_{vd\tau} = 0$) and generate no revenue contribution in the objective function. Given that after completing the refueling task the servicer is located in Orbit 2 and the combined transfer to Orbit 3 requires approximately 5 time steps, neither repair task can be initiated within a 2-step window after notification and should therefore be excluded from the optimal plan.

Observed: In the optimized solution, the refueling task at Customer 1 remains scheduled at its nominal activation time ($t = 219$), while both repair tasks at Customer 2 are *omitted* from the mission timeline. No violations of the service window constraint occur, and all other feasibility conditions remain satisfied. The total revenue decreases accordingly from \$12.5M in the baseline run to \$2.5M, reflecting the exclusion of the unreachable repair tasks. The resulting optimized schedule is illustrated in Figure 7.4, and the executed services are listed in Table 7.11. The corresponding economic evolution over the Rolling Horizon iterations, expressed as cumulative net profit, is shown in Figure 7.3.

Verdict: *Pass.* The solver correctly identifies that the repair tasks cannot be executed within their shortened service windows and omits them from the optimal schedule. This confirms that service window constraints are strictly enforced and that the framework does not generate infeasible or temporally inconsistent task assignments under time-limited conditions.

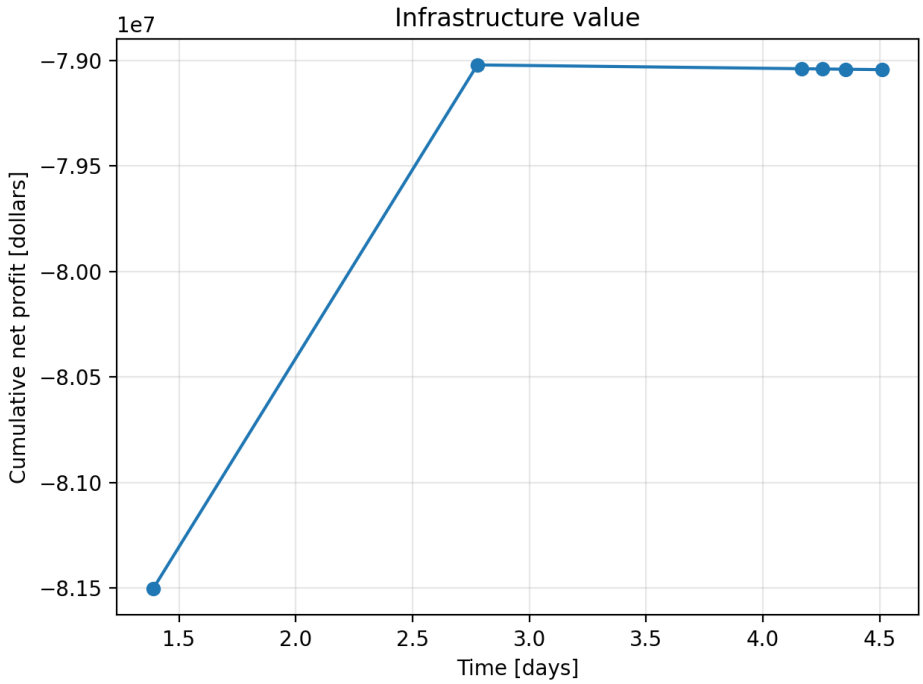


Figure 7.3: Cumulative net profit evolution over the Rolling Horizon iterations (EP1).

Table 7.11: Executed service tasks for Servicer d1 in scenario EP1, ordered by execution start time. Activation times refer to task availability, while execution times correspond to the scheduled service start.

Service	Activation [min]	Start [min]	End [min]	Delay [min]	Duration [min]
Refueling of Customer 1	3500	3504	3584	≈0	80

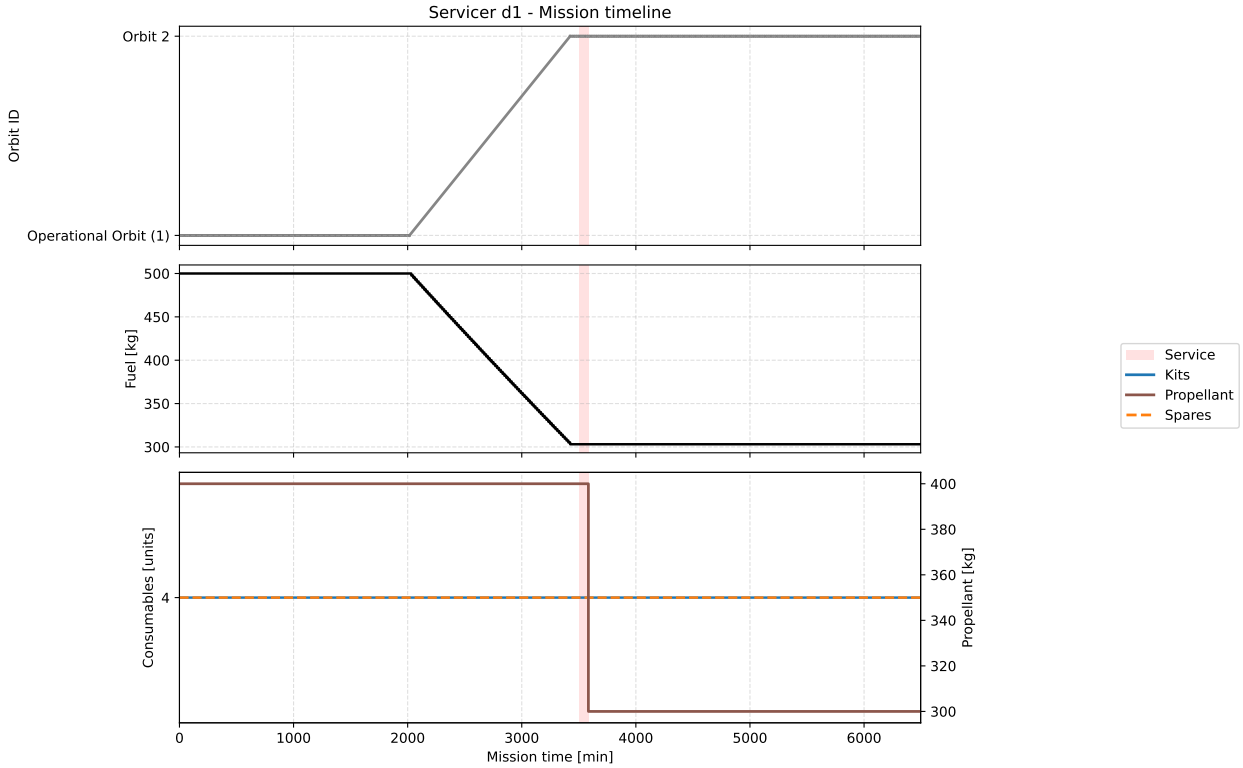


Figure 7.4: Servicer d1 - mission timeline and resource evolution for EP1 (tight service windows).

EP2 - Tool Availability $\Rightarrow$ Task Blocking

Setup: The second edge probe evaluates the correct enforcement of tool-availability constraints, ensuring that only tasks for which the servicer carries the required equipment can be executed. The same parameters as in the baseline run are retained, except that the servicer is now assumed to be launched without the tool required to perform repair operations (T3). As a result, the servicer is capable of conducting refueling, inspection, and de-orbiting tasks, but no repair activities.

Expectation: Eq. (6.23) enforces that a task can only be assigned to servicer if the required tool type is available on board. If the necessary tools are missing, the binary assignment variable $h_{vd\tau}$ must remain zero, and the task should not appear in the optimal schedule or contribute to the objective function. Since the servicer in this probe lacks the tools associated with repair services, both repair tasks at Customer 2 should be automatically excluded from the optimization, while other feasible tasks (e.g., refueling) remain unaffected.

Observed: In the optimized solution, the refueling task at Customer 1 is executed identically to the baseline and starts at its nominal activation time ($t = 219$), while both repair tasks at Customer 2 are *omitted* from the service list. All other feasibility conditions remain satisfied, and the resulting trajectory, fuel consumption, and resource evolution are the same as in EP1 (cf. Figure 7.4 and Table 7.11). The total revenue again decreases from \$12.5M in the baseline run to \$2.5M, consistent with the exclusion of the repair services. Since tool T3 is not included in this configuration, its associated launch and PDM costs are also omitted. The corresponding economic evolution over the Rolling Horizon iterations, expressed as cumulative net profit, is shown in Figure 7.5.

Verdict: *Pass.* The solver correctly prevents the assignment of repair tasks when the required tools are unavailable, validating the enforcement of the tool-availability constraint. This confirms that the framework correctly restricts feasible task allocations based on onboard equipment configuration and does not generate logically inconsistent service plans.

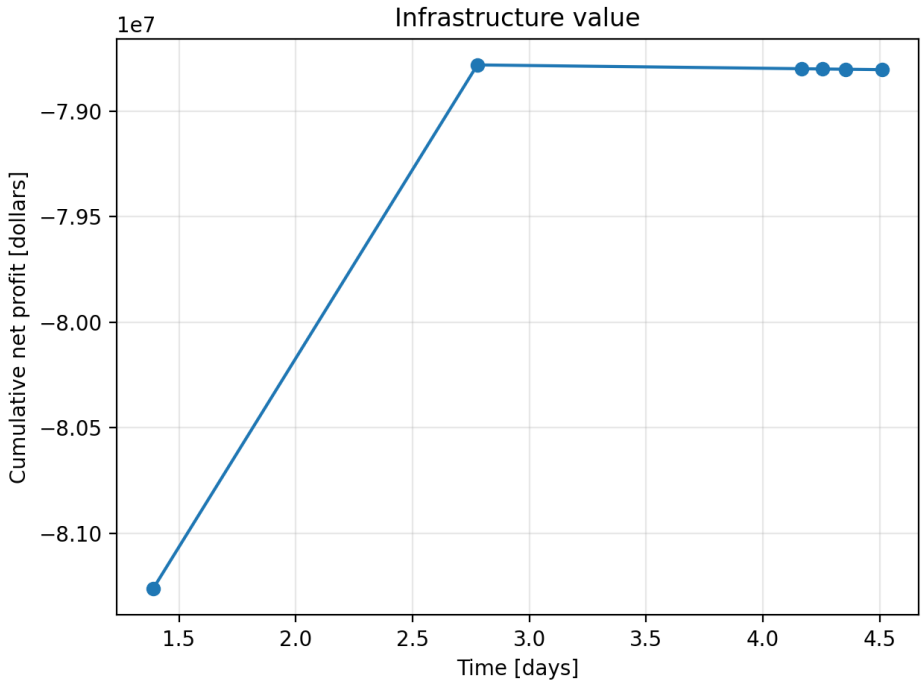


Figure 7.5: Cumulative net profit evolution over the Rolling Horizon iterations (EP2).

EP3 - Simultaneous Tasks ⇒ Optimal Selection

Setup: The third edge probe verifies the solver's decision-making logic when confronted with mutually exclusive tasks that cannot both be executed. This test examines whether the optimization correctly prioritizes the more profitable option when only one service can be performed due to temporal overlap and single-task execution constraints. All parameters remain identical to the baseline configuration, except that Customer 2 is assigned a deterministic *inspection* task instead of repair tasks. Its inter-occurrence time is set equal to that of the refueling task (3500 min according to Table 7.6), such that both deterministic tasks (refueling at Customer 1 and inspection at Customer 2) are activated simultaneously. Furthermore, their service windows are restricted to 2 time steps (32 minutes) to prevent sequential execution. Since the two services occur in different orbits at the same time, it is physically impossible for the servicer to perform both concurrently. The model must therefore decide which task to execute based solely on the objective function that maximizes total profit, with task revenue serving as the determining factor.

Expectation: When multiple feasible services overlap in time, the MILP should select the combination that yields the highest overall objective value as defined in Eq. (6.42). Given the revenue structure of Table 7.6, the solver is expected to select the refueling task at Customer 1 ($\$2.5$ M) and drop the less profitable inspection task at Customer 2 ($\$1.67$ M).

Observed: In the optimized solution, the solver assigns the servicer to perform the refueling task at Customer 1 at its activation time step ($t = 219$), while the inspection task at Customer 2 is *omitted* from the service list. No timing or feasibility violations occur, and the trajectory remains identical to that of EP1 and EP2 (cf. Figure 7.4 and Table 7.11). The total revenue equals $2.5M, consistent with the execution of the single, more profitable refueling task and the cumulative net profit evolution is the same as in EP1 (cf. Figure 7.3)

Verdict: *Pass.* The solver correctly prioritizes the higher-value deterministic task when two services are mutually exclusive in time, confirming that the objective function consistently drives optimal task selection under profit maximization. This demonstrates the coherent interaction between the task assignment logic, feasibility constraints, and the objective formulation within the MILP model.

EP4a - Limited Fuel Availability $\Rightarrow$ Fuel-Constrained Operations

Setup: The first part of this edge probe investigates the model's behavior when the servicer begins its mission with insufficient onboard propellant to visit all customer satellites and complete all deterministic service requests. The configuration mirrors the baseline verification scenario, except for a deliberate reduction in initial resources and a slight adjustment in customer assignments and task timing.

The servicer retains the same characteristics and tool configuration as in the baseline case, with a total propellant tank capacity of 500 kg and a dry mass of 1500 kg. However, its initial onboard resources are reduced as follows: the fuel load is set to 375 kg (instead of 500 kg), the number of spare parts is reduced from four to three units, and the onboard propellant dedicated to refueling services is limited to 300 kg. The number of de-orbit kits remains unchanged.

Customer 1 departs from the same orbit and node as in the baseline scenario (Orbit 2, node 9) and continues to generate deterministic `refueling` tasks with an inter-occurrence time of 3500 min, identical to the baseline configuration. Customer 2, on the other hand, is now positioned in Orbit 3 at node 14 and is assigned a deterministic `inspection` task with an inter-occurrence time of 6600 min. Each service window spans two discrete time steps, defining the temporal flexibility within which each service must be initiated. No orbital depot is included in this configuration, preventing any mid-mission resupplying opportunities.

The simulation parameters used for this test case are summarized in Table 7.12.

Table 7.12: Simulation parameters for EP4a (fuel-limited scenario).

Parameter	Value	Units
Time step size (Δt)	16	min
Scheduling horizon	7500	min
Planning horizon	3600	min
Maximum idle time	3536	min

Under these settings, the three deterministic tasks planned within the scheduling horizon are:

- `Refueling` task for Customer 1 at 3500 min = 219 steps,
- `Inspection` task for Customer 2 at 6600 min = 412 steps,
- `Refueling` task for Customer 1 at 7000 min = 438 steps.

Expectation: With only 375 kg of available fuel, the servicer does not possess sufficient propellant to execute all inter-orbital maneuvers required to service both customers and return to complete the second refueling. The optimizer is therefore expected to prioritize the most profitable sequence of tasks within the available fuel budget. Given the revenue structure in Table 7.6, refueling services at Customer 1 ($2.5M each) are more profitable than the inspection task at Customer 2 ($1.67M). Hence, the optimal strategy is expected to perform both refueling tasks at Customer 1 and omit the less profitable inspection at Customer 2.

Observed: In the optimized solution, the servicer performs both deterministic `refueling` tasks for Customer 1 precisely at their scheduled activation times, beginning at time step 219 ($t = 3504$ min) and 438 ($t = 7008$ min), each with a duration of five time steps (80 min). As expected, the `inspection` task for Customer 2 is omitted due to insufficient propellant to complete the inter-orbital transfer required to reach and return from Orbit 3. The resulting mission plan remains fully compliant with all timing, fuel-balance, and capacity constraints, and no feasibility violations are detected throughout the scheduling horizon. The total revenue amounts to $5.0M, corresponding to the successful execution of the two high-value refueling operations. The optimized mission trajectory and corresponding resource evolution are presented in Figure 7.6, while the executed services are summarized in Table 7.13.

Table 7.13: Executed service tasks for Servicer d1 in scenario EP4a, ordered by execution start time. Activation times refer to task availability, while execution times correspond to the scheduled service start.

Service	Activation [min]	Start [min]	End [min]	Delay [min]	Duration [min]
1st refueling of Customer 1	3500	3504	3584	≈0	80
2nd refueling of Customer 1	7000	7008	7088	≈0	80

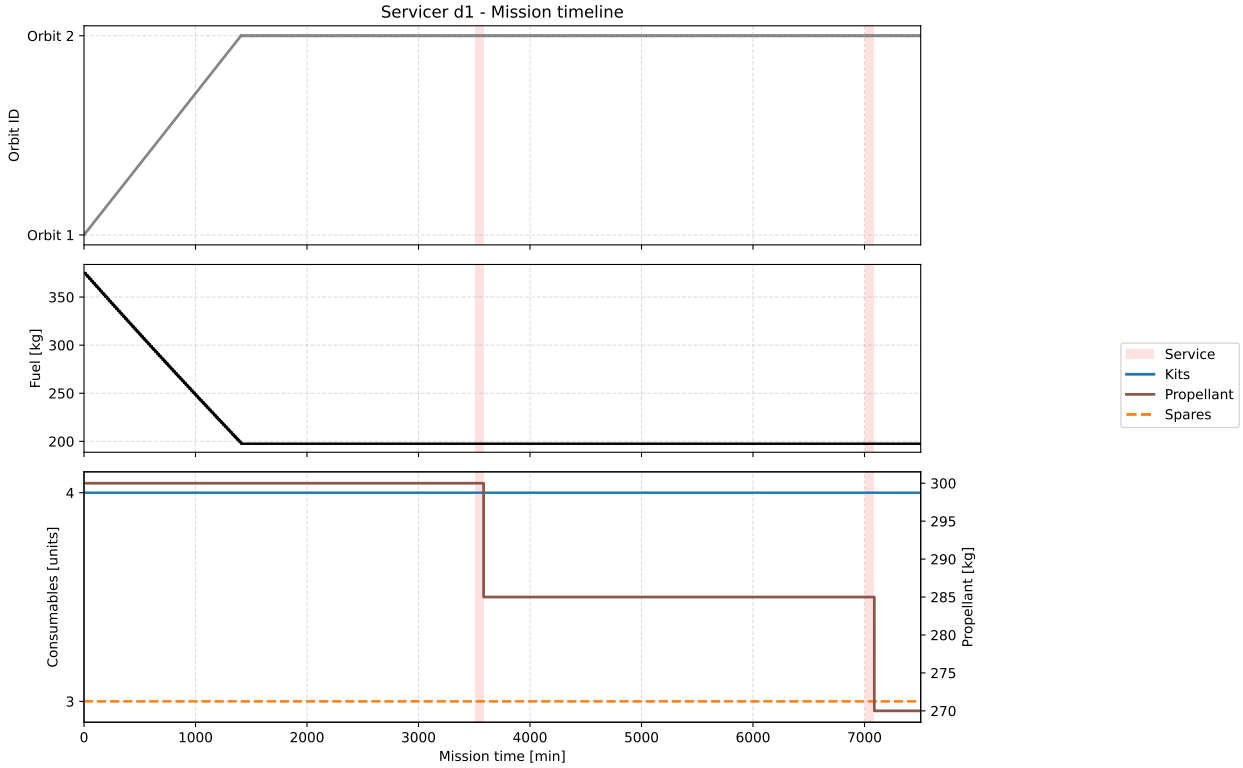


Figure 7.6: Servicer d1 - mission timeline and resource evolution for EP4a (fuel-limited scenario without depot).

Verdict: *Pass.* The solver correctly prioritizes the more profitable `refueling` services over the lower-value `inspection` task when constrained by limited fuel availability, achieving the expected mission outcome. This behavior validates that the MILP formulation consistently enforces fuel capacity and balance constraints while optimizing for profit. It also confirms the correct interaction between the economic objective, resource limitations, and task-selection logic, demonstrating that the framework behaves rationally under conditions of resource scarcity.

EP4b - Depot Introduction ⇒ Verification of Resupplying Dynamics

Setup: The second part of this edge probe extends the previous fuel-limited scenario (EP4a) by introducing an orbital depot to verify whether the optimization framework correctly identifies and exploits in-flight resupplying opportunities when available. All parameters, including simulation settings and planned tasks, remain identical to EP4a, except for the addition of one orbital depot positioned in the operational Orbit 1 at node 5.

The depot follows a purely coasting trajectory along its orbit and provides continuous resupply capability whenever the servicer traverses a corresponding resupplying arc. Depots are modeled as passive assets without propulsion or maneuvering capability and are assumed to have unlimited fuel and consumables for the purpose of this verification. The maximum quantity of propellant that can be transferred on a given resupplying arc is constrained by Eq. (6.1), where RF_d is the number of time steps required for a complete refueling. For this verification, RF_d is set to unity ($RF_d = 1$), allowing the servicer to replenish up to its full fuel capacity during any individual resupply event.

Expectation: Compared to the fuel-limited case in EP4a, the introduction of a depot is expected to

restore the servicer's ability to complete all planned tasks within the scheduling horizon. By enabling mid-mission refueling, the optimizer should schedule a rendezvous with the depot to replenish propellant and thereby recover full operational capacity. After resupply, the servicer is expected to execute the entire task sequence, two `refueling` operations for Customer 1 and one `inspection` for Customer 2, without violating fuel-balance, time, or service-window constraints. The total revenue should therefore increase from \$5.0M in EP4a, where the inspection was omitted, to \$6.67M in EP4b, reflecting successful completion of all three deterministic tasks following depot refueling.

Observed: In the optimized solution, the servicer initiates its mission by performing the first deterministic `refueling` task at Customer 1 immediately at its activation time, starting at time step 219 ($t = 3504$ min) and concluding at time step 224 ($t = 3584$ min). After completing this initial service, the servicer transfers to the orbital depot, where it performs a one-step refueling maneuver between time steps 317 ($t = 5072$ min) and 318 ($t = 5088$ min). During this event, the onboard propellant is replenished to the maximum fuel capacity of 500 kg, restoring full operational capability and enabling the continuation of subsequent mission tasks. Following the depot resupply, the servicer proceeds to execute the `inspection` task at Customer 2, beginning at time step 412 ($t = 6592$ min) and concluding at time step 417 ($t = 6672$ min). Subsequently, it performs the final `refueling` at Customer 1 from time step 438 ($t = 7008$ min) to time step 443 ($t = 7088$ min). All services are initiated precisely at their activation times, without delay, and the mission remains fully compliant with all refueling-rate, fuel-balance, and timing constraints throughout the scheduling horizon. The total revenue amounts to \$6.67M, corresponding to the successful completion of all deterministic tasks. Although the overall economic outcome remains negative mainly due to the high PDM costs, the incremental revenue relative to EP4a confirms that the optimizer effectively exploits the orbital depot to restore full mission capability. The resulting optimized trajectory and the evolution of onboard fuel and service commodities are both illustrated in Figure 7.8, which clearly highlights the intermediate depot refueling event followed by continued service execution. The full sequence of executed services and resupply events is summarized in Table 7.14, and the corresponding cumulative economic evolution over the Rolling Horizon iterations is shown in Figure 7.7.

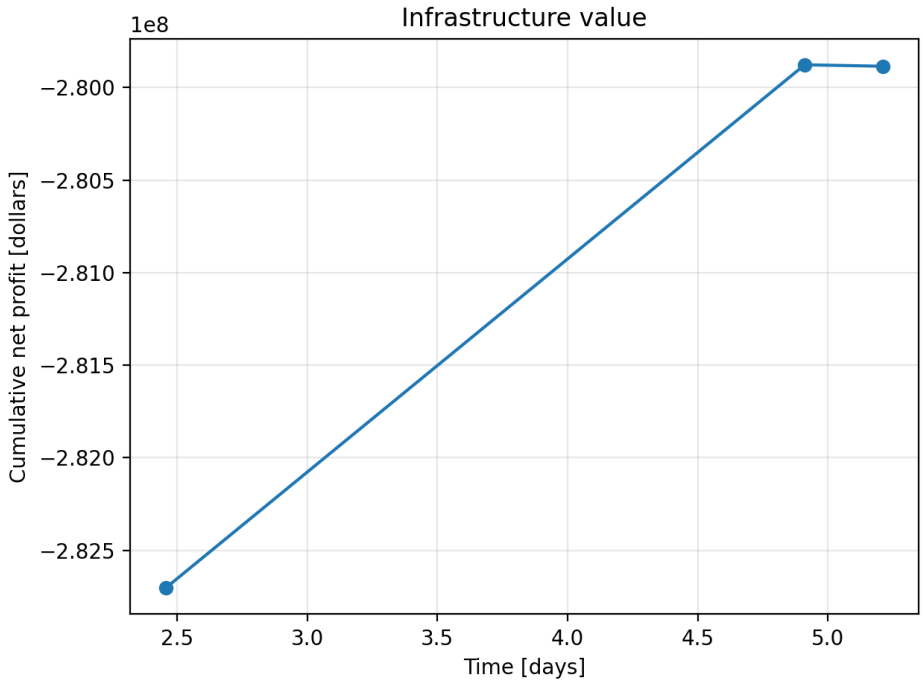


Figure 7.7: Cumulative net profit evolution over the Rolling Horizon iterations (EP4b).

Table 7.14: Executed service tasks and resupply events for Servicer d1 in scenario EP4b, ordered by execution start time. Activation times refer to task availability, while execution times correspond to the scheduled event start.

Event	Activation [min]	Start [min]	End [min]	Delay [min]	Duration [min]
1st refueling of Customer 1	3500	3504	3584	≈0	80
Resupply event	–	5072	5088	–	16
Inspection of Customer 2	6600	6592	6672	≈0	80
2nd refueling of Customer 1	7000	7008	7088	≈0	80

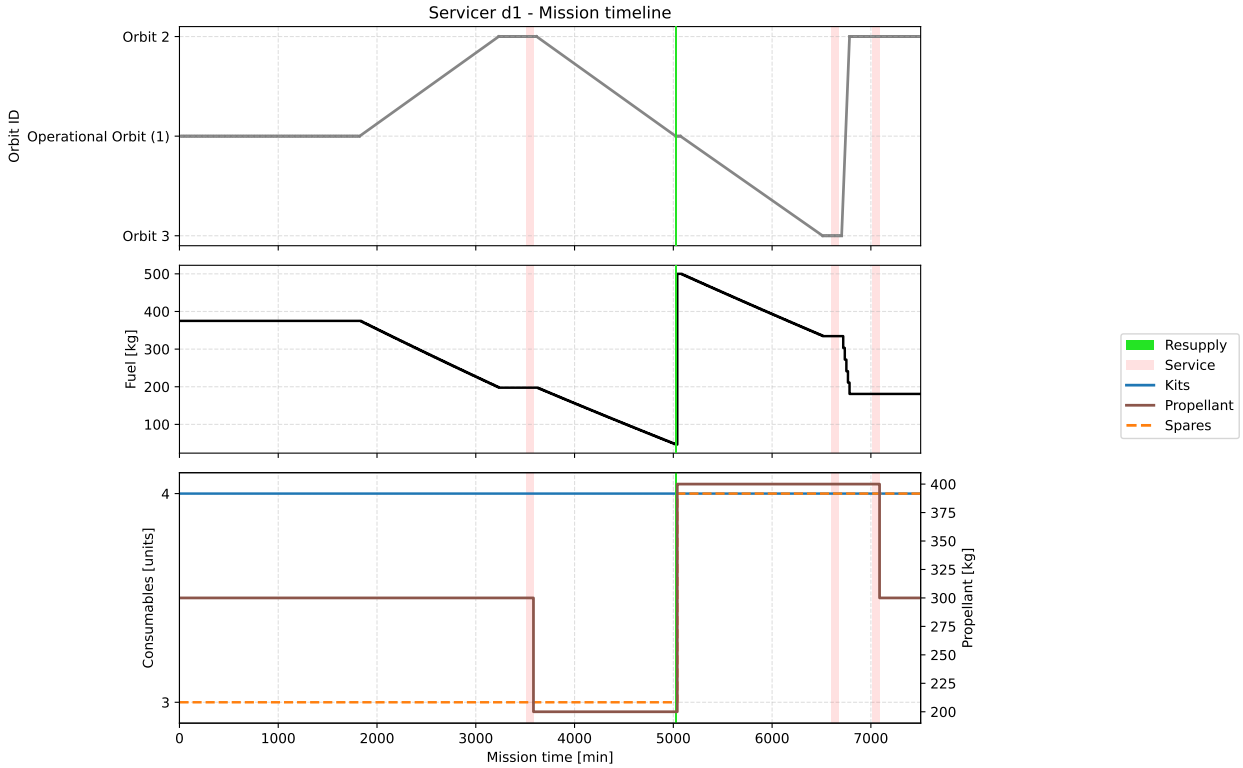


Figure 7.8: Servicer d1 - mission timeline and resource evolution for EP4b (resupply-enabled scenario with orbital depot).

Conservation of mass and resources (EP4b): The fuel trajectory shown in Figure 7.8 (middle) satisfies the fuel balance equation (6.27) and remains within the capacity bounds of Eq. (6.24) throughout the scheduling horizon. Since all other resource and mass-conservation aspects were already verified in the baseline scenario, the focus here is placed on the refueling behavior introduced through the depot interaction.

During a resupplying arc, which corresponds to a coasting transfer, no thrusting occurs and thus $\text{burn}_{dt} = 0$. Under this condition, the fuel balance simplifies to the previous fuel level plus the refueled amount,

$$f_{dt} \;=\; f_{d,t-1} \;+\; \sum_{a \in \mathcal{A}} r_{d,a,t},$$

where $r_{d,a,t}$ represents the refueling volume transferred to servicer d along arc a at time t. At the time of the depot rendezvous ($t = 317$ steps), the servicer's remaining fuel is approximately 47.8 kg. According to Eq. (6.1) and with $RF_d = 1$, a full replenishment is permitted within a single time step. Consequently, after traversing the depot arc ($t = 317 \rightarrow 318$), the fuel level is restored to its maximum tank capacity,

$$f_{d,318} \;=\; F_d^{\text{cap}} \;=\; 500 \text{ kg},$$

which corresponds to a refueling quantity of approximately 452.2 kg. This confirms that the refueling process behaves as expected.

The commodities trajectory in Figure 7.8 (bottom) verifies the stepwise inventory logic of Eqs. (6.19)-(6.22), bounded by the capacity limits in Eq. (6.16). As detailed in the baseline scenario, onboard inventories either decrease due to task-related consumption or are replenished to capacity during a resupply event. When a resupplying arc is completed, the balance condition forces

$$q_{d,m,t} = Q^{\text{cap}}_{d,m}.$$

At the time of the depot rendezvous, the onboard customer propellant amounts to 200 kg and the spares inventory to 3 units. Both quantities are replenished to their respective maximum payload capacities upon resupplying completion, consistent with the model equations and the resource trajectories shown in Figure 7.8.

Economic consistency (EP4b): The overall economic outcome of this verification scenario is evaluated according to the profit-maximizing objective function in Eq. (6.39), which aggregates all revenue and cost components over the scheduling horizon. Compared to the baseline scenario, EP4b introduces an orbital depot, thereby activating additional economic terms associated with resupplying activities and depot operations. Each term is briefly verified below.

Launch costs ($J_{L,\text{dep}}$): The total launch cost includes a component associated with the depot, accounting for both its dry mass and the initial onboard commodities as defined in Eq. (6.31). Although the depot is modeled as having unlimited resupply capability during operations, for the purpose of launch cost accounting only the actual mass of commodities used for resupplying during the mission is considered. Accordingly, the depot launch expenditure is given by:

$$J_{L,\text{dep}} = \underbrace{1410}_{\text{launch cost per kg}} \times \Big(1 \times \underbrace{3000}_{\text{depot dry mass}} + \underbrace{200 + 1 \times 86}_{\text{resupplied commodities}} + \underbrace{452.201}_{\text{refueled propellant mass}}\Big) = 5{,}270{,}863 \text{ USD}.$$

All depot and commodity parameters are taken from Tables 7.4 and 7.7, respectively. Including the servicer launch cost of $4{,}002{,}426$ USD, the total launch expenditure amounts to:

$$J_{\text{launch}} = 4{,}002{,}426 + 5{,}270{,}863 = 9{,}273{,}289 \text{ USD}.$$

Purchase, Development, and Manufacturing Costs ($J_{\text{pdm,dep}}$): The PDM cost term also encompasses the manufacturing cost of the orbital depot as well as the purchase of its initial onboard fuel and other commodities, as defined in Eq. (6.34). Given that the servicer accounts for a PDM value of $76{,}907{,}250$ USD, the cumulative PDM value of $277{,}143{,}256$ USD reported by the solver is consistent with:

$$J_{\text{pdm,dep}} = \underbrace{200{,}000{,}000}_{\text{depot manufacturing}} + \underbrace{452.2 \times 230}_{\text{fuel purchases}} + \underbrace{200 \times 230 + 1 \times 86{,}000}_{\text{commodity purchases}} = 200{,}236{,}006 \text{ USD}.$$

All depot and commodity parameters are taken from Tables 7.4 and 7.7, respectively.

Operating costs ($J_{\text{ops,dep}}$): Depot operating costs are calculated in the same manner as for the servicer, following Eq. (6.36a), using the daily operating rate specified in Table 7.4.

Net profit (J_{net}): The overall net profit, obtained by aggregating all revenue and cost components according to Eq. (6.39), is expressed as:

$$\begin{aligned} J_{\text{net}} &= \underbrace{6{,}666{,}667}_{\text{revenues}} - \underbrace{9{,}273{,}289}_{\text{launch costs}} - \underbrace{277{,}143{,}256}_{\text{PDM costs}} - \underbrace{135{,}489}_{\text{operating costs}} \\ &= -279{,}885{,}367 \text{ USD}. \end{aligned}$$

The mission therefore yields a negative profit, primarily driven by the high PDM expenses and the initial launch costs. Despite the negative outcome, the result demonstrates the internal consistency of the economic formulation, confirming that the depot-related cost terms are correctly integrated within the overall objective structure.

Verdict: *Pass.* The solver autonomously schedules the depot rendezvous at an optimal point in the mission sequence, restoring the servicer's full propellant capacity and enabling the completion of all deterministic tasks that were infeasible under the fuel-limited configuration (EP4a). This behavior confirms the correct implementation of resource-capacity and balance constraints, refueling rate limits, and the logical integration of depot interactions within the MILP formulation. Moreover, the consistent economic evolution and absence of constraint violations demonstrate that the framework robustly couples operational feasibility with profit maximization once the resupply functionality is activated.

EP5 - Multiple Servicers $\Rightarrow$ Cost-Aware Assignment and Collision Avoidance

Setup: The fifth edge probe verifies the solver's ability to (i) assign service tasks to the servicer with the lowest-cost trajectory and (ii) prevent collisions or simultaneous servicing at the same node and time step, even when multiple services are activated simultaneously at the same customer. All parameters remain identical to the baseline configuration, except for task assignments and the introduction of a second servicer ($d2$) operating in parallel with the original one ($d1$). Both servicers are identical in performance, physical characteristics, and resource capacity, but begin at different orbital locations to induce distinct transfer costs and path durations. Servicer $d1$ starts in Orbit 1 at node 1, while Servicer $d2$ begins in Orbit 3 at node 14. Both are initialized with full propellant and onboard commodities to ensure that assignment decisions depend solely on trajectory cost and temporal feasibility rather than on initial inventory levels.

To test both cost-aware selection and node-time exclusivity, Customer 1 is assigned two deterministic tasks, `refueling` and `inspection`, with identical inter-occurrence times and, thus, activation at $t = 3500$ min (219 steps). This configuration forces the optimizer to decide which servicer should perform each service and to ensure that both tasks, although co-located and simultaneous in activation, are executed without temporal overlap at the same node. Customer 2 is excluded from this test to isolate the multiple-servicer interaction dynamics.

Expectation: The optimizer should assign each service to the servicer with the lowest overall transfer cost while enforcing strict spatial and temporal exclusivity. Given the initial orbital positions, Servicer $d2$, starting at node 14, has a shorter and less fuel-intensive trajectory to Customer 1 (Orbit 2, starting from node 9) than Servicer $d1$, which would require a longer and more costly transfer from node 1. Consequently, $d2$ is expected to perform the first of the two simultaneous services and, since the service window is sufficiently wide, also the second one sequentially. Although both tasks are activated at $t = 3500$ min (219 steps), they cannot be executed concurrently, neither by the same servicer, as prohibited by constraint Eq. (6.14), nor by two different servicers at the same node and time, as constrained also by Eq. (6.15). Therefore, one service should be delayed by the duration of the first task within its 300-step service window to maintain temporal separation and prevent overlapping operations. No spatial or temporal collisions are expected, and all resource and feasibility constraints should remain satisfied.

Observed: In the optimized solution, the solver assigns both tasks at Customer 1 to Servicer $d2$, which begins in close proximity to the customer's orbit. Servicer $d2$ executes the `refueling` task first, starting at time step 219 ($t = 3504$ min) and completing it at time step 224 ($t = 3584$ min). Immediately thereafter, it performs the `inspection` task sequentially from time step 224 ($t = 3584$ min) to 229 ($t = 3664$ min). Servicer $d1$ remains idle throughout the mission, as its longer and more fuel-intensive trajectory from Orbit 1 to Orbit 2 makes it less cost-efficient. This outcome confirms that the optimizer autonomously selected the servicer with the lowest transfer cost and correctly enforced temporal separation between the two services at the same node. Furthermore, no spatial collisions occurred, and all resource-balance and feasibility constraints were satisfied. The total revenue equals \$4.17M, corresponding to the successful completion of both deterministic tasks. The optimized mission sequence and resource evolution for Servicer $d2$ are illustrated in Figure 7.9, and the executed services are summarized in Table 7.15.

Table 7.15: Executed service tasks for Servicer d2 in scenario EP5, ordered by execution start time. Activation times refer to task availability, while execution times correspond to the scheduled service start.

Service	Activation [min]	Start [min]	End [min]	Delay [min]	Duration [min]
Refueling of Customer 1	3500	3504	3584	≈0	80
Inspection of Customer 1	3500	3584	3664	84	80

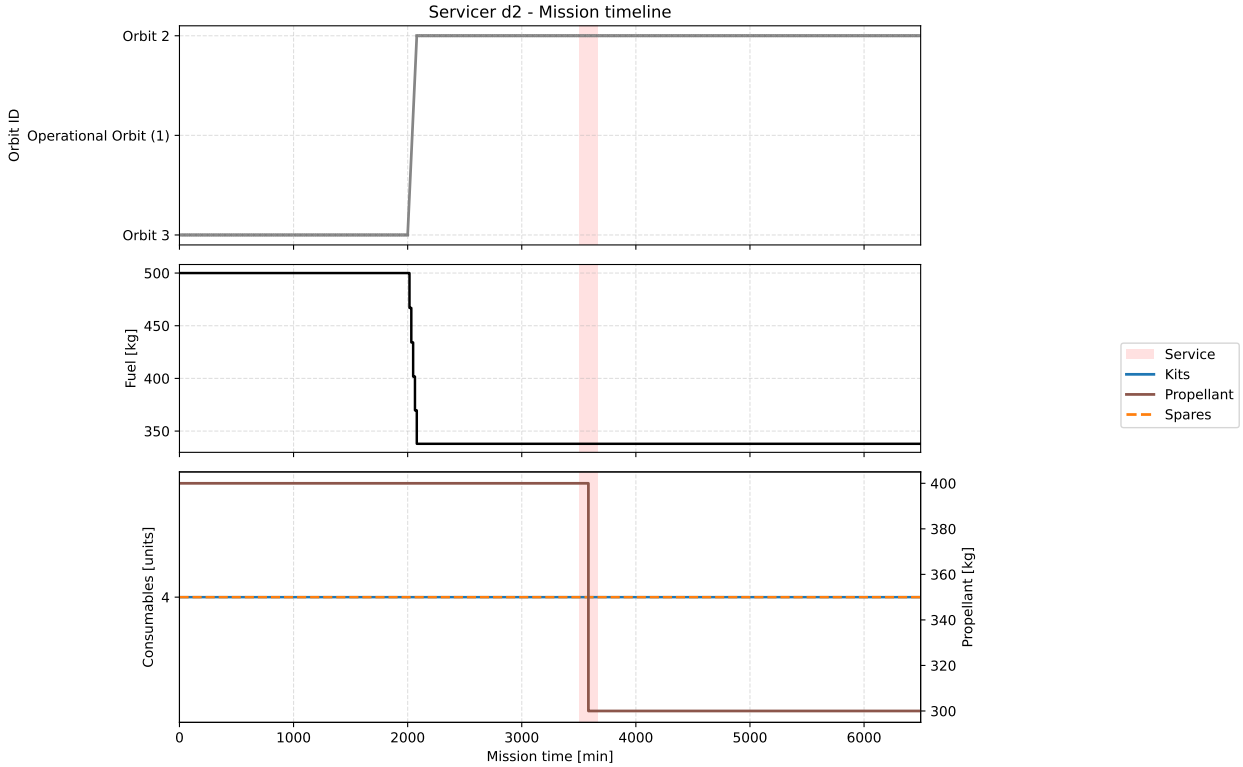


Figure 7.9: Servicer d2 - mission timeline and resource evolution for EP5 (multiple-servicer scenario).

Verdict: *Pass.* The solver correctly assigns both services to the most cost-efficient servicer ($d2$) while maintaining complete temporal separation between operations at the same node. The results demonstrate that the MILP formulation successfully integrates cost-aware task assignment and node-time exclusivity, ensuring collision-free multi-servicer coordination.

8

Case Studies

This chapter applies the verified IOS optimization framework to a series of representative case studies that demonstrate its operational and strategic capabilities. Building upon the verified model from the previous chapter, the analyses presented here explore how the framework supports decision-making across different temporal and functional dimensions of IOS logistics.

The chapter begins by introducing the standardized **case study setup**, which defines the orbital environment, system components (servicers, depots, and customer satellites), task generation processes, and the tuning of the Rolling Horizon parameters. This unified configuration ensures consistency across all subsequent simulations and enables direct comparison of results under varying operational and architectural assumptions.

Based on this configuration, the remainder of the chapter is devoted to two types of case studies:

- **Operational Case Study:** Demonstrates the framework's ability to generate effective operational decisions under uncertainty. This includes adaptive task assignment, trajectory design, resource management, and responsiveness to stochastic task arrivals in a single-servicer scenario.
- **Strategic Case Studies:** Assess the framework's ability to inform long-term architectural planning by exploring key trade-offs that shape the scalability and economic viability of IOS infrastructures. The strategic analyses investigate two complementary dimensions: (i) how the availability of zero, one, or two orbital depots affects service performance and profitability across different demand levels; (ii) how a single versatile servicer compares against a fleet of four specialized servicers, each dedicated to one service type, in markets of varying intensity. These analyses evaluate the robustness and adaptability of competing IOS architectures under evolving market conditions.

Together, these case studies transition the framework from verified functionality to applied evaluation, highlighting its capacity to support both real-time operational scheduling and high-level strategic planning for future IOS systems in Low Earth Orbit.

8.1. Case Study Setup

This section outlines the configuration adopted for all case studies, establishing the physical, operational, and computational environment in which the proposed IOS framework is applied. The setup defines the orbital network, system components, task models, and optimization parameters that together represent a realistic LEO servicing scenario. It provides the basis for generating, scheduling, and executing service operations within a consistent modeling structure.

The section is organized as follows. First, the construction of the orbital network is detailed, including the definition of orbits, nodes, and maneuver arcs that form the spatial backbone of the optimization model. Next, the system modeling assumptions for servicers, depots, and customer satellites are presented, followed by the specification of service types, task parameters, and commodity data. Finally, the Rolling Horizon parameters governing the temporal structure of the optimization are introduced.

This setup ensures consistency across all case studies while still allowing controlled variations for targeted experimental analyses.

Network Construction

Orbits and Nodes: As in the framework verification setup, the case study setup also begins by defining the orbits and nodes that form the foundation of the network. Each orbit is characterized by its corresponding orbital parameters, serving as the spatial structure on which customers and therefore tasks are distributed. A dedicated operational orbit is defined for the depots, from which the servicers are initially deployed. The remaining orbits host customer satellites, where service tasks are executed. Multiple customers may share the same orbit.

To construct the orbital environment for the case studies, a representative dataset of active SSO satellites was retrieved from Space-Track (U.S. Space Command, 2025). Eight distinct SSOs corresponding to different payloads were selected, and their orbital parameters were used to define the orbits considered in the simulations. The RAAN values were normalized to a common epoch, and all orbits were approximated as circular with zero eccentricity. For each orbit, three nodes were defined, evenly spaced at 120° intervals in argument of latitude. Thus, within a given orbit, the nodes share identical orbital parameters except for their argument of latitude, which takes the values 0°, 120° and 240°.

Given that the orbital periods of the selected orbits range from approximately 96 to 99 minutes, the simulation time step, Δt, is set to 32 minutes so that traversal between two adjacent nodes within the same orbit corresponds approximately to a single time step. Table 8.1 summarizes the orbital parameters of the selected orbits and the resulting 24 nodes considered in all case studies. The table also indicates the reference payload for each orbit and the corresponding inter-node separation in minutes, which was uniformly approximated to Δt (32 minutes).

Table 8.1: Orbital parameters for the nodes used in the case studies.

Orbit	Nodes	a (km)	i (deg)	RAAN (deg)	Based on	Period (min)	Separation (min/deg)
1	1–3	6983.606	97.8204	314.6031	TIANHUI 5A	96.8	32.3 / 120.0
2	4–6	6983.606	97.8203	314.6069	TIANHUI 5B	96.8	32.3 / 120.0
3	7–9	6981.003	97.8099	314.6351	LT-1 01B	96.7	32.2 / 120.0
4	10–12	6981.068	97.8108	314.6503	LT-1 01A	96.7	32.2 / 120.0
5	13–15	6983.668	97.8196	314.7819	TIANHUI 5C	96.8	32.3 / 120.0
6	16–18	6983.669	97.8194	314.7882	TIANHUI 5D	96.8	32.3 / 120.0
7	19–21	7073.893	98.1815	314.7941	SENTINEL 1A	98.7	32.9 / 120.0
8	22–24	7028.668	98.0069	314.8111	HAIYANG 4A	97.7	32.6 / 120.0

Phasing Maneuver Parameter Tuning: The next step involves tuning the maneuver parameters required for constructing the network arcs. Beginning with the phasing maneuvers, and consequently the phasing arcs, the key parameter to calibrate is N_{max}, which defines the maximum number of revolutions allowed for a phasing transfer. Two criteria are considered in this tuning process: (i) the differential RAAN drift occurring during a phasing maneuver relative to the target, and (ii) the diminishing ΔV benefit obtained by increasing N.

During a multi-revolution phasing maneuver, a *RAAN offset* accumulates between the servicer and the customer satellite due to the mismatch in their J_2-induced RAAN drift rates. As discussed in Chapter 4, this framework neglects RAAN drift during phasing for simplicity; consequently, any RAAN offset generated during these maneuvers is neither corrected nor explicitly accounted for afterwards. The underlying assumption is that, as long as the accumulated RAAN error remains sufficiently small, it can later be corrected during the rendezvous stage, an element not modeled here but compatible with future extensions of the framework. To ensure this, the maximum number of phasing revolutions is capped so that the resulting differential RAAN separation remains within an acceptable tolerance. For the case studies considered, this tolerance is set to $\Delta\Omega_{\text{tol}} = 0.095°$.

In parallel, the trade-off between decreasing ΔV and increasing maneuver duration with higher revolution counts is analyzed. Beyond a certain point, extending the phasing duration yields marginal ΔV improvements while unnecessarily enlarging the solution space for the optimizer. To avoid this, a minimum relative-benefit threshold is imposed: an additional revolution is accepted only if it yields at least a 20% reduction in ΔV compared to the previous option, thereby preventing unnecessary growth of the solution space and keeping the optimization computationally tractable. Figure 8.1 illustrates this trend for a representative phasing case (Node 1 $\rightarrow$ Node 3). As the number of phasing revolutions increases,

the required ΔV decreases rapidly at first and then gradually flattens, with the relative improvement falling below 10% after about eleven revolutions, indicating diminishing returns for higher values of N.

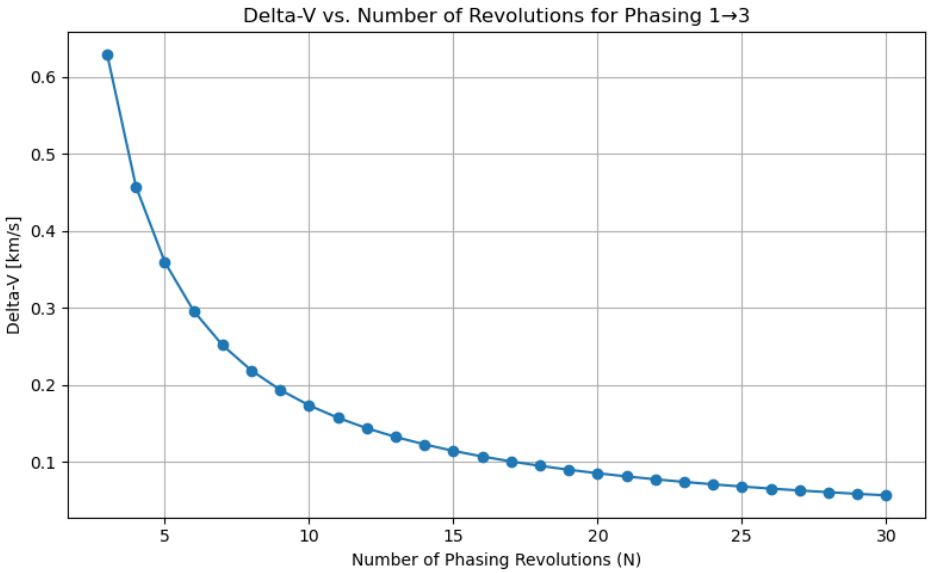


Figure 8.1: ΔV as a function of the number of phasing revolutions for the transfer between Node 1 and Node 3.

The thresholds on differential RAAN drift and ΔV flattening are evaluated for every node pair where a phasing maneuver is feasible, resulting in a specific upper bound $N_{\text{max}}^{(i,j)}$ for each pair of nodes. Because these limits vary across orbits, a single global N_{max} must be selected for network construction. To ensure full feasibility and consistency across all orbits, the most conservative approach is adopted: the global N_{max} is set equal to the minimum value among all pairwise limits,

$$N_{\text{max}} = \min_{(i,j)} N_{\text{max}}^{(i,j)},$$

thus guaranteeing that both criteria are satisfied for every possible phasing maneuver in the network. After applying this tuning to the node set defined in Table 8.1, the resulting global value is $N_{\text{max}} = 5$, which is used throughout all case studies. In practice, this tuning limits phasing maneuvers to short, fuel-efficient transfers while preventing excessive RAAN drift and unnecessary enlargement of the optimization problem.

Combined Maneuver Parameter Tuning: The tuning of the combined maneuver parameters involves defining two key quantities: the RAAN difference threshold $\Delta\Omega_{\text{TH}}$ and the maximum drift duration T_{max}. The process begins by analyzing the distribution of RAAN separations $\Delta\Omega$ across all orbit pairs extracted from the mission dataset. The RAAN separations are normalized to the interval $[0, 180°]$, ensuring consistent treatment of directionality and magnitude, after which a percentile choice is applied to define the threshold $\Delta\Omega_{\text{TH}}$. This threshold controls how often the full drift capacity is used: for pairs with $\Delta\Omega > \Delta\Omega_{\text{TH}}$, the maneuver duration is set to the maximum allowed time T_{max}, while for smaller RAAN differences, the allocated drift time scales linearly with the ratio $\Delta\Omega/\Delta\Omega_{\text{TH}}$. The percentile choice therefore dictates the proportion of orbit pairs that fully exploit the drift capacity (e.g., percentile $= 0.70$ results in approximately 30% of pairs using the full drift duration, representing a balanced compromise between maneuver duration and fuel efficiency).

Once the threshold is fixed, the algorithm evaluates all possible orbit pairs to determine the minimum physically feasible drift time required to close the RAAN gap within the allowed altitude window for operational LEO orbits, here defined between 400 km and 2000 km. For each pair, the altitude giving the maximum differential drift in the appropriate direction is selected, and the corresponding closure time is determined according to the threshold rule, ensuring that the RAAN gap can be closed within the final time allocated to that pair. The global drift-time cap T_{max} is then set equal to the largest of

these per-pair minimum times, ensuring that all RAAN separations can be closed within the selected threshold and altitude constraints.

This approach guarantees physical feasibility across all orbit pairs while allowing the user to balance maneuver efficiency and duration through the choice of $\Delta\Omega_{\mathsf{TH}}$. Lower thresholds lead to longer, more fuel-efficient drifts characterized by gentle differential precession rates, whereas higher thresholds yield shorter, faster drifts with stronger differential precession requiring higher ΔV. After applying this tuning to the node set defined in Table 8.1, the resulting values for the combined maneuver parameters are $\Delta\Omega_{\mathsf{TH}} = 0.175^\circ$ and $T_{\mathsf{max}} = 1.9297$ days, which are adopted for all subsequent case studies.

For clarity, Table 8.2 summarizes the key numerical parameters governing the network construction and maneuver modeling that are fixed across all case studies. These values result directly from the tuning procedures described above and define the temporal resolution, maneuver feasibility bounds, and RAAN-drift control assumptions adopted throughout the chapter.

Table 8.2: Key network and maneuver parameters used in all case studies.

Parameter	Value	Role in the framework
Time step Δt	32 min	Temporal discretization of the network; aligned with inter-node spacing along each orbit.
Maximum phasing revolutions N_{max}	5	Upper bound on phasing maneuver duration, limiting differential RAAN drift and controlling network size.
RAAN drift tolerance during phasing $\Delta\Omega_{\mathsf{tol}}$	0.095°	Maximum admissible RAAN offset accumulated during phasing maneuvers.
RAAN difference threshold $\Delta\Omega_{\mathsf{TH}}$	0.175°	Threshold separating partial and full-duration RAAN drift maneuvers.
Maximum drift duration T_{max}	1.93 days	Upper bound on RAAN drift time ensuring physical feasibility across all orbit pairs.

Using the calibrated maneuver parameters and a sufficiently large planning horizon such that no arc exceeds its duration, the resulting network comprises 200 arcs in total: 24 coasting maneuvers, 56 combined maneuvers, and 120 phasing maneuvers. A summary of the generated arcs is provided in Table 1 in the Appendix. The column “Count” indicates the number of arcs in each category, while the final three columns report the minimum, mean, and maximum values for the corresponding parameters.

System Modeling and Assumptions

This section defines the system model and parameter set used throughout the case studies. The objective is to establish a generalized, reproducible IOS mission model suitable for LEO that captures the architecture (servicers, orbital depots), customer tasks (deterministic and stochastic), consumables and tools, and the economic assumptions used in the optimization.

Servicer Modeling: The parameters for **servicer sizing** are derived from M. A. Luu and Hastings (2022). In that study, the authors compare mass data from previous IOS missions and select a dry mass of 1,000 kg, which represents a relatively low value. This reduction is justified by the introduction of the pod concept, which decreases the inert mass that must be carried throughout the mission. Following the same rationale, since the inclusion of an orbital depot concept similarly reduces inert mass requirements, the present work adopts the same baseline for a *specialized* servicer class, which carries only the tools necessary to perform a single service type. In addition, a *versatile* servicer class is defined, capable of executing all service types and thus representing a larger, multi-purpose vehicle. Its dry mass is scaled up by 50% to 1,500 kg, following the same proportional increase applied in Sarton du Jonchay et al. (2021) to reflect expanded capability and onboard equipment.

Assuming a typical propellant fraction of approximately 50% of the wet mass, consistent with operational satellite statistics, and adopting wet masses comparable to the OSAM-1 (On-orbit Servicing, Assembly, and Manufacturing-1) mission (around 4,000 kg), the *specialized* servicer is defined with a total wet mass of 3,000 kg, while the *versatile* variant is scaled to 4,500 kg. These values correspond to

propellant capacities of 1,500 kg and 2,250 kg, respectively. The remaining mass budget is allocated to payload other than fuel, amounting to approximately 500 kg for the specialized and 750 kg for the versatile servicer. Given the integration of orbital depots within the architecture, only a limited payload mass must be carried onboard, as essential consumables can be resupplied when needed. For the specialized servicers, the available payload capacity is fully dedicated to the commodity and tool(s) required for their specific service type. In contrast, the versatile servicers allocate part of their payload capacity to the integrated tools, with the remaining volume distributed among spare parts, de-orbit modules, and customer propellant.

For chemical propulsion, a monopropellant system with a **specific impulse** of 230 s is selected, representing a mid-range value within the 200-235 s interval reported by M. A. Luu and Hastings (2022) for monopropellant systems. **Servicer PDM costs** are estimated using the cost formulation proposed by M. A. Luu and Hastings (2022), which relates the total cost to the servicer dry mass, since the cost of other components are accounted for separately, through a scaling factor of 5.07×10^{-2}M$\,/\,$kg. **Operating costs** are modeled as a fixed daily expense primarily driven by ground-crew salaries. As these costs are largely independent of orbital altitude and servicer size, identical values are assumed for both servicer types, following the GEO-based example of Sarton du Jonchay et al. (2021).

Fuel **resupply operations** at orbital depots are modeled as requiring a single time step for a complete refueling ($RF_d = 1$), enabling the servicer to replenish its full propellant capacity within a single resupply event. This simplification is justified by the relatively short duration of propellant transfer compared with the timescales of orbital transfers and the overall mission timeline, particularly once IOS operations become routine. Nevertheless, the framework remains flexible and can accommodate a specified refueling duration if a more detailed and realistic modeling approach is required in future analyses.

Table 8.3 summarizes the final servicer parameters adopted for the case studies, providing examples for the different types of *specialized* servicers and for a *versatile* servicer equipped for all service types.

Table 8.3: Servicer parameters used in the case studies.

Servicer type	Tools	Dry mass [kg]	Fuel cap. [kg]	Impulse [s]	PDM cost [M$]	Op. cost [$/day]	Cap. spares	Cap. prop. [kg]	Cap. d. modules
Specialized - Refueling	T1	1,000	1,500	230	50.70	13,000	0	400	0
Specialized - Inspection	T2	1,000	1,500	230	50.70	13,000	0	0	0
Specialized - Repair	T3	1,000	1,500	230	50.70	13,000	5	0	0
Specialized - ADR	T3,T4	1,000	1,500	230	50.70	13,000	0	0	10
Versatile	T1-T4	1,500	2,250	230	76.05	13,000	3	88	3

Depot Modeling: Regarding the parameters adopted for depot modeling, the operating costs are again represented as a fixed daily expense primarily driven by ground-crew salaries and are taken from Sarton du Jonchay et al. (2021). The depot PDM cost is also derived from the same source. Considering that the GEO mission analyzed in that study involved a larger, heavier depot designed to support longer mission durations and a greater number of serviced customers, the present work scales this value down by 75%, resulting in an estimated PDM cost of approximately 50M$. Consistent with the assumptions in Choi and Ho (2025), the depot dry mass is set to 1,500 kg.

The parameters associated with depot modeling are summarized in Table 8.4.

Table 8.4: Depot parameters used in the case studies.

Parameter	Value	Units
Depot operating cost	$13K	USD/day
Depot development & manufacturing cost	$50M	USD
Depot dry mass	1,500	kg

Customer and Task Modeling: The modeling of customer satellites is based on the Iridium NEXT constellation, as it represents a well-documented example of a large commercial LEO constellation with publicly available data on spacecraft design, mass, and propellant budget. These satellites are medium-sized, with a launch mass of approximately 860 kg each, making them representative of average LEO targets for servicing operations (Clark, 2016). Within the proposed framework, each customer

satellite can be assigned the modeled deterministic service types, refueling and inspection, and can also trigger random services, i.e., repair or ADR.

The **deterministic task generation** follows the servicing logic of the constellation. Since the Iridium NEXT satellites are designed for an operational lifetime of approximately 15 years (Clark, 2016), the **refueling service** should have an inter-occurrence time equal to this lifetime, resulting in a single refueling event per refueling customer within each mission horizon. The servicing campaign is assumed to be scheduled such that both the refueling service window and the nominal end-of-life of the constellation fall within the simulated time-frame; in practice, the inter-occurrence time is therefore chosen sufficiently large to ensure this requirement.

Inspection tasks are modeled as recurring deterministic events on a yearly basis to support preventive maintenance and early anomaly detection. As with refueling, a long inter-occurrence time ensures that each inspection task is activated only once per inspection customer during the simulation.

To avoid unrealistic synchronization, where all deterministic tasks of a given type would otherwise activate simultaneously across all eligible customers, the framework introduces a systematic temporal offset to each deterministic activation time. The offset is defined as:

$$\text{customer\_offset} = \text{start\_offset} + (\text{customer\_id} - 1) \times \text{offset\_step},$$

such that the activation of deterministic tasks is staggered across customers, with the offset increasing by a fixed increment for each successive customer. The values of *offset_step* and *start_offset* are selected individually for each case study.

This mechanism generates phased yet deterministic activation times across the fleet, ensuring that refueling and inspection tasks do not occur simultaneously for all customers. It preserves the single-occurrence nature of these tasks within the mission horizon while introducing temporal diversity that more accurately reflects realistic servicing operations.

Stochastic, or random services, repair and ADR, are modeled based on empirical failure data from the first-generation Iridium constellation. That constellation consisted of 95 satellites with a nominal design lifetime of 8 years, out of which 30 satellites experienced failures and remained in orbit, leading to an overall failure rate of approximately 32% (M. A. Luu & Hastings, 2022). Of these failed satellites, it is assumed that 50% would be repairable, while the remaining 50% would require de-orbiting via ADR. These proportions are used to derive mean inter-occurrence times for stochastic task generation in the framework.

At the fleet level, the 95-satellite constellation experienced 30 failures over 8 years, corresponding to a failure rate of $\lambda_{\text{all}} = 3.75\ \text{yr}^{-1}$. This yields a mean inter-occurrence time of approximately 0.2667 years (97.4 days). When split by service type, this corresponds to $\lambda_{\text{repair/ADR}} = 1.875\ \text{yr}^{-1}$, or a mean inter-occurrence time of 0.533 years (194.8 days).

At the per-satellite level, dividing these rates by 95 yields $\lambda_{\text{all,sat}} = 0.0395\ \text{yr}^{-1}$, $\lambda_{\text{repair,sat}} = \lambda_{\text{ADR,sat}} = 0.01974\ \text{yr}^{-1}$. The corresponding mean inter-occurrence times per satellite are therefore:

- **Repair:** 50.66 years ($\approx 26{,}626{,}900$ min),
- **ADR:** 50.66 years ($\approx 26{,}626{,}900$ min).

These inter-occurrence times are directly implemented as parameters for the Poisson process governing stochastic task generation in the case studies, ensuring a realistic representation of random failure-driven demand within the LEO servicing environment. To maintain realism, once an ADR task is assigned to a customer satellite, no further tasks are generated for that satellite, as it needs to be de-orbited and therefore removed from the active constellation.

Building upon these assumptions, the **remaining task parameters** used in the case studies are defined to ensure physical and economic consistency with real in-orbit servicing operations. The ***revenue*** and ***delay penalty*** values for GEO satellite servicing are taken from Sarton du Jonchay et al. (2021), and subsequently scaled for LEO operations according to the ratio between GEO and LEO customer satellite costs. For GEO-class satellites, a typical cost of approximately $200 million per unit (Verstraete et al., 2018) was used as the baseline, while for LEO commercial constellations, the Iridium NEXT satellites were adopted as reference, with an estimated production and launch cost of $33.2 million per satellite (Clark, 2016). This yields a scaling factor of roughly one-sixth between GEO and LEO

service revenues and penalties. The *ADR* service is modeled scaling the revenue and delay parameters of the *retirement* task defined in the GEO study, as both represent similar end-of-life disposal operations.

The ***service duration*** of each operation was informed by the ADRIOS ClearSpace-1 mission concept, which provides a representative timeline for in-orbit servicing activities. According to these estimates, inspection operations require on the order of two weeks, whereas more complex activities, such as refueling, repair, or debris removal, take roughly five weeks, including the closing, proximity operations, and capture or docking phases. The actual hands-on servicing time is assumed comparatively small and thus neglected in this approximation (Woicke et al., 2025). However, adopting such multi-week durations directly would make the optimization problem computationally intractable. To preserve solvability while retaining the relative ordering of task complexities, the service durations are uniformly scaled down: inspection tasks are modeled as lasting two time steps, whereas refueling, repair, and debris removal tasks last five time steps.

The ***service window*** for each task was assumed to be predefined through agreements with customers, providing flexibility for scheduling within operational planning horizons.

The ***service consumables*** are defined according to the nature of each operation: one undifferentiated spare part is required for repair services, one de-orbit module/kit for ADR, no consumables for inspection, and a specified amount of propellant for refueling operations. The propellant mass allocated to refueling is based on the typical fuel budget of Iridium-class satellites, as reported in Iridium Constellation LLC (2013a). Since approximately 5.6 kg of propellant are originally assigned to station-keeping and collision-avoidance maneuvers, an equivalent quantity is required to renew the nominal operational lifetime. An additional margin is included to restore residuals and any previously consumed de-orbit fuel, resulting in a total refueling quantity of about 15 kg.

The parameters associated with each task type are summarized in Table 8.5. It should be noted that the listed inter-occurrence times are defined on a per-satellite basis; when extended to an entire constellation, the aggregate frequency of task occurrences increases proportionally to the number of satellites.

Table 8.5: Task parameters for case studies. Service duration and service window are expressed in time steps (Δt) and inter-occurrence time in minutes.

Type	Task	Revenue	Delay penalty	Duration	Service window	(Mean) Inter-occurrence time	Commodities
Deterministic	Inspection	$1.67M	$0.83K	2	215	Varies w/ SH	—
Deterministic	Refueling	$2.50M	$16.67K	5	215	Varies w/ SH	prop.: 15 kg
Random	Repair	$5.00M	$16.67K	5	215	26626900	spares: 1 unit
Random	ADR	$1.67M	—	5	215	26626900	de-orbit kit: 1 unit

Commodity Modeling: Finally, the different commodities modeled in the framework require the definition of their respective masses and costs. Starting with the tools carried by the servicers to perform the various service types, these are modeled following Sarton du Jonchay et al. (2021), where tools T1-T4 (cf. 2.2.1) are treated as generic equipment with an assumed mass of 100 kg and a unit cost of $100,000 for optimization purposes, rather than as GEO-specific hardware. This generic representation is retained for the LEO case studies presented in this work, for which the same parameter values are adopted.

Regarding propellant, a distinction is made between servicer and customer fuel in terms of their purpose and utilization. Nevertheless, both are modeled as monopropellant hydrazine, consistent with M. A. Luu and Hastings (2022) for servicers and with Iridium Constellation LLC (2013b) for customer satellites. Accordingly, the same unit cost of $230 per kilogram is applied to both, following Sarton du Jonchay et al. (2021).

For spare parts, given that a modeled Iridium NEXT customer satellite has a launch mass of approximately 860 kg, a complete replacement would correspond to 100% of this mass, whereas repair activities are expected to involve only a fraction of it. Accordingly, the spare mass required for repair operations is assumed to represent 10% of the spacecraft mass, resulting in an estimated 86 kg of generic spare material per repair operation (M. A. Luu & Hastings, 2022). This assumption ensures consistency in the modeling of repair consumables within the framework, while allowing for future refinement through more detailed component-level modeling. The manufacturing cost of spare parts is set at $1,000 per kilogram, following Sarton du Jonchay et al. (2021).

Finally, for de-orbit systems, the modeling is based on the drag-sail module configuration described by Shukla and Sawant (2024). The reference system (AirDragMod) was developed for small satellites and is used here as a representative design concept. Each unit is assumed to have a mass of approximately 1.4 kg and an estimated cost of $11,000.

The parameters associated with the modeled commodities are summarized in Table 8.6. Additionally, the launch cost is set to $1,410 per kilogram, consistent with a Falcon 9 heavy LEO launch ride share as reported by M. A. Luu and Hastings (2022).

Table 8.6: Commodity purchase costs and masses for case studies.

Commodity	Purchase cost	Unit	Mass per unit [kg]
Spares	$86K	per unit	86
Propellant	$230	per kg	1
De-orbit kit	$11K	per unit	1.4
T1	$100K	per unit	100
T2	$100K	per unit	100
T3	$100K	per unit	100
T4	$100K	per unit	100

Propagation of the Iridium-Based Assumptions: The adoption of the Iridium NEXT constellation as a reference for customer modeling serves as a unifying baseline for the definition of task demand, physical requirements, and economic parameters in the case studies. In particular, this choice propagates into the assumed inter-occurrence times of service tasks, the scaling of service revenues and delay penalties, the propellant quantities required for refueling operations, the propellant type used by customer spacecraft, and the mass and cost of consumables such as spare parts and de-orbit kits. These parameters are derived from publicly available Iridium-related data where possible and complemented by simplifying assumptions when detailed information is unavailable, with the primary objective of constructing a coherent and internally consistent task scenario rather than reproducing exact operational values.

As a consequence, the numerical results obtained in the following case studies, such as absolute profit levels, task completion counts, or break-even points for infrastructure elements, are inherently sensitive to this parameterization and should be interpreted as scenario-dependent. However, the intent of the case studies is not to provide globally valid economic estimates for IOS operations, but to demonstrate the capabilities of the proposed optimization framework when applied to a realistic constellation-scale servicing problem. The qualitative behaviors observed, including the interaction between stochastic task arrivals and Rolling Horizon replanning, the impact of depot availability on feasibility and resource utilization, and the trade-offs between servicer flexibility and specialization, are expected to remain valid under alternative customer models, provided that consistent physical and economic inputs are supplied. In a real mission design context, the framework is therefore intended to be instantiated with operator-specific data, replacing the Iridium-based approximations used here for demonstration purposes.

Rolling Horizon Parameterization

To support the Rolling Horizon optimization framework, the main simulation parameters, *Scheduling Horizon*, *Planning Horizon*, and *Maximum Idle Time*, are tuned for each case study as follows.

The *Scheduling Horizon* defines the overall mission duration, corresponding to the intended time span of the servicing campaign.

The *Planning Horizon* defines the “look-ahead” window of each optimization cycle and must be sufficiently long to capture key downstream interactions. In particular, it must span at least one full refueling cycle, including the downstream benefits it enables, such as additional services that become feasible after refueling. Otherwise, the optimizer would overlook profitable refueling-service sequences. Accordingly, the Planning Horizon is set to include the longest possible transfer to a depot (i.e., the sum of the longest combined and phasing maneuver durations), one time step for refueling, the same return duration, and an additional buffer to allow further service opportunities to enter the window. Since transfers to customers are already included within refueling cycles and any service that extends beyond the

horizon is forced to continue in the next iteration through the boundary constraints (cf. 6.3.5), this configuration ensures that all operationally relevant chains are fully represented within each optimization cycle.

Unlike conventional implementations that use a fixed *Control Horizon*, this framework adopts a *Maximum Idle Time* parameter. It defines the longest interval the model is allowed to operate without re-optimization and conceptually should reflect the natural frequency at which new information becomes available, such as the arrival of new service requests or task updates.

A shorter Maximum Idle Time results in more frequent replanning, improving responsiveness and adaptability but increasing the computational burden. Conversely, a longer value enhances stability but reduces responsiveness.

Therefore, the Maximum Idle Time must balance these competing considerations. In this work, it is linked to the cadence of deterministic task generation, since the introduction of a new deterministic task can modify the optimal servicing sequence and influence downstream planning decisions. Furthermore, because the framework already triggers immediate re-optimization whenever a random (stochastic) task is activated, the Maximum Idle Time effectively governs only those periods in which no unexpected events occur.

Accordingly, the offset between deterministic task activations is used as the basis for selecting the Maximum Idle Time. This choice ensures that replanning occurs on a timescale consistent with the rate at which new deterministic information enters the system, while avoiding unnecessary computational overhead.

Fuel Safeguard Parameter Tuning: In addition to the temporal parameters governing the Rolling Horizon framework, the case studies adopt a fixed tuning for the fuel safeguard mechanism introduced and discussed in detail in Chapter 6, Section 6.4.7. As formulated there, the fuel safeguard is implemented as a soft penalty in the objective function that becomes active when a servicer's propellant level drops below a predefined threshold, discouraging critically low end-of-horizon fuel states without imposing a hard feasibility constraint.

For all case studies, the fuel threshold is set to 500 kg, representing a conservative reserve level relative to typical maneuvering requirements and ensuring that servicers retain sufficient capability to reach a depot and perform a refueling operation in subsequent Rolling Horizon iterations. The associated penalty weight is fixed at $\lambda_{\text{fuel}} = 10^4$, selected in relation to the economic structure of the task catalog and the objective function implemented in the MILP. With this value, small violations of the fuel threshold remain economically acceptable when offset by sufficiently high service revenues, allowing the optimizer to prioritize profitable tasks in the short term. As the magnitude of the violation increases, however, the cumulative fuel penalty rapidly outweighs both delay penalties and marginal service revenues.

This behavior biases the optimizer toward scheduling depot refueling maneuvers or postponing services before the servicer becomes critically under-fueled. In particular, the tuning strongly disfavors solutions that would leave the servicer with insufficient fuel to reach a depot and refuel, thereby preventing the execution of any further services in subsequent planning horizons.

This tuning was found to yield stable Rolling Horizon behavior across all tested scenarios, preventing horizon-induced fuel depletion while avoiding systematic over-refueling. Since the safeguard is one-sided and does not reward fuel levels above the threshold, it does not introduce incentives for excess propellant retention. The same parameter values are used consistently throughout all operational and strategic case studies to ensure comparability of results.

8.2. Operational Routing and Scheduling Under Uncertainty

This first case study evaluates the framework's ability to support operational decision-making under uncertainty. The objective is to demonstrate how the proposed Rolling Horizon optimization process enables an IOS operator to dynamically route and schedule a servicer in response to both deterministic service needs and stochastic failure-driven requests.

The operational setting reflects a realistic scenario in which an IOS operator must deliver a set of pre-agreed deterministic services, such as refueling operations contracted in advance by satellite operators, while simultaneously remaining prepared to address unpredictable failures across a large customer base. When such stochastic events occur and repair or active debris removal tasks are

needed, the IOS operator must promptly adjust the mission plan to accommodate the new demand. In practice, an IOS operator would initialize the optimization tool over a single Planning Horizon using only the set of known deterministic service requests. As stochastic service requests arise during execution, the mission plan must be re-optimized using the current orbital state of the servicing infrastructure as the updated initial condition, while augmenting the demand set with the newly activated tasks. This case study illustrates and validates the framework's ability to support such an operational workflow by continuously replanning through successive Rolling Horizon iterations, thereby capturing both the uncertainty of service arrivals and the need for timely, adaptive decision-making.

It is assumed that all servicing infrastructure elements are initially deployed in the operational orbit (Orbit 1), where depots will stay and operate. Servicers depart from this orbit to execute their assigned tasks across the customer orbits. Nonetheless, the framework is fully flexible and allows users to specify any initial state of the IOS infrastructure.

In this operational case study, a single *versatile* servicer is simulated across multiple planning horizons. Equipped with all tools required to perform any service type, the servicer begins at Node 1 of the operational orbit, while one depot is initially positioned at Node 3 of the same orbit. The customer database comprises 1,730 satellites: all are eligible for stochastic repair or ADR services, while seven primary customers, one placed in each of the dedicated customer orbits (Orbits 2 to 8), additionally require deterministic refueling or inspection services.

Figure 8.2 provides a schematic overview of the orbital network and the initial spatial configuration of the servicing infrastructure and primary customers considered in this case study. Orbits are shown ordered by increasing altitude but are not to scale, and no additional orbital parameters are represented.

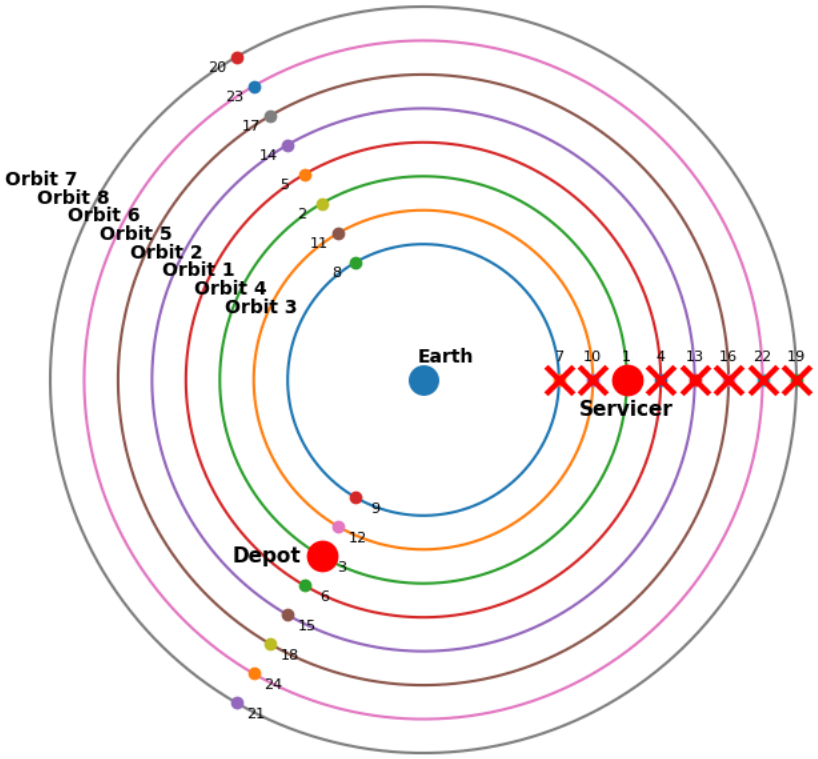


Figure 8.2: Schematic representation of the orbital network and initial configuration for the operational case study. Eight orbits are shown, ordered by increasing altitude and discretized into three nodes each. Primary customers are indicated by red crosses at their initial orbital positions. The servicer and the depot are initially positioned in the operational orbit (Orbit 1) at Nodes 1 and 3, respectively.

The operational scenario is summarized in Table 8.7, and the simulation parameters used to implement the Rolling Horizon optimization for this case study are provided in Table 8.8. Under these assumptions and parameters, the service tasks that can be precomputed over the full Scheduling Hori-

zon are enumerated in Table 8.9 in chronological order.

Table 8.7: Scenario definition for the operational case study.

Element	Description
Servicing architecture	One versatile servicer equipped with all service tools, initially located at Orbit 1 / Node 1
Depot configuration	One depot initially located at Orbit 1 / Node 3
Customer fleet	1,730 customer satellites eligible for stochastic repair and active debris removal services
Primary customers	7 out of 1,730 customer satellites, one located in each customer orbit (Orbits 2-8), requiring deterministic services
Deterministic service types	Refueling and inspection
Primary customer initial nodes	Refueling customers are initially located at Nodes 4, 10, 16, and 22 Inspection customers are initially located at Nodes 7, 13, and 19
Operational orbit	Orbit 1, hosting the depot and serving as the initial orbit for the servicer

Table 8.8: Simulation parameters used in the operational case study.

Parameter	Value	Units
Time step size (Δt)	32	min
Scheduling horizon	18,000	min
Planning horizon	4,500	min
Maximum idle time	1,000	min

Table 8.9: Deterministic and stochastic service requests considered in the operational case study, ordered by activation time.

Task type	Customer	Orbit	Time [min]	Time step	Resource requirement
Refueling	1	2	500	16	15 kg propellant
Inspection	2	3	3200	100	–
Refueling	3	4	4900	153	15 kg propellant
Inspection	4	5	7600	238	–
Refueling	5	6	9300	291	15 kg propellant
ADR	197	5	10811	338	1 de-orbit kit
Inspection	6	7	12000	375	–
Refueling	7	8	13700	428	15 kg propellant
Repair	63	4	15232	476	1 spare unit

This case study illustrates the framework's ability to:

- dynamically assign and sequence tasks throughout the planning horizon,
- generate feasible routing decisions using the precomputed maneuver network,
- handle the stochastic arrival of random service requests,
- manage limited propellant and consumables,
- coordinate depot access for refueling and commodity resupply, and
- ensure full compliance with service windows, task durations, required tools, and resource constraints.

Results and Discussion

The optimization framework is executed on the DelftBlue supercomputing cluster ((DHPC), 2024) under an academic allocation, using Gurobi Optimizer 12.0.3 (linux64). Computations run on Intel® Xeon®

Gold 6248R CPUs at 3.00 GHz with up to 12 threads per optimization. The solution for the full operational case study is obtained in approximately two hours using a solver configuration tailored to improve MILP performance under both memory and time constraints. For each RH iteration, disk-based node storage is enabled to handle the large search tree efficiently, and a moderate level of presolve and cut generation is activated, together with variable aggregation. A low heuristic intensity and a concurrent MILP strategy are used to diversify the search process. The algorithm is initialized with `Method = 1` (dual simplex), and the search is guided by `MIPFocus = 2` to prioritize bound improvement. A relative optimality gap of 0.1% and a wall-clock time limit of 1 200s are imposed per RH iteration, under which the solver successfully produced the solution presented hereafter.

Solver configuration and solution quality: It is important to emphasize that, relative to the verification case studies presented in Chapter 7, the underlying optimization framework, network construction, Rolling Horizon logic, and constraint set remain entirely unchanged in the present operational case study. The only methodological difference lies in the solver configuration: whereas the verification experiments enforced a zero optimality gap at each iteration, the operational case study, and the subsequent strategic analyses, adopt a nonzero relative MILP gap together with a strict per-iteration wall-clock time limit to ensure computational tractability across a large number of RH solves. This adjustment is motivated solely by practical computational considerations and does not reflect a limitation of the formulation itself; with sufficient computational resources, the same zero-gap configuration could be retained. Consequently, the formal verification of model correctness and logical consistency does not need to be repeated here. Instead, it is sufficient to demonstrate that the adopted solver tolerances do not materially degrade solution quality, thereby preserving the validity and interpretability of the operational results.

Across the full operational simulation, 17 out of 19 Rolling Horizon MILPs converged to the prescribed optimality tolerance of 0.1% within the imposed wall-clock limit of 1,200s. In the remaining two iterations, the solver reached the time limit and terminated with final relative gaps of approximately 1.9%. To assess whether these larger gaps materially affect the reported operational outcome, the solver configuration was selectively tightened for these two iterations by imposing a stricter optimality tolerance (0.05%) and a substantially extended time limit (5,000s), while keeping all inputs, system states, and RH parameters identical to the baseline operational run. Under this targeted re-solve, all Rolling Horizon iterations converged within the imposed optimality tolerance. The resulting solution exhibits a modest improvement in the overall objective value, which can be directly attributed to reduced accumulated delay costs. While the sequence of executed services and depot resupply events remains unchanged, the tightened solver configuration enables a slightly earlier scheduling of the final depot visit and the last two service operations. These temporal adjustments do not alter the operational structure or feasibility of the mission plan, but they reduce delay penalties and thereby improve the objective value.

Importantly, the qualitative operational behavior illustrated in Figure 8.3 and the service sequence reported in Table 8.10 remain unchanged. The observed differences are confined to minor timing refinements in the later stages of the mission and do not affect service selection, ordering, or resource-feasibility decisions. This confirms that the baseline operational solution already captures the correct operational structure, and that the residual optimality gap primarily influences fine-grained scheduling within service windows rather than mission-level decisions. Any remaining suboptimality at the level of the full Scheduling Horizon is therefore attributable to the inherent limitations of the Rolling Horizon approach and demand uncertainty, rather than to solver convergence issues. In the case studies that follow, MILP gaps of this magnitude for a limited number of iterations are therefore considered acceptable.

Having established the robustness and near-optimality of the obtained solutions, the following discussion focuses on the resulting operational behavior and mission execution.

Figure 8.3 and Table 8.10 should be read jointly to interpret the operational outcome of the case study. The figure illustrates the optimized mission timeline of Servicer d1 together with the time evolution of onboard fuel and service commodities across the full Scheduling Horizon, while the table reports the same sequence of executed service tasks and depot resupply events shown in the figure in a structured and quantitative form, ordered according to their occurrence along the timeline and annotated with their corresponding start and end times. For each service, the associated activation time, previ-

ously introduced in Table 8.9, is also reported, enabling the reader to directly relate task availability, execution timing, and the resulting delays. Since the objective of this case study is to highlight the operational behavior of the framework under uncertainty, cumulative economic performance across Rolling Horizon iterations is not discussed here.

Only a small subset of the customer fleet exhibits service needs within the analyzed horizon. Although the total customer database comprises 1,730 satellites, only nine generate service requests, and the servicer ultimately visits eight of them in the optimal solution. The optimization framework automatically focuses on these active customers, which substantially reduces the problem size and ensures computational tractability without compromising the quality of the solution.

Table 8.10: Executed service and depot resupply events for Servicer d1 in the operational case study, ordered by execution start time. Activation times refer to task availability, while execution times correspond to the scheduled service start.

Event	Activation [min]	Start [min]	End [min]	Delay [min]	Duration [min]
Refueling of Customer 1	500	1248	1408	748	160
Depot resupply (1)	–	2112	2144	–	32
Inspection of Customer 2	3200	3200	3264	0	64
Depot resupply (2)	–	4064	4096	–	32
Refueling of Customer 3	4900	4896	5056	≈0	160
Inspection of Customer 4	7600	8832	8896	1232	64
Refueling of Customer 5	9300	9952	10112	652	160
Inspection of Customer 6	12000	12000	12064	0	64
ADR for Customer 197	10811	12768	12928	1957	160
Depot resupply (3)	–	16160	16192	–	32
Repair of Customer 63	15232	17024	17184	1792	160

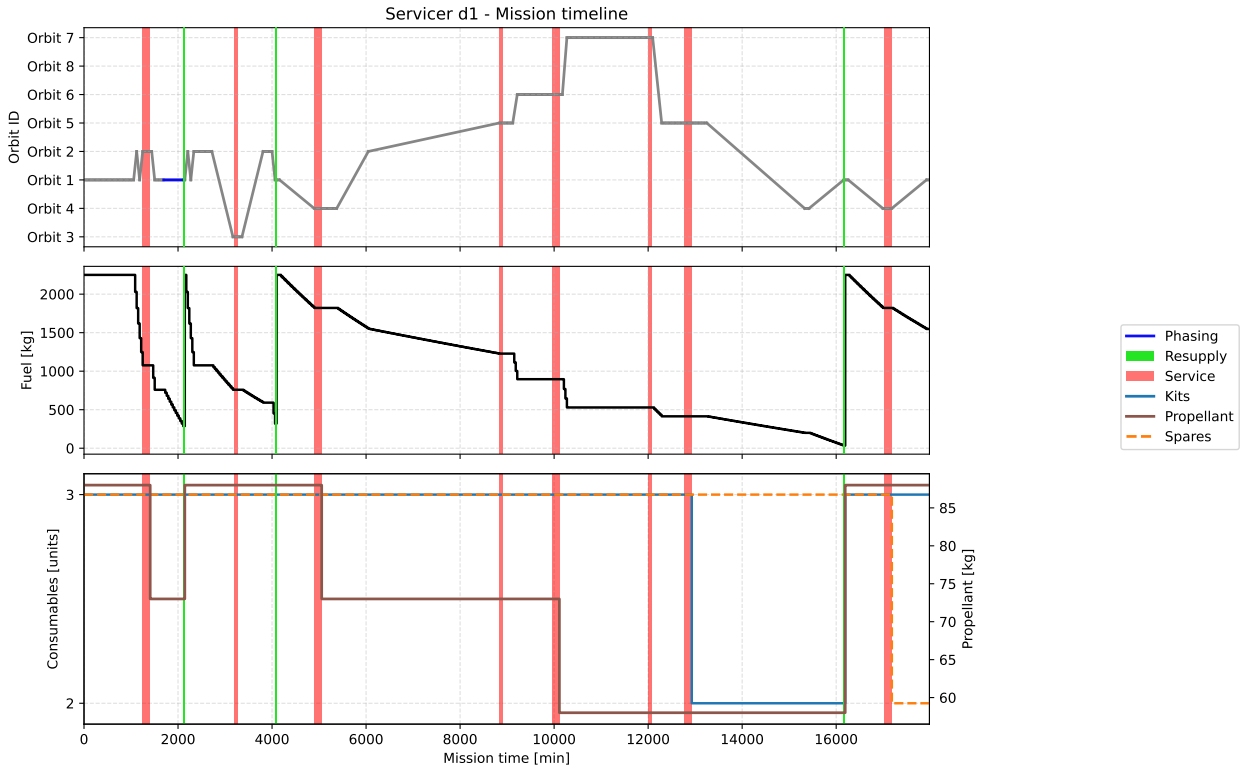


Figure 8.3: Servicer d1: mission timeline and resource evolution for the operational case study.

The optimized mission plan exhibits a clear operational pattern: the servicer repeatedly transits between the active customer orbits, performs the requested services, and subsequently returns to the

depot to replenish propellant and consumables. As illustrated in Figure 8.3, onboard fuel decreases during orbital maneuvers and is fully restored following each depot rendezvous. Consumable usage follows an analogous stepwise behavior: inventories of customer propellant, spare units, and de-orbit kits decrease discretely upon service completion and are subsequently fully replenished during depot visits. It is noteworthy that, for the considered service demand, orbital geometry, and servicer configuration, *propellant* emerges as the dominant operational constraint. This is evidenced by the substantial fuel expenditure incurred over the mission, which significantly exceeds the consumption of other service commodities, and by the necessity of multiple depot resupply events to enable completion of the scheduled services.

The temporal structure of Figure 8.3 also reflects modeling choices made to compress the operational scenario. Because the orbits used in this study are relatively close to one another and the service and resupply durations were shortened, the overall mission timeline appears accelerated relative to a physically realistic campaign. This intentional scaling enables the visualization and evaluation of multiple operational cycles within a reduced temporal window.

The compact orbital geometry adopted in the case studies gives rise to an additional emergent behavior in the optimized trajectories. Although the network explicitly includes propellant-efficient multi-revolution phasing arcs, the optimizer may instead exploit nearby orbits as temporary phasing layers. Because the customer orbits are tightly clustered in semi-major axis, inclination, and RAAN, transfers between adjacent orbits via the combined maneuvers embedded in the network incur only a small ΔV, comparable to that of the multi-revolution phasing options. At the same time, the small relative RAAN offsets between neighboring orbits can be closed over short time spans through these combined maneuvers. Consequently, when the additional propellant expenditure does not significantly affect the economic objective while enabling improved or feasible node-time alignment, the optimizer may, under Rolling Horizon optimality-gap limits, select short orbit-hopping detours as an economically near-equivalent alternative for synchronization with a target orbit.

This behavior is visible in the mission timeline, for example in the brief transfer from Orbit 2 to Orbit 1 and subsequently back to Orbit 2 prior to the first service event. In such cases, the optimizer effectively uses a neighboring orbit as an opportunistic phasing layer, exploiting the small geometric separation to reduce the residual phasing gap before rendezvous. This behavior is specific to the compact orbital configuration considered in the case studies and would not arise in scenarios with more widely spaced orbits. If the RAAN, altitude, or inclination differences between orbits were larger, the associated plane-alignment and cross-track penalties would dominate, rendering such detours both time- and cost-prohibitive and therefore unattractive in optimal service sequences. In the present configuration, however, the impact on the objective function remains marginal: the orbit-hopping detours introduce only minor delays and limited additional propellant consumption, neither of which materially affects the economic outcome. As a result, and within the wall-clock and optimality-gap limits imposed on each Rolling Horizon iteration, the optimizer may converge to these economically near-equivalent trajectories when they offer a computationally efficient route to a feasible rendezvous. Overall, the tight clustering of the operational orbits enables the optimizer to leverage neighboring orbits as temporary phasing layers, giving rise to the orbit-hopping patterns observed in the mission timeline.

In addition to exploiting the orbital geometry, the framework actively leverages the flexibility provided by wide service windows to reorder tasks in a cost-efficient manner. Rather than following the chronological order of task activations, the optimizer selects a servicing sequence that jointly minimizes maneuvering effort, total propellant expenditure, subject to the fuel safeguard mechanism, and accumulated delay penalties. This behavior is clearly illustrated by the servicing order of the deterministic inspection at Orbit 7 (Customer 6) and the stochastic debris removal task at Orbit 5 (Customer 197). Although the debris removal task becomes available earlier, the optimizer schedules the inspection of Customer 6 first ($t = 12000$–12064 min), followed by the ADR task at Customer 197 ($t = 12768$–12928 min). This reordering is driven by two complementary considerations. First, debris removal services do not incur delay penalties, whereas deterministic inspections do, making it economically advantageous to prioritize the latter. Second, the resulting orbital sequence reduces maneuvering effort, as the transfer to Orbit 7 followed by a descent to Orbit 5 aligns naturally with the subsequent trajectory toward the depot. By servicing Customer 6 first, the optimizer avoids unnecessary delay costs while simultaneously reducing propellant consumption, all while remaining well within the wide service window of the ADR task. The service windows therefore provide sufficient temporal slack to enable such reordering without compromising feasibility.

The same temporal flexibility also accommodates the required depot resupply events. The depot returns at $t = 2112$–2144 min, $t = 4064$–4096 min, and $t = 16160$–16192 min are positioned at times that support, rather than disrupt, the downstream servicing sequence. If service windows were narrow, these interruptions could render otherwise feasible tasks infeasible. Instead, the resupply events restore the servicer's propellant reserves at strategically advantageous moments, enabling fuel-efficient execution of subsequent services.

Overall, the final servicing sequence shown in Figure 8.3 illustrates the combined effect of geometric structure, service-window flexibility, and resource constraints on the resulting mission plan. Rather than adhering strictly to activation chronology, the optimization process constructs operationally feasible schedules that balance maneuvering effort, propellant usage, and delay penalties to achieve higher overall efficiency and economic performance.

Ultimately, the servicer achieves a high level of operational effectiveness, successfully completing 8 out of the 9 requested services within the analyzed horizon, corresponding to a service completion rate of approximately 89%. The only service request not executed within the horizon is the deterministic refueling task for Customer 7. This outcome is not the result of infeasibility but follows directly from the interaction between resource constraints, task economics, and the fuel safeguard mechanism embedded in the MILP. After completing the debris removal task for Customer 197, the remaining onboard propellant is insufficient to support both the refueling of Customer 7 and the subsequent repair task for Customer 63 without violating fuel constraints. To avoid incurring a large fuel shortage penalty at horizon closure, the optimizer schedules a depot refueling instead, preserving propellant reserves for continued operations in the subsequent mission period. Given the limited remaining time, the optimizer then prioritizes the repair of Customer 63, which offers a substantially higher revenue than the refueling task for Customer 7. Consequently, the refueling service is deliberately dropped in favor of a solution that maximizes overall net profit while maintaining fuel robustness beyond the current Scheduling Horizon.

Furthermore, the average service delay is 13.3 h. Part of this delay is an intentional consequence of the cost-driven sequencing strategy adopted by the optimizer. By exploiting wide service windows, the framework clusters geometrically favorable tasks into successive sequences, reducing maneuvering effort and enabling depot refueling at advantageous times to increase overall service throughput. Services are further reordered to minimize cumulative delay penalties, with priority given to tasks for which deferred execution is most costly. As a result, some delay is deliberately accepted for tasks as a trade-off to improve overall mission efficiency and profitability. At the same time, the observed delays are also partly attributable to the computational limits imposed on each Rolling Horizon iteration. To maintain tractability across a large number of MILP solves, each iteration is executed under a strict wall-clock time budget and a nonzero optimality tolerance. These limits prevent the optimizer from exhaustively exploring deeper regions of the search tree, where alternative schedules with lower delay might be identified.

Finally, it should be noted that Rolling Horizon schemes do not guarantee global optimality, since each optimization step operates over a finite look-ahead window and relies on partial information about future service demands and system evolution. The resulting performance is inherently sensitive to the choice of parameters such as the Planning Horizon and the Control Horizon, which determine how far ahead the optimizer can anticipate and commit to decisions. In this work, these parameters were selected and tuned through a combination of analytical reasoning and empirical experimentation. Nonetheless, additional refinement or joint optimization may further improve the overall performance of the Rolling Horizon method.

Overall, the results demonstrate that the proposed framework adapts robustly to evolving operational conditions and maintains feasibility across the entire horizon. The Rolling Horizon scheme successfully accommodates stochastic task arrivals, dynamic resource constraints, and repeated resupply cycles, illustrating its suitability as a real-time or near-real-time decision-support tool for In-Orbit Servicing operations in Low Earth Orbit.

8.3. Long-Term Strategic Analysis

The optimization framework is not limited to operational routing and scheduling; it can also support long-term strategic planning. In this capacity, it enables the evaluation of how different IOS architectures perform over extended periods and how exogenous factors, such as varying levels of service

demand, affect their economic performance. For new entrants in the IOS market, such analyses are essential for designing infrastructures that remain competitive under fluctuating demand conditions and capable of generating sustainable profits as the market evolves (Sarton du Jonchay et al., 2021, 2022). Conducting such evaluations requires exploring a broad architectural design space, including servicer configurations, depot number and placement, and the associated operational strategies. Furthermore, uncertainty in service demand necessitates a planning mechanism that accounts for stochastic events; this function is provided by the Rolling Horizon scheme integrated into the framework.

In this context, the following case study illustrates the strategic value of the framework. Executing the optimization over an extended scheduling horizon enables systematic comparisons across IOS architectures and provides insight into their relative advantages and limitations. Architectural performance is primarily assessed through cumulative profits generated over each simulation and the system value attained at the end of the Scheduling Horizon. To provide additional insight in the presence of high upfront investment costs, complementary indicators capturing service revenue evolution, infrastructure utilization, and marginal service capability are also considered.

For the strategic evaluation, the framework is run over a scheduling horizon of 30 days. Although an even longer horizon could capture additional long-term effects, the chosen duration offers a practical balance between computational tractability and the ability to extract meaningful strategic insights.

Given that the IOS market is projected to evolve and expand over time, as supported by market studies and demand forecasts, the strategic case study evaluates multiple representative demand scenarios to determine how different IOS architectures perform across varying market conditions. Specifically, three market conditions are examined, corresponding to progressively higher levels of service demand as defined in Table 8.11. As in the operational case study, a distinction is made between the full customer fleet and a subset of *primary customers*. Primary customers represent satellites with pre-agreed deterministic service requirements (refueling or inspection), while all customers in the fleet are eligible to generate stochastic repair and active debris removal requests.

Table 8.11: Comparison of demand scenarios considered in the long-term strategic analysis.

Element	**Low demand**	**Medium demand**	**High demand**
Customer fleet	865 satellites	1,730 satellites	2,595 satellites
Primary customers	7 out of 865 (1 per Orbit 2-8)	14 out of 1,730 (2 per Orbit 2-8)	21 out of 2,595 (3 per Orbit 2-8)
Primary customer initial nodes (refueling)	Nodes 4, 10, 16, and 22	Nodes 4, 9, 10, 15, 16, 21, and 22	Nodes 4, 5, 9, 10, 11, 15, 16, 17, 21, 22, and 23
Primary customer initial nodes (inspection)	Nodes 7, 13, and 19	Nodes 6, 7, 12, 13, 18, 19, and 24	Nodes 6, 7, 8, 12, 13, 14, 18, 19, 20, and 24

All strategic demand scenarios are evaluated on the same orbital network and node discretization introduced in the operational case study (so as a graphical reference see and compare with Figure 8.2).

The task sets associated with each demand scenario used in the strategic simulations are precomputed over the 30-day Scheduling Horizon and listed in Tables 8.12-8.14, ordered by activation time and including the corresponding resource requirements.

Table 8.12: Deterministic and stochastic service requests considered in the low-demand strategic scenario, ordered by activation time.

Task type	Customer	Orbit	Time [min]	Time step	Resource requirement
Refueling	1	2	3000	94	15 kg propellant
Debris removal	197	5	10811	338	1 de-orbit kit
Refueling	3	4	13000	406	15 kg propellant
Inspection	2	3	15000	469	–
Repair	63	4	15232	476	1 spare unit
Refueling	5	6	23000	719	15 kg propellant
Inspection	4	5	25000	781	–
Refueling	7	8	33000	1031	15 kg propellant
Inspection	6	7	35000	1094	–

Table 8.13: Deterministic and stochastic service requests considered in the medium-demand strategic scenario, ordered by activation time.

Task type	Customer	Orbit	Time [min]	Time step	Resource requirement
Refueling	1	2	3000	94	15 kg propellant
Refueling	3	4	8000	250	15 kg propellant
Debris removal	197	5	10811	338	1 de-orbit kit
Inspection	2	3	12500	391	–
Refueling	5	6	13000	406	15 kg propellant
Repair	63	4	15232	476	1 spare unit
Inspection	4	5	17500	547	–
Refueling	7	8	18000	562	15 kg propellant
Debris removal	1271	7	18004	563	1 de-orbit kit
Debris removal	1460	6	18205	569	1 de-orbit kit
Inspection	6	7	22500	703	–
Refueling	9	3	23000	719	15 kg propellant
Inspection	8	2	27500	859	–
Refueling	11	5	28000	875	15 kg propellant
Repair	1281	7	28940	904	1 spare unit
Inspection	10	4	32500	1016	–
Refueling	13	7	33000	1031	15 kg propellant
Repair	1493	4	34135	1067	1 spare unit
Inspection	12	6	37500	1172	–
Debris removal	1376	7	38958	1217	1 de-orbit kit
Inspection	14	8	42500	1328	–

To facilitate a high-level comparison across demand scenarios, Table 8.15 summarizes the total number of service requests considered in each case and distinguishes between deterministic tasks associated with primary customers and stochastic repair or debris removal requests. The table highlights the rapid growth in both overall task volume and stochastic demand as market size increases.

Building on these demand scenarios, the strategic case study conducts two complementary analyses:

1. **Depot number trade-off analysis:** The impact of including zero, one, or two orbital depots is examined across all demand levels.
2. **Servicer fleet configuration comparison:** For each market condition, a single versatile servicer is compared against a distributed architecture composed of four specialized servicers, each dedicated to one of the modeled service types.

Together, these analyses provide a comprehensive evaluation of strategic design trade-offs and enable the identification of IOS architectures that remain resilient, cost-effective, and economically viable across diverse market conditions. In all simulations, it is assumed that both depots and servicers are initially deployed in the operational orbit (Orbit 1), with depots remaining there and servicers departing from Orbit 1 to execute their assigned tasks across the customer orbits.

8.3.1. Impact of Depot Number Across Market Scenarios

A larger number of depots naturally increases both the initial investment and the ongoing operating costs of an IOS architecture. Under low-demand conditions, these additional costs may not be offset by the limited service revenue available over the short mission horizon considered here. However, from a strategic perspective, additional infrastructure may enhance operational capability and enable a system to capture a greater share of future service opportunities as demand increases.

This trade-off motivates the following analysis, which examines how depot number affects the performance of an IOS system operating with a single versatile servicer across different demand levels. Performance is evaluated primarily in terms of economic outcomes, while additional utilization- and capability-based metrics are used to clarify how increasing infrastructure affects service execution and value creation. Three architectural configurations are considered, differing only in the number of available depots, zero, one, or two, while keeping the servicer platform, its tooling, and its initial deployment

Table 8.14: Deterministic and stochastic service requests considered in the high-demand strategic scenario, ordered by activation time.

Task type	Customer	Orbit	Time [min]	Time step	Resource requirement
Debris removal	2171	7	2459	77	1 de-orbit kit
Refueling	1	2	3000	94	15 kg propellant
Repair	1756	7	4519	141	1 spare unit
Refueling	3	4	6000	188	15 kg propellant
Debris removal	2047	8	7650	239	1 de-orbit kit
Refueling	5	6	9000	281	15 kg propellant
Debris removal	197	5	10811	338	1 de-orbit kit
Inspection	2	3	11500	359	–
Refueling	7	8	12000	375	15 kg propellant
Inspection	4	5	14500	453	–
Refueling	9	3	15000	469	15 kg propellant
Repair	63	4	15232	476	1 spare unit
Inspection	6	7	17500	547	–
Refueling	11	5	18000	562	15 kg propellant
Debris removal	1271	7	18004	563	1 de-orbit kit
Debris removal	1460	6	18205	569	1 de-orbit kit
Debris removal	2244	5	19173	599	1 de-orbit kit
Inspection	8	2	20500	641	–
Refueling	13	7	21000	656	15 kg propellant
Inspection	10	4	23500	734	–
Refueling	15	2	24000	750	15 kg propellant
Inspection	12	6	26500	828	–
Refueling	17	4	27000	844	15 kg propellant
Repair	1281	7	28940	904	1 spare unit
Inspection	14	8	29500	922	–
Refueling	19	6	30000	938	15 kg propellant
Inspection	16	3	32500	1016	–
Refueling	21	8	33000	1031	15 kg propellant
Repair	1493	4	34135	1067	1 spare unit
Inspection	18	5	35500	1109	–
Inspection	20	7	38500	1203	–
Debris removal	1376	7	38958	1217	1 de-orbit kit
Repair	2500	6	42128	1317	1 spare unit

Table 8.15: Summary of service-task volumes across demand scenarios.

Scenario	Total tasks	Deterministic	Stochastic
Low demand	9	7	2
Medium demand	21	14	7
High demand	33	21	12

(Node 1 of Orbit 1) constant across all cases. Each architecture is evaluated under the three demand conditions defined earlier (low, medium, and high), resulting in a total of nine simulation runs.

The characteristics of the three architectural configurations are summarized in Table 8.16.

Table 8.16: IOS architecture configurations used in the depot-number trade-off analysis.

Architecture	No Depot	One Depot	Two Depots
Number of Servicers	1	1	1
Servicer Architecture	Versatile (4 tools)	Versatile (4 tools)	Versatile (4 tools)
Servicer Starting Node	Node 1	Node 1	Node 1
Depot Configuration	None	1 depot at Node 3	2 depots at Nodes 2 and 3

The simulation parameters used in the Rolling Horizon implementation, which are held constant across all depot configurations, are summarized in Table 8.17. The Planning Horizon and maximum idle time are varied across demand scenarios. Shorter planning horizons are adopted under higher demand to limit problem size and enable tighter MILP optimality gaps in denser decision spaces, while longer horizons are retained under low-demand conditions to provide sufficient look-ahead when service opportunities are sparse. The maximum idle time is adjusted consistently with demand intensity to reflect more frequent task arrivals.

Table 8.17: Simulation parameters used for the depot-number trade-off analysis.

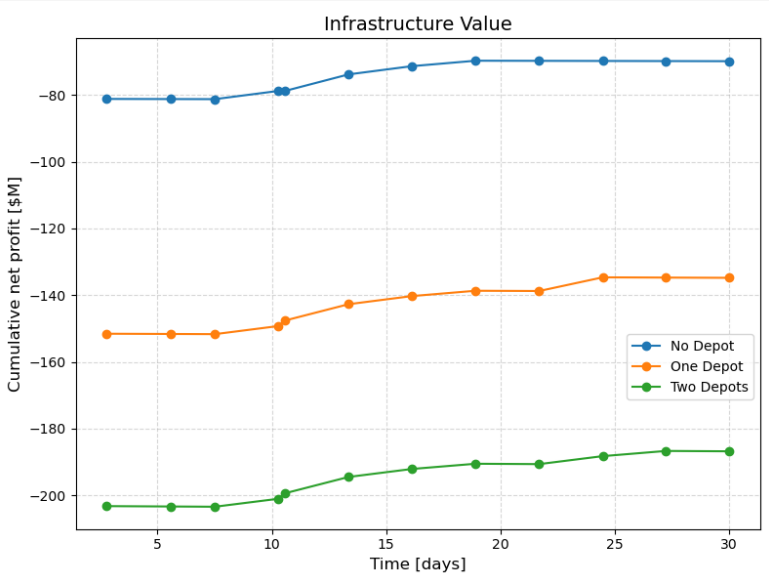
Parameter	Value	Units
Time step size (Δt)	32	min
Scheduling horizon	43,200	min
Planning horizon (low demand)	10,000	min
Planning horizon (medium demand)	8,000	min
Planning horizon (high demand)	4,500	min
Maximum idle time (low demand)	4,000	min
Maximum idle time (medium demand)	2,000	min
Maximum idle time (high demand)	1,000	min

Results and Discussion

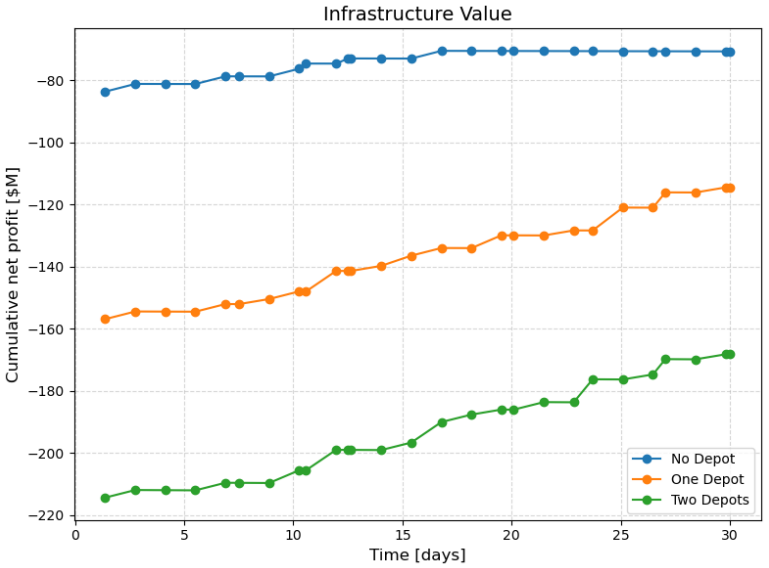
The simulations for the depot-number trade-off analysis are executed using the same computational setup described in the operational case study (see Section 8.2), including the DelftBlue supercomputing environment and the Gurobi 12.0.3 solver configuration. The only adjustment concerns the computational budget allocated to each Rolling Horizon iteration: instead of the 1 200 s wall-clock limit used in the operational case study, a higher limit of 3 000 s per iteration is imposed to account for the larger decision space. All other solver settings, including the presolve strategy, cut generation, node-file storage, and heuristic intensity, remain unchanged. Under this configuration, the solver successfully generated the solutions analyzed in the following discussion.

Across the nine simulation runs, a total of 270 MILP optimizations were solved, with each full simulation requiring on average approximately 9h to complete. This corresponds to an average solution time of about 1,080s per MILP iteration. The majority of iterations (approximately 72%) reached proven optimality within the imposed time limit, while the remaining 28% terminated due to the time constraint. For the latter, solution quality remained high, with an average MILP gap of only about 2.2%. This indicates that even when the solver is unable to fully close the optimality gap within the available computational budget, the Rolling Horizon framework consistently produces near-optimal solutions suitable for strategic analysis.

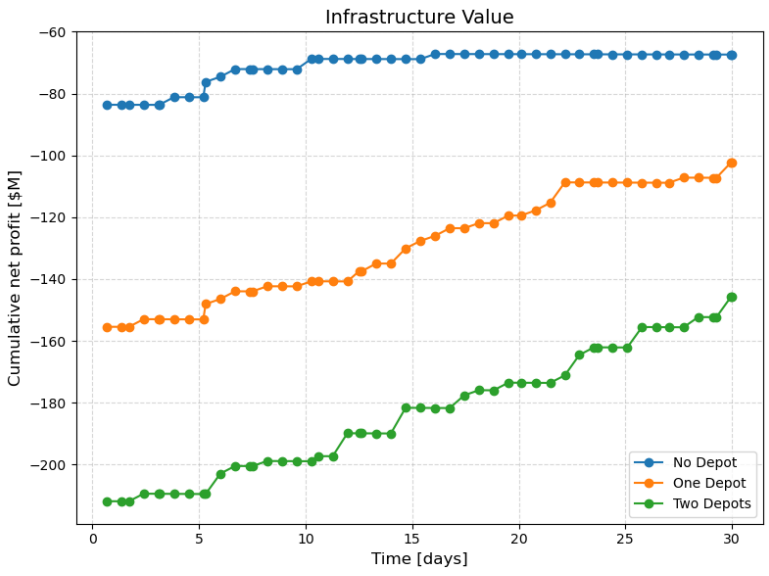
Unless stated otherwise, the graphical results presented in the following figures report time-evolving quantities at the resolution of the Rolling Horizon scheme, with each point on a curve corresponding to the outcome of a single MILP optimization at a given replanning iteration.



(a) Low-demand scenario



(b) Medium-demand scenario



(c) High-demand scenario

Figure 8.4: Cumulative net profit over the 30-day horizon across demand scenarios under the three depot configurations.

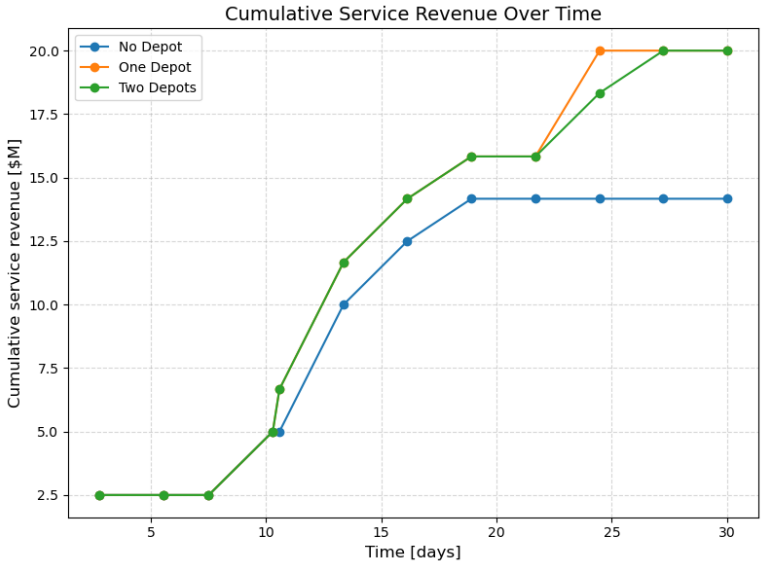
Figures 8.4a, 8.4b, and 8.4c compare the evolution of cumulative net profit over the 30-day mission horizon for the three depot infrastructure configurations (no depot, one depot, and two depots) under low-, medium-, and high-demand scenarios. Together, these nine simulation runs allow for a joint assessment of infrastructure value across varying market conditions.

When interpreting these profit trajectories, it is useful to distinguish between two aspects: the absolute infrastructure value reflected by the cumulative profit level, and the slope of the curves, which captures ongoing operational performance and value generation. Across all demand scenarios, initial costs, including launch and PDM expenses, strongly weigh down the cumulative profit profiles, resulting in a systematic downward offset that persists throughout the horizon. As a result, none of the architectures reaches break-even within the considered 30-day horizon, regardless of demand level. Because additional depot infrastructure entails higher initial and operating costs, the no-depot configuration consistently exhibits the least negative cumulative profit. This outcome highlights the dominant influence of initial investments under the chosen economic assumptions and limited mission duration.

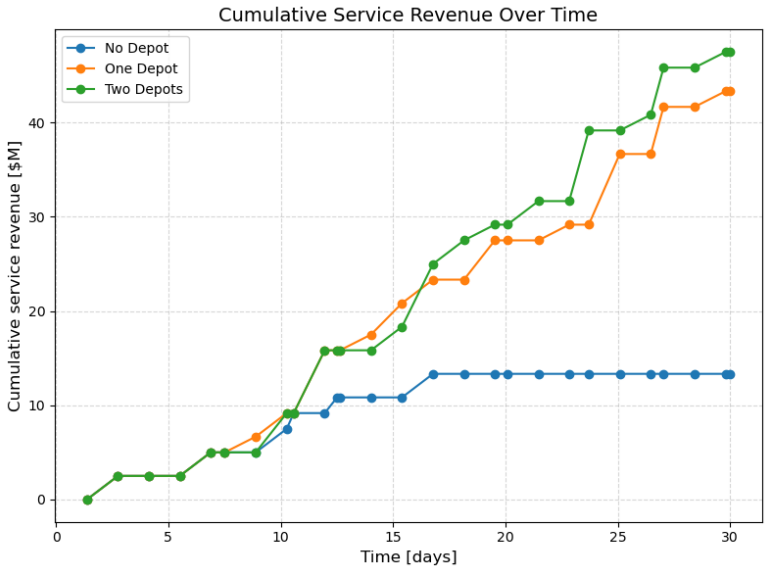
Nevertheless, important operational insights can be drawn from the slopes of the profit curves. Under all demand scenarios, the no-depot configuration initially exhibits positive profit accumulation but eventually reaches a plateau. This behavior indicates that service execution ceases once the servicer's initial fuel reserves are depleted: without access to an orbital depot, the servicer can no longer maneuver to remaining customers and perform additional services. As market demand increases, this plateau is reached earlier, since a larger number of available tasks accelerates resource depletion. In contrast, depot-enabled architectures retain operational capability throughout the mission horizon. Periodic resupply restores maneuvering and servicing capacity, enabling continued service execution and sustained value generation under all demand conditions.

In the low-demand scenario, revenues accumulate slowly even for the one- and two-depot configurations, reflecting the limited availability of service opportunities, and the profit trajectories of the different depot architectures remain nearly parallel. In this regime, the presence of depots does not translate into significantly higher value creation, as there is insufficient demand to fully exploit the additional operational capability. In the medium- and high-demand scenarios, depot-enabled architectures exhibit steeper profit slopes, indicating that value is accrued at a faster rate throughout the mission. This effect becomes increasingly pronounced as demand grows: profit trajectories are steeper in the high-demand case than in the medium-demand case, demonstrating the ability of depot-equipped systems to more effectively capitalize on increased service demand.

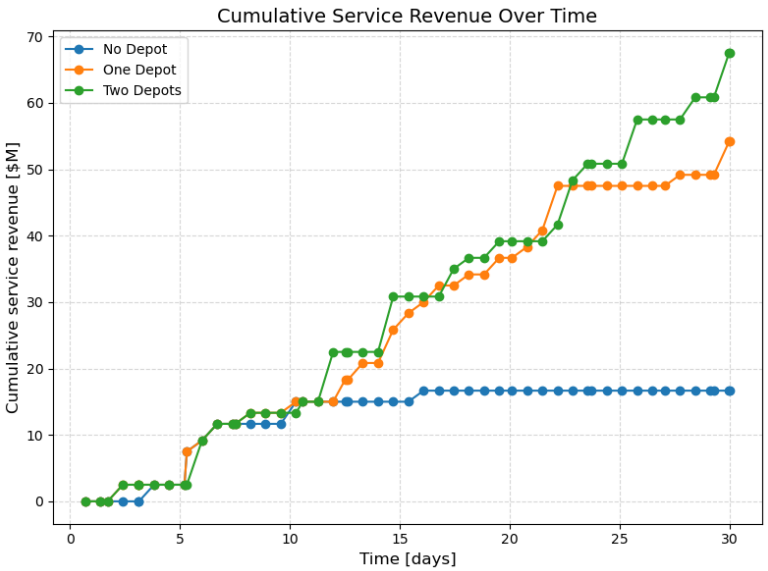
These results suggest that, under a refined economic model and sufficiently extended mission durations, economies of scale may allow depot-enabled architectures to surpass the no-depot configuration in cumulative profit and potentially achieve positive net profit. Before exploring this possibility further, additional metrics derived from this case study are analyzed in the following. The slope differences are driven primarily by sustained service execution and are therefore examined more directly through revenue and throughput metrics below.



(a) Low-demand scenario



(b) Medium-demand scenario



(c) High-demand scenario

Figure 8.5: Cumulative service revenue over the 30-day horizon across demand scenarios under the three depot configurations.

Net profit is composed of initial investment costs, service revenues, delay costs, and operating costs. Because initial investment costs dominate net infrastructure value and scale directly and deterministically with the amount of deployed infrastructure, and because operating costs represent a comparatively smaller but similarly deterministic contribution that increases with the number of depots, it is informative to isolate service revenues and delay costs from these effects. Figures 8.5a, 8.5b, and 8.5c therefore report cumulative service revenue over time for each depot architecture under the three demand scenarios.

In the low-demand scenario, revenue trajectories are very similar across architectures, with only some divergence emerging toward the end of the mission, as servicer resources become depleted in the no-depot architecture and further servicing is no longer possible. The no-depot configuration completes 5 out of 9 service tasks, corresponding to a completion rate of 55.6% and a total revenue of 14.2M$, while both the one- and two-depot configurations complete 8 out of 9 tasks, corresponding to a completion rate of 88.9% and a total revenue of 20M$. The limited availability of service opportunities constrains revenue generation, such that additional infrastructure yields only marginal gains.

Under medium-demand conditions, differences in revenue generation across architectures become more pronounced. The no-depot configuration completes 6 out of 21 service tasks, corresponding to a completion rate of 28.6% and total revenues of 13.3M$. Introducing a single depot increases throughput to 17 completed tasks (81% completion rate), yielding total revenues of 43.3M$. The two-depot architecture further improves performance, completing 19 out of 21 tasks (90.5% completion rate) and generating revenues of 47.5M$. These results indicate that depot-enabled architectures are increasingly able to translate additional operational flexibility into higher service throughput as demand rises. Compared to the profit trajectories, the revenue-based comparison highlights a clearer separation between the one- and two-depot configurations, with the second depot providing a slight advantage under medium-demand conditions.

These effects are strongest in the high-demand scenario. Here, the no-depot configuration completes only 7 out of 33 service tasks, corresponding to a completion rate of 21.2% and a total revenue of 16.7M$. By contrast, the one-depot configuration completes 21 out of 33 tasks (63.6%) with a total revenue of 54.2M$, while the two-depot configuration completes 26 out of 33 tasks (78.8%) with a total revenue of 67.5M$. These results demonstrate that, when sufficient service opportunities are available, additional depot infrastructure translates directly into higher service throughput and substantially greater revenue generation.

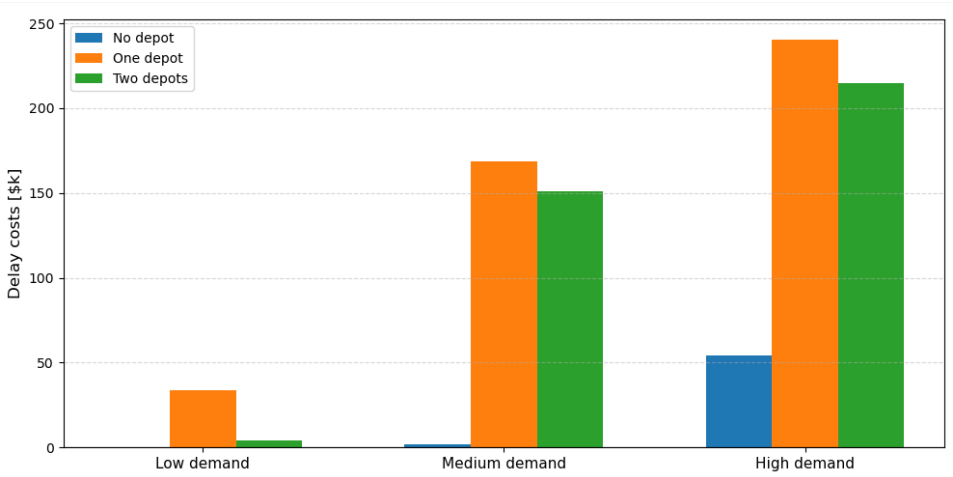
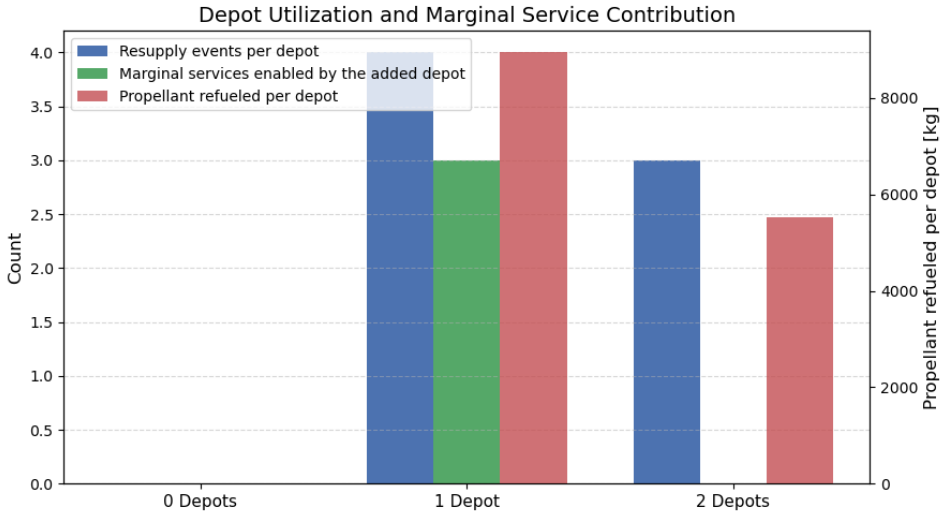


Figure 8.6: Aggregate delay costs incurred over the 30-day horizon under the three depot configurations and demand scenarios.

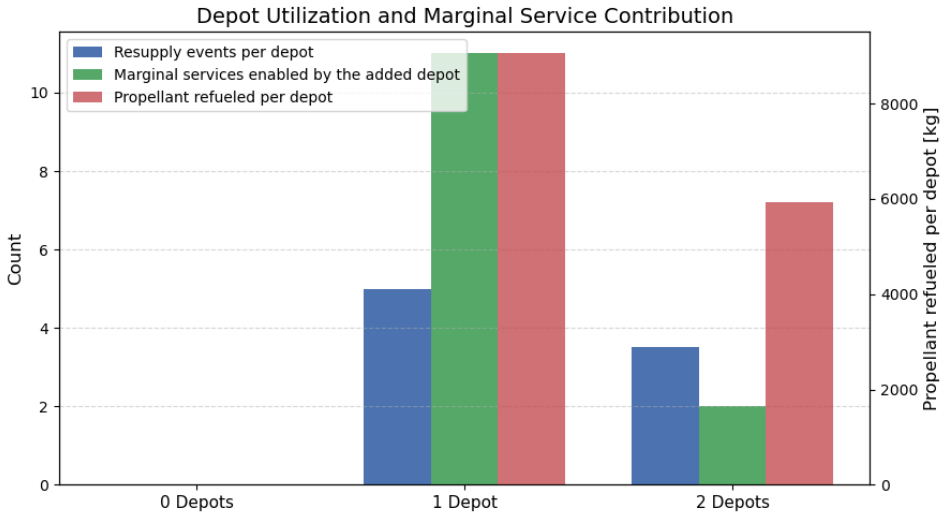
Isolating delay-related costs, Figure 8.6 reports the aggregate delay costs incurred under each depot configuration and demand scenario. Because delay costs are more sensitive to solver near-optimality than revenue-based metrics, particularly in Rolling Horizon iterations terminating with small but non-zero MILP gaps, the figure is interpreted qualitatively to highlight structural trends rather than precise quantitative differences.

At first glance, the no-depot configuration exhibits the lowest delay costs across all demand scenarios. This outcome does not reflect superior temporal performance, but rather the limited service capability of the architecture: with few services executed, there are fewer opportunities for delays to accumulate. Once depot-enabled architectures are considered, delay costs increase markedly, reflecting substantially higher service throughput and sustained operational activity.

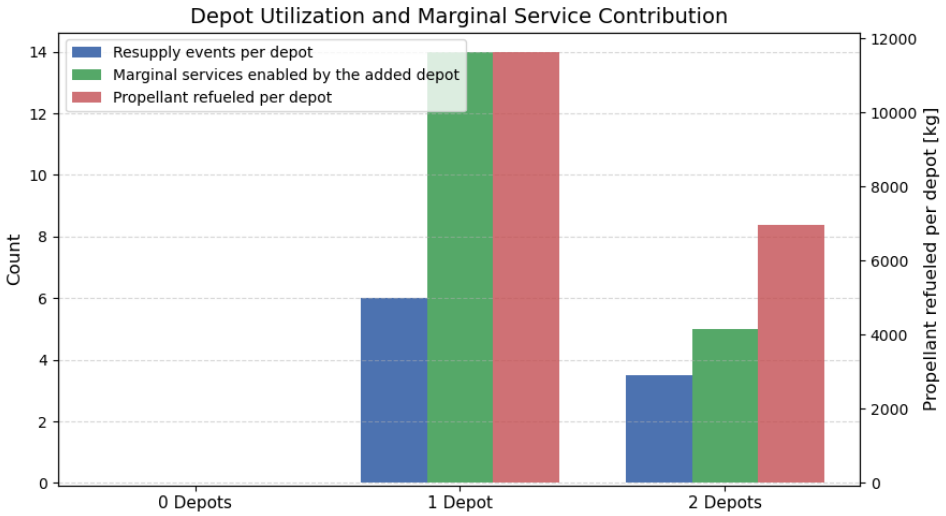
Comparing depot-enabled configurations, the two-depot architecture consistently incurs lower aggregate delay costs than the one-depot configuration, despite completing a larger number of services overall. This indicates that the availability of an additional depot reduces waiting times and scheduling congestion by improving access to resupply and increasing temporal and spatial flexibility in service execution. Finally, delay costs increase with demand level across all architectures, as a larger number of services are executed, increasing exposure to potential delays.



(a) Low-demand scenario



(b) Medium-demand scenario



(c) High-demand scenario

Figure 8.7: Depot utilization and marginal service contribution across demand scenarios. Resupplying events normalized per deployed depot and marginal services enabled by each additional depot are reported on the left axis, while propellant throughput normalized per deployed depot is shown on the right axis.

To further assess whether the deployed infrastructure is effectively exploited, and to better understand the differences between one- and two-depot architectures, Figures 8.7a, 8.7b, and 8.7c report depot utilization indicators together with the marginal service contribution of each additional depot under each demand scenario.

In the low-demand scenario, the first depot is actively utilized, both in terms of resupply events and propellant throughput, and enables a measurable increase in completed services. By contrast, the two-depot architecture exhibits lower utilization per depot, and the second depot does not enable any additional service completions. This indicates that the second depot is largely redundant under low-demand conditions.

As demand increases, depot utilization rises for both depot configurations. These utilization indicators are, however, partly influenced by the resupply policy embedded in the Rolling Horizon framework, which may trigger preemptive refueling to preserve operational flexibility even when the acquired fuel is not fully used and translated into additional services within the mission horizon. For this reason, utilization metrics are interpreted jointly with the marginal number of services enabled by each additional depot. Under medium- and high-demand conditions, the marginal service contribution associated with depot deployment increases, and the second depot exhibits a strictly positive contribution. Nevertheless, the figures consistently show that both utilization per depot and marginal service contribution remain higher in the one-depot configuration than in the two-depot configuration, even at high demand levels. This pattern indicates diminishing marginal returns from additional depots: while a first depot substantially enhances service feasibility and throughput, a second depot is used less intensively and enables fewer incremental services relative to its capacity. Within the examined demand range and resupply policy, these results suggest that the primary value of an additional depot lies in redundancy and operational flexibility rather than proportional gains in serviced demand.

To isolate the economic performance generated *during the mission*, revenues, operating costs, and delay costs are combined while excluding initial investment costs. Under this metric, in the low-demand scenario the no-depot configuration generates approximately 13.8M$, compared to 19.2M$ for the one-depot architecture and 18.8M$ for the two-depot architecture. The introduction of the first depot therefore increases economic gains by approximately 5.4M$ relative to the no-depot case, whereas the second depot yields negative marginal gain compared to the one-depot configuration. This confirms that, under low-demand conditions, the second depot represents structural overcapacity. Even in the one-depot architecture, the modest increase in service throughput would require an extended operational period to offset the associated capital and operating expenditures.

For the medium-demand scenario, the first depot increases economic gains by approximately 29.5M$, while the second depot contributes an additional 3.7M$. In the high-demand scenario, the first depot increases economic gains by approximately 36.9M$, with the second depot adding further 13M$. Although the marginal benefit of the second depot increases with demand, it remains substantially lower than the value introduced by the first depot. Again, this asymmetry suggests diminishing marginal returns to additional depot deployment.

To consolidate the results discussed above, Table 8.18 summarizes the key performance indicators across demand scenarios and depot configurations, highlighting service throughput, economic performance, and marginal effects of additional depot deployment.

Table 8.18: Summary of performance metrics for the depot-number trade-off analysis across demand scenarios. Economic gains exclude initial investment costs.

Scenario	Depots	Completed services	Revenue [M$]	Delay [k$]	Econ. Gain [M$]
Low demand	0	5 / 9	14.2	0.4	13.8
	1	8 / 9	20.0	33.7	19.2
	2	8 / 9	20.0	3.9	18.8
Medium demand	0	6 / 21	13.3	1.9	12.9
	1	17 / 21	43.3	168.8	42.4
	2	19 / 21	47.5	151.1	46.1
High demand	0	7 / 33	16.7	54.1	16.2
	1	21 / 33	54.2	240.2	53.1
	2	26 / 33	67.5	214.6	66.1

Break-even considerations under high-demand conditions: Under the baseline economic assumptions of the case study, none of the considered architectures reaches break-even within the simulated horizon, even in the high-demand scenario. This motivates an examination of how break-even could plausibly be achieved through adjustments to selected economic assumptions, while remaining consistent with anticipated developments in the IOS market.

First, the initial investment costs associated with servicers and orbital depots are treated conservatively in the case study, as they are fully allocated to a single mission sequence. In practice, both servicers and depots are reusable assets capable of supporting multiple successive IOS campaigns over their operational lifetime. Amortizing these infrastructure costs across several mission cycles substantially reduces the effective upfront cost per mission, thereby improving long-term profitability without affecting operational performance.

Second, revenues associated with stochastic, failure-driven services are likely underestimated. Unlike routine servicing, these interventions address time-critical anomalies that pose severe economic and systemic risks. As IOS markets mature, the value of such services is expected to be linked to avoided losses rather than direct service costs. Under this perspective, emergency repairs, anomaly resolution, and debris-removal actions command significantly higher economic value than deterministic servicing activities.

Third, revenues from deterministic services may also be conservative. The case study adopts fixed service prices that do not explicitly account for evolving satellite architectures, increasing constellation densities, or the growing dependence on uninterrupted space-based infrastructure. As reliance on satellite services increases, so does the willingness to pay for guaranteed availability and lifetime extension, suggesting additional upside potential in deterministic service pricing.

At first glance, the revenue levels required to reach break-even may appear unrealistically high if interpreted as direct payments by individual satellite operators. However, such a framing overlooks the broader transformation of business models enabled by IOS, particularly its natural coupling with the satellite insurance market. At present, only high-value spacecraft are typically insured, and insurance premiums remain high because many in-orbit failures are effectively irreversible. By increasing mission recoverability, IOS has the potential to reduce insurance premiums and to be integrated into satellite insurance frameworks, allowing servicing costs to be covered through insurance-based mechanisms rather than direct operator payments. Although significant challenges remain in aligning insurers, IOS providers, and regulatory frameworks, recent literature suggests that legislative support and standardized contractual structures could enable the adoption of such models (Mainelli et al., 2023).

Finally, IOS activities address a collective interest shared across the entire space ecosystem: the mitigation of space debris. The benefits of debris removal and collision-risk reduction extend beyond the serviced asset, improving safety and sustainability for operators, insurers, governments, and downstream service users alike. As these benefits constitute a system-level externality rather than a purely private good, they further motivate collective funding and governance mechanisms. Under such arrangements, IOS revenues may reasonably exceed what individual operators would be willing or able to pay in isolation, supporting higher service valuations in high-demand scenarios.

Thus, under these revised conditions, cumulative net profit trajectories are re-evaluated for the high-demand scenario assuming (i) that initial investment costs, namely launch and PDM costs of servicers and depots, are amortized over two mission campaigns and therefore reduced by half, and (ii) that revenues associated with all service tasks are doubled. Figure 8.8 reports the resulting cumulative net profit over the 30-day horizon for the no-depot, one-depot, and two-depot architectures.

Under these assumptions, break-even is effectively achieved within the simulated horizon for both depot-based architectures. The one-depot configuration reaches positive cumulative profit first, crossing the break-even threshold earlier in the horizon at around 22 days. This reflects its lower initial infrastructure burden combined with a substantial improvement in operational efficiency once depot-supported servicing becomes available. The earlier break-even of the one-depot architecture highlights its suitability as an entry-level IOS infrastructure under high-demand conditions, where capital constraints remain relevant despite favorable revenue environments.

The two-depot architecture reaches break-even later, at approximately 28 days, due to its higher upfront infrastructure and operating costs. Over the simulated horizon, the cumulative profit trajectories of the one- and two-depot configurations converge and ultimately terminate at very similar values, indicating that the additional operational flexibility provided by a second depot is largely offset by its higher initial investment within the 30-day window. This outcome highlights the trade-off between su-

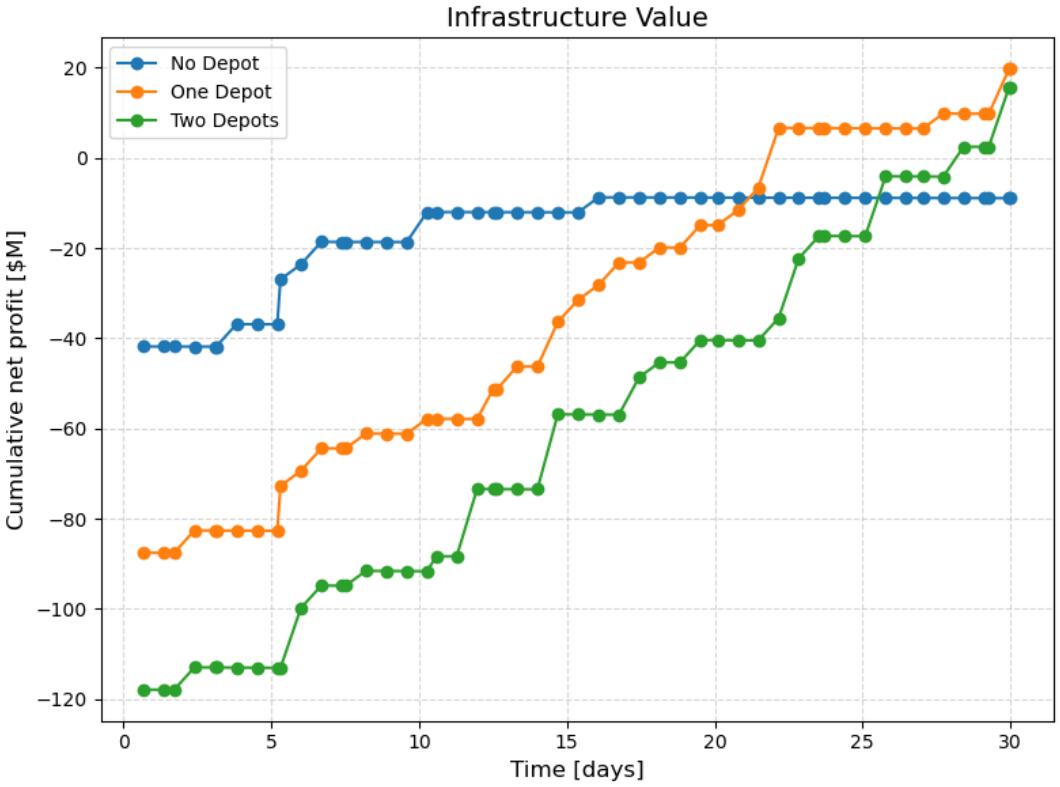


Figure 8.8: Cumulative net profit over the 30-day horizon for the high-demand scenario under revised economic assumptions.

perior long-term operational flexibility and economic recovery time: while the second depot enhances servicing capability, its economic advantage does not fully materialize over the limited mission duration considered here.

Overall, the results demonstrate that the economic attractiveness of depot-enabled IOS architectures is strongly contingent on market demand. Only under sufficiently high demand levels does the increase in service throughput enabled by orbital resupply begin to compensate for the substantial PDM, launch, and operating costs associated with depot infrastructure. In such market conditions, early investment in a single depot, and, under more favorable circumstances, in additional depots, may become economically attractive. By contrast, under low-demand conditions, depot deployment leads to structural overcapacity, characterized by limited utilization and negligible or negative marginal economic benefit.

Across all demand scenarios, a consistent pattern of diminishing marginal returns emerges as additional depots are introduced. While the first depot yields a substantial improvement in operational continuity, service throughput, and economic performance, the second depot contributes a markedly smaller incremental gain, even under high-demand conditions. This asymmetry suggests that the deployment of additional depots beyond the first requires more careful strategic justification, particularly with respect to whether prevailing economic conditions and mission timelines allow sufficient recovery of the associated upfront investments. Moreover, the marginal value of additional depots depends critically on how effectively they can be exploited by the operational environment. The mission planning problem with depots is intrinsically sensitive to constellation geometry and to depot placement. In this context, alternative design choices, such as repositioning a second depot to a high-demand customer orbit rather than collocating it in the nominal operational orbit, may significantly affect utilization rates and marginal economic contribution, and therefore warrant dedicated strategic analysis beyond the scope of the present study.

It is important to emphasize that the quantitative outcomes reported in this case study are inherently dependent on the adopted scenario configuration and economic assumptions, including the Iridium-

inspired constellation model, cost parameters, and the selected mission duration. This dependence does not constitute a limitation of the proposed framework. Rather, the objective of this strategic analysis is not to derive absolute conclusions regarding optimal depot numbers or universal profitability thresholds, but to demonstrate that the developed mission planning framework enables systematic, quantitative exploration of IOS architectural trade-offs. From an operator's perspective, the framework serves as a decision-support tool that can be instantiated with case-specific inputs to evaluate long-term infrastructure value. In this way, the framework supports consistent comparison of competing IOS architectures and reinforces its value as a flexible and extensible tool for strategic IOS infrastructure assessment.

The following section presents a second example of the type of strategic analysis enabled by the proposed framework, focusing on the trade-off between servicer versatility and fleet specialization under varying market demand.

8.3.2. Comparison of Servicer Fleet Architectures under Varying Demand

Deploying a single versatile servicer capable of executing all service types can reduce both initial investment and operating costs, particularly in low-demand environments where service opportunities are sparse and parallelization is limited. As demand increases, however, such a centralized architecture may become a bottleneck, constraining service throughput and limiting the system's ability to exploit concurrent service opportunities. In this regime, introducing specialization by deploying multiple smaller servicers, each dedicated to a specific service category, may improve operational efficiency through parallel execution, albeit at the cost of higher upfront investment and increased fleet-level operating complexity.

This trade-off motivates the following analysis, which compares the performance of two contrasting servicer fleet architectures under varying market conditions. The first configuration employs a single versatile servicer capable of executing all service types, while the second adopts a distributed fleet composed of four specialized servicers, each dedicated to a specific service category. Aside from the servicer fleet composition, all other system components are held constant across the two architectures, including the presence of a single orbital depot initially positioned at Node 3 of Orbit 1.

Each architecture is evaluated under the three demand scenarios introduced earlier, low, medium, and high, resulting in a total of six simulation runs. This setup enables a direct assessment of how versatility versus specialization affects operational efficiency, service throughput, and economic performance as demand scales. The defining characteristics of both fleet configurations are summarized in Table 8.19.

Table 8.19: Definitions of the versatile and specialized servicer fleet architectures.

Architecture	Versatile Fleet	Specialized Fleet
Number of Servicers	1	4
Servicer Architecture	Versatile (4 tools)	Specialized (service-specific tools)
Servicer Starting Nodes	Node 1	2 at Node 1, 1 at Node 2, 1 at Node 3
Depot Configuration	1 depot at Node 3	1 depot at Node 3

The simulation parameters used in the Rolling Horizon implementation, which are held constant across both servicer fleet architectures, are summarized in Table 8.20. As in the previous case study, the Planning Horizon and maximum idle time are adjusted across demand scenarios to reflect differences in task density and to maintain computational tractability under higher demand.

Results and Discussion

The simulations for the servicer configuration analysis are carried out using the same computational setup described in the depot-number trade-off analysis. The DelftBlue supercomputing environment and the Gurobi 12.0.3 solver configuration are retained without modification.

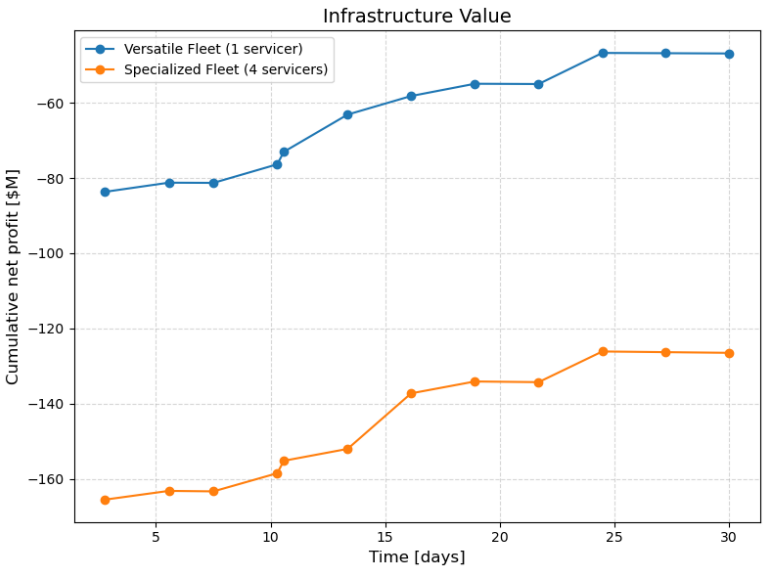
Across the six simulation runs, a total of 180 MILP optimizations were solved, with each full simulation requiring on average approximately 10h to complete. This corresponds to an average solution time of about 1200s per MILP iteration. The majority of iterations (approximately 69%) reached proven optimality within the imposed time limit, while the remaining 31% terminated due to the time constraint. For the latter, solution quality remained high, with an average MILP gap of only about 1.85%. This

Table 8.20: Simulation parameters used for the servicer fleet architecture comparison.

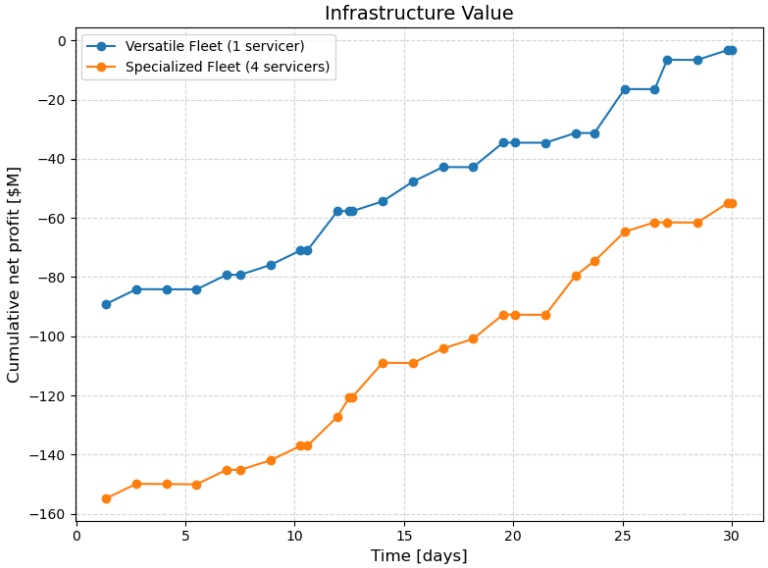
Parameter	Value	Units
Time step size (Δt)	32	min
Scheduling horizon	43,200	min
Planning horizon (low demand)	10,000	min
Planning horizon (medium demand)	6,000	min
Planning horizon (high demand)	4,500	min
Maximum idle time (low demand)	4,000	min
Maximum idle time (medium demand)	2,000	min
Maximum idle time (high demand)	1,000	min

indicates that even when the solver is unable to fully close the optimality gap within the available computational budget, the Rolling Horizon framework consistently produces near-optimal solutions suitable for strategic analysis.

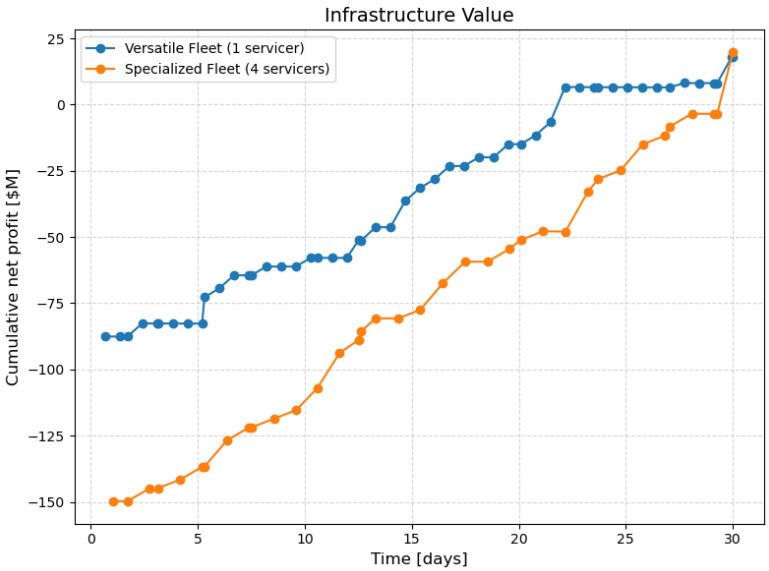
Unless stated otherwise, the graphical results presented in the following figures report time-evolving quantities at the resolution of the Rolling Horizon scheme, with each point on a curve corresponding to the outcome of a single MILP optimization at a given replanning iteration.



(a) Low-demand scenario

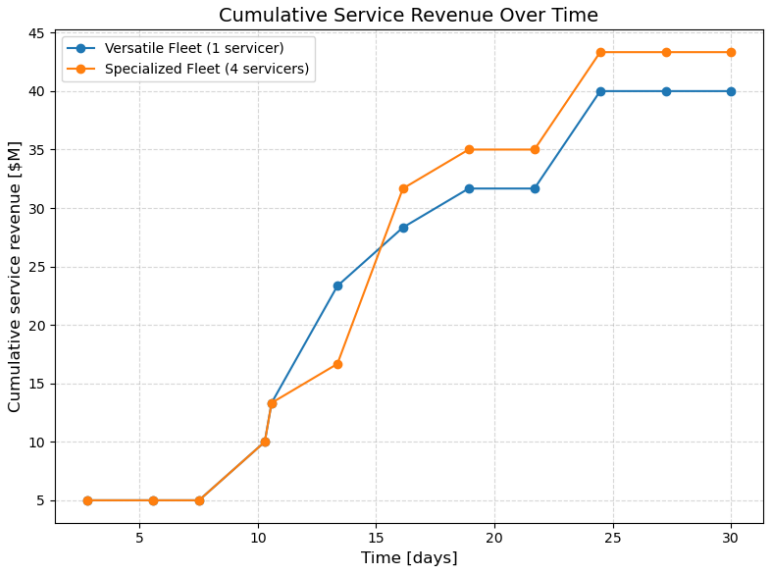


(b) Medium-demand scenario

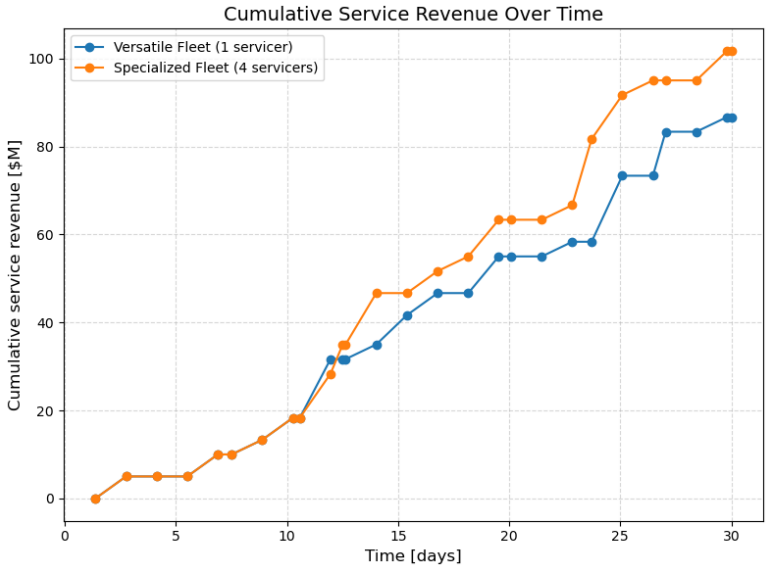


(c) High-demand scenario

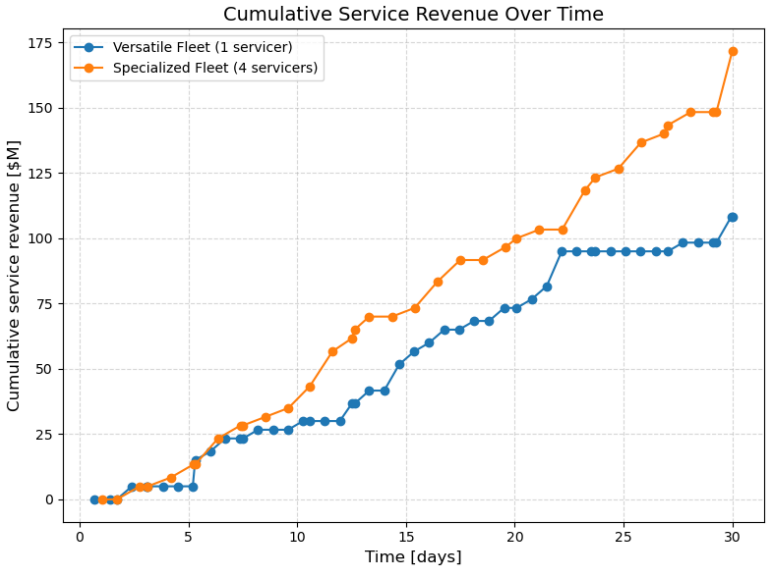
Figure 8.9: Cumulative net profit over the 30-day horizon across demand scenarios under the two servicer fleet configurations.



(a) Low-demand scenario



(b) Medium-demand scenario



(c) High-demand scenario

Figure 8.10: Cumulative service revenue over the 30-day horizon across demand scenarios under the two servicer fleet configurations.

Figures 8.9a, 8.9b, and 8.9c illustrate the evolution of cumulative net profit over the 30-day mission horizon for the versatile and specialized servicer fleet architectures under low-, medium-, and high-demand conditions, respectively. All results are evaluated under the revised economic assumptions introduced in the previous strategic analysis (Section 8.3.1), including amortized initial investment costs and increased service revenues. For the specialized fleet architecture, it is assumed that launch and PDM costs are amortized across three successive mission sequences, reflecting the higher re-usability potential of distributed fleets. Figures 8.10a, 8.10b, and 8.10c present the corresponding cumulative service revenue trajectories over the same horizon.

Under low-demand conditions, both fleet architectures exhibit similar revenue trajectories throughout the mission horizon (Figure 8.10a). The limited number of available service opportunities constrains achievable throughput, such that the additional parallelization capability of the specialized fleet remains largely underutilized. Over the full horizon, the versatile architecture completes 8 out of 9 available services, generating a cumulative service revenue of approximately 40.0M$, while the specialized fleet completes all 9 services and achieves a slightly higher cumulative revenue of 43.3M$. This difference indicates only a marginal advantage for fleet specialization in terms of gross revenue generation under low-demand conditions.

When costs are accounted for, the versatile single-servicer architecture consistently achieves higher cumulative net profit under low-demand conditions (Figure 8.9a). Although the specialized fleet generates a marginally higher cumulative service revenue, this advantage is barely perceptible once translated into net profit: the two profit trajectories remain closely aligned and exhibit an almost constant offset throughout the horizon. By the end of the mission, neither architecture reaches a positive infrastructure value. This outcome reflects the substantially lower upfront and operating costs associated with a single servicer and confirms that, in sparse demand environments, fleet specialization results in structural overcapacity rather than improved economic performance.

In the medium-demand scenario, differences between the two architectures become more pronounced. As shown in Figure 8.10b, the specialized fleet achieves a steeper revenue trajectory, completing all 21 available services and reaching a cumulative service revenue of approximately 101.7M$ by the end of the horizon. The versatile architecture, by comparison, completes 17 out of 21 services, generating a cumulative revenue of 86.7M$. This divergence reflects the ability of the specialized fleet to execute multiple services in parallel and to reduce idle time between tasks belonging to different service categories. In contrast, the single versatile servicer increasingly operates as a bottleneck, limiting achievable throughput despite the availability of sufficient demand.

These revenue gains partially offset the higher infrastructure costs of the specialized fleet. Consistent with Figure 8.9b, the versatile architecture maintains higher cumulative net profit over the mission horizon, but the gap between the two profit trajectories narrows progressively over time. This convergence indicates that the superior operational performance of the specialized fleet increasingly compensates for its higher capital and operating expenditures as demand grows. Nevertheless, under medium-demand conditions the specialized architecture does not surpass the versatile configuration in cumulative net profit, and neither architecture reaches break-even within the simulated horizon, although the versatile fleet approaches it closely toward the end of the mission.

The contrast between the two architectures is strongest under high-demand conditions. As shown in Figure 8.10c, the specialized fleet consistently outperforms the versatile configuration in cumulative service revenue, completing all 33 available services and reaching approximately 171.7M$ by the end of the horizon, compared to 26 services completed and a cumulative revenue of 108.3M$ for the single-servicer architecture. In this regime, demand density is sufficiently high for the specialized fleet's parallel execution capability to be continuously exploited, enabling sustained revenue accumulation through a higher effective service throughput over the full mission duration.

This operational advantage progressively offsets the higher capital and operating costs of the specialized architecture, leading to a pronounced convergence in cumulative net profit (Figure 8.9c). The versatile configuration initially maintains a higher infrastructure value due to its lower fixed cost base and reaches break-even earlier in the mission. However, as revenue accumulation accelerates for the specialized fleet, the profit gap closes steadily, and by the end of the 30-day horizon the specialized architecture achieves a slightly higher cumulative net profit.

This late crossover indicates that, under sufficiently high demand and sustained operations, fleet specialization can overcome its initial cost disadvantage and become economically competitive with, and eventually superior to, a single versatile servicer. At the same time, the marginal nature and

late timing of the overtake suggest that, under the current mission scenario, this advantage is highly sensitive to mission duration and demand persistence, underscoring the importance of aligning fleet architecture decisions with long-term operational expectations.

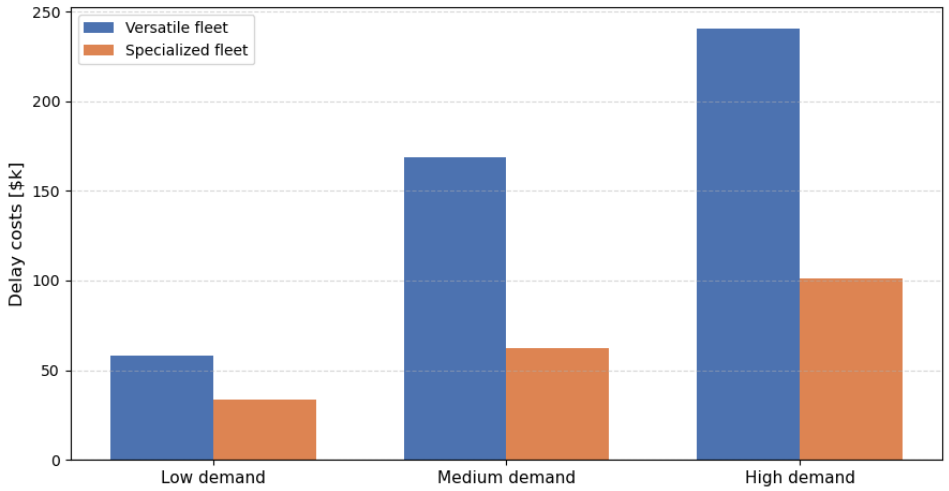


Figure 8.11: Aggregate delay costs incurred over the 30-day horizon for the versatile and specialized servicer fleet architectures under varying demand scenarios.

Isolating delay-related costs, Figure 8.11 reports the aggregate delay costs incurred by the versatile and specialized fleet architectures across demand scenarios. As in the previous case study, delay costs are interpreted qualitatively, as they are more sensitive to solver near-optimality effects in Rolling Horizon iterations than revenue-based metrics.

Across all demand scenarios, the specialized fleet consistently incurs lower aggregate delay costs than the versatile single-servicer architecture. This difference reflects the ability of the specialized fleet to execute services in parallel and to alleviate scheduling congestion, particularly when multiple tasks of different service categories are simultaneously active. By contrast, the versatile architecture concentrates all service execution on a single servicer, increasing waiting times and delay accumulation as demand rises.

Delay costs increase with demand level for both architectures, reflecting the higher number of executed services and greater exposure to scheduling conflicts. However, the relative gap between the two configurations widens as demand increases, indicating that the benefits of parallelism become more pronounced in higher-demand regimes. Under low demand, delay costs remain modest for both architectures, as limited task availability constrains congestion effects. In medium- and high-demand scenarios, the specialized fleet's ability to distribute workload across multiple servicers leads to a systematic reduction in delay penalties despite its higher overall activity level.

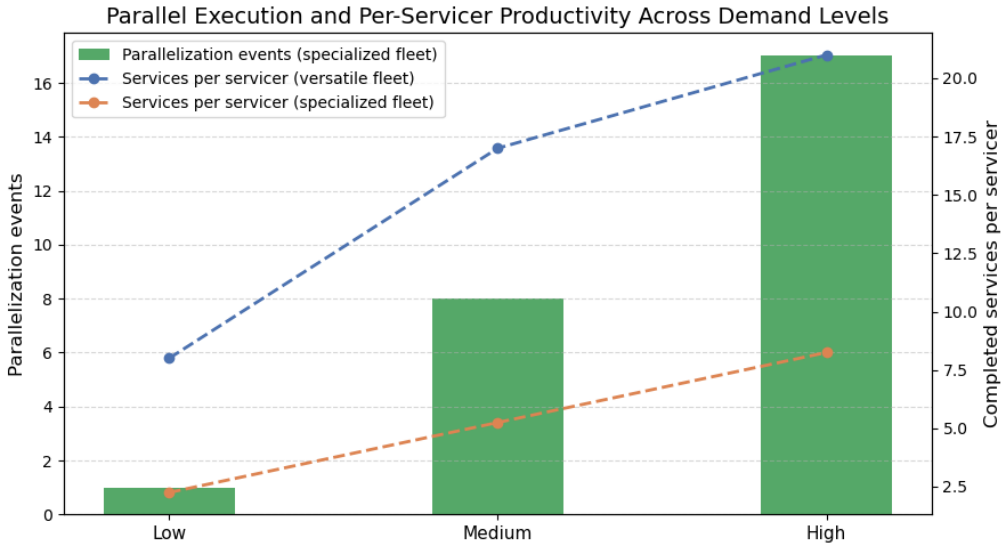


Figure 8.12: Parallel execution events and average per-servicer productivity across demand scenarios for the versatile and specialized fleet architectures.

To further illustrate the operational mechanisms underlying the revenue and profit differences discussed above, Figure 8.12 reports indicators related to service parallelization and per-servicer productivity across demand scenarios. The figure combines the number of parallel execution events enabled by the specialized fleet with the average number of completed services per servicer for both architectures.

Two services are classified as parallel if they are scheduled such that a single versatile servicer would be unable to complete one service and subsequently initiate the other at the scheduled time, once the required orbital transfers between service locations are taken into account. Under this definition, parallelization does not necessarily imply the execution of additional services, but may instead reflect the avoidance of additional waiting times and delay accumulation.

Under low-demand conditions, parallel execution opportunities are scarce, and per-servicer productivity in the specialized fleet remains significantly lower than in the versatile architecture, even though the specialized fleet completes one additional service overall. In this regime, the ability of the specialized fleet to execute services concurrently is rarely exploited, which is consistent with the marginal differences observed earlier in revenue generation and cumulative profit.

As demand increases, the number of parallel execution events enabled by the specialized fleet rises sharply, reflecting the growing prevalence of overlapping service opportunities across different service categories. At the same time, per-servicer productivity exhibits contrasting trends across the two architectures. For the versatile architecture, the number of completed services per servicer increases rapidly from low to medium demand, but grows more modestly from medium to high demand, indicating progressive saturation of the single-servicer capacity. By contrast, per-servicer productivity in the specialized fleet increases more gradually but consistently across all demand levels, as workload is distributed across multiple servicers and does not exhibit signs of saturation within the considered horizon. The simultaneous increase in parallel execution events and steady growth in per-servicer productivity indicates that the specialized fleet continues to exploit additional service opportunities as demand grows.

The lack of saturation and the persistent gap in per-servicer productivity relative to the versatile architecture can suggest remaining headroom for further utilization under higher demand levels or longer operational horizons. These observations motivate exploration of alternative fleet sizes and compositions, including intermediate configurations. For example, two or three specialized servicers covering subsets of service types may achieve a more favorable balance between parallel execution capability and servicer utilization, potentially yielding higher overall efficiency under sufficiently high demand.

To consolidate the results discussed above, Table 8.21 summarizes the key performance indicators across demand scenarios and servicer fleet architectures, highlighting service throughput, revenue

generation, delay behavior, and economic performance during mission execution.

Table 8.21: Summary of performance metrics for the servicer fleet architecture comparison across demand scenarios. Economic gains exclude initial investment costs.

Scenario	Architecture	Completed services	Revenue [M$]	Delay [k$]	Econ. Gain [M$]
Low demand	Versatile	8 / 9	40.0	58.1	39.2
	Specialized	9 / 9	43.3	33.7	41.3
Medium demand	Versatile	17 / 21	86.7	168.8	85.8
	Specialized	21 / 21	101.7	62.2	99.7
High demand	Versatile	26 / 33	108.3	240.2	107.3
	Specialized	33 / 33	171.7	101.1	169.6

Overall, the results demonstrate that the relative attractiveness of versatile and specialized servicer fleet architectures is strongly demand-dependent. Under low-demand conditions, a single versatile servicer achieves higher economic efficiency due to its lower initial investment and operating costs, combined with the limited availability of parallel execution opportunities. In this regime, fleet specialization leads to under-utilization of servicing assets and does not translate into proportional economic gains.

As demand increases, specialization enables higher service throughput, reduced delay accumulation, and faster revenue generation by alleviating single-servicer bottlenecks. Similar to the depot-number trade-off analyzed in the previous case study, the benefits of additional infrastructure materialize only when the added operational capacity can be effectively exploited; fleet specialization therefore becomes economically attractive only once demand is sufficiently high. Under medium- and high-demand conditions, these operational advantages progressively offset the higher initial and operating costs of the specialized fleet, leading to a convergence in economic performance between the two architectures over the simulated horizon.

Under high-demand conditions, this convergence culminates in a late crossover in cumulative net profit, with the specialized configuration slightly outperforming the versatile architecture by the end of the mission despite its higher cost base. Beyond this quantitative effect, specialization introduces an important qualitative advantage: by distributing operations across multiple assets, it reduces exposure to single-point failures, whereas failure of a lone versatile servicer would halt all servicing activity.

Although the specialized configuration does not exhibit strong economic dominance within the considered mission duration and scenario, its ability to match and marginally exceed the infrastructure value of the versatile architecture under high demand, combined with the benefits of concurrency, resilience, and scalability, motivates further exploration of alternative specialized fleet sizes and compositions.

However, these conclusions should not be interpreted as general prescriptions for optimal servicer fleet design. The relative economic performance of versatile and specialized architectures is highly sensitive to assumptions regarding servicer design, resupply logistics, and market characteristics. Alternative design choices, operational strategies, or economic assumptions may therefore lead to different comparative outcomes.

As in the previous case study, the objective of this strategic analysis is not to identify universally optimal IOS architectures, but to demonstrate the flexibility and analytical value of the proposed mission planning framework. By integrating orbital dynamics, logistics constraints, and stochastic service demand within a unified Rolling Horizon MILP formulation, the framework enables systematic, quantitative exploration of long-term servicer fleet design trade-offs across a wide range of scenarios. From an operator's perspective, the framework can be instantiated with case-specific assumptions to assess long-term operational and economic performance and to evaluate how alternative architectures perform under evolving technological and market conditions. While the results presented here represent only a subset of the possible design space, they illustrate how the framework can support informed decision-making and guide the development of scalable and economically resilient IOS infrastructures under uncertain demand.

9

Conclusion

This thesis develops an integrated optimization framework for planning and evaluating many-to-many In-Orbit Servicing operations in Low Earth Orbit. By combining a fully dynamic time-expanded logistics network, a Mixed-Integer Linear Programming formulation, and a Rolling Horizon decision-making architecture, the proposed framework jointly optimizes target sequencing, trajectory selection, resource allocation, and economic outcomes in the presence of stochastic service demand. It brings together realistic orbital dynamics, operational constraints, and infrastructure-level economics into a unified decision-support tool applicable to both operational routing and scheduling and long-term strategic assessment.

Research Questions Revisited

The research questions formulated in Chapter 1 provide a natural structure for synthesizing the main findings of this thesis.

RQ1: How can orbital motion, servicing opportunities, and IOS logistics be represented within a unified time-expanded network that preserves operational realism while remaining computationally tractable?

This thesis demonstrates that a fully dynamic discrete time-expanded network can effectively represent the spatiotemporal evolution of all system elements in a many-to-many IOS architecture while preserving computational tractability. By propagating the orbital motion of customers, servicers, and depots directly into the network and incorporating a tailored set of impulsive maneuver options, including coasting, multi-revolution phasing, and J_2-assisted cross-orbit drift transfers, the framework yields a physically consistent representation of rendezvous opportunities in Sun-synchronous orbits. Trajectory choices are discretized through the use of multiarcs, allowing the optimizer to select among alternative transfer realizations characterized by distinct time–fuel trade-offs grounded in established orbital mechanics formulations. This construction preserves linearity while embedding realistic orbital behavior directly into the logistics network, enabling seamless integration with mixed-integer optimization.

RQ2: How can IOS mission planning be formulated as a Mixed-Integer Linear Programming problem that jointly optimizes trajectory design, target sequencing, and resource allocation?

Building on the dynamic network representation, the thesis extends classical space logistics MILP formulations with IOS-specific constraints and decision variables. The formulation explicitly captures service windows and durations, consumable usage, depot resupply operations, tool compatibility, and infrastructure constraints across multiple servicers and depots. An integrated economic model is incorporated through a single profit-maximizing objective function that accounts for service revenues, operating costs, launch and PDM costs, and delay penalties. As a result, trajectory selection and target sequencing are jointly optimized alongside resource flows and economic performance. The case studies confirm that this formulation produces mission-realistic solutions that remain consistent with orbital dynamics, servicing logic, and economic objectives, thereby enabling coherent operational and strategic planning within a unified optimization framework.

RQ3: How can uncertainty in service demand be incorporated into an IOS mission planning framework through a Rolling Horizon approach, and what limitations does this introduce in terms of optimality, stability, and computational tractability?

Uncertainty in service demand is incorporated through a Rolling Horizon optimization scheme in which stochastic repair and debris-removal requests, modeled as Poisson processes, are revealed dynamically and integrated into successive planning iterations. This approach enables adaptive decision-making over extended mission durations, allowing schedules to be revised as new tasks appear while managing finite consumables, coordinating depot resupply operations, and preserving temporal consistency through explicit propagation of boundary states between iterations.

At the same time, the analysis highlights inherent limitations of Rolling Horizon optimization. Because each iteration optimizes over a finite look-ahead window, the resulting solutions exhibit myopic behavior and do not guarantee global optimality. Performance is sensitive to the choice of the Planning Horizon, which governs the trade-off between computational tractability and anticipatory capability, as well as to the Control Horizon, which mediates the balance between responsiveness to new information and decision stability across iterations.

RQ4: What operational and strategic insights can be derived from the proposed optimization framework with respect to IOS mission execution, infrastructure design choices, and their interaction with different service demand regimes?

After establishing correctness and internal consistency through a dedicated verification stage, the subsequent case studies demonstrate the applicability of the framework to both operational and strategic planning contexts.

At the operational level, the framework provides an IOS operator with a concrete, executable mission plan in the form of a time-ordered sequence of actions, including orbital transfers, servicing operations, and depot resupply events. In the presence of both deterministic and stochastic service demands, the Rolling Horizon architecture continuously updates this plan as new repair and debris-removal requests become available, while preserving feasibility with respect to service windows, maneuver constraints, and finite onboard resources.

The resulting mission plans demonstrate that effective IOS operations do not follow task activation order, but instead emerge from a prioritization of services based on orbital geometry, economic urgency, and downstream resource implications. The optimizer deliberately reorders tasks to cluster geometrically favorable transfers, exploit wide service windows, and position depot resupply events at points that sustain future operational capability. For an operator, this translates into clear guidance on *when* to service which customer, *which maneuvers* to execute between services, and *when* to interrupt operations for resupply in order to maximize overall mission value.

At the same time, the operational analysis highlights practical limitations that are directly relevant for mission execution. Under strict computational budgets, residual sub-optimality manifests primarily through increased accumulated delay rather than incorrect service selection or infeasible plans. This indicates that, in realistic operational settings, Rolling Horizon planning yields robust high-level mission structures and reliable action sequences, while fine-grained timing optimality may be traded off to ensure responsiveness and tractability. From an operator's perspective, this implies that the framework is well suited for real-time or near-real-time decision support, where adaptability and feasibility are prioritized over globally optimal timing in the presence of uncertainty.

At the strategic level, the framework enables systematic and quantitative assessment of how IOS infrastructure design choices interact with service demand intensity over extended mission horizons. The depot-number trade-off analysis shows that while the introduction of orbital depots substantially enhances operational flexibility and service throughput, particularly under medium- and high-demand conditions, these benefits come at the cost of significant upfront and operating expenditures. Within the 30-day horizon and mission scenario considered, these investments are only partially recovered; consequently, the economic attractiveness of additional depots depends critically on demand level, mission duration, and the underlying economic parameters. Under revised economic assumptions, however, depot-based architectures reach break-even performance, with the single-depot configuration emerging as the most economically favorable.

Similarly, the comparison between a single versatile servicer and a distributed fleet of specialized servicers indicates that increased parallelization capability does not automatically translate into higher

profitability. Although specialization improves service throughput and reduces delay accumulation as demand increases, these operational advantages are offset by higher infrastructure and operating costs within the simulated horizon and mission scenario, resulting in comparable economic performance only under high-demand conditions.

Taken together, these findings demonstrate that the performance and economic viability of IOS infrastructures are strongly demand-dependent and sensitive to strategic design choices. Rather than identifying universally optimal architectures, the case studies highlight the value of the proposed framework as a decision-support tool for exploring architectural trade-offs under case-specific inputs, quantifying marginal returns on additional infrastructure, and assessing how operational flexibility, resilience, and economic performance evolve under uncertain and changing market conditions.

Key Outcomes and Limitations

The results of this thesis demonstrate that the proposed framework provides a coherent and operationally meaningful approach to planning and evaluating many-to-many In-Orbit Servicing missions in Low Earth Orbit. By integrating orbital dynamics, logistics decisions, and economic evaluation within a single optimization environment, the framework bridges the gap between operational mission planning and long-term strategic assessment under uncertain service demand.

The key outcomes of this work can be summarized as follows:

- the development of a physically grounded, time-expanded network representation that captures orbital motion, servicing opportunities, and logistical interactions in Sun-synchronous LEO while remaining compatible with mixed-integer optimization;
- a mission-realistic MILP formulation that jointly optimizes trajectory selection, service sequencing, resource allocation, and economic performance across multiple servicers and depots;
- the implementation of a Rolling Horizon planning architecture that enables adaptive mission planning under stochastic service demand, producing consistent and executable action sequences while balancing computational tractability with solution quality;
- quantitative operational and strategic insights into IOS mission execution and infrastructure design, encompassing routing, scheduling, and resupply timing at the operational level, and the marginal value, utilization, and economic viability of alternative architectures under varying demand regimes at the strategic level;

At the same time, several limitations of the proposed framework must be acknowledged. The primary constraint arises from computational scalability. The combination of a fully time-expanded network capturing the orbital motion of all servicing elements and customers, embedded maneuver representations, mixed-integer decision variables and constraints accounting for IOS logistics, and Rolling Horizon replanning leads to large and computationally demanding optimization problems whose size grows rapidly with scenario complexity. As a result, even for moderate-scale instances, practical limits on memory usage and solution time necessitate compromises in temporal discretization, planning horizon length, mission duration, problem scale (in terms of servicers, depots, and customers), and optimality tolerances. These constraints delimit the size and complexity of scenarios that can be explored within a given computational budget and primarily affect solution granularity and fine-grained scheduling optimality, such as accumulated delays, rather than mission feasibility or structural validity.

A second limitation concerns the Rolling Horizon implementation itself. By construction, Rolling Horizon optimization does not guarantee global optimality and is sensitive to the choice of planning horizon length, potentially exhibiting myopic behavior. While this trade-off is intrinsic to adaptive planning under uncertainty, it influences solution quality and stability and must be carefully managed when interpreting operational results.

Furthermore, the current framework is restricted to LEO operations, with simplified representations of rendezvous execution, service processes, and resupply logistics. While these modeling choices are appropriate for the objectives of this thesis and enable tractable optimization, they abstract away several aspects of operational detail and limit the direct applicability of the framework to other orbital regimes.

Overall, this thesis establishes a modular, extensible, and physically grounded optimization environment for mission planning in many-to-many IOS systems. By integrating orbital mechanics, dynamic logistics, stochastic demand modeling, and economic assessment within a single decision-support tool,

the framework provides a foundation for both operational routing and scheduling as well as strategic architectural evaluation in the emerging LEO servicing market. The subsequent *Future Work* section outlines directions to enhance scalability, realism, and scope, further expanding the applicability and decision-support value of the proposed methodology.

9.1. Future Work

Future research should primarily address the computational challenges encountered when solving larger and more complex mission instances. Even for moderate-scale scenarios, significant compromises were required due to memory and time limitations, highlighting the inherent difficulty of the underlying optimization problem. Several directions can be pursued to improve scalability. One promising approach is the development of dedicated heuristics capable of producing high-quality solutions over longer scheduling horizons and for larger networks involving multiple servicers and orbital depots. Another possibility is to reformulate the time-expanded network using event-driven or variable-resolution discretization schemes, which may substantially reduce the number of decision points while preserving the temporal fidelity required for accurate maneuver modeling.

A second line of improvement concerns the extension of the maneuver model. The present framework relies exclusively on impulsive maneuvers tailored to near-circular LEO operations. Incorporating additional maneuver types, or moving beyond impulsive dynamics altogether, would broaden the range of feasible transfer strategies. In particular, representing low-thrust maneuvers could enable more fuel-efficient servicer architectures, albeit at the cost of increased trajectory complexity and longer transfer times.

To enhance operational realism, future work should integrate more detailed representations of rendezvous timing and costs, service execution processes, and resupply operations. The economic model could likewise be expanded and grounded in LEO-specific market assessments as new data becomes available.

The IOS planning problem could also be formulated as a multi-objective optimization problem. This would allow simultaneous consideration of multiple performance criteria, such as minimizing total mission time, reducing propellant expenditure, and maximizing net profit, thereby providing a richer depiction of the trade-offs inherent in designing and operating future servicing infrastructures.

Furthermore, incorporating a sustainability dimension into the optimization process represents a promising opportunity. As debris populations in LEO continue to rise, mission planning tools must increasingly account for the long-term environmental implications of servicing decisions. One approach is to embed a sustainability metric within a multi-objective formulation, enabling the optimizer not only to maximize economic performance but also to favor actions that mitigate debris-related risks. In particular, the selection of targets for Active Debris Removal could be guided by a debris index quantifying each object's contribution to collision probability, cascade potential, or long-term debris proliferation. Integrating such a metric into the objective function would allow the framework to explicitly balance economic performance with environmental responsibility, encouraging the removal of high-risk objects and supporting more sustainable operations in Low Earth Orbit.

Finally, extending the framework beyond LEO represents an important step toward increasing its applicability. Supporting missions in MEO, GEO, cislunar space, or multi-regime architectures would require refining the perturbation model to include additional effects beyond J_2, such as atmospheric drag, solar radiation pressure, and third-body perturbations. Such extensions would broaden the operational scope of the planning tool and enable its application to a wider range of mission concepts and infrastructure designs.

References

Adimurthy, V., Prasad, M. Y. S., & Shivakumar, S. K. (2007). Space mission planning and operations. *Current Science*, *93*(12), 1791–1811.

Arney, D., Sutherland, R., Mulvaney, J., Steinkoenig, D., Stockdale, C., & Farley, M. (2021). *On-orbit servicing, assembly, and manufacturing (osam) state of play* (tech. rep.). NASA.

Arney, D. C., & Wilhite, A. W. (2014). Modeling space system architectures with graph theory. *Journal of Spacecraft and Rockets*, *51*(5), 1413–1429. https://doi.org/10.2514/1.A32578

Aziz, S. (2013). Development and verification of ground-based tele-robotics operations concept for dextre. *Acta Astronautica*, *86*, 1–9. https://doi.org/https://doi.org/10.1016/j.actaastro.2011.11.004

Bo, X., & Feng, Q. (2011). Research on constellation refueling based on formation flying. *Acta Astronautica*, *68*(11), 1987–1995. https://doi.org/https://doi.org/10.1016/j.actaastro.2010.11.012

Brettle, H., Forshaw, J., Auburn, J., Blackerby, C., & Okada, N. (2019). Towards a future debris removal service: Evolution of an adr business model. *70th International Astronautical Congress*.

Cafolla, D. (2023). Performance analysis of torveastro, an outer space service robot. *Design Advances in Aerospace Robotics*, 90–99.

Cavaciuti, A. J., Heying, J. H., & Davis, J. (2022). In-space servicing, assembly, and manufacturing for the new space economy. *Aerospace Center for Space Policy and Strategy*, 2022–07.

Cerf, M. (2013). Multiple space debris collecting mission—debris selection and trajectory optimization. *Journal of Optimization Theory and Applications*, *156*(3), 761–796. https://doi.org/10.1007/s10957-012-0130-6

Chen, H., & Ho, K. (2018). Integrated space logistics mission planning and spacecraft design with mixed-integer nonlinear programming. *Journal of Spacecraft and Rockets*, *55*(2), 365–381. https://doi.org/10.2514/1.A33905

Chen, H., Lee, H. W., & Ho, K. (2019). Space transportation system and mission planning for regular interplanetary missions. *Journal of Spacecraft and Rockets*, *56*(1), 12–20. https://doi.org/10.2514/1.A34168

Chen, H., Sarton du Jonchay, T., Hou, L., & Ho, K. (2020). Integrated in-situ resource utilization system design and logistics for mars exploration. *Acta Astronautica*, *170*, 80–92. https://doi.org/https://doi.org/10.1016/j.actaastro.2020.01.031

Chiu, S. (2019). Promoting international co-operation in the age of global space governance – a study on on-orbit servicing operations. *Acta Astronautica*, *161*, 375–381. https://doi.org/https://doi.org/10.1016/j.actaastro.2018.07.019

Choi, E., & Ho, K. (2025). Orbital depot location optimization for satellite constellation servicing with low-thrust transfers. *Journal of Spacecraft and Rockets*, 1–10. https://doi.org/10.2514/1.A36386

Clark, S. (2016). *Iridium satellites rolling off assembly line in arizona*. Spaceflight Now. https://spaceflightnow.com/2016/07/13/iridium-satellites-rolling-off-assembly-line-in-arizona/

Curtis, H. D. (2020). *Orbital mechanics for engineering students* (4th ed.). Butterworth-Heinemann.

(DHPC), D. H. P. C. C. (2024). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2

Dishan, Q., Chuan, H., Jin, L., & Manhao, M. (2013). A dynamic scheduling method of earth-observing satellites by employing rolling horizon strategy. *The Scientific World Journal*, *2013*(1), 304047. https://doi.org/10.1155/2013/304047

Du, B., Zhao, Y., Dutta, A., Yu, J., & Chen, X. (2015). Optimal scheduling of multispacecraft refueling based on cooperative maneuver. *Advances in Space Research*, *55*(12), 2808–2819. https://doi.org/https://doi.org/10.1016/j.asr.2015.02.025

Dubanchet, V., Romero, J. B., Gregertsen, K. N., Austad, H., Gancet, J., Natusiewicz, K., Vinals, J., Guerra, G., Rekleitis, G., Paraskevas, I., et al. (2020). Eross project-european autonomous robotic vehicle for on-orbit servicing. *International Symposium on Artificial Intelligence, Robotics and Automation in Space,(i-SAIRAS'20)*.

Duke, H. (2021). On-orbit servicing. *Center for Strategic and International Studies: Washington, DC, USA*.

Ellery, A., Kreisel, J., & Sommer, B. (2008). The case for robotic on-orbit servicing of spacecraft: Spacecraft reliability is a myth. *Acta Astronautica*, *63*(5), 632–648. https://doi.org/https://doi.org/10.1016/j.actaastro.2008.01.042

Florescu, I. (2014). *Probability and stochastic processes*. John Wiley & Sons.

Froehlich, A. (2020). *On-orbit servicing: Next generation of space activities* (Vol. 26). Springer.

Grogan, P., Yue, H., & Weck, O. D. (2011). Space logistics modeling and simulation analysis using spacenet: Four application cases. In *Aiaa space 2011 conference & exposition* (p. 7346). https://doi.org/10.2514/6.2011-7346

Han, C., Zhang, S., & Wang, X. (2019). On-orbit servicing of geosynchronous satellites based on low-thrust transfers considering perturbations. *Acta Astronautica*, *159*, 658–675. https://doi.org/https://doi.org/10.1016/j.actaastro.2019.01.041

Han, P., Guo, Y., Li, C., Zhi, H., & Lv, Y. (2022). Multiple geo satellites on-orbit repairing mission planning using large neighborhood search-adaptive genetic algorithm. *Advances in Space Research*, *70*(2), 286–302. https://doi.org/https://doi.org/10.1016/j.asr.2022.04.034

Hastings, D. E., Putbrese, B. L., & La Tour, P. A. (2016). When will on-orbit servicing be part of the space enterprise? *Acta Astronautica*, *127*, 655–666. https://doi.org/https://doi.org/10.1016/j.actaastro.2016.07.007

Herron, M. (2023). *The development of in-space servicing, assembly, and manufacturing technology: Drivers, challenges, and policy implications*. The Pardee RAND Graduate School.

Ho, K., de Weck, O. L., Hoffman, J. A., & Shishko, R. (2014). Dynamic modeling and optimization for space logistics using time-expanded networks. *Acta Astronautica*, *105*(2), 428–443. https://doi.org/https://doi.org/10.1016/j.actaastro.2014.10.026

Ho, K., de Weck, O. L., Hoffman, J. A., & Shishko, R. (2016). Campaign-level dynamic network modelling for spaceflight logistics for the flexible path concept. *Acta Astronautica*, *123*, 51–61. https://doi.org/10.1016/j.actaastro.2016.03.006

Horsham, G. A. (2003). *Envisioning a 21st century, national, spacecraft servicing and protection infrastructure and demand potential: A logical development of the earth orbit economy* (tech. rep.). NASA.

Iridium Constellation LLC. (2013a). *Iridium next engineering statement* (tech. rep. No. SAT-MOD-20131227-00141) (Appendix 1). Federal Communications Commission (FCC). https://fcc.report/

Iridium Constellation LLC. (2013b). *Iridium next orbital debris mitigation plan* (tech. rep. No. SAT-MOD-20131227-00141) (Exhibit C). Federal Communications Commission (FCC). https://fcc.report/

Ishimatsu, T., de Weck, O. L., Hoffman, J. A., Ohkami, Y., & Shishko, R. (2016). Generalized multicommodity network flow model for the earth–moon–mars logistics system. *Journal of Spacecraft and Rockets*, *53*(1), 25–38. https://doi.org/10.2514/1.A33235

Jing, Y., Chen, X.-q., & Chen, L.-h. (2014). Biobjective planning of geo debris removal mission with multiple servicing spacecrafts. *Acta Astronautica*, *105*(1), 311–320. https://doi.org/https://doi.org/10.1016/j.actaastro.2014.10.003

Kopanos, G. M., & Pistikopoulos, E. N. (2014). Reactive scheduling by a multiparametric programming rolling horizon framework: A case of a network of combined heat and power units. *Industrial & Engineering Chemistry Research*, *53*(11), 4366–4386. https://doi.org/10.1021/ie402393s

Kreisel, J. (2002). On-orbit servicing of satellites (oos): Its potential market & impact. *proceedings of 7th ESA Workshop on Advanced Space Technologies for Robotics and Automation 'ASTRA*.

Lallo, M. D. (2012). Experience with the hubble space telescope: 20 years of an archetype. *Optical Engineering*, *51*(1), 011011. https://doi.org/10.1117/1.OE.51.1.011011

Li, W.-J., Cheng, D.-Y., Liu, X.-G., Wang, Y.-B., Shi, W.-H., Tang, Z.-X., Gao, F., Zeng, F.-M., Chai, H.-Y., Luo, W.-B., Cong, Q., & Gao, Z.-L. (2019). On-orbit service (oos) of spacecraft: A

review of engineering developments. *Progress in Aerospace Sciences*, *108*, 32–120. https://doi.org/https://doi.org/10.1016/j.paerosci.2019.01.004

Lillie, C. F. (2006). On-orbit assembly and servicing of future space observatories. *Space Telescopes and Instrumentation I: Optical, Infrared, and Millimeter*, *6265*. https://doi.org/10.1117/12.672528

Luo, Y.-Z., Tang, G.-J., Lei, Y.-J., & Li, H.-Y. (2007). Optimization of multiple-impulse, multiple-revolution, rendezvous- phasing maneuvers. *Journal of Guidance, Control, and Dynamics*, *30*(4), 946–952. https://doi.org/10.2514/1.25620

Luu, M., & Hastings, D. E. (2021). Review of on-orbit servicing considerations for low-earth orbit constellations. *ASCEND 2021*. https://doi.org/10.2514/6.2021-4207

Luu, M. A., & Hastings, D. E. (2022). On-orbit servicing system architectures for proliferated low-earth-orbit constellations. *Journal of Spacecraft and Rockets*, *59*(6), 1946–1965. https://doi.org/10.2514/1.A35393

Ma, B., Jiang, Z., Liu, Y., & Xie, Z. (2023). Advances in space robots for on-orbit servicing: A comprehensive review. *Advanced Intelligent Systems*, *5*(8), 2200397. https://doi.org/https://doi.org/10.1002/aisy.202200397

Mainelli, M., Vermont, C., Mathewson, P., Lecas, M., Eagleson, D., & Selka, H. (2023). In-orbit servicing and insurance markets: A symbiotic approach. *Proceedings of the International Astronautical Congress*.

Mason, A. (2023). *In-orbit satellite market: 7th edition*. Analysys Mason. https://www.analysysmason.com/research/content/reports/in-orbit-satellite-7th-nsr/

Nanjangud, A., Blacker, P. C., Bandyopadhyay, S., & Gao, Y. (2018). Robotics and ai-enabled on-orbit operations with future generation of small satellites. *Proceedings of the IEEE*, *106*(3), 429–439. https://doi.org/10.1109/JPROC.2018.2794829

Northern Sky Research. (2022, February). *Nsr's in-orbit services report projects $14.3 billion in revenues as non-geo constellations grow demand* (Press release).

Ogilvie, A., Allport, J., Hannah, M., & Lymer, J. (2008). Autonomous robotic operations for on-orbit satellite servicing. *Sensors and Systems for Space Applications II*, *6958*, 695809. https://doi.org/10.1117/12.784081

Opromolla, R., Grishko, D., Auburn, J., Bevilacqua, R., Buinhas, L., Cassady, J., Jäger, M., Jankovic, M., Rodriguez, J., Perino, M. A., & Bastida-Virgili, B. (2024). Future in-orbit servicing operations in the space traffic management context. *Acta Astronautica*, *220*, 469–477. https://doi.org/https://doi.org/10.1016/j.actaastro.2024.05.007

Roberts, T. G., & Kaplan, S. (2022). Space launch to low earth orbit: How much does it cost?

Sarton du Jonchay, T., Chen, H., Gunasekara, O., & Ho, K. (2021). Framework for modeling and optimization of on-orbit servicing operations under demand uncertainties. *Journal of Spacecraft and Rockets*, *58*(4), 1157–1173. https://doi.org/10.2514/1.A34978

Sarton du Jonchay, T., Chen, H., Isaji, M., Shimane, Y., & Ho, K. (2022). On-orbit servicing optimization framework with high- and low-thrust propulsion tradeoff. *Journal of Spacecraft and Rockets*, *59*(1), 33–48. https://doi.org/10.2514/1.A35094

Sarton du Jonchay, T., & Ho, K. (2017). Quantification of the responsiveness of on-orbit servicing infrastructure for modularized earth-orbiting platforms. *Acta Astronautica*, *132*, 192–203. https://doi.org/https://doi.org/10.1016/j.actaastro.2016.12.021

Satellite database. (2021). *Union of Concerned Scientists*.

Sethi, S., & Sorger, G. (1991). A theory of rolling horizon decision making. *Annals of Operations Research*, *29*(1), 387–415. https://doi.org/10.1007/BF02283607

Shen, H., & Tsiotras, P. (2012). Optimal scheduling for servicing multiple satellites in a circular constellation. In *Aiaa/aas astrodynamics specialist conference and exhibit*. https://doi.org/10.2514/6.2002-4907

Shukla, A., & Sawant, P. (2024). Plug-and-play drag sail module for leo satellites: Implementation and early testing of airdragmod (adm). https://doi.org/10.48550/arXiv.2408.13562

Silvente, J., Kopanos, G. M., Pistikopoulos, E. N., & Espuña, A. (2015). A rolling horizon optimization framework for the simultaneous energy supply and demand planning in microgrids. *Applied Energy*, *155*, 485–501. https://doi.org/10.1016/j.apenergy.2015.05.090

Sorenson, S. E., & Nurre Pinkley, S. G. (2023). Multi-orbit routing and scheduling of refuellable on-orbit servicing space robots. *Computers & Industrial Engineering*, *176*, 108852. https://doi.org/10.1016/j.cie.2022.108852

Sullivan, B. R., Akin, D. L., & Roesler, G. (2015). A parametric investigation of satellite servicing requirements, revenues and options in geostationary orbit. In *Aiaa space 2015 conference and exposition*. https://doi.org/10.2514/6.2015-4477

Taylor, C., Song, M., Klabjan, D., deWeck, O., & Simchi-Levi, D. (2007). A mathematical model for interplanetary logistics. *Logistics Spectrum*, *41*(1), 23–33.

U.S. Space Command. (2025). *Space-track.org*. https://www.space-track.org/

Vallado, D. A. (2001). *Fundamentals of astrodynamics and applications* (Vol. 12). Springer Science & Business Media.

Verstraete, A. W., Anderson, D., St. Louis, N. M., & Hudson, J. (2018). Geosynchronous earth orbit robotic servicer mission design. *Journal of Spacecraft and Rockets*, *55*(6), 1444–1452. https://doi.org/10.2514/1.A33945

Wakker, K. F. (2015). *Fundamentals of astrodynamics* [TU Delft Repository]. Delft University of Technology.

Woicke, S., Jipp, J., Steimle, C., Pokrupa, N., & Metrailler, L. (2025). Adrios clearspace-1: In-orbit demonstration of the removal of a non-cooperative spacecraft. *Proceedings of the 9th European Conference on Space Debris*.

Yao, W., Chen, X., Huang, Y., & van Tooren, M. (2013). On-orbit servicing system assessment and optimization methods based on lifecycle simulation under mixed aleatory and epistemic uncertainties. *Acta Astronautica*, *87*, 107–126. https://doi.org/https://doi.org/10.1016/j.actaastro.2013.02.005

Zhang, J., Luo, Y., & Tang, G. (2012). Hybrid planning for LEO long-duration multi-spacecraft rendezvous mission. *Science China Technological Sciences*, *55*(1), 233–243. https://doi.org/10.1007/s11431-011-4662-z

Zhang, T.-J., Yang, Y.-K., Wang, B.-H., Li, Z., Shen, H.-X., & Li, H.-N. (2019). Optimal scheduling for location geosynchronous satellites refueling problem. *Acta Astronautica*, *163*, 264–271. https://doi.org/https://doi.org/10.1016/j.actaastro.2019.01.024

Zhao, S., Gurfil, P., & Zhang, J. (2016). Optimal servicing of geostationary satellites considering earth's triaxiality and lunisolar effects. *Journal of Guidance, Control, and Dynamics*, *39*(10), 2219–2231. https://doi.org/10.2514/1.G001424

Appendices

Table 1: Arc summary by start/end orbit (IDs) and maneuver type.

Maneuver	Start orbit	End orbit	Count	Time (h)			ΔV (km/s)			ΔV per hour (km/s/h)		
				Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
Coasting	1	1	3	0.54	0.54	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	2	2	3	0.54	0.54	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	3	3	3	0.54	0.54	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	4	4	3	0.54	0.54	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	5	5	3	0.54	0.54	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	6	6	3	0.54	0.54	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	7	7	3	0.55	0.55	0.55	0.00	0.00	0.00	0.00	0.00	0.00
Coasting	8	8	3	0.54	0.54	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Combined Maneuver	1	2	1	1.06	1.06	1.06	0.23	0.23	0.23	0.21	0.21	0.21
Combined Maneuver	1	3	1	8.47	8.47	8.47	0.23	0.23	0.23	0.03	0.03	0.03
Combined Maneuver	1	4	1	12.44	12.44	12.44	0.23	0.23	0.23	0.02	0.02	0.02
Combined Maneuver	1	5	1	46.31	46.31	46.31	0.21	0.21	0.21	0.00	0.00	0.00
Combined Maneuver	1	6	1	46.31	46.31	46.31	0.21	0.21	0.21	0.00	0.00	0.00
Combined Maneuver	1	7	1	46.31	46.31	46.31	0.27	0.27	0.27	0.01	0.01	0.01
Combined Maneuver	1	8	1	46.31	46.31	46.31	0.25	0.25	0.25	0.01	0.01	0.01
Combined Maneuver	2	1	1	1.06	1.06	1.06	0.23	0.23	0.23	0.21	0.21	0.21
Combined Maneuver	2	3	1	7.41	7.41	7.41	0.23	0.23	0.23	0.03	0.03	0.03
Combined Maneuver	2	4	1	11.38	11.38	11.38	0.23	0.23	0.23	0.02	0.02	0.02
Combined Maneuver	2	5	1	46.31	46.31	46.31	0.20	0.20	0.20	0.00	0.00	0.00
Combined Maneuver	2	6	1	46.31	46.31	46.31	0.21	0.21	0.21	0.00	0.00	0.00
Combined Maneuver	2	7	1	46.31	46.31	46.31	0.27	0.27	0.27	0.01	0.01	0.01
Combined Maneuver	2	8	1	46.31	46.31	46.31	0.25	0.25	0.25	0.01	0.01	0.01
Combined Maneuver	3	1	1	8.47	8.47	8.47	0.14	0.14	0.14	0.02	0.02	0.02
Combined Maneuver	3	2	1	7.41	7.41	7.41	0.13	0.13	0.13	0.02	0.02	0.02
Combined Maneuver	3	4	1	3.97	3.97	3.97	0.22	0.22	0.22	0.06	0.06	0.06
Combined Maneuver	3	5	1	38.90	38.90	38.90	0.20	0.20	0.20	0.01	0.01	0.01
Combined Maneuver	3	6	1	40.49	40.49	40.49	0.20	0.20	0.20	0.01	0.01	0.01
Combined Maneuver	3	7	1	42.08	42.08	42.08	0.26	0.26	0.26	0.01	0.01	0.01
Combined Maneuver	3	8	1	46.31	46.31	46.31	0.23	0.23	0.23	0.01	0.01	0.01
Combined Maneuver	4	1	1	12.44	12.44	12.44	0.16	0.16	0.16	0.01	0.01	0.01
Combined Maneuver	4	2	1	11.38	11.38	11.38	0.16	0.16	0.16	0.01	0.01	0.01
Combined Maneuver	4	3	1	3.97	3.97	3.97	0.07	0.07	0.07	0.02	0.02	0.02
Combined Maneuver	4	5	1	34.93	34.93	34.93	0.21	0.21	0.21	0.01	0.01	0.01
Combined Maneuver	4	6	1	36.52	36.52	36.52	0.20	0.20	0.20	0.01	0.01	0.01
Combined Maneuver	4	7	1	38.11	38.11	38.11	0.26	0.26	0.26	0.01	0.01	0.01
Combined Maneuver	4	8	1	42.61	42.61	42.61	0.23	0.23	0.23	0.01	0.01	0.01
Combined Maneuver	5	1	1	46.31	46.31	46.31	0.20	0.20	0.20	0.00	0.00	0.00
Combined Maneuver	5	2	1	46.31	46.31	46.31	0.19	0.19	0.19	0.00	0.00	0.00
Combined Maneuver	5	3	1	38.90	38.90	38.90	0.19	0.19	0.19	0.00	0.00	0.00
Combined Maneuver	5	4	1	34.93	34.93	34.93	0.19	0.19	0.19	0.01	0.01	0.01
Combined Maneuver	5	6	1	1.59	1.59	1.59	0.23	0.23	0.23	0.14	0.14	0.14
Combined Maneuver	5	7	1	3.18	3.18	3.18	0.28	0.28	0.28	0.09	0.09	0.09
Combined Maneuver	5	8	1	7.67	7.67	7.67	0.25	0.25	0.25	0.03	0.03	0.03
Combined Maneuver	6	1	1	46.31	46.31	46.31	0.21	0.21	0.21	0.00	0.00	0.00
Combined Maneuver	6	2	1	46.31	46.31	46.31	0.20	0.20	0.20	0.00	0.00	0.00
Combined Maneuver	6	3	1	40.49	40.49	40.49	0.19	0.19	0.19	0.00	0.00	0.00
Combined Maneuver	6	4	1	36.52	36.52	36.52	0.19	0.19	0.19	0.01	0.01	0.01
Combined Maneuver	6	5	1	1.59	1.59	1.59	0.13	0.13	0.13	0.08	0.08	0.08
Combined Maneuver	6	7	1	1.59	1.59	1.59	0.28	0.28	0.28	0.18	0.18	0.18
Combined Maneuver	6	8	1	6.09	6.09	6.09	0.25	0.25	0.25	0.04	0.04	0.04
Combined Maneuver	7	1	1	46.31	46.31	46.31	0.27	0.27	0.27	0.01	0.01	0.01
Combined Maneuver	7	2	1	46.31	46.31	46.31	0.26	0.26	0.26	0.01	0.01	0.01
Combined Maneuver	7	3	1	42.08	42.08	42.08	0.25	0.25	0.25	0.01	0.01	0.01
Combined Maneuver	7	4	1	38.11	38.11	38.11	0.25	0.25	0.25	0.01	0.01	0.01
Combined Maneuver	7	5	1	3.18	3.18	3.18	0.10	0.10	0.10	0.03	0.03	0.03
Combined Maneuver	7	6	1	1.59	1.59	1.59	0.11	0.11	0.11	0.07	0.07	0.07
Combined Maneuver	7	8	1	4.50	4.50	4.50	0.27	0.27	0.27	0.06	0.06	0.06
Combined Maneuver	8	1	1	46.31	46.31	46.31	0.26	0.26	0.26	0.01	0.01	0.01
Combined Maneuver	8	2	1	46.31	46.31	46.31	0.26	0.26	0.26	0.01	0.01	0.01
Combined Maneuver	8	3	1	46.31	46.31	46.31	0.23	0.23	0.23	0.00	0.00	0.00
Combined Maneuver	8	4	1	42.61	42.61	42.61	0.22	0.22	0.22	0.01	0.01	0.01
Combined Maneuver	8	5	1	7.67	7.67	7.67	0.16	0.16	0.16	0.02	0.02	0.02
Combined Maneuver	8	6	1	6.09	6.09	6.09	0.14	0.14	0.14	0.02	0.02	0.02
Combined Maneuver	8	7	1	4.50	4.50	4.50	0.07	0.07	0.07	0.02	0.02	0.02
Phasing	1	1	15	2.15	5.38	8.60	0.31	0.64	1.27	0.04	0.19	0.59
Phasing	2	2	15	2.15	5.38	8.60	0.31	0.64	1.27	0.04	0.19	0.59
Phasing	3	3	15	2.15	5.37	8.60	0.31	0.64	1.27	0.04	0.19	0.59
Phasing	4	4	15	2.15	5.37	8.60	0.31	0.64	1.27	0.04	0.19	0.59
Phasing	5	5	15	2.15	5.38	8.60	0.31	0.64	1.27	0.04	0.19	0.59
Phasing	6	6	15	2.15	5.38	8.60	0.31	0.64	1.27	0.04	0.19	0.59
Phasing	7	7	15	2.19	5.48	8.77	0.31	0.63	1.26	0.04	0.19	0.57
Phasing	8	8	15	2.17	5.43	8.69	0.31	0.64	1.26	0.04	0.19	0.58