A HIGH-LEVEL FRAMEWORK FOR THE AUTONOMOUS REFUELING OF SATELLITE CONSTELLATIONS

A Thesis
Presented to
The Academic Faculty

by

Alexandros Salazar

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Aerospace Engineering

Georgia Institute of Technology
May 2007

A HIGH-LEVEL FRAMEWORK FOR THE AUTONOMOUS REFUELING OF SATELLITE CONSTELLATIONS

Approved by:

Prof. Panagiotis Tsiotras, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Robert Braun
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Craig Tovey
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: 12 January 2007

For my family.

ACKNOWLEDGEMENTS

First, I want to thank Professor Tsiotras, my advisor. I do not know whether other professors give their students the sort of latitude he gave me, but I thank Professor Tsiotras for the times he let me alter the direction of this work. I thank him for holding me to high, demanding standards, that my work might withstand scrutiny. And I thank him for his generous support for two and a half years—my best efforts would not have been as fruitful otherwise. I also thank Professors Braun and Tovey, the other members of my committee, for providing both useful and illuminating feedback. One of my best memories from Tech will be sitting with my committee and discussing the content of this work—I was, and still am, amazed at the breadth of their knowledge and at the unexpected insights that I heard in that room.

I extend my thanks to DramaTech and all my friends there—too many to name here, but every one of them a treasure. It is with great sadness that I also salute the memory of Greg Abbott, the artistic director of DramaTech for over twenty years, who passed away too soon on December 1, 2006. To him and to DT I owe the pleasure, when I was not a student, of having been a painter and a king, a producer and a beast. I am eternally grateful for those times of joy and art.

Two special lines of mention: first, a thank you to Alana, for her tart and good advice, for her kindness cloaked in acid, and her friendship over time. And, of course, my love to Brittany and her laughter, her tolerant smile and joyous eyes. Thank you for your constancy and calmness, and that quiet strength you have.

And last (not least!) I give thanks to my family. To my parents, for bearing with three years of my ambivalence before I finally set a course, and for giving me throughout my life unending wisdom, strength, support and love. To my brother Antonio, for bearing, as my roommate, with both ramblings and well-meant-yet-pompous counsel, and for still managing to teach me several lessons of his own. And to my brother Emilio, for inspiring me, through his own successes (so much greater than my own) to reach higher than I had before. You have made me, in small or in great part, the man I am today. I hope you are half as proud as I am grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Satellite constellations are an increasingly attractive option for many commercial and military applications; they provide a robust and distributed method of accomplishing the goals of expensive monolithic satellites. Among the many challenges that satellite constellations engender (challenges in control, coordination, disposal, and other areas), refueling is of particular interest because of the many methods one can use to refuel a constellation and the lifetime implications on the satellites.

The present work presents a methodology for carrying out peer-to-peer refueling maneuvers within a constellation. Peer-to-peer (P2P) refueling can be of great value both in cases where a satellite unexpectedly consumes more fuel than it was alloted, and as part of a mixed refueling strategy that will include an outside tanker bringing fuel to the constellation. Without considering mixed-refueling, we formulate the peer-to-peer refueling problem as an assignment problem that seeks to guarantee that all satellites will have the fuel they need to be functional until the next refueling, while concurrently minimizing the cost in fuel that the refueling maneuvers entail. The assignment problem is then solved via auctions, which, by virtue of their distributed nature, can easily and effectively be implemented on a constellation without jeopardizing any robustness properties.

Taking as a given that the P2P assignment problem has been solved, and that it has produced some matching among fuel deficient and fuel sufficient satellites, we then seek to sequence those prescribed maneuvers in the most effective manner. The idea is that while a constellation can be expected to have some redundancy, enough satellites leaving their assigned orbital slots will eventually make it impossible for the constellation to function. To tackle this problem, we define a wide class of operability conditions, and present three algorithms that intelligently schedule the maneuvers. We then briefly show how combining the matching and scheduling problems yields a complete methodology for organizing P2P satellite refueling operations.

CHAPTER I

INTRODUCTION AND OVERVIEW OF THE WORK

This thesis presents a high-level framework for the autonomous refueling of satellite constellations in a peer-to-peer setting. Specifically, we address the following question: given certain parameters of fuel sufficiency, what are the best way and the best time to carry out refueling operations within a constellation, assuming that the satellites are refueling each other without any assistance from a tug or from an external refueling spacecraft?

The purpose of this work is to develop the following elements of the framework:

1. A clear definition of fuel need for satellites.
2. A distributed way of assigning refueling satellites to fuel-deficient satellites.
3. An intelligent method of organizing the refueling maneuvers so as to interfere as little as possible with the functioning of the constellation.

In this chapter, we place this study in the historical context of satellite studies, and then provide a more detailed overview of the work that we aim to accomplish.

1.1 Historical Perspective

Satellite technology has, since its inception, essentially relied on the paradigm that one highly sophisticated satellite, thoroughly tested and designed specifically for a given purpose, is the best usage of resources for any work that needs to be done from space. This is not to say that no one gave any thought to distributed satellite systems. As early as the 1970s, papers can be found that explore the concept of space constellations [11]. However, even then, the constellations were only considered when there was no other alternative. An example of such a system is the GPS system, which would be impossible without the constellation of satellites that gives at all times several reference points to calculate location.

Recent years have seen a change in this line of thought. The clearest evidence of this can be seen in the telecommunications field. For decades, powerful geostationary satellites have been the norm, essentially since Arthur C. Clarke (of *2001: A Space Odyssey* fame) first proposed the concept in his 1945 paper "Extra-Terrestrial Relays: Can Rocket Stations Give World-wide Radio Coverage?" [17] In the 1990s, however, thought was given to the replacement of such high-powered machines with constellations of low-power satellites in low-Earth orbit (LEO), as witnessed by [36, 32, 24], to name a few.

Satellite Internet is not the only application that envisions the use of coordinated constellations. Some applications, such as oceanic altimetry [45], would rely on the distributed structure of satellite constellations to improve current results, which are obtained via single satellites or groups of satellites acting independently of each other. Others are entirely novel in their approach: for example, the concept of interferometric observatories in orbit [25, 16]. Finally, we should not neglect to mention the potential satellite constellations have in military applications, such as missile defense [2]. But all have in common that they push the boundaries of what can be expected and demanded from satellites.

The challenges that are brought about by this new focus on satellite constellations are numerous and novel. First and foremost, control of such constellations is an issue, since they must be able to coordinate in order to achieve multiple tasks. A great deal of research in recent years has therefore been devoted to optimal control of constellations, see for example [61]. Other challenges include initialization, [50], as well as the nontrivial question of disposal [3], which must be addressed if constellations with high numbers of satellites are to come in common use.

We focus this work on a different aspect of the satellite constellation concept. All the applications mentioned above share one common characteristic: they are essentially open-ended. There is no time-limit to how long one may wish to measure the levels of the ocean (especially amidst the growing concern about global warming), or carry Internet traffic, or, for that matter, observe the outer reaches of space. However the reliability and lifetime of satellites can be limited by their onboard fuel. Indeed, fuel consumption is inevitable even for the most static geostationary satellites, and is an acute issue when dealing with satellite formations, which may need to reconfigure regularly to transition from one task to the next. As another example, gradual deployment of constellations leads to space maneuvers of the satellites that are already operational when the new satellites are added in, so as to maximize the efficiency of the constellations [18]. Since such constellations will usually reside in LEO, atmospheric drag will lead to fuel consumption. Thus, the constellation (after reconfiguration) will be made up of two sets of satellites with differing amounts of fuel. Therefore, it is natural to wonder whether one may practically refuel satellites in orbit, and more specifically, whether it is possible to refuel only some of the satellites, with the fuel of the other satellites. That is the question that we address in this work.

1.2 Satellite Servicing

The idea of on-orbit satellite servicing is in itself nothing new. As far back as the early- to mid-1980s, proposals were made that included repair, refueling, and replacement of active satellites [4, 62], and by the early 1990s, studies were being conducted to evaluate the potential benefits of on-orbit refueling of US Air Force spacecraft [57].

On-orbit servicing is not, however, merely a conceptual notion. The almost legendary repair of the Hubble Space Telescope in 1993 is the prime (though not earliest [49]) example both of a successful servicing mission, and of all the reasons why incorporating servicing abilities into any design can be beneficial. It illustrates human error and its effects, even under the most exacting standards of accuracy in manufacturing. The reason the Hubble was unable to focus properly was the spherical aberration of the main mirror, which had been painstakingly polished over a period of years. Astoundingly, there were fully three serious failures in the process: first, the primary mirror contractor misread an instrument, second, it discarded conflicting results of the backup instruments as flawed, and third, the NASA overseers did not inquire into the reason for this dismissal. Thus, even one of the most carefully constructed devices in NASA's history was not immune from human error.

It was therefore fortunate that the Hubble was designed with servicing missions in mind. While there was a delay of three years until the telescope became fully operational, the first servicing mission repaired the optical system so the originally planned definition could be achieved, and did so without the need for extraneous trips. This type of flexibility is precisely what servicing capabilities can afford—in addition to additional flexibility to allow for upgrades, as discussed in [49].

Servicing itself can take many forms. The very word, "servicing," is used with various meanings in the literature, as discussed in [49], but for our purposes, we consider "servicing" to be any modification of the constellation, either for upkeep or upgrade purposes, that occurs after the constellation first goes online. At one end of the spectrum we can consider "servicing" operations that require no physical motion at all: for example, rerouting traffic in a communications constellation in LEO to compensate for the failure of a node in the communications network (i.e. a satellite). Such a problem was considered in [58], which also considered the design problem of building a network that could tolerate the failure of one of the nodes.

At the other end lie servicing operations that involve the complete, permanent reconfiguration of entire constellations. An example of this type of servicing was studied by the authors of [18], who considered the problem of optimally reconfiguring a constellation into another, larger constellation. Specifically, the problem was as follows: given a mission objective, an existing constellation, and a number of satellites that are to be added to this constellation, what is the optimal way to reconfigure and augment the the constellation in order to achieve the mission objective? As much work exists on optimal configurations for efficient terrestrial coverage by satellite constellations, the authors focused on optimally assigning orbiting and ground satellites to slots in the final (known) constellation. Their approach, of particular interest because of its similarity to the approach we take to peer-to-peer refueling in Chapter 4, was to create an assignment problem by seeking to minimize the ΔV incurred during the transfers.

In this thesis, we consider a different kind of servicing problem, namely the satellite constellation refueling problem. We consider this from a very high level of analysis, ignoring several challenging technical problems. In particular, we assume that satellites are able to rendezvous autonomously, and that they can exchange fuel in a microgravity environment. Neither of these assumptions is technically realistic: both the proximity and terminal phases of automated docking operations are still under active research [38, 64], while great improvements can still be made in microgravity fluid transfer [15, 14]. Finally, we also neglect spacecraft dynamics in the problem formulation, which we approach as an orbital mechanics problem only. Nevertheless, these are "merely" technical obstacles, and do not undermine the validity of the results presented herein; they only put in question the timetable of their potential usefulness, and perhaps the numerical exactitude of the simulations.

1.3 Satellite Constellation Refueling

Unlike other servicing problems, the question of refueling constellations was not specifically considered until the recent work of Shen and Tsiotras in [51, 60, 53, 54]. Constellation refueling can be achieved via two main methods: outside refueling, and peer-to-peer (P2P) refueling. The idea behind outside refueling is simply that a tanker spacecraft of some sort will travel to those satellites that need refueling and provide them with the necessary fuel. The main problem treated in [51] was the scheduling of those visits so as to minimize fuel consumption. The results presented are mainly heuristic, but they provide a very good guideline: the visits should be made sequentially or semi-sequentually along the orbit. Sequentially here means that the refueling spacecraft will refuel one satellite, and then move on to the next one along the line (measured by angular separation). Semi-sequentially means that at some point during the maneuver, the refueling satellite will jump to a satellite other than the nearest one, but will then continue to visit the remaining satellites sequentially.

It is clear that all constellations will eventually need outside refueling. However, it is possible that fuel will be consumed in an uneven manner within the constellation. For example, in a constellation of ten satellites, six satellites might be primarily needed for support while the remaining four are expected to actively change their orientation, or even their orbits, on a regular basis. In this case, the four moving satellites will consume a lot more fuel than the six static ones. While it is possible to send a tanker spacecraft to refuel those four satellites every time they run out of fuel, or even to swap in a fully fueled spare when one of the four most active satellites runs out of fuel, it may be more fuel-efficient to use the extra fuel from the less active satellites to provide the moving satellites, at the right time, with what they need, and only send a refueling spacecraft when all the satellites need to be refueled. It is this approach to refueling that we term peer-to-peer (P2P) refueling.

1.3.1 Peer-to-Peer Refueling

To study the possibilities of P2P refueling, a specific formulation is needed. What exactly is the objective in refueling those satellites? How do we determine that a satellite is in need of fuel, and how do we determine that it has fuel to spare? The work in [60, 53, 54] focuses on fuel equalization within a constellation: all satellites are considered to be identical, and the goal is to make sure they all have as close to the same amount of fuel as possible. It turns out, as we will discuss in Chapter 2, that in a circular constellation, the fuel cost of a rendezvous maneuver between two constellation satellites can be made arbitrarily small, given enough time. Therefore, the work in [53] focused on minimizing the total deviation from the constellation average while ignoring the fuel cost. This led to a simple and elegant theorem, namely that in this instance the symmetric matching was optimal. By symmetric matching, we mean matching the satellite with the most fuel to that with the least fuel, and so on.

In most realistic cases, the time granted for a refueling maneuver will not be sufficiently large to warrant the assumption of zero fuel cost. In [60, 54], the authors studied the problem of averaging fuel in this more complex situation. In order to include some fuel conservation elements into the problem, they focused on minimizing the deviation from the initial average. This led to a non-bipartite matching problem whose solution required implementing the Edmonds algorithm for general maximum matchings.

The relative costs of refueling satellites with an external refueling spacecraft as compared to refueling them via a peer-to-peer methodology were studied in [59] in the context of a mixed refueling strategy. Specifically, the authors proposed that while some external refueling was needed (since the fuel has to arrive to the constellation somehow), one way of distributing the fuel would be to distribute fuel via the refueling tanker to only some of the satellites, and use these satellites to refuel the remaining satellites. Again, the context was one of obtaining a constellation whose satellites had nearly equal fuel. It was shown in [59] that the use of a mixed-refueling strategy could indeed lead to fuel savings.

The work in [60, 53, 54] and [59] assumed that all refueling maneuvers were time-constrained. Specifically, the authors assumed that all refueling maneuvers started at the same time, and had to be completed by a set time. In addition, it was assumed that single-pairings would be used, that is, a given satellite could only refuel one other. Moreover the satellite that left its orbit would take half the allotted time for the forward trip, and the other half for the return trip. As an extension of this work, attention was given to asynchronous P2P maneuvers as part of mixed refueling strategies, that is to say, situations where the refueling maneuvers may start at different times [19]. In addition, the case where the forward and return trips were allotted different times were studied. The results provided further lowering of cost, and thus made mixed refueling strategies even more attractive and competitive with respect to single-tanker refueling. Nevertheless, these studies focused on the original formulation of the problem given in [60, 54], where the idea was to equalize the fuel distribution among the satellites, and as such are not necessarily applicable to the formulation proposed in this thesis. We do not conduct similar studies here, but we believe, due to the cost-lowering mechanism we introduce in Chapter 5, that this method is competitive as well.

We close this section by pointing out that, in this thesis, we are ignoring the risk involved in moving several satellites at once. It can be argued that it is better, if at all possible, to have each fuel sufficient satellite refuel as many satellites as possible. We are aware of no work in the literature that quantifies the risk of a maneuver to a satellite, so we are not able to carry out studies in that respect, nor is it the purpose of this thesis to do so. The problem of having one satellite rendezvous with several fuel-deficient satellites is an interesting problem for future study, but we do not concern ourselves with it in this work.

1.4 Thesis Objectives

As mentioned above, all the work on P2P constellation refueling so far has built around the formulation first presented in [53] and expanded thereafter. This formulation suffers from two large drawbacks: it assumes identical satellites, and it does not minimize the fuel cost directly. In this work, we propose an alternate formulation of the P2P refueling problem which can be used both to obtain an emergency solution in case of unexpected fuel consumption and as a building block we can use to design a mixed-refueling strategy. This formulation can be applied to constellations with differing satellites, and is split into two parts: the first deals with minimizing the refueling cost, and the second deals with minimizing the time the constellation is offline during the maneuvers. The combined results provide a framework for studying optimal P2P refueling.

1.4.1 Peer-to-peer Matching

The first part of the problem deals with matching satellites so as to have some of them refuel the others. We will present a notion of fuel sufficiency that is set by the constellation operators, possibly based on the expected fuel need until the next external refueling. This allows us to bypass a complicated definition of fuel sufficient and fuel deficient satellites, as they are decided ahead of time. Given the threshold of fuel sufficiency for each satellite, we will define a pure fuel cost minimization problem by defining fuel deficient and fuel sufficient satellites (for refueling purposes) in such a way that the associated constellation graph will naturally be a bipartite graph.

Since the graph is bipartite, we will show that the problem of minimizing fuel consumption while guaranteeing that all fuel deficient satellites become fuel sufficient after the maneuvers are completed becomes an assignment problem. Since the assignment problem is easily solvable via many methods, we choose the one most appropriate to the structure of a satellite constellation: the method of auction algorithms. By virtue of their easily distributable nature, these algorithms take full advantage of the satellite constellations structure, and help maintain its robustness by avoiding reliance on a single central computing satellite. The final result will be a matching between fuel sufficient and fuel deficient satellites; a matching, moreover, which will be the cheapest matching in terms of fuel.

1.4.2 Maneuver Scheduling

It would make sense that a satellite constellation, made up as it is of many satellites, would have some inherent redundancy and would be robust to failure of some of its components. Ideally, any one of the satellites could fail and the constellation would be able to continue functioning. Nevertheless, the number of satellites that leave their orbital slots during a refueling maneuver (and that are therefore useless to the constellation during that time) may be quite a bit larger than this robustness will compensate for. As such, it becomes a matter of interest to minimize the time that the constellation will be unable to function by properly sequencing the maneuvers dictated by the matching obtained as a solution to the assignment problem discussed above.

The second part of this thesis treats this problem, which is quite different from the standard scheduling problems seen in the literature. We prove certain theorems which allow us to search for globally optimal solutions while restricting ourselves to a finite set. This set is still very large, however, and we therefore are limited to heuristics when we search it. Nevertheless, we propose three algorithms, each of which has desirable properties either in terms of the downtime of the constellation, or in terms of the fuel consumption. Indeed, we show that with proper sequencing, we may add time to the maneuvers obtained from the matching, and thus lower their fuel cost.

1.5 Thesis Outline

This thesis is divided into six Chapters including this Introduction. The remaining Chapters cover, respectively:

Chapter 2: An introduction to the basics of the concepts used throughout the thesis. In particular, a quick review of optimal two-impulse satellite rendezvous, a brief introduction to assignment problems as linear programs, and an overview of the scheduling literature as relevant to the scheduling problem to be developed in Chapter 5. This chapter will serve both as a foundational chapter and as a literature review.

Chapter 3: The formulation of the P2P refueling problem as an assignment problem, including calculations of the "benefits" as fuel cost.

Chapter 4: The solution of the assignment problem via auction methods. The various forward auction algorithms that can be used to solve this assignment problem are presented, with particular attention being paid to the asynchronous auction algorithm, which is robust to communication failures and delays; numerical examples for the various algorithms are given.

Chapter 5: A formulation for the scheduling problem, and heuristic algorithms to solve it. Conditions for operability of a constellation are developed, two scheduling heuristics and one fuel-cost reduction algorithm are introduced, and their desirable properties proven.

Chapter 6: The conclusions we can draw based on this work, as well as some proposed future directions of study for autonomous peer-to-peer refueling.

CHAPTER II

BACKGROUND AND LITERATURE REVIEW

This thesis is based on several disparate subjects, and as such may not be accessible to those familiar with any one of them without some introduction to the others. This chapter is therefore dedicated to presenting the rudiments of knowledge required to place the contributions of this work in context, as well as a literature review of each of the areas. We begin by a section on optimal two-impulse maneuvers, and discuss phasing maneuvers, which yield the optimal N-revolution transfer for co-orbital satellites in a circular orbit. We then briefly discuss linear programming in general and the assignment problem in particular, presenting some of the methods for the solution of the problem that were not used in this work. Finally, we present an short overview of scheduling problems.

2.1 Optimal Two-Impulse Transfers

The simplest problem in gravitational mechanics is the two-body problem: analyzing the relative motion of two masses under the influence of gravity—masses such as the earth and the sun. It was observed by Kepler [29] and mathematically confirmed by Newton [37] that the planets moved around the sun in ellipses, that the line joining the planet to the sun sweeps equal areas in equal times, and that the square of the period is proportional to the cube of the semi-major axis. Newton in fact generalized these observation, known as Keplers Three Laws of Planetary Motion, to conic sections, which occur depending on the energy-level of the orbit: ellipses are the lowest energy orbits, parabolas are the singular-point orbits, and hyperbolas are high-energy orbits [6].

In the context of satellite rendezvous, hyperbolas and parabolas, being higher energy orbits, are too expensive in terms of fuel to be used as trajectories, so we will limit the discussion to elliptical orbits. Even more restrictive, we will only consider two-impulse trajectories, since the fuel conservation resulting from calculating optimal trajectories with more than two impulses is offset by the greater computational burden [55]. We are therefore concerned with the Lambert problem, also known as the Gauss problem [6]: given two fixed points P_1 , P_2 in space at a distance r_1 and r_2 from an attracting body M , a direction of flight around M, and a time of flight t_f assumed to be long enough to allow elliptical orbits, determine the transfer orbit that will take a spacecraft from P_1 to P_2 . Direction of flight is important because given any two points in space and a mass around which they are orbiting, there are two elliptical orbits that will take a spacecraft from P_1 to P_2 in time t_f, each with a different direction of motion.

Since satellites occupy well-known orbits, the Lambert problem is exactly a rendezvous problem: P_1 is the initial location at a time t_0 of the satellite that will move, which we term the active satellite, and P_2 is the final location at time $t_0 + t_f$ of the satellite that does not move (in the rotating reference frame of the orbit), which we term the passive satellite. The time of flight was shown by Lagrange (who was confirming a hypothesis by Lambert) to depend strictly on three quantities [44]: the semi-major axis of the transfer orbit, the sum of the distance of each of the two points from the central attracting body, and the distance separating those two points. Mathematically,

$$\sqrt{\mu} t_f = a^{3/2}[2N\pi + \alpha - \beta - (\sin\alpha - \sin\beta)], \tag{1}$$

where a is the semi-major axis of the transfer orbit, μ is a gravitational parameter depending on the mass of body M, and α and β are defined as:

$$\sin\frac{\alpha}{2} = \sqrt{\frac{s}{2a}}, \qquad \sin\frac{\beta}{2} = \sqrt{\frac{s-d}{2a}}, \tag{2}$$

where d is the third side of the triangle $P_1 P_2 M$ (the so-called space triangle), and $s = (r_1 + r_2 + d)/2$ [43]. In [42], the author provides both the method of quadrant determination of the angles α and β and a geometrical interpretation of their meaning.

The original Lambert theorem was derived for the case where the spacecraft stays on the transfer orbit less than a full revolution [44]. The more general case, known as the N-revolution transfer orbit problem was studied by Prussing [43]. In this case, for each transfer time t_f , $2N + 1$ possible orbits exist, where N is the maximum number of revolutions that can be completed along any transfer orbit. There are N more orbits than the maximum number of revolutions because for each $N \geq 1,$ there are two possible orbits: one with large eccentricity, and one with small eccentricity.

A standard measure of the cost of a maneuver is the velocity increment ΔV required to carry it out—specifically, it is the total impulsive change in velocity that must be applied to a spacecraft in order to move it from its initial orbit to its final orbit. The authors showed in [52] that the ΔV

required to transfer a spacecraft from a circular orbit passing through point P_1 onto a given transfer orbit (one of the $2N + 1$ possible ones), and back onto a circular orbit passing through point P_2 depends only on the semi-major axis of the transfer orbit, given a fixed transfer time. Based on this observation, the authors developed an algorithm to find the optimal two-impulse orbit and its associated ΔV by checking only two of the $2N + 1$ possibilities shown to exist in [43].

We note two things. First, what complicates the Lambert problem is the specific, pre-set time of flight and the initial and final positions. If the time of flight and final location on the target orbit are free, the optimal two-impulse transfer between two circular orbits is given by the Hohman transfer, which consists of two apsidal impulses. Here, apsidal impulses mean impulses at the periapsis and apoapsis of the transfer orbit, parallel to the velocity of the spacecraft. Second, all these studies were based on numerical solutions to the implicit Equation (1). There is no known closed form solution to either the zero revolution or N-revolution Lambert problem, except in the specific case where P_1 , P_2 and t_f are chosen so as to allow a Hohman transfer; thus, an extension of Battin's method for solving the Gauss problem [7] was introduced by the authors of [52] in [56] in order to be able to solve N-revolution Lambert problems.

Despite these computational complexities, the authors showed numerically in [52] that given enough time, a coasting rule could be applied which would lead, in the case where $r_1 \neq r_2$, to Hohman transfers. If insufficient time was allotted, the coasting rule could still be applied to reduce the calculated cost of a fixed end-time transfer. Because in this thesis we limit our numerical studies to circular constellations (although the methodology we develop is not limited to them), we pay careful attention to the special case where $r_1 = r_2$, described in the next section.

2.1.1 Calculation of ΔV: $r_1 = r_2$

There is a simple solution to the Lambert problem in the case where $r_1 = r_2$ and the location of P_2 and time t_f are selected so that $P_2 = P_1$ and t_f allows an integral number of revolutions of an orbit with slightly higher or slightly lower apoapsis or periapsis, respectively. In this case, a closed form solution does exist for the optimal transfer in the majority of cases, namely the phasing transfer. Like the Hohman transfer, this type of maneuver also uses tangential initial and final impulses. The idea is to move the active satellite to an elliptical orbit by either raising the apoapsis or lowering the periapsis by an appropriate amount, and then waiting for the passive satellite to rendezvous with the active satellite at the point where the active satellite's new orbit tangentially intersects the original orbit. Thus, the rendezvous occurs at the same point in space as the departure. The return maneuver follows the same principle in reverse.

We mention in passing that the Euler-Hill equations, which are very useful to describe the motion of objects that are close to each other in orbit, do not apply in this particular case. In particular, the angular separation of the two satellites makes linearization problematic, while the phasing solution expounded below is an exact solution.

Computing the cost of a phasing transfer is much simpler than computing the cost of a multiple-revolution Lambert transfer and can be done analytically. Specifically, the ΔV for satellite i to rendezvous with satellite j via phasing, denoted by ΔV_{ij}^{i}, can be obtained analytically as follows [33]

$$\Delta V_{ij}^{i} = \sqrt{\frac{\mu}{r}} \left| \sqrt{2 - \left(\frac{z + \kappa}{z - \tilde{\theta}_{ij}/2\pi} \right)} - 1 \right|, \tag{3}$$

$$z = \left\lfloor \frac{t_{ij}^{i}}{2\pi} \sqrt{\frac{\mu}{r^3}} + \frac{\tilde{\theta}_{ij}}{2\pi} \right\rfloor, \tag{4}$$

where r is the radius of the orbit and

$$\tilde{\theta}_{ij} = \begin{cases} \theta_{ij} \bmod 2\pi, & \text{if } -\pi \leq \theta_{ij} \bmod 2\pi \leq \pi, \\ \theta_{ij} \bmod 2\pi - 2\pi, & \text{if } \theta_{ij} \bmod 2\pi > \pi, \\ \theta_{ij} \bmod 2\pi + 2\pi, & \text{if } \theta_{ij} \bmod 2\pi < -\pi, \end{cases} \tag{5}$$

and where,

$$\kappa = \begin{cases} -1 & \text{if } a > r \text{ and } \tilde{\theta}_{ij} > 0, \\ 1 & \text{if } a < r \text{ and } \tilde{\theta}_{ij} < 0, \\ 0 & \text{otherwise}, \end{cases} \tag{6}$$

where a is the semi-major axis of the transfer orbit.

While the above calculations are useful, it is important to keep in mind that the phasing maneuver is not always the optimal two-impulse transfer between two satellites. As shown in Figure 1, there are small segments in the time constraints where the general Lambert-based solution (dashed

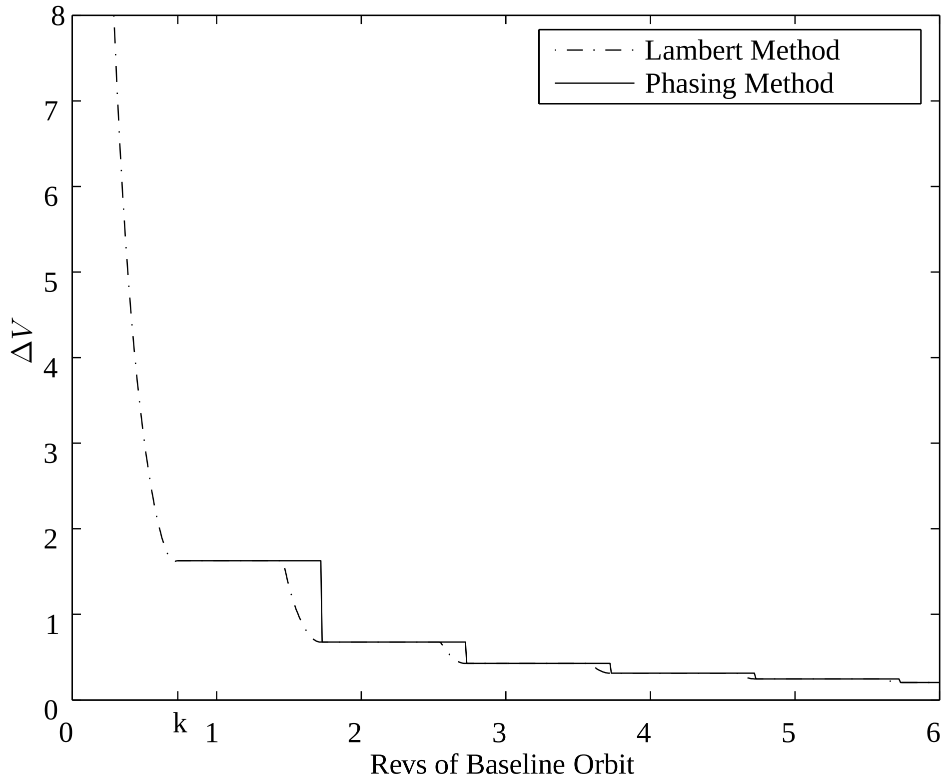

Figure 1: Comparison of ΔV between the general (Lambert-transfer) and phasing rendezvous, for a separation angle $\theta_{ij} = 100°$. For the majority of cases a phasing transfer is optimal. Even in cases when it is not optimal, the degree of sub-optimality is small.

line) is less expensive than the phasing solution (solid line). In addition, if the time allotted to the maneuver is less than t_{min}, a phasing maneuver is not even possible. However, it is also clear that the phasing solution is optimal in the vast majority of cases, thus making it a reasonable approximation of the real cost, especially when a large number of revolutions is allowed. If needed, both the phasing and the Lambert-based ΔV can be computed, and the lowest one chosen.

We now move on to the second area that this thesis touches, namely, linear programming.

2.2 Linear Programming and Assignment Problems

Linear programming (LP) is the study of solutions to problems of the form:

$$\min \quad c^T x \tag{7}$$

$$\text{such that:} \quad Ax \geq b \tag{8}$$

where $c, x \in \mathbb{R}^m,\ b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m\times n}$. Linear Programming is thus an optimization problem. While the problem above seems rigid, it is in fact a specific form of a general class of problems and has very wide applications [10]. To name a few, for completeness, linear programming can be used to:

- Schedule shifts for nurses at a hospital.
- Find optimal network flows to carry a commodity from a set of sources to a set of sinks.
- Create a balanced diet with foods of known nutritional content (the so-called diet problem, also sometimes referred to as the "virtual carrot" problem).
- Assign n people to n jobs so that the cost of labor is minimized.

Those four applications barely scratch the surface of the uses of LP. It was of great practical significance, therefore, to develop efficient algorithms to solve these problems, since no closed-form solutions are known. The most general and powerful of these algorithms is known as the simplex method, and was introduced by George P. Danzig in 1947 [10]. The idea of the algorithm is based on the geometry of the problem.

To begin with, we note that the closed set defined by

$$Ax \geq b \tag{9}$$

defines a polyhedron in $\mathbb{R}^n$, as each of the inequalities, when set to equality, defines a plane. The polyhedron may be feasible or infeasible: feasible if it contains at least one point, infeasible otherwise. In addition, it may be bounded or unbounded. If bounded, it is referred to as a polytope. For simplicity of exposition, we will restrict discussion of the simplex method to polytopes, although the algorithm is quite capable of detecting an unbounded polyhedron and, where applicable, state that the solution is unbounded. Note that the polytopes defined as above are all closed. It is shown in [10], which is the main source for this brief exposition, that they are also convex, which is critical to many of the proofs of the validity of the simplex algorithm, as well as those, perhaps more important, about duality theory.

Geometrically, minimizing the function $c^T x$ over a polytope P is equivalent to taking a hyperplane perpendicular to the vector c, and translating that plane as far as it can go in the c direction while still having at least one point on the plane also in P. Such a point is a solution to the optimization problem. Our 3-dimensional intuition generalizes well in this particular instance, and we have two easily understood cases:

1. There is a unique solution, i.e., it is possible to push the hyperplane far enough so that only one point is both on the hyperplane and in the polytope. This point is one of a family of points called basic feasible solutions in the context of the simplex method, and more simply vertices in the geometrical sense. They have the property that at least n of the constraints are active, meaning that they hold at equality.

2. There are infinitely many solutions. In three dimensions, this is equivalent to there being an edge or a face of the polytope that is parallel to the hyperplane defined by c. Note that in this case, there are still basic feasible solutions in the solution set: the vertices at the limits of the line or plane.

From the geometrical intuition above, we can explain the simplex method as a method of first, finding a vertex of P (i.e. a basic feasible solution) and second, traversing the vertices until an optimal vertex if sound. The simplex method, then, follows the edges of the polytope from vertex to vertex until it finds a one that it can verify is an optimal solution. At each vertex, the simplex applies a specific rule, known as a pivoting rule, in order to select which of the possible vertices to visit next. While for each problem, a specific pivoting rule may be more appropriate, there currently is no way to know a priori which pivoting rule is appropriate for which problem.

This leads into one of the problems of the simplex algorithm: while it is certainly effective, it has poor guaranteed performance. In particular, Klee and Minty created examples of transformed cubes in $\mathbb{R}^n$ that would lead the simplex algorithm, using a greedy pivoting rule, to visit all $2n$ vertices of the cube before reaching an optimal solution [31]. Since then, researchers have found pathological examples of polyhedrons for each pivoting rule currently in use that will force the simplex method to visit all vertices.

A different approach, that of continuous interior point methods, was first introduced by Khachiyan in 1979 in the form of the ellipsoid algorithm [30]. This algorithm was revolutionary in that it was a provably polynomial algorithm (meaning that it found a solution quickly, in the sense of complexity theory), but its practical performance was very poor compared to the simplex method (meaning that it found a solution slowly, in the sense of usefulness theory). A better algorithm was proposed by Kamarkar in 1984 [28]. This method was the first competitive solution method for LP problems with a polynomial running time guarantee. By competitive, we mean that the practical implementation of the algorithm surpassed the speed of the simplex implementations of the time by a factor of 50—certainly a tremendous gain [65]. The resulting competition between the two methods has led to great increases in performance of the various implementations of both interior-point and simplex-like algorithms.

2.2.1 Duality

Duality theory is an integral part of mathematical programming, and its particularly elegant results in the case of linear programming form the basis of both the theoretical refinement of LP theory and the development of highly efficient implementations of the simplex method. Briefly, it can be shown that every linear program has a dual problem. In the case of a problem in the form (7)-(8), the dual problem is given by:

$$\max \quad p^T b \tag{10}$$

$$\text{subject to:} \quad p^T = c^T \tag{11}$$

$$p \geq 0. \tag{12}$$

The dual problem has several important properties, namely:

1. The dual of a minimization problem is a maximization problem, and vice versa.

2. The dual of the dual of a problem is the problem itself.

3. If $x \in P$ and $p \in Q$, where Q is the dual polyhedron (Clark's theorem states that if P is a polytope, Q must be an unbounded polyhedron, assuming both are feasible), we have:

   $$c^T x \geq p^T b, \tag{13}$$

   a result known as *weak duality.*

4. If x and p are optimal solutions to the primal and dual problem respectively, then

$$c^T x = p^T b \tag{14}$$

a result known as *strong duality*.

5. If x and p are optimal solutions to the primal and dual problems, respectively, then

$$p_i(a_i^T x - b_i) = 0 \qquad \forall i = 1, 2, \ldots, n \tag{15}$$

$$(c_j - p^T A_j)x_j = 0 \qquad \forall j = 1, 2, \ldots, m \tag{16}$$

where

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & \cdots & A_n \end{bmatrix}, \tag{17}$$

a result known as *complementary slackness.*

We take particular note of complementary slackness, as it is a crucial element of the auction algorithm which we describe in detail in Chapter 4. All these results are well-established in the field, and were introduced by von Neumann in 1947 and Gale, Kunh, and Tucker in 1951, the last two being equally well-known for the Karush-Kunh-Tucker (KKT) conditions of nonlinear programming.

2.2.2 The Assignment Problem

A field closely related to linear programming is that of Integer Programming (IP), which can, in its most general form, be expressed as the same problem as given by (7)-(8), with the added restriction that all elements of x must be integer. This simple modification results in tremendous complication of the problem. In particular, this problem can be shown to belong to the class of NP-hard problems, which contains some of the most challenging combinatorial problems, including well known examples such as the Hamiltonian Cycle and Traveling Salesman problems.

Nevertheless, not all instances of the general IP problem are NP-hard. In particular, the assignment problem is a special case whose properties make it tractable. Consider a set of m persons

$\mathcal{D} = \{i : i = 1, \ldots, m\}$ that has to be assigned to a set of n objects $\mathcal{S} = \{j : j = 1, \ldots, n\}$, where $m \leq n$, such that $(i, j) \in \mathcal{E}_f$, where $\mathcal{E}_f$ denotes the set of all allowable pairs. For every person i the set $\mathcal{N}(i)$ consists of all objects that person i can be assigned to, i.e. $\mathcal{N}(i) = \{j : (i, j) \in E_f\}$. For each object $j \in N(i)$ there is a benefit a_{ij} for matching person i with object j . The objective is to find the person/object pairs (i, j_i) so that all persons are assigned only one object, and such that the total benefit $\sum_{i \in \mathcal{D}} a_{ij_i}$ is maximized among all possible person/object pairs. It is also assumed that each object can be assigned to only one person.

In practice, we can assume that a_{ij} is defined for each object, since we can assign a very small number (essentially $-\infty$) to all a_{ij} such that $j \notin \mathcal{N}(i)$. In the standard case where $n = m$, the problem takes the following form:

$$\max \quad \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_{ij} \tag{18}$$

$$\text{such that:} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, 2, \ldots, n \tag{19}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, 2, \ldots, n \tag{20}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \tag{21}$$

The first set of conditions insures that no person is assigned to more than one object, while the second insures that no object is assigned to more than one person. The last condition, the integrality condition, ensures that half a person isn't assigned to a third of an object.

The history of the solutions of assignment problems is rich and interesting. The first solution is Kuhn's Hungarian method, based on the work of Hungarian mathematicians D. König and E. Egervary, which is already strongly polynomial. It was used as the foundation of seminal work in transportation and general minimum flow problems [20].

During the 1950s the work of Hoffman and Kruskal [23] proved, among other things, that the integrality constraints of the assignment problem were unnecessary, and could be replaced by simple nonnegativity constraints. In other words, the assignment problem is in fact a linear programming problem that is guaranteed to have integral solutions, because of the special structure of the constraint matrix—specifically, the matrix is unimodular [23, 10]. Combined with the fact that the right-hand-side constraints are integral, this suffices to rewrite the problem as an LP.

Assignment problems, however, are specialized problems even within the LP category: they can be viewed as network flow problems. It is well known that polynomial time algorithms, such as the Ford-Fulkerson max-flow algorithm, exist to solve such problems [39], and those algorithms can also be applied to the assignment problem. But most importantly for this work, a method of solving the assignment problem via an auction-like mechanism, known as the auction algorithm, was introduced by Bertsekas in 1979 [8]. This method, which also has a strongly polynomial running time for the symmetric assignment problem, is the method we selected to solve the peer-to-peer matching problem, for reasons which we will describe in more detail in Chapter 4.

2.3 Scheduling

The last topic that this thesis touches on in its attempt to create a methodology for P2P satellite refueling is that of scheduling. The problem will be shown to arise quite naturally as we try to limit the effect of the refueling process on constellation operations. Much like the two topics sketched above, scheduling is a vast field. Unlike, however, those topics, there is not a central problem that can be termed the "scheduling problem" from which all else derives. Rather, a vast class of problems can be termed scheduling problems, even when their mathematical structure is markedly different.

The first class of scheduling problems relevant to the present research is the class of job shop scheduling problems. A job shop scheduling problem consists of a set of *jobs* $\{J_1, J_2, \ldots, J_n\}$ to be processed through m *machines* $\{M_1, M_2, \ldots, M_m\}$. Each job must be processed through each machine once and only once (note that the time that any machine needs to spend processing any given job can be set to zero). Each operation is denoted o_{ij} , referring to the i-th job's operation on the j-th machine [21]. Each job typically also has a due date d_i. The problem is then to schedule the jobs so as to optimize some metric.

There are many possible metrics for the job-shop scheduling problem. To name a few, we can try to minimize the makespan (total time to complete all jobs), the total lateness (sum of the differences between the due dates and the termination dates of all jobs), the total tardiness (the sum of the positive differences between the due dates and the termination dates of all jobs), the average lateness, or the average tardiness. Many more examples of target metrics can be found in [21].

An additional element of the problem is that some (or all) the operations for each job may have precedence constraints, i.e. o_{ij_1} may need to be completed before o_{ij_2} may begin. Further expansions of the model can be included, such as ready-dates (a particular operation may not begin before a specified time).

Job-scheduling problems have been shown to be NP-hard [41] except in specific cases [26] of limited interest. There are several general approaches to job-shop scheduling. The main one uses disjuntive graphs, which represent the job schedule as a flow problem that includes both directed conjunctive arcs denoting precedence relations between operations and disjunctive arcs, denoting operations that cannot concurrently be carried out because they both use the same machine. This approach was first proposed by Roy and Sussman in 1964 [46, 40], and sought to find ways of replacing the disjunctions by conjunctive arcs, thus obtaining a sequence of jobs for each machine. Another option has been to formulate the problem as a mixed-integer program (MIP), by which we mean a LP-type problem in which some, but not all, the variables are required to be integer.

More recently, constraint satisfaction programming (CSP) has been studied as a way to approach many scheduling problems, and in particular the job-scheduling problem. In [27, 35], the authors propose a repair technique for scheduling problems (though not necessarily job-scheduling problems) based on the empirical behavior of neural nets to improve initial solutions. Another novel approach from the past decade consists of the use of reinforcement learning to train neural networks to detect domain-specific regularities in a vast class of job-scheduling problems [67, 66], i.e. training the neural network to spot and exploit the special structure that each job-scheduling subclass of problems may have. These learning- and repair-based techniques have been successful in speeding up the performance of scheduling algorithms. This is particularly important since the general CSP problem is NP-complete and computationally intractable.

The second class of scheduling problems that bear resemblance to the scheduling problem of Chapter 5 is the server upgrade problem encountered in Internet applications [1, 22]. This problem has in fact more in common with our problem than the job scheduling problem, whose main relevance to this work is the investigation of repair techniques. The server upgrade problem can be stated as follows: given a number of servers running an application online, how can they be upgraded to the next version of the application without interfering with the operations the servers are carrying out?

The solutions proposed in [1, 22] do not revolve around scheduling alone—indeed, the main line of research seeks to find ways to update each server without affecting its operation, not, as a scheduling problem would imply, around finding a way to take servers down without significantly affecting the operation of the online application that is run on those servers. Thus, as we will discuss in Chapter 5, the scheduling problems do not model our problem adequately, while the server upgrade problems try to solve it in ways that are not open to us. This motivates the development of new methods to attack our specific problem.

2.4 Summary

This chapter gave a brief overview and literature survey of the three main fields that form the basis for this thesis: two-impulse orbital transfers, linear programming and assignment problems, and scheduling theory. With the foundations set here, we are ready to move on to the development of the peer-to-peer refueling methodology that is the core of this thesis.

CHAPTER III

THE PEER-TO-PEER REFUELING PROBLEM

This chapter is concerned with the formulation of the P2P refueling problem. We will show that it is reasonable to model the fuel needs of individual satellites so that a natural bipartition is created among them, dividing the constellation into fuel sufficient and fuel deficient satellites. We will then formulate the refueling problem as an assignment problem, which is a format particularly suited for solution via auctions, as will be seen in Chapter 4.

For the purposes of the next two chapters, we will assume that all the refueling maneuvers are to start at the same time. Therefore, all the data for the satellites is given either at the initial time, or at times specified relative to the initial time. We now explain the core concept of P2P refueling.

If we have a constellation of satellites, some satellites may have fuel to spare, while others may have too little fuel (precise terminology will be introduced later in the chapter). The question then arises: is it possible to refuel those satellites which have too little fuel using the extra fuel of the other satellites? If so, we will want to do this as cheaply as possible, i.e. with as little fuel consumption as possible, and within a certain time frame. This last is a reasonable condition, since even a very robust constellation may have to go off line for such a refueling maneuver, and we would like to keep such disturbances in performance to a minimum.

Several issues arise immediately. Firstly, what is meant by “too little” and “too much” fuel? The work in [53, 54, 60] sought to minimize the deviation from the average fuel of the constellation at the end of the maneuvers. Therefore, satellites with more than the average amount of fuel had “too much” while those with less than the average amount of fuel had “too little.” While this makes sense in the case of identical satellites, it is not a useful standard in cases where some satellites may have huge fuel capacities, while others may have very small ones. The formulation presented in this work, based on fuel need, has wider application to general constellations.

Another question is how to organize the refueling process itself. One could, for example, pool all the extra fuel in as few satellites as possible and have those satellites make the rounds to refuel those in need. Alternatively, one could have each satellite with "too much" fuel refuel as many needy satellites as it can on its own. Finally, we may require ahead of time that less than half the satellites be needy, and have them be refueled by pairing them up with surplus-carrying satellites. This approach is the one used in this work, under the key assumption that each satellite must be in its own slot in order for the constellation to function properly.

Given how we plan to approach the refueling problem, we can define a *refueling transaction* as an operation involving two satellites which starts and ends with both satellites in their original slots, with some sort of a rendezvous and a fuel transfer occurring in the interim. There are two main ways of organizing a rendezvous: one of the satellites goes to the other satellite's slot, or both move and rendezvous in a third location. In this work, we use the first approach, but note that research is being carried out to determine whether there would be any significant benefit the second approach.

The final issue that must be addressed is that of constraints. It makes sense that there would be a time constraint on the refueling operation. Other constraints may include conditions on when satellites can move, or even prohibition of some satellites moving at all. For the purposes of this chapter, we assume that the only constraint is the time constraint. Additional constraints will be introduced when we tackle the scheduling problem, in Chapter 5.

3.1 The Feasible Constellation Graph

We begin by considering a constellation $\mathcal{C} = \{s_k : k = 1, 2, \dots, N\}$ of N satellites. These satellites are not necessarily identical and not necessarily on the same orbit; they are assumed to be a part of the constellation because they are, in some way, required for the constellation to perform its function. Each satellite has a dry mass m_k, a specific impulse $I_{\mathrm{sp}k}$, a fuel capacity $\bar{f}_k$, and fuel content f_k^- at the time when refueling is about to start. We now introduce the notion of the fuel need of a satellite.

Suppose a constellation gets refueled at regular time intervals. It is possible, either because of unexpected fuel expenses, or because the constellation was designed to take advantage of P2P refueling methods, that at some point in time, some of the satellites will no longer have enough fuel to remain operational until the next refueling. The amount of fuel a satellite s_k must have to remain operational until the next refueling is called the *fuel need* of the satellites, and is denoted by $\underline{f}_k$. It is clear, then, that at any given time, there are two types of satellites in the constellation: *fuel deficient* satellites, meaning satellites such that

$$f_k^- < \underline{f}_k, \tag{22}$$

and *fuel sufficient* satellites, meaning satellites such that

$$f_k^- \geq \underline{f}_k. \tag{23}$$

We can then partition the constellation $\mathcal{C}$ into two disjoint sets: that of fuel deficient and that of fuel sufficient satellites. Specifically, let $\mathcal{D} = \{i : i = 1, 2, \ldots, m\}$ denote an index set of all fuel-deficient satellites, and $\mathcal{S} = \{j : j = 1, 2, \ldots, n\}$ denote an index set of all fuel-sufficient satellites, where there exist two one-to-one maps ϕ and ψ such that $\phi(i) \mapsto k_i$ and $\psi(j) \mapsto k_j$ and $\phi(i) \neq \psi(j)$ for any i, j. Note that, by definition, $N = m + n$.

To facilitate the formulation of the P2P refueling problem, we next introduce the notion of a constellation graph. As first discussed in [60, 53, 54], the vertex set of the constellation graph consists of the satellites, while its edge set consists of the potential pairings. In our case, the sets $\mathcal{D}$ and $\mathcal{S}$ induce a natural bi-partition on the set $\mathcal{C}$, and we can define the bipartite graph $\mathcal{G} = \{\phi(\mathcal{D}) \cup \psi(\mathcal{S}), \mathcal{E}\}$, where the notation $\phi(\mathcal{D})$ denotes the mapping of all the elements of $\mathcal{D}$ into $\mathcal{C}$, and where the set of edges $\mathcal{E} = \{(\phi(i), \psi(j)) : i \in \mathcal{D}, j \in \mathcal{S}\}$ has as vertices pairs of fuel sufficient and fuel deficient satellites. Note that this graph is a complete bi-partite graph.

It is clear that not all pairs in this complete graph need represent a physically achievable maneuver. For example, one of the fuel sufficient satellites might be exactly fuel sufficient, that is, condition (23) holds at equality. Clearly, then, this satellite cannot afford to rendezvous with any other satellite, much less give up fuel. To account for such situations, we introduce some terminology. In a given refueling transaction, the *active* satellite is the satellite which leaves its orbit to rendezvous with the other, *passive*, satellite. We can define two necessary conditions for feasibility of a pair (i, j) as follows, where the subscripts i and j refer to members of the sets $\mathcal{D}$ and $\mathcal{S}$ respectively (this can be done without ambiguity since the maps ϕ and ψ are one-to-one):

1. One of the satellites in the pair must have enough fuel to initiate a rendezvous. Defining p_{ij}^i as the fuel cost of satellite i leaving its orbital slot and transferring to that of satellite j, and p_{ji}^j analogously, this means that either $p_{ij}^i \leq f_i$ (i is active), or $p_{ji}^j \leq f_j$ (j is active).

2. There must be enough fuel in the satellites so that after the refueling transaction is over, both satellites are fuel sufficient. Defining c_{ij} as the total fuel cost of the maneuver (i.e., the orbital transfers), the fuel in a feasible pair must satisfy :

$$f_i^- + f_j^- - c_{ij} \geq \underline{f}_i + \underline{f}_j \tag{24}$$

Expressions for the quantities defined above will be calculated in detail in a subsequent section. Suffice it to say, for now, that they are well defined for every case. With that in mind, we partition the set $\mathcal{E}$ as follows:

$$\mathcal{A} = \{(\phi(i), \psi(j)) \in \mathcal{E} : (i, j) \text{ is feasible and } i \text{ can be active}\}, \tag{25a}$$

$$\mathcal{P} = \{(\phi(i), \psi(j)) \in \mathcal{E} : (i, j) \text{ is feasible and } j \text{ can be active}\}, \tag{25b}$$

$$\mathcal{U} = \{(\phi(i), \psi(j)) \in \mathcal{E} : (i, j) \text{ is infeasible}\}, \tag{25c}$$

where a pair is considered infeasible if it fails either of the two conditions set out above. Note that in general $\mathcal{A} \cap \mathcal{P} \neq \varnothing$, since it is possible for both satellites to have enough fuel to initiate a rendezvous.

Given the above, we can define the *feasible edge set* as $\mathcal{E}_f = \{(\phi(i), \psi(j)) \in \mathcal{A} \cup \mathcal{P}\}$, and the *feasible graph* as $\mathcal{G}_f = \{\phi(\mathcal{D}) \cup \psi(\mathcal{S}), \mathcal{E}_f\}$. The graph $\mathcal{G}_f$ is a representation of all the possible pairings of satellites. The optimal matching, i.e., the matching that will cost the least fuel, must be chosen from the pairs contained in this graph.

3.2 The Peer-to-Peer Assignment Problem

Now that the feasible constellation graph is well defined, we turn our attention to the assignment problem proper. Recall that we are trying to match the satellites on a one-to-one basis so as to minimize the fuel consumption of the whole refueling process. A time constraint, as discussed at the beginning of this chapter, is implicit in the cost c_{ij} associated with a pair (i, j), as was seen in Section 2.1.1.

Let $\mathcal{N}(i) = \{j \in \mathcal{S} : (i, j) \in \mathcal{E}_f\}$ denote the set of fuel sufficient satellites that can perform a refueling maneuver with $i \in \mathcal{D}$, where for the rest of the chapter we write, by abuse of notation, i and j instead of $\phi(i)$ and $\psi(j)$ in all expressions. Because we allow only one fuel exchange per satellite, the product of the P2P refueling strategy will be a matching $\mathcal{M} \subset \mathcal{E}_f$ that minimizes the fuel consumption of the maneuvers. By matching, we mean a set of edges of $\mathcal{E}_f$ such that no two edges share a vertex.

The problem of finding the optimal matching $\mathcal{M}^*$ can be formulated as an integer program over the bipartite graph $\mathcal{G}_f$ as follows

$$\text{Maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} -c_{ij}x_{ij} \tag{26}$$

$$\text{Subject to:} \quad \sum_{j=1}^{n} x_{ij} = 1, \quad \forall\, i \in \mathcal{D}, \tag{27}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall\, j \in \mathcal{S}, \tag{28}$$

$$x_{ij} = 1 \implies j \in \mathcal{N}(i), \quad i \in \mathcal{D}. \tag{29}$$

$$x_{ij} \in \{0, 1\}, \quad \forall\, i, j. \tag{30}$$

Equation (27) enforces the condition that every fuel deficient satellite must be paired with exactly one fuel sufficient satellite. Inequality (28) enforces the condition that every fuel sufficient satellite can be paired with at most one fuel deficient satellite. Equation (29) simply states that all pairs considered are feasible. Note that for a problem to be feasible, necessarily $n \geq m$.

An integrality constraint on x_{ij} is not needed since it is known that assignment problems always have integral solutions [10]. In other words, one may replace (30) with the condition $x_{ij} \geq 0$, without loss of generality.

The above is an assignment problem, which allows us to use many methods to solve it. We present the auction algorithm in Chapter 4 as a possible candidate. For now, we turn our attention to the detailed calculation of expressions for c_{ij} and the p^i_{ij} and p^j_{ji} quantities on which the sets $\mathcal{N}(i)$ depend.

3.3 *Fuel Cost Calculations*

Consider two satellites $i \in \mathcal{D}$ and $j \in \mathcal{S}$ in circular orbits of (possibly different) radii r_i and r_j, and separated initially by an angle θ_{ij}, as shown in Figure 2.

We assume that the satellites will rendezvous by applying a two-impulse rendezvous strategy. Specifically, if satellite i is active, it will apply impulses to rendezvous with satellite j within a given

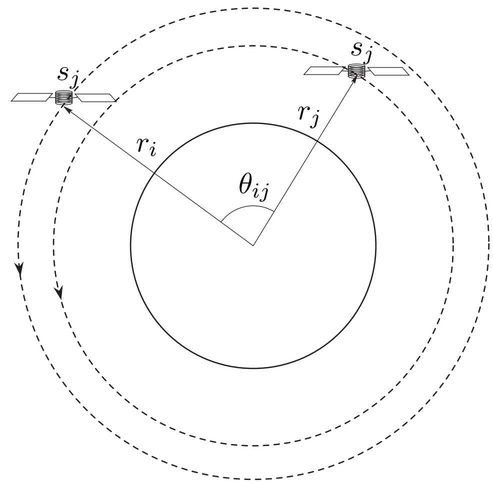

Figure 2: A basic rendezvous configuration.

amount of time t^i_{ij}. Let $\Delta V^i_{ij} = \Delta V(t^i_{ij}, \theta_{ij})$ denote the two-impulse velocity increment required for satellite i to leave its orbital slot and rendezvous with satellite j under the time constraint t^i_{ij}. Since we discussed calculation of the ΔV quantities in the previous chapter, we will assume that they are known.

Given the required ΔV^i_{ij} for satellite i to visit satellite j, the corresponding fuel cost p^i_{ij} can be computed from [12]

$$p^i_{ij} = (m_i + f^-_i)\left(1 - e^{-\Delta V^i_{ij}/\sigma_i}\right), \tag{31}$$

where $\sigma_i = g_0 I_{\text{sp}i}$ and g_0 is the acceleration of gravity at the surface of the Earth.

The return fuel cost p^i_{ji} can be similarly computed from

$$p^i_{ji} = (m_i + f^-_i - p^i_{ij} - g_{ij})\left(1 - e^{-\Delta V^i_{ji}/\sigma_i}\right), \tag{32}$$

where g_{ij} denotes the amount of fuel transferred from i to j. Note that, in general, $\Delta V^i_{ij} \neq \Delta V^i_{ji} = \Delta V(t^i_{ji}, 2\pi - \theta_{ij})$, even if $t^i_{ij} = t^i_{ji}$; see, for example, [52].

From (31) and (32), it is clear that in order to minimize $p^i_{ij} + p^i_{ji}$ the quantity g_{ij} must be maximized, subject to the constraint

$$g_{ij} \leq f^-_i - p^i_{ij} - p^i_{ji} - \underline{f}_i, \tag{33}$$

so that $f^+_i \equiv f^-_i - p^i_{ij} - p^i_{ji} - g_{ij} \geq \underline{f}_i$. Clearly, the optimal value is $g^*_{ij} = f^-_i - p^i_{ij} - p^i_{ji} - \underline{f}_i$, and the corresponding value for p^i_{ji} is

$$p^i_{ji} = (m_i + \underline{f}_i + p^i_{ji}) \left(1 - e^{-\Delta V^i_{ji}/\sigma_i}\right),$$

or, upon rearrangement

$$p^i_{ji} = (m_i + \underline{f}_i) \left(1 - e^{-\Delta V^i_{ji}/\sigma_i}\right) e^{\Delta V^i_{ji}/\sigma_i}. \tag{34}$$

Equations (31) and (34) give the optimal fuel cost in the case where satellite i is active, and under the assumptions that i has enough fuel to complete the first leg of the rendezvous and that there is enough fuel in j to transfer over g_{ji} units of fuel to i.

Similarly, when satellite j is active, the cost for the forward trip is given by

$$p^j_{ji} = (m_j + f^-_j) \left(1 - e^{-\Delta V^j_{ji}/\sigma_j}\right), \tag{35}$$

whereas the return transfer cost is given by

$$p^j_{ij} = (m_j + f^-_j - p^j_{ji} - g_{ji}) \left(1 - e^{-\Delta V^j_{ij}/\sigma_j}\right). \tag{36}$$

In this case g_{ji}'s value is limited by the fuel requirement for j to be able to make its return trip, and by the amount of fuel that satellite i is capable of accepting. Now, equation (36) implies that g_{ji} must be maximized in order to minimize $p^j_{ji} + p^j_{ij}$. Recalling that $\bar{f}_i$ is the maximum fuel capacity of satellite s_i, the optimal transfer fuel from j to i, in this case is given by

$$g^*_{ji} = \min\{f^-_j - \underline{f}_j - p^j_{ij} - p^j_{ji}\ ,\ \bar{f}_i - f^-_i\}, \tag{37}$$

which leads to the following expression for p^j_{ij}

$$p^j_{ij} = \begin{cases} (m_j + f^-_j - p^j_{ji} - \bar{f}_i + f^-_i) \left(1 - e^{-\Delta V^j_{ij}/\sigma_j}\right), \\ \qquad \text{if} \quad g^*_{ji} = \bar{f}_i - f^-_i, \\ \\ (m_j + \underline{f}_j) \left(1 - e^{-\Delta V^j_{ij}/\sigma_j}\right) e^{\Delta V^j_{ij}/\sigma_j}, \\ \qquad \text{if} \quad g^*_{ji} = f^-_j - \underline{f}_j - p^j_{ij}. \end{cases} \tag{38}$$

Finally, the cost c_{ij} assigned to satellite pair (i, j) is given by

$$c_{ij} = \min\{p^i_{ij} + p^i_{ji}, p^j_{ji} + p^j_{ij}\}, \tag{39}$$

assuming both satellites can be active. Obviously, if only one can be active, the corresponding cost ($p^i_{ij} + p^i_{ji}$ or its counterpart) will be the fuel cost of the maneuver.

With the results above, we can fully form the feasible constellation graph over which the assignment problem is formulated. We note that all the calculations depend on the ΔV required to carry out the two-impulse maneuvers, which we discussed in Chapter 2.

3.4 Summary

We now have completely formulated the P2P refueling problem as a well-defined (though possibly infeasible—this discussion is deferred for the next chapter) assignment problem, given in (26)-(30). Since the assignment problem is well studied, many solutions exist. We are therefore interested in the solution that is most likely to play to the strengths of a satellites constellation. This is the purpose of the next chapter.

CHAPTER IV

SOLUTION OF THE PEER-TO-PEER REFUELING PROBLEM VIA AUCTIONS

As discussed in Chapter 3, the P2P refueling problem can be formulated as an assignment problem. The fact that the assignment problem is a linear programming problem opens the door to the many solution methods developed for linear programs. The simplex method, by far the most popular method for solving general LPs, is one possible approach to the P2P refueling problem. Nevertheless, it is unwieldy, s mentioned previously, especially in light of the specialized algorithms that exist for specialized LPs. The assignment problem in particular can be reformulated as a network flow problem, and such problems admit solutions via specialized algorithms that exploit the unique structure of the constraints. However, one method in particular, which was initially developed specifically for the assignment problem, has great promise as a solution approach to the satellite P2P refueling problem, namely the solution of the assignment problem via auctions.

Auction algorithms have two inherent advantages over the simplex method: firstly they are guaranteed terminate in polynomial time (the simplex method *generally* terminates quickly, but can require an exponential number of iterations under some circumstances), and secondly they admit a distributed implementation—ideal for a constellation where we would prefer not to depend on any one satellite too much. These two advantages are counterbalanced by the fact that optimality of the solution obtained via auctions is only guaranteed within bounds. However, since those bounds can be made arbitrarily small, this disadvantage is slight, and may even work to our benefit, as will be discussed later. Another advantage, however, is that the auction algorithm can be implemented just as easily for the asymmetric assignment problem that the P2P refueling problem generates as for the symmetric case. The cost is the strongly polynomial running time, which can no longer be guaranteed, since the scaling method used to guarantee a strongly polynomial running time for the symmetric case can no longer be applied [8]. Nevertheless, the algorithm maintains a weakly polynomial running time.

Our choice to study the auction algorithm is not, ultimately, dictated by clear computational advantages over other methods. It is, rather, the distributable nature of the algorithm that is of greatest appeal. In this thesis, we try, when possible, to use methods that can take advantage of the multiple satellites, i.e. the multiple processors, that we have available in the constellation to design a methodology that can be carried out by the constellation *independently* of the ground control system. The auction algorithm offers precisely such a solution to the matching problem: a solution where all that is required in terms of central computation is for one satellite to keep track of which satellites are assigned and which are not, so as to declare the matching phase terminated (the scheduling phase is discussed in Chapter 5).

There are several ways of implementing auction algorithms for a constellation. First, we may apply a serial version of the algorithm, known as a Gauss-Seidel implementation. Second, we may apply a parallel version of the algorithm known as a Jacobi implementation. Both of these algorithms admit *asynchronous implementations*, meaning that the information used for each iteration does not have to be up-to-date, subject to certain mild constraints. This last property is very desirable, especially if there is uncertainty about the speed or reliability of communications.

In this chapter, we will describe both the serial and the parallel implementations of the auction algorithm, and present the solution to a sample problem obtained through both methods.

4.1 The Serial and Parallel Auction Algorithms

The auction algorithm is a method of solving the assignment problem that is rooted in economic auction processes. Our presentation of this algorithm closely matches that given in [8]. To better understand how the auction algorithm works, recall that the symmetric assignment problem is given by:

$$\max \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} \tag{40}$$

$$\text{such that:} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, 2, \ldots, n \tag{41}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, 2, \ldots, n \tag{42}$$

$$x_{ij} \geq 0, \tag{43}$$

where a_{ij} represents the benefit of matching person i with object j, when there are n persons and n objects. The mathematical form of this problem is the same as that of a network flow problem, a fact that will be useful shortly. Now imagine trying to match persons with objects via some sort of market mechanism. We can then assign to each object a *price* π_j. Each object then offers a *profit* $a_{ij} - \pi_j$ to person i, and each person i will seek to be assigned to the object j_i which yields the greatest profit, that is:

$$a_{ij_i} - \pi_{j_i} = \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j\}. \tag{44}$$

The condition (44) is a form of the *complementary slackness condition* we discussed in Chapter 2, which we abbreviate as CS-condition. To see this, we write th dual of the assignment problem (40)-(43) as [10]:

$$\min \quad \sum_{i=1}^{n} \rho_i + \sum_{j=1}^{n} \pi_j \tag{45}$$

$$\text{such that:} \quad \rho_i + \pi_j \geq a_{ij} \quad \forall i, j, \tag{46}$$

where we have ignored the sets $\mathcal{N}(i)$ because they can always be simulated by setting a_{ij} to a very low number in case $j \neq \mathcal{N}(i)$.

Now suppose the values of $\pi_i, \pi_2, \ldots, \pi_n$ are set. The value of $\sum_{i=1}^{n} \rho_i$ is then minimized if each of the ρ_i is set to the smallest value that will still satisfy (46). This smallest value we can write as

$$\rho_i = \max_{j=1,\ldots,n} \{a_{ij} - \pi_j\}. \tag{47}$$

Now recalling that the assignment problem is a special type of network flow problem, there are two complementary slackness conditions that must be satisfied: first, flow must be conserved, and second, if there is positive flow on arc (i, j), the dual variables at the nodes i and j, ρ_i and π_j must satisfy:

$$\pi_j - a_{ij} = -\rho_i = -\max_k \{a_{ik} - \pi_k\}, \tag{48}$$

$$a_{ij} - \pi_j = \max_k \{a_{ik} - \pi_j\}, \tag{49}$$

which is exactly (44), without the arc constraints (which are a product of $\mathcal{N}(i)$). As for the flow conservation constraints, which form the second part of the complementary slackness conditions for network flow problems, any feasible assignment will satisfy them [10]. In light of this and the discussion of complementary slackness in Chapter 2, it is clear that if all matched pairs fulfill the CS-condition, the solution will be optimal.

Returning to the auction algorithm, at the beginning of each iteration the same object may be desired by more than one person, and thus a bidding mechanism is introduced, whereby every object that is bid on by more than one person chooses the highest bidder and then raises its price to that offered by the highest bidder. This is the basis for naming this and related algorithms auction algorithms. For its part, the CS condition should hold with all assigned pairs at the beginning of each iteration.

This strict enforcing of the CS condition, however, may lead to cycling. This problem appears in cases where an object is equally desirable by two persons, and neither one is willing to raise its bid to get the object, leading to the object being assigned to each of the two persons alternatively with each bidding iteration. To avoid cycling one introduces a minimum biding amount, but this has the effect of destroying CS. However, introducing the *ε-complementary slackness condition*, or ε-CS condition for short,

$$a_{ij_i} - \pi_{j_i} \geq \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j\} - \varepsilon, \tag{50}$$

where $\varepsilon > 0$ allows us to still maintain optimality guarantees. Specifically, while the minimum bidding increment ε prevents cycling, the final assignment is guaranteed to be optimal within $m\varepsilon$ as long as ε-CS holds for all the pairs in a final matching [9].

Below we summarize the main steps of the bidding process during an auction algorithm. For simplicity, we limit our presentation to the forward auction, which is the simplest and most straightforward implementation.

Initialization At the beginning of the algorithm, no person is assigned. In addition, and without loss of generality, we let $\pi_j = 0$ for all $j \in \mathcal{S}$. This last is a sufficient condition to guarantee the successful termination of the algorithm in the case of an asymmetric problem. It is unnecessary if the problem is symmetric. At the beginning of each subsequent iteration, every person $i \in \mathcal{D}$ is either assigned or unassigned. For every person assigned to an object j_i, the ε-CS condition (50) is satisfied.

Termination The algorithm terminates when all persons are assigned.

Bidding Phase Let $\mathcal{V} \subseteq \mathcal{D}$ be a nonempty subset of persons that are unassigned at the beginning of a given iteration. Each person $i \in \mathcal{V}$ finds an object j_i which offers maximal profit,

$$j_i \in \arg\max_{j \in \mathcal{N}(i)} \{a_{ji} - \pi_j\}, \tag{51}$$

and computes a bidding increment

$$\gamma_i = v_i - w_i + \varepsilon, \tag{52}$$

where v_i is the largest object profit, and w_i is the second largest object profit,

$$v_i = \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j\}, \tag{53}$$

$$w_i = \max_{\substack{j \in \mathcal{N}(i) \\ j \neq j_i}} \{a_{ij} - \pi_j\}. \tag{54}$$

If j_i is the only object in $\mathcal{N}(i)$, then we define $w_i = -\infty$, (in practice, a number much smaller than v_i). Obviously, if $\mathcal{N}(i)$ is empty, the problem is infeasible.

Assignment Phase Each object j that is selected by a nonempty subset $\mathcal{K}(j)$ of persons in $\mathcal{V}$ determines the highest bidder according to

$$i_j \in \arg\max_{i \in \mathcal{K}(j)} \gamma_i, \tag{55}$$

raises its prices by the highest bidding increment $\max_{i \in \mathcal{K}(j)} \gamma_i$, and gets assigned to the highest bidder i_j. The person that was assigned to j at the beginning of the iteration, if any, becomes unassigned.

It is clear that the above algorithm can be applied to the P2P refueling problem, which is, in general, an asymmetric assignment problem. In the formulation above, we simply replace a_{ij} with $-c_{ij}$. The serial algorithm, as well as the parallel and asynchronous implementations, are guaranteed to terminate if the problem is feasible. This is not guaranteed in the P2P refueling problem. However, since the solution to the feasibility problem is the same for all implementations, we defer the discussion of feasibility issues to Section 4.2.1. For now, we content ourselves with noting that the problem exists.

We now draw attention to the set $\mathcal{V}$. It is noteworthy that it is completely free, as long as it is nonempty. The serial (Gauss-Seidel) algorithm is simply the case where at each iteration, only one person is included in $\mathcal{V}$, i.e. $|\mathcal{V}| = 1$. The parallel (Jacobi) case is then the case where *every unassigned person* is included in $\mathcal{V}$. In between these two extremes, other implementations may be attempted, so-called block-Gauss-Seidel implementations [9]. These three options essentially present the gamut of forward auction algorithms for the case where complete information is available. The case where information is lost or delayed, known as the asynchronous case, requires a slightly different approach that we turn to next.

4.2 The Asynchronous Auction Algorithm

As we saw, the auction algorithm presented in the previous section assumes that the most current information is known by all the satellites that are bidding at each iteration. This is not necessarily realistic, since communication problems between (to use our example) the satellites can delay or even drop the information during transit. Fortunately, the auction algorithm is robust to this and other situations. In particular, formulated slightly differently, we can make guarantees that the algorithm will converge in cases where:

1. All unassigned persons will eventually bid.

2. All persons will eventually get updated information.

These two assumptions will be given mathematical precision in the discussion that follows. We note here that the presentation of the asynchronous auction algorithm follows very closely that given in [9].

We assume that time is discretized by bids: events (changes in the state of the assignments) occur only at the end of the bidding and assignment calculations. This does not cause a loss of generality, since we make two key assumptions further on which, by allowing for satellites to not be ready to bid at a given time, allow us to model calculation delays as an inability to bid. The calculations themselves are referred to as iterations. At the beginning of an iteration, we denote by

$\mathcal{V}(t) \subseteq \mathcal{D}$ the set of persons that are unassigned at time t. We further assume that of this set, a subset $\mathcal{W}(t) \subseteq \mathcal{V}(t)$ is ready to bid. We do not assume that the members of $\mathcal{W}(t)$ have up-to-date pricing information. Instead, we denote the pricing information at time t as $\pi(t)$, a vector of length n, where $\pi_j(t)$ is the price associated with object $j \in \mathcal{S}$ at time t, and we assume that each person $i \in \mathcal{W}(t)$ is aware of a price for a given object j given by $\pi(\tau_{ij}(t))$ for some $\tau_{ij}(t) \leq t$. This price information is used in calculating any bids submitted at time t. Finally, we define $r_j(t)$ as the index of the person that object j is assigned to at time t. We now make two key assumptions:

1. $\forall i \in \mathcal{V}(t),\ \exists\, t' \geq t : i \in \mathcal{W}(t')$. This is an expression of the first condition set out above: all the satellites that are unassigned at a time t will eventually submit a bid.

2. $\tau_{ij}(t) \to \infty$, as $t \to \infty,\ \forall i, j$. This is the mathematical statement of the second condition above: each satellite eventually gets an update on the pricing information.

As mentioned, the two assumptions above completely model any delay in the calculations or communications that might occur. Delays in the transmission of the pricing information are modeled via the $\tau_{ij}(t)$ function. Delays in the sending of the information are modeled via the set $\mathcal{W}(t)$. Delays due to the computations (i.e., the computer had up-to-date information, but the bids are no longer up-to-date because time elapsed between the time the computer started the computations and the time it submitted its bid) can be modeled via a combination $\tau_{ij}(t)$ and $\mathcal{W}(t)$. All other delays are simply a combination of these three. Because of this, we can take any delays into account by modeling them *a priori* and assume that all calculations happen instantaneously.

Once again, the key element in the algorithm is the ε-CS condition, which is enforced at every step. Specifically, for each pair (i, j_i) in an assignment at time t, the following must hold[9]:

$$a_{ij_i} - \pi_{j_i}(t) \geq \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j(t)\} - \varepsilon, \tag{56}$$

where $\varepsilon > 0$. We now state the algorithm.

Initialization: Set $t = 0$. At the beginning of the algorithm, no persons are assigned to any objects, so $\mathcal{V}(0) = \mathcal{D}$, and $\pi(0) = 0$ (again, only truly required for the asymmetric assignment problem). At each subsequent iteration, all assigned pairs satisfy (56).

Termination: The algorithm terminates when $\mathcal{V}(t) = \varnothing$.

Bidding: Each person $i \in \mathcal{W}(t)$ calculates the maximum value among all objects $j \in \mathcal{N}(i)$,

$$v_i(t) = \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j(\tau_{ij}(t))\}, \tag{57}$$

an object that yields that maximum value,

$$j_i(t) = \arg \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j(\tau_{ij}(t))\}, \tag{58}$$

the second best value among all objects $j \in \mathcal{N}(i)$,

$$w_i(t) = \max_{\substack{j \in \mathcal{N}(i) \\ j \neq j_i(t)}} \{a_{ij} - \pi_j(\bar{\tau}_{ij}(t))\}, \tag{59}$$

where $\bar{\tau}_{ij}(t) \geq \tau_{ij}(t)$ and a bid for object j_i

$$\beta_i(t) = a_{ij_i(t)} - w_i(t) + \varepsilon. \tag{60}$$

Each person then submits their bid to the appropriate object. If $j_i(t)$ is the only object in $\mathcal{N}(i)$, then we define $w_i(t) = -\infty$, and subsequently $\beta_i(t) = +\infty$. Note that it is possible at any time t for $\mathcal{W}(t)$ to be empty. As long as the first assumption is true, this does not affect the convergence of the algorithm except in speed.

Assignment: Each object receives bids from a (possibly empty) set of persons $B_j(t) = \{i \in \mathcal{W}(t) : j_i(t) = j\}$. If $B_j(t) \neq \varnothing$, each object j determines the highest bid

$$b_i(t) = \max_{i \in B_j(t)} \beta_i(t), \tag{61}$$

and a person submitting such a bid,

$$i_j(t) = \arg \max_{i \in B_j(t)} \beta_i(t). \tag{62}$$

Object j then updates the pair $(\pi_j(t), r_j(t))$ according to the following rule:

$$(\pi_j(t+1), r_j(t+1)) = \begin{cases} (b_j(t), i_j(t)) & \text{if } b_j(t) \geq \pi_j(t) + \varepsilon \\ (\pi_j(t), r_j(t)) & \text{otherwise.} \end{cases} \tag{63}$$

The above algorithm is identical to that presented in [9], and the results shown therein hold. In particular:

1. The algorithm is guaranteed to terminate if the problem is feasible.

2. The final assignment benefit is guaranteed to be within $m\varepsilon$ of the optimal assignment benefit, where $m = |\mathcal{D}|$.

The only issue remaining is that of feasibility, which will be discussed in the next section. Before moving on to that, however, we show that the algorithm as stated here is equivalent to the algorithm stated in the previous section in the case where $\mathcal{W}(t) = \mathcal{V}(t) = \mathcal{V}$, and $\tau_{ij}(t) = \bar{\tau}_{ij}(t) = t$ for all i, j. Firstly, the complementary slackness condition (56) is seen to be equivalent to (50) by inspection. Since the only event that changes t in this case is an iteration, we see that the vector $\pi(t)$ is the same as the vector π in Section 4.1, which changes with each iteration. Therefore (51) and (58) are equivalent, as are (53) and (57) and (54) and (59).

Some care must be taken when looking at the bids. In the first implementation, the bid γ_i is added on to the price p_j. In other words, it is a proposed *increase in the price*. In the asynchronous implementation, however, the bid β_i is the proposed new price. This can be seen by the differing conditions for reassignment: in the first implementation, being the highest proposed increase is enough, since there is a minimum bid. In the second, being the highest is combined with the requirement that the price increase by at least ε (63). However, looking in detail at what the bid is in the second case, we see that we can write:

$$\begin{aligned}
\beta_i &= a_{ij_i} - w_i + \varepsilon \\
&= a_{ij_i} + \pi_{j_i} - \pi_{j_i} - w_i + \varepsilon \\
&= \pi_{j_i} + v_i - w_i + \varepsilon \\
&= \pi_{j_i} + \gamma_i,
\end{aligned}$$

from where it is seen that if a $b_j = \beta_{i_j}$, the final price object j will be $\pi_j + \gamma_{i_j}$, which is the same as what it would be in the first implementation, since $b_j = \beta_i$ only if $\gamma_{i_j} \geq \gamma_i$ for all i.

4.2.1 Feasibility

As mentioned before, there is no *a priori* guarantee that a given problem is feasible. This particularly true in the specific case of satellite constellation P2P refueling problems. If the problem is infeasible and there are no methods for detecting it, the algorithm will go on indefinitely, since the set $\mathcal{W}(t)$ will never be empty. Fortunately, several ways of detecting infeasibility exist, all of which are easy to implement assuming *some* foreknowledge by the satellites of the fuel capacity of the whole constellation.

For example, there is a minimum bound on $v_i(t)$ for any feasible problem given by [8]:

$$v_i(t) \geq -(2n-1)C - (n-1)\varepsilon, \tag{64}$$

for all $i \in \mathcal{D}$ and for all $t \geq 0$, in case the algorithm is initialized with $\pi_j(0) = 0$, where $C = \max_{(i,j)\in\mathcal{E}_f} |a_{ij}|$. Since for an infeasible problem, at least one object will receive an infinite number of bids, at least one $v_i(t)$ will drop below this lower bound. Substituting C with the total fuel capacity of the constellation $\sum_k \bar{f}_k$ (where $\bar{f}_k$ is the maximum fuel capacity of the k-th satellite in constellation), we have a condition that can easily be assumed known *a priori* by each satellite and moreover is an upper bound on C, since no feasible cost can be greater than the total fuel content of the constellation. Again, since an infinite number of bids will be submitted, the value of v_i will eventually drop below this new lower bound as well–although it may take longer to detect infeasibility in this fashion.

Another approach to solving an infeasible problem can be adopted by understanding what situations can cause a problem to be infeasible. In the specific case of the P2P assignment problem, there are three such situations. Firstly, there may be more fuel deficient than fuel sufficient satellites. Since our assumptions state that a fuel sufficient satellite may refuel at most one fuel deficient satellite, this clearly means the problem has no solution. This is a situation that can easily be checked for, however. Secondly, the set $\mathcal{N}(i)$ may be empty for one or more satellites i, meaning that no fuel sufficient satellite can replenish them. This, again, is easily checked for.

The third and last situation is more complicated and is best explained by referring to Figure 3. There it is seen that it is possible for too many fuel deficient satellites to be able to exchange fuel with too few fuel sufficient satellites. This situation is not easily checked for, but a solution can be created by realizing that it is the sparsity of the constellation graph that causes it. By adding imaginary edges between infeasible pairs, as shown in Figure 4, and assigning to them a very small benefit (essentially negative infinity), we can insure that the algorithm will terminate in every case,

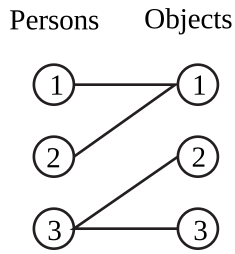

Figure 3: An infeasible problem. For every person i, $\mathcal{N}(i) \neq \varnothing$, but the fact that $\mathcal{N}(1) = \mathcal{N}(2) = \{1\}$ implies that one of them must remain unassigned.

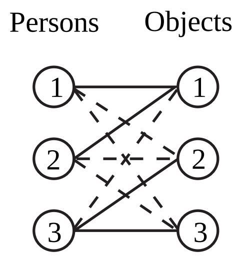

Figure 4: The infeasible problem from Figure 3 can be made feasible by augmenting it with highly-weighted edges (dashed lines).

but will have a final benefit that is very low (infinitely so) if the problem is infeasible.

The problem with the above implementation is that it takes a possibly sparse graph and transforms it into a fully populated graph. The auction algorithm is known to be particularly fast in the case of sparse graphs, and therefore it is counter-productive to suppress this advantage in order to detect infeasibility. Nevertheless, it should be possible to add no more than n edges and still resolve feasibility issues; in Figure 3, one would need only add a low-cost edge between element 2 on the left and element 2 on the right. In addition, if the alternative detection method is used in the P2P refueling problem, it is not known how much longer the inflated lower bound on v_i will take to reach than the standard lower bound C. It is therefore not possible to say, *a priori*, which of the two methods is better suited for the P2P refueling problem.

4.3 Simulation Results

To illustrate the various aspects of the auction methodology, we present here two examples that we will solve via the serial implementation of the auction algorithm. We will then solve the second example via a parallel implementation, and furthermore simulate some communication losses and

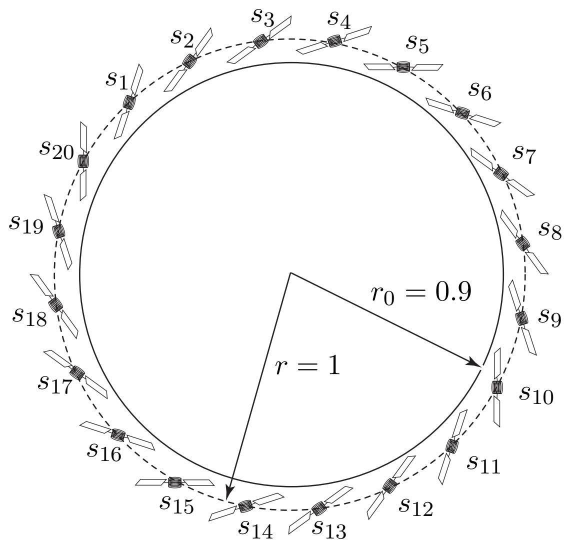

Figure 5: A constellation with 20 evenly distributed identical satellites.

Table 1: Satellite data.

Dry Mass (m_k)	50 units
Fuel Capacity ($\bar{f}_k$)	100 units
Specific Impulse ($I_{\mathrm{sp}k}$)	197 s

test out the asynchronous implementation. Both of our examples will deal with constellations of 20 evenly-spaced satellites along the same circular orbit, as shown in Figure 5. The basic information of each satellite is the same, and is given in Table 1. It is assumed that the surface of the planet is at 0.9 times the orbit radius.

We mention here that the calculations for the ΔV quantities were carried out in a normalized fashion, where the radius of the constellation was taken as the unit of length, and the period of the constellation as the unit of time. This allowed us to take arbitrary units for mass and fuel. The fuel content data for the two constellations is provided in Tables 2 and 3.

We note that in the first example, only three satellites are fuel deficient, while in the second example, fully half the constellation is in need of fuel. Moreover, the ten satellites that are fuel deficient in Example 2 occupy sequential slots in the orbit. This somewhat artificial configuration is used observe what solutions are obtained when the satellites do not have the option of going to a fuel satellite close by.

Table 2: Satellite fuel specifics for Example 1.

Satellite	$\overline{f}_k$	$\underline{f}_k$	Satellite	$\overline{f}_k$	$\underline{f}_k$
1	83.1	8.5	11	0.3	6.4
2	37.2	16.7	12	78.8	24.6
3	33.8	27.1	13	14.8	19.1
4	23.4	12.1	14	58.3	6.2
5	32.6	11.9	15	70.6	29.9
6	28.0	19.3	16	82.4	23.1
7	29.1	29.0	17	38.9	0.1
8	40.6	6.3	18	42.9	9.8
9	2.4	22.9	19	78.5	13.1
10	44.1	2.8	20	71.0	28.5

Table 3: Satellite fuel specifics for Example 2.

Satellite	$\overline{f}_k$	$\underline{f}_k$	Satellite	$\overline{f}_k$	$\underline{f}_k$
1	12	30	11	44	30
2	9	30	12	75	30
3	26	30	13	52	30
4	0	30	14	97	30
5	23	30	15	80	30
6	29	30	16	58	30
7	29	30	17	82	30
8	23	30	18	48	30
9	13	30	19	60	30
10	14	30	20	95	30

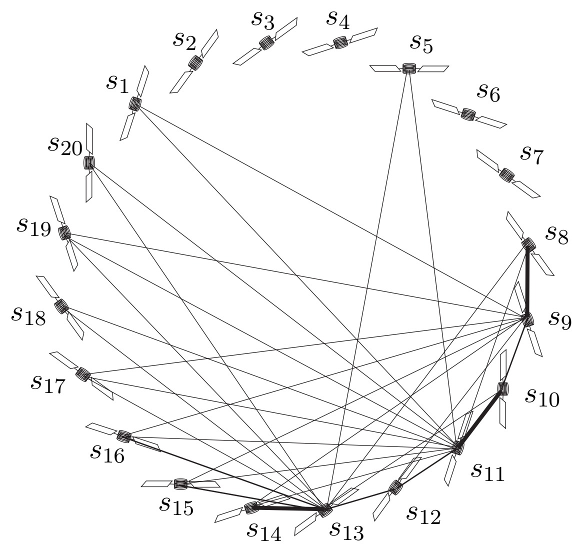

Figure 6: The constellation graph for Example 1. The optimal matching is shown in bold.

4.3.1 Solutions via the Serial Algorithm

The serial and parallel implementations of the auction algorithm are very similar, differing only in the set $\mathcal{V}$ of unassigned satellites that bids at each iteration. They were implemented essentially as outlined in Section 4.1, and require little additional discussion. The feasible constellation graph for Example 1 is shown in Figure 6, with the bold lines indicating the final matching obtained via the auction algorithm. The arrows point from the active to the passive satellite, while the thin lines indicate all the possible pairings.

The details of the results are given in Table 4. We note that the amount of fuel in the active satellite is equal to the minimum required amount of fuel, i.e. $f_k^+ = \underline{f}_k$ if s_k is the active satellite in the exchange. This makes sense, since the active satellite must burn fuel to return to its original slot, and should therefore be as light as possible for maximum fuel efficiency.

We also note that in two out of three cases, the active satellite is the *fuel deficient* satellite. This makes sense, since the fuel deficient satellite has less fuel and therefore is lighter—but this is not a general rule, since sometimes the fuel deficient satellite will not have enough fuel to initialize, or there may even be another pairing that ends up being less expensive when the fuel sufficient satellite is the active one. However, the cost of the fuel sufficient satellite carrying out the rendezvous maneuver is usually greater, and is avoided. Example 2 will further confirm this observation, as we

Table 4: Details of the fuel-optimal matching for Example 1.

$i \in \mathcal{D}$	$j \in \mathcal{S}$	f_i^+	f_j^+	g_{ji}^*	Active
9	8	22.9	18.3881	22.2119	9
11	10	39.6438	2.8	39.3438	10
13	14	19.1	52.2368	6.0632	13

are about to see.

For the record, the total cost of this matching is 5.43 units of fuel, or just 0.71% of the constellation fuel content, and the result is obtained after 4 iterations of the algorithm. The ε parameter was taken to be 0.25, and therefore the matching is guaranteed to be within 0.75 fuel units of optimal.

The feasible constellation graph for Example 2 is shown in Figure 7, following the same conventions as previously. We note from this graph that each fuel deficient satellite tries to match itself with a satellite that is as close to it as possible. This of course has the effect of leaving satellite s_5 to travel halfway across the constellation to meet its match. In addition, we can observe that satellite s_{18} can only be matched with one of the fuel deficient satellites, satellite s_3. Since half the constellation is fuel deficient, this forces that possible pairing to be the final pairing. Moreover, we can see from the pattern of the matches that the optimal pairings seem to want to pair fuel deficient satellites with fuel sufficient satellites as close to them as possible.

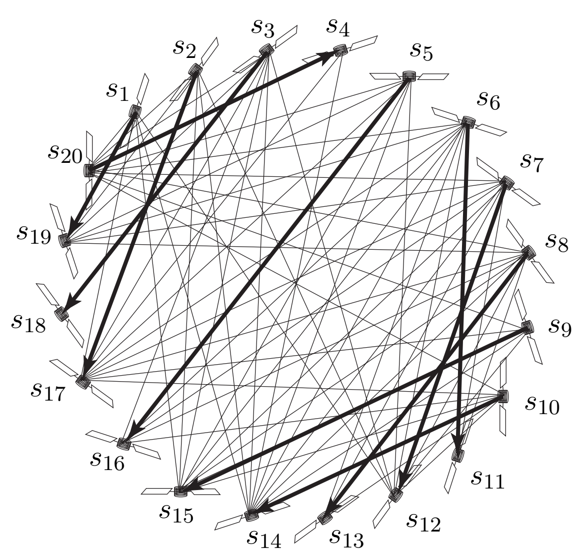

Figure 7: The constellation graph for Example 2. The bold lines indicate the final optimal pairings, obtained with the serial auction algorithm. The arrows point from the active to the passive satellite.

The numerical data regarding this matching is given in Table 5, where we see the same pattern

Table 5: Details of the fuel-optimal matching for Example 2.

$i \in \mathcal{D}$	$j \in \mathcal{S}$	f_i^+	f_j^+	g_{ji}^*	Active
1	19	30.0	37.90	22.09	1
2	17	30.0	51.09	30.90	2
3	18	30.0	32.93	15.06	3
4	20	52.76	30.0	52.76	20
5	16	30.0	31.80	26.19	5
6	11	30.0	32.30	11.69	6
7	12	30.0	63.30	11.69	7
8	13	30.0	34.69	17.30	8
9	15	30.0	51.34	28.65	9
10	14	30.0	73.27	23.72	10

repeated as in Table 4: the active satellite is the fuel deficient satellite, except where the fuel deficient satellite cannot initiate the maneuver. An extreme case of this is satellite s_4 in this example, which Table 3 indicates has no fuel at all. Also, again we note that the active satellite has exactly the amount of fuel that it needs at the end of the transaction.

Again, we note that the fuel cost of this matching is is 107.56 units of fuel, which corresponds to 12.37% of the total initial fuel content in the constellation (869 units). The solution in this example was obtained after 108 iterations.

4.3.2 Solutions via the Parallel and Asynchronous Algorithms

The first sample problem above is simple—almost trivial—and is only given to illustrate that the algorithm works. The second problem is a prime candidate for running a simulation of parallel calculations. As was mentioned in Section 4.1, the only difference between the serial and parallel implementations of the algorithm is the choice of set $\mathcal{V}$ of unassigned satellites that bid at each iteration. The asynchronous situation that we simulated, however, was a somewhat simplified version of the algorithm, which we describe now.

In implementing the asynchronous algorithm, we sought to simulate only loss of pricing information due to communication failures. In addition, we assumed that the pricing information was carried in a single, non-divisible packet. Finally, we assumed that all satellites were ready to bid at every iteration. Mathematically, this translates into the following for all times t:

$$\mathcal{W}(t) = \mathcal{V}(t), \tag{65}$$

$$\tau_{ij_1}(t) = \tau_{ij_2}(t) = \tau_i(t), \quad \forall j_1, j_2 \in \mathcal{S}. \tag{66}$$

To simulate the packet loss, we express $\tau_i(t)$ as follows:

$$\tau_i(t+1) = \begin{cases} t+1 & \text{if } X(t) \leq P_i, \\ \\ \tau_i(t) & \text{if } X(t) > P_i, \end{cases} \tag{67}$$

where $X(t)$ is a uniformly distributed random variable on $[0, 1]$ and P_i is a probability threshold. The meaning of the expression above is that satellite i will either get the full list of up-to-date prices (i.e., the vector $\pi(t)$), or it will keep its former price vector. Therefore, we do not simulate delayed information. In addition, we set $P_i = P$ for all the satellites, thereby assigning the same threshold to each satellite.

Running the code in parallel yielded a bit of a surprise: after taking only 40 iterations, the code converged to a different matching. As shown in Figure 8, the satellites assigned to satellites s_9 and s_{10} were switched, as were those for satellites s_6 and s_7. This matching had a cost of 107.5772, which, though a little higher than that of the first algorithm, is still well within the tolerance guaranteed by the algorithm. Moreover, running the asynchronous algorithm with different randomly generated rates of update (from 10% to 90% probability of data loss) yielded additional near-optimal matching matchings, all within tolerance. The best matching obtained was the one shown in Figure 9, with only the assignments for 9 and 10 being switched relative to the Gauss-Seidel type solution. The cost of this matching was 107.5523, and the matching was obtained after 48 iterations, with a price update probability of 0.9.

The reason that several matchings can be obtained via different update probabilities and different implementations lies with the fact that none of these matching is guaranteed to be optimal. In point of actual fact, we only guarantee the optimality to within $m\varepsilon$. If more than one matching meet this tolerance, there is no matching that the algorithm will converge to *a priori*: all the matchings can be obtained if the simulations are run with some randomizing factor (such as the probability thresholds). Even in the case of a Gauss-Seidel serial implementation, what will determine which of the sub-optimal matchings the algorithm will converge to is the order in which the satellites bid.

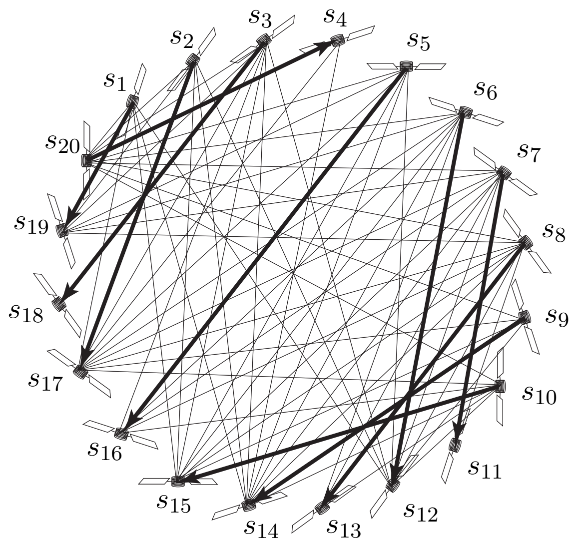

Figure 8: The constellation graph for Example 2, with the matching from the parallel version of the algorithm.

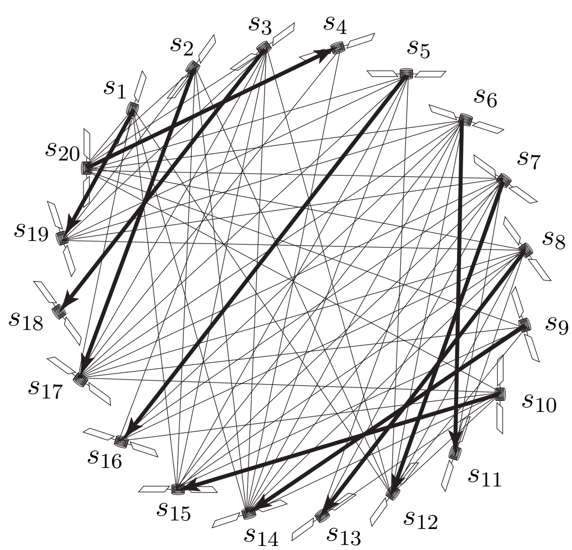

Figure 9: The constellation graph for Example 2, with the best matching obtained with the asynchronous parallel version of the algorithm.

Table 6: Average number of iterations to termination with respect to update probability.

Probability	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Iterations	222.2	152.4	111.9	86.4	70.7	60.5	53.2	47.7	44.0

Nevertheless, it is instructive to look at the best matching obtained via multiple runs of the algorithm. While it is not guaranteed to be the optimal, a good case can be made for this by noticing that each satellite tries to be matched to a satellite as close to it as possible (note the pattern of the links from satellites $s_6 - s_{10}$ to satellites $s_{11} - s_{15}$). Since the results in Figure 6 indicate that satellites prefer to be matched with other satellites that are close (in an angular separation sense), this furthering of the pattern reinforces the idea that the last matching is indeed the optimal one. Note that the pattern is broken in the links between satellites $s_1 - s_5$ and $s_{16} - s_{20}$ because satellite s_{18} can only be paired up with satellite s_3.

4.3.2.1 Performance of the Asynchronous Algorithm

To get a better idea of the response of the auction algorithm to asynchronous bids, we ran two separate tests. First, we ran a test with the same base case given in Example 2 above and averaged the number of iterations for 1000 runs of the algorithm at $P = 0.1, 0.2, \ldots, 0.9$. The average results are given in Table 6.

We note that even in a situation where there is only a ten percent probability that updated price data is received, the algorithm, on average, runs only four and a half times slower than the fully parallel algorithm. This is very good performance, but may be skewed by the fact that we modeled information *loss* rather than information *delay*. In other words, there is no situation where $\tau_i(t+1) = \tau_i(t-q)$ for some integer $q \geq 1$. This means that when the satellite receives an update, the update is the newest price vector, $\pi(t)$. The effects of receiving an older price vector, say $\pi(t-q)$ for some $q \geq 1$, are not included in the current simulation.

In order to check that this promising performance is not an aberration, we ran three more examples of randomly generated constellations. The configuration of the constellations was identical to the previous one, but the fuel content for each satellite was randomly chosen so that each satellite had a 0.4 probability of being fuel deficient. Mathematically, we the fuel content was determined as

Table 7: Average number of iterations to termination with respect to update probability.

P	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Constellation 1	37.4	28.2	22.4	18.9	16.2	14.5	13.5	12.8	12.3	12.0
Constellation 2	18.1	13.4	10.4	8.1	7.0	6.0	5.2	4.8	4.3	4.0
Constellation 3	6.5	5.0	4.2	3.5	3.0	2.6	2.4	2.2	2.1	2.0

follows:

$$f_k^- = Y(k)\bar{f}_k, \quad \underline{f}_k = 0.4\bar{f}_k \tag{68}$$

where $Y(k)$ is a uniformly distributed random variable on $[0,1]$, and $\bar{f}_k = 100$ units of fuel.

The results for the run are given in Table 7. Here again we note the same pattern as before, where lower update probability results in longer running times, but the increase is not excessive, even for low update probabilities.

Finally, to illustrate the behavior of the algorithm over a more complete range of probabilities, we ran one randomly generated constellation at 1000 runs for probability P on the interval $[0.1, 1]$. The results are presented in Figure 10, where it is clear that the run time increases as the probability of update goes to zero.

As can be seen from these results, there is good indication that the the auction algorithm is very well suited for this particular application, since even high rates of data loss do not detract from relatively fast convergence. However, it must be noted that several instances of randomly generated constellations fell prey to price wars. While this problem can be alleviated by the use of forward-reverse auctions, it is not entirely avoided. In addition, the ε-scaling method[8], the preferred method of avoiding price wars, is not readily applicable to asymmetric auctions as far as we know. However, every time the straightforward auction algorithm terminated within a reasonable time, the running times associated with the asynchronous algorithm were of the same order of magnitude even under unfavorable communication conditions.

4.4 Summary

This chapter introduced auction algorithms as a distributed method for solving the assignment problem resulting from the P2P refueling formulation given in Chapter 3. Because of their distributed

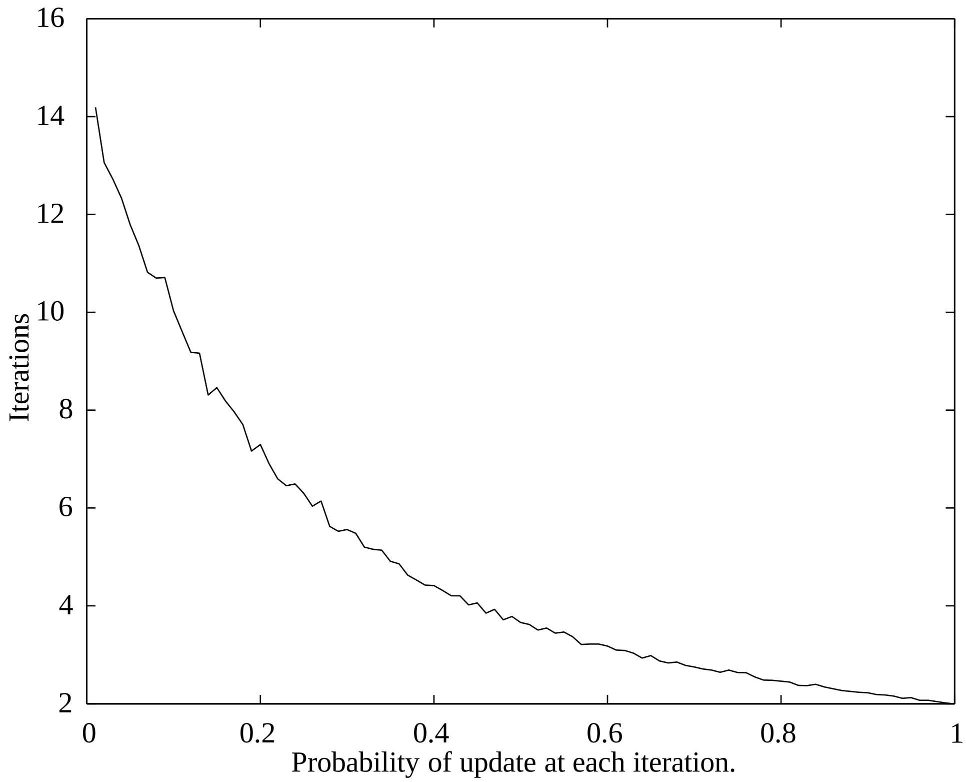

Figure 10: The number of iterations compared with the probability of each element of the set $\mathcal{D}$ having updated price information after each iteration.

nature, illustrated in parallel and asynchronous implementations, these algorithms are prime candidates for implementation in a constellation. One advantage that we did not discuss, however, is the very sub-optimality of the algorithm.

Since the auction algorithm is guaranteed to be optimal within bounds, it can be run with a high error margin, and therefore quickly, in order to obtain quick estimates. In particular, if all that is sought is to push the fuel cost of the refueling maneuver below a certain threshold, quick estimates can be run, and perhaps even yield results that are below the threshold, even if they are not optimal.

Ultimately, though, it is the robustness and the distributed implementations of auctions that present the most incentive for implementing them on real-life constellations. Since they exhibit strong performance even under very adverse circumstances, they will be able to exploit and reinforce the main advantages of satellites constellations: distribution of resources, and redundancy.

With this chapter, we close the section of the thesis dedicated to the matching part of the problem: if we setup the assignment problem of Chapter 3 and solve it with the methods of Chapter 4, we will obtain an optimal (to within $n\varepsilon$) matching on the bipartite constellation graph. We will therefore have a set of maneuvers that must be carried out. This is the setup for the scheduling problem, which we will treat in the next chapter.

CHAPTER V

THE SCHEDULING PROBLEM

We have, in the past two chapters, established a method for assigning fuel-deficient satellites to fuel-sufficient satellites for refueling purposes. In essence, we have found an optimal set of maneuvers that will guarantee that, upon execution, the constellation will be fuel sufficient. Nevertheless, the implicit assumption, so far, has been that all maneuvers would be carried out at once. Since each maneuver requires a satellite to leave its orbital slot, this presents several problems.

First, we may reasonably assume that a satellite can only carry out its function from its orbital slot. If nothing else, communications would be setup based on each satellite's assigned location. At the very least, satellite maneuvers would require redrawn communication pathways. At worst, they will signify a constellation communications breakdown. Since some functions will also be dependent on the location of the satellite (for example, oceanographic measurements as described in [45]), even maintaining communications may not guarantee constellation functionality.

Second, we must take into account the notion of redundancy. Indeed, since a constellation consists of several physically independent satellites, it is reasonable to assume that the designers would have some sort of redundancy built in that would take advantage of the distributed physical nature of constellations. Part of it may be simply multiple data backups, providing data redundancy, but another part may be functional redundancy, where certain satellites may be lost without adversely affecting the constellation's ability to do its job. Even so, in the examples of the previous chapters we introduced situations where fully half the constellation had to move to remedy the fuel deficiency problems. Such a level of redundancy is unreasonable, given the cost of building and launching satellites. Thus, executing all the maneuvers at the same time may lead to the constellation (rather than individual satellites) being unable to carry out its function.

Third, the carrying out of all maneuvers simultaneously is far riskier. Indeed, every maneuver in space carries with it some element of risk—from a thruster misfiring to an unexpected collision with an uncharted object that damages the unlucky satellite's critical systems. While we have mentioned in Chapter 1 that we leave the risk analysis as a subject for later study, it is still clear that there is a risk involved, and that carrying out all maneuvers at once greatly magnifies this risk. In particular, if a satellite is damaged in transit and all the other maneuvers are being carried out simultaneously, there is no way to correct for the new redundancy state of the constellation, or to avoid moving satellites in order to reduce the risk of mid-transfer collision with space debris, for example. On the other hand, if the maneuvers are staggered, it may be possible to cancel or reschedule a maneuver to solve or alleviate the problem.

For these and other reasons, we will dedicate this chapter to the consideration of maneuver scheduling. From this point forward, the matching for the refueling maneuvers is taken to be given, obtained either via the methods of the previous chapters, or the methods described in previous work on the subject [60]. While we may change certain aspects of the maneuvers, the most important aspect, the active-passive satellite pair of each maneuver, will remain constant.

The matching calculations were carried out under a time constraint that applied to each individual maneuver: no maneuver would last more than a certain, predetermined, duration. The scheduling calculations are also time-constrained (otherwise the trivial solution is to order the maneuvers and execute them sequentially without overlap). However, the time allotted to the completion of all maneuvers will be assumed to be longer than that given the individual active satellites to carry out of their respective maneuvers. The main requirement will then be that all maneuvers be carried out sometime during this longer time interval. We will focus on finding an intelligent way of scheduling maneuvers around constellation operability conditions. Predictably, we will begin by defining those conditions.

For the rest of the chapter, we will make the following assumptions, which we justified above for the most part:

1. A satellite is able to carry out its function if and only if it is in its original slot.

2. A constellation is functional if and only if all of its operability conditions are fulfilled.

After precisely defining various types of operability conditions, we will seek a way of scheduling the maneuvers over a period of time so as to minimize the downtime caused by the maneuvers.

We note that, in its simplest form, our problem is different from standard scheduling problems. Most scheduling problems, as discussed in Chapter 2, come with constraints such as processor constraints (number of operations that can be executing simultaneously), precedence constraints (one operation must be carried out before another), and release date constraints (an operation cannot start before a certain point in time), to name a few [40]. In contrast, we do not have any hard constraints limiting the number of maneuvers that can concurrently be carried out, nor precedence and release date constraints. Soft constraints instead are used to define our cost function, which is also different from the standard cost functions considered in the scheduling literature. We distinguish soft constraints from hard constraints as follows: "hard" constraints are constraints that cannot be violated in a valid solution; "soft" constraints are constraints that can be violated, but may increase the cost.

Scheduling theory usually deals with minimization of the makespan of a set of operations (the time when the last operation is completed), the tardiness (the weighed sum of the difference between the operations' completion times and their respective deadlines), the number of tardy jobs, or other metrics that implicitly assume the system is fully functional at all times. We, on the other hand, allow ourselves to break our system's functionality, and seek to minimize the amount of time we do so, while treating the deadline as the only hard constraint. Thus, our cost function is the amount of time the constellation is not functional during the refueling process.

A scheduling problem that bears somewhat more of a resemblance to the one developed in this chapter is the server upgrade problem, which seeks to update software or hardware on a group of servers without destroying the functionality of the online-application that they are running. The similarities exist because most of the time, the guiding constraint of those problems is one of time: all the hardware must be upgraded within a limited period of time, if for no other reason than it might become obsolete otherwise (imagine upgrading the servers in Google's server farms one by one). At the same time, it is of the utmost importance that the service provided to the customer should continue. In the realm of Internet software, the issue is often attacked through innovative software upgrade techniques, known as dynamic software updating [22]. They often manage to sidestep the scheduling problem entirely. Not so for operations that need outages, such as electrical power transmission networks [34], or even more generally, outages in distributed environments of any kind [13]. Approaches differ, with the authors in [34] adopting genetic programming as the method of choice, while those in [35, 5] advocate the use of constraint satisfaction programming (CSP). Since genetic programming is too complex for the relatively small computational capabilities of a satellite constellation, and since CSP is in general NP-complete, we decided to pursue heuristic methods to approach our specific problem.

Here we will develop a heuristic to schedule maneuvers subject to constellation constraints. We begin by defining a constellation and a set of operability constraints in Section 5.1. In Section 5.2, we define the interference scheduling problem and prove some of its important properties—in particular, we will prove that we can search a finite set of maneuver configurations and still find, without loss of generality, a globally optimal schedule, even when maneuvers are allowed to start at any point in a time continuum. In Section 5.3 we propose a distributable heuristic to obtain a good initial solution to the interference scheduling problem, and then present a repair technique to improve such initial solutions, based on similar techniques used in constraint satisfaction programming (CSP) [35]. We conclude the section by presenting a method to reduce fuel cost, where possible, given an existing constellation schedule. We combine all these results in Section 5.4 to propose a methodology for organizing P2P refueling in a satellite constellation, from matching to scheduling. Finally, in Section 5.5 we present numerical examples illustrating the methods developed in the preceding sections.

Because there are many new notions to be defined, we introduce new notation in this chapter. Some effort has been expended in making it consistent with the notation of previous chapters, but the present notation should be taken and understood on its own terms.

5.1 Satellite Constellations and Operability Conditions

We begin by defining what we mean by a satellite constellation. We may assume that an intelligent design will have some built-in redundancy in satellite functionality, and that a single satellite will not break the constellation. Given that, we become concerned with how many satellites it will take to break down the constellation, and, presumably, in what combinations. For the purposes of this thesis, we admit constellations with several operational goals and requirements, but we assume that if any of those requirements are not met, the constellation as a whole is considered not operational.

5.1.1 General Constellation Notation.

We define a satellite constellation as a time-varying triplet $\mathcal{C} = \{S, \Phi, \sigma_{t\geq 0}\}$ such that $S = \{s_i : i = 0, 1, \ldots, N\}$ is the set of N satellites in the constellation plus a fictitious satellite s_0, $\Phi = \{\phi_i \in [0, 2\pi) : i = 0, 1, \ldots, N, i \neq j \implies \phi_i \neq \phi_j\}$ is the set of slots, or locations along the various orbits where satellites can be located, plus a fictitious slot ϕ_0, and $\sigma_t : \Phi \mapsto S$ is a set-valued map assigning to each orbital slot the satellites in that slot at time t. If a slot ϕ_i is empty at time t, we write $\sigma_t(\phi_i) = s_0$. The function $\sigma^{-1} : S \mapsto \Phi$ is also a map, which assigns to each satellite the slot that it occupies, where we write $\sigma^{-1}(s_i) = \phi_0$ if the satellite is in transfer between slots. We assume that in a nominal constellation $\sigma_0(\phi_i) = s_i$. Finally, as a notational convenience, if $\Phi_\ell \subseteq \Phi$, we write

$$\sigma_t(\Phi_\ell) = \bigcup_{\phi_i \in \Phi_\ell} \sigma_t(\phi_i).$$

We denote by $\mathcal{O}_{t,\ell}$ the satellites from a subset $S_\ell \subseteq S$ of satellites that are in their original slots at time t, and by $\mathcal{O}_t$ all the satellites that are in their original slots at time t. Finally, we denote by $\mathcal{X}_t = \{S, \Phi, \sigma_t\}$ the state of the constellation at time t. In the constellation, each satellite s_i, it is assumed, has a function to perform. We assume in this formulation that a satellite is able to perform its function if and only if it is in its original slot.

We define a *refueling maneuver* as a pentuplet $r_k = \{s_{i_k}, \phi_{i_k}, s_{j_k}, \phi_{j_k}, \delta t_k\}$ consisting of the active satellite s_{i_k}, its slot ϕ_{i_k}, the passive satellite s_{j_k}, the slot ϕ_{j_k} of the passive satellite , and the time δt_k the active satellite will be away from its slot while carrying out the maneuver. We also define the set $\mathcal{R} = \{r_k : k = 1, \ldots N_m\}$ as the set of all maneuvers that are to be scheduled.

5.1.2 Operability Conditions

As discussed, there must be some minimum operability requirements in terms of the presence of satellites in slots from which they are able to perform their function, which in our case are the original slots. We propose the following three general operability conditions to model these requirements:

1. Outside world connectivity requirements. Outside bases/spacecraft must be able to connect to

the constellation at all (or certain) times. We only consider the case where the connection must be maintained at all times, but a relaxed condition can easily be accounted for by introducing a time-varying element in the value $p(w_k)$ defined further down.

2. Skeleton crew requirements. Sometimes, it may be required that a minimum number of satellites from a given subset be operational. There is of course no need to limit this to one subset.

3. In-constellation connectivity requirements. Certain satellite skeleton crews may need to be guaranteed intercommunication (i.e., communications within that subset) among operational satellites of the subset, either via line-of-sight or via relays with other satellites, in order to ensure functionality.

Note that the in-constellation connectivity requirements do not make sense without associated skeleton crew requirements, which moreover must require at least two satellites from their respective subsets to be in their slots. One need only consider the case where only one of the satellites in a given subset is operational to see how the requirement becomes nonsensical otherwise. The case where none are operational is another such example.

5.1.2.1 Outside World Connectivity

Suppose, as is very likely, that a constellation must be able to communicate with Earth, or with a space station, or even with another constellation. Each one of those will have access, at any given time, to a certain number of slots in the constellation, which may or may not be occupied. It is reasonable to assume that a minimum number of those slots must be occupied for proper communication to occur: one may be enough most of the time, but some cases (such as, for instance, interferometric constellations) may require more. At any rate, this type of situation must be modeled, as a constellation that is unable to relay its results to the outside world is effectively unable to carry out its mission.

Suppose that there are m_w external communication linkups, by which we mean open communication sockets, each with a different purpose, and each of which requires a certain number n_{w_k} of satellites to be in the slots that are accessible to the socket at a given time. In addition, suppose those satellites must all belong to a given subset $S_{w_k} \subseteq S$ $(k = 1, 2, \ldots, m_w)$. Denote the set of linkups

γ_{w_k} by $\Gamma = \{\gamma_{w_k} : k = 1, 2, \ldots, m_w\}$, and denote by $\Xi_t : \Gamma \mapsto 2^{\Phi}$ the time-varying set-valued map that returns for each linkup the satellites in the constellation that are accessible to it at time t, that is

$$\Xi_t(\gamma_{w_k}) = \sigma_t(\{\phi_i \in \Phi : \phi_i \text{ is accessible to } \gamma_{w_k} \text{ at time } t\}). \tag{69}$$

If we write $\Xi_{t,w_k} = \Xi_t(\gamma_{w_k}) \cap \mathcal{O}_{t,w_k}$, the outside world connectivity requirement can then be expressed by requiring that for a given t:

$$|\Xi_{t,w_k}| \geq p(\gamma_{w_k}) \qquad \forall w_k \tag{70}$$

where $|\cdot|$ represents the cardinality of a set and $p(\gamma(w_k))$ denotes the minimum number of satellites that must be connected to linkup γ_{w_k} from subset S_{w_k}. Note that while we do not address more complex situations where a linkup might need a certain number of satellites from each of a number of subsets, these can be modeled with a straightforward expansion of notation.

5.1.2.2 *Skeleton Crew Requirements*

A skeleton crew requirement is simply a requirement that a certain number of satellites from a given subset be operational at any given time. Given n_c such requirements, this can be handled seamlessly in the outside world requirements by introducing fictitious external linkups γ_{c_k} $(k = 1, 2, \ldots, n_c)$. These fictitious linkups have an unchanging list of accessible slots $\Xi(\gamma_{c_k})$, corresponding to the subset of slots for which the skeleton crew requirement is defined (since a satellite is only operational in its original slot, slots and satellites can be used interchangeably). We then use these sets and the required number of satellites in the skeleton crew $p(\gamma_{c_k})$ to write conditions completely equivalent to condition (70).

5.1.2.3 *In-constellation Connectivity*

In order to discuss in-constellation connectivity requirements, we first introduce the notion of constellation graphs. In previous chapters, the constellation graph has referred to the graph consisting of satellites as the vertices and possible refueling pairings as the edges. However, we must here modify the notion to be able to capture more diverse and dynamical situations. We will no longer be concerned only with the graphs representing the state of the entire constellation, but also with graphs representing the state of certain subsets of the satellites, which may be time-varying. Therefore, let $S_k \subseteq S$ be a set of satellites that share some property (say a communications method, such as infrared, radio, etc.) that induces a property k between ordered pairs $(s_i, s_j) \in \mathcal{E}_k$ where $\mathcal{E}_k \subseteq S_k \times S_k$ (say the ability to communicate directly via this communications method). The *satellite subset graph of property k at time t* is the directed graph defined by $\mathcal{G}_{t,k} = (\mathcal{O}_{t,k}, \mathcal{E}_{t,k})$, where $\mathcal{E}_{t,k} = \{(s_i, s_j) \in \mathcal{E}_k : s_i, s_j \in \mathcal{O}_{t,k}\}$. Thus, $\mathcal{G}_t$ is a time-dependent graph with vertex set $\mathcal{O}_{t,k}$ consisting of the satellites concerned with property k that are present in their original slot, and edge set $\mathcal{E}_{t,k}$ consisting of all pairs of satellites that are both in their slot and have property k between them. The *nominal graph of property k*, defined as the graph when all satellites are operational, is denoted by $\mathcal{G}_{0,k}$, since we assume that at time $t = 0$, all the satellites are in their original slots.

The in-constellation connectivity requirements assume, as implied above, that communication of one type or another (and probably of several types) is accounted for in the constellation. We can thus consider the presence of m_a communications networks N_{a_k} $(k = 1, 2, \ldots, m_a)$ in the constellation, each associated with a protocol a_k, a subset $S_{a_k} \subseteq S$ $(k = 1, 2, \ldots, m_a)$ of satellites that can send and receive packets using that protocol, and with a set of pairs $\mathcal{E}_{a_k} \subseteq S_{a_k} \times S_{a_k}$ whose elements denote pairs that can directly communicate with each other using that protocol. As mentioned before, the in-constellation connectivity requirements must be paired with skeleton crews to make sense. As such, assume we have m_c skeleton crews with corresponding fictitious linkups and satellite subsets γ_{r_s}, S_{r_s} $(s = 1, 2, \ldots, m_c)$, and let $\nu : \Gamma_{m_c} \mapsto \mathcal{N}$, where Γ_{m_c} denotes the set of all in-constellation connectivity fictitious linkups and $\mathcal{N}$ denotes the set of all communication networks. The in-constellation connectivity requirement at time t is then that for each skeleton crew linkup γ_{r_s}, the satellites $\Xi_{r_s,t}$ be able to communicate with each other via the network $\nu(\gamma_{r_s})$. In other words, all $s_i \in \Xi_{r_s,t}$ must belong to the same connected subgraph of $\mathcal{G}_{t,a_k}$, where $\nu(\gamma_{r_s}) = N_{a_k}$.

5.1.3 Run-time of Verification.

We end this section by examining the run-time required for verification of requirements. Suppose we have m total requirements, which will include the outside-world connectivity, skeleton crew, and in-constellation connectivity requirements. We can state the following proposition:

Proposition 5.1.1. *Verification of constellation operability is carried out in at most* $O(mn^2)$.

Proof. We begin by considering verifying the skeleton crew requirements. Each has a subset of satellites that must be checked for presence in the constellation. We may reasonably assume that this is a constant-time operation for each verification; we thus have a linear time process. Counting the number of satellites is concurrent, and so does not add to the runtime. Thus, each skeleton crew requirement can be verified in $O(n)$.

The case of the outside-world connectivity requirements is identical to that of skeleton crews, with the difference that the map Ξ_{t,w_k} is time-varying. This does not affect verification, since it occurs for a specified time, and thus each outside-world connectivity requirement can be verified in $O(n)$.

Lastly, the case of in-constellation connectivity requirements is verified in quadratic time. To see this, we note the well-known fact that finding all the connected vertices of a graph starting from a given vertex can be carried out in $O(|V| + |E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges. In the worst-case scenario, $|E| = |N|^2 = n^2$, yielding a run-time of $O(n^2)$ for in-constellation connectivity verification.

The worst-case scenario in terms of requirements is clearly that all requirements be in-constellation connectivity requirements. This corresponds to a run-time of $O(mn^2)$, as described above. $\square$

The worst-case run-time of the verification is quadratic in the number of satellites, but this is only true in cases where the communications graphs are both complete and involve all the satellites in the constellation. While the latter may be likely, the former is highly unlikely, since a lot of foreseeable applications of constellations deal with low-earth-orbit (LEO) constellations, where line of sight limits the angular separation of two satellites that wish to communicate directly. Thus the run-time is likely to be much faster than this worst-case scenario, more on a par with $O(mn)$.

5.2 The Interference Scheduling Problem

With the operability conditions well-defined, we introduce a scheduling problem that can be stated as follows: given a constellation and a set of maneuvers that must be carried out by individual satellites, what is the best way to schedule the operations so that their interference with constellation operability is minimized? We first formulate this problem precisely, and then introduce a certain class of schedules, which we will call *anchored schedules*, that have the property that for an arbitrarily given schedule, there always exists an anchored schedule incurring equal or lower cost. This result will be the driving justification for the algorithms given in Section 5.3.

5.2.1 Mathematical Notation and Problem Formulation

We begin by quantifying the functionality of the constellation. We introduce the constellation's satellite state vector at time t, $y_t \in \{0,1\}^N$, defined on the satellites as:

$$y_{t,i} = \begin{cases} 1 & \text{if } s_i \in \mathcal{O}_t, \\ 0 & \text{otherwise.} \end{cases} \tag{71}$$

We next define a vector valued function of constellation state, the *requirement vector* $\omega : \{0,1\}^N \mapsto \{0,1\}^{N_\omega}$, where $N_\omega = m_w + n_c + m_c$ is the total number of operability requirements, by:

$$\omega_j(z) = \begin{cases} 1 & \text{if operability requirement } j \text{ is satisfied under vector,} z \\ 0 & \text{otherwise.} \end{cases} \tag{72}$$

where z is a state of the constellation. It is clear that $y_t = \mathbf{1} \implies \omega(y_t) = \mathbf{1}$. In addition, note that given two states y_t^1 and y_t^2, which are identical except that for one i, $y_{t,i}^1 = 1$ and $y_{t,i}^2 = 0$, we have

$$\omega_j(y_t^2) \leq \omega_j(y_t^1) \tag{73}$$

for all j. This is a mathematical expression of the fact that a satellite being away from its slot can only hurt constellation operability, consistent with the notion that satellites are only operational in their original slots. We then define the *constellation operability function* $X : \{0,1\}^{N_\omega} \mapsto \{0,1\}$ by:

$$X = \begin{cases} 1 & \text{if the constellation is operational,} \\ 0 & \text{otherwise.} \end{cases} \tag{74}$$

As mentioned before, we assume that if a single requirement is violated, the constellation is inoperative. We thus have

$$X(y_t) = \min_j \ \omega_j(y_t). \tag{75}$$

Note that we have defined both ω and X to be time-invariant, which precludes the time-varying outside world connectivity requirements. We do this for simplicity, since extension to include them is straightforward.

By a *schedule* of $\mathcal{R}$ on $[0, T_f]$ we mean a map $\psi : \mathcal{R} \mapsto [0, T_f]$, where $\psi(r_k)$ denotes the initialization time of maneuver $r_k \in \mathcal{R}$ such that:

$$\max_{r_k \in \mathcal{R}} \{\psi(r_k) + \delta t_k\} \leq T_f. \tag{76}$$

We also introduce the notation ψ_t, defined as a schedule such that

$$\psi_t(r_k) = \psi(r_k) + t, \quad \forall r_k \in \mathcal{R}, \tag{77}$$

denoting a translation in time of the entire schedule by t units. If a schedule ψ may not be translated, we call it a *fixed schedule.* For convenience, we set $\psi = \psi_0$.

A schedule ψ is the *superposition* or *sum* of two schedules $\psi^1 : \mathcal{R}^1 \mapsto [0, T_{f_1}],\ \psi^2 : \mathcal{R}^2 \mapsto [0, T_{f_2}]$, where $\mathcal{R}^1, \mathcal{R}^2$ partition $\mathcal{R}$, if $\psi : \mathcal{R} \mapsto [0, \max\{T_{f_1}, T_{f_2}\}]$, and

$$\psi(r_k) = \begin{cases} \psi^1(r_k) & \text{if } r_k \in \mathcal{R}^1 \\ \psi^2(r_k) & \text{if } r_k \in \mathcal{R}^2 \end{cases}, \tag{78}$$

We will write $\psi^1 + \psi^2$ to denote the sum of two schedules. By extension, we define the notion of *subtraction* of two schedules. A schedule ψ is said to be the *difference* of two schedules $\psi^1 : \mathcal{R}^1 \mapsto [0, T_{f_1}], \psi^2 : \mathcal{R}^2 \mapsto [0, T_{f_2}]$, denoted by $\psi = \psi^1 - \psi^2$, if and only if $\psi + \psi^2 = \psi^1$.

It is useful for schedule analysis to define two sets $\Psi = B(\psi) \cup E(\psi)$ and $\bar{\Psi} = B(\psi) \cup E(\psi) \cup \{0, T_f\}$ such that:

$$t \in B(\psi) \iff \exists k : \psi(r_k) = t, \tag{79}$$

$$t \in E(\psi) \iff \exists k : \psi(r_k) + \delta t_k = t. \tag{80}$$

Note that it is possible for a time t to be both in $B(\psi)$ and $E(\psi)$. Since Ψ contains all the beginning and end points of all operations in the schedule, this simply means that it is possible for one operation (or more) to end where another (or more) begins. Without loss of generality, we may assume that the elements $\tau_i \in \Psi$ are ordered so that

$$\tau_1 < \tau_2 < \tau_3 < \ldots < \tau_{N_\Psi}, \tag{81}$$

where N_Ψ is the number of elements in Ψ. By extension, we assume that the elements $\bar{\tau}_i \in \bar{\Psi}$ are ordered, but in addition, we have

$$0 = \bar{\tau}_1 < \bar{\tau}_2 < \bar{\tau}_3 < \ldots < \bar{\tau}_{N_{\bar{\Psi}}} = T_f. \tag{82}$$

By construction, every time interval $[\bar{\tau}_i, \bar{\tau}_{i+1})$ corresponds to a constant state vector, which we denote by y^i. For each schedule ψ, we can therefore define an operability vector x as

$$x_i = X(y^i), \quad i = 1, \ldots, N_{\bar{\Psi}} - 1. \tag{83}$$

Using $I_i = [\bar{\tau}_i, \bar{\tau}_{i+1})$, the downtime of the system due to a schedule ψ on $[0, T_f]$ can then be obtained by the following relation:

$$\mathsf{Tdwn}(\psi) = T_f - \sum_{i=1}^{N_{\bar{\Psi}}-1} x_i \ell(I_i), \tag{84}$$

where $\ell(I_i) = \bar{\tau}_{i+1} - \bar{\tau}_i$. The full problem can then be formulated simply as:

$$\text{Minimize:} \quad \mathsf{Tdown}(\psi) \tag{85}$$

$$\text{Subject to:} \quad \psi(r_k) \geq 0 \qquad \forall k = 1, 2, \ldots, N_m \tag{86}$$

$$\psi(r_k) + \delta t_k \leq T_f \qquad \forall k = 1, 2 \ldots, N_m. \tag{87}$$

We denote an optimal schedule by ψ^*. In addition, we adopt the convention that for any fixed schedule ψ, $\psi(r_k) = 0$ for some $r_k \in \mathcal{R}$. Finally, before studying the properties of schedules, we introduce two very important notions for what is to follow. First, an operation $r_k \in \mathcal{R}$ is called *anchored by schedule* $\psi : \mathcal{R} \mapsto [0, T_f]$ if there exists a sequence of operations $\{r_{k_i}\}_{i=1}^{n_k}$ such that:

1. $\psi(r_{k_1}) = 0$ or $\psi(r_{k_1}) + \delta t_{k_1} = T_f$.

2. For any $1 < i + 1 < n_k$, at least one of the following holds:

   (a) $\psi(r_{k_i}) = \psi(r_{k_{i+1}})$.

   (b) $\psi(r_{k_i}) = \psi(r_{k_{i+1}}) + \delta t_{k_{i+1}}$

   (c) $\psi(r_{k_i}) + \delta t_{k_i} = \psi(r_{k_{i+1}})$

   (d) $\psi(r_{k_i}) + \delta t_{k_i} = \psi(r_{k_{i+1}}) + \delta t_{k_{i+1}}$

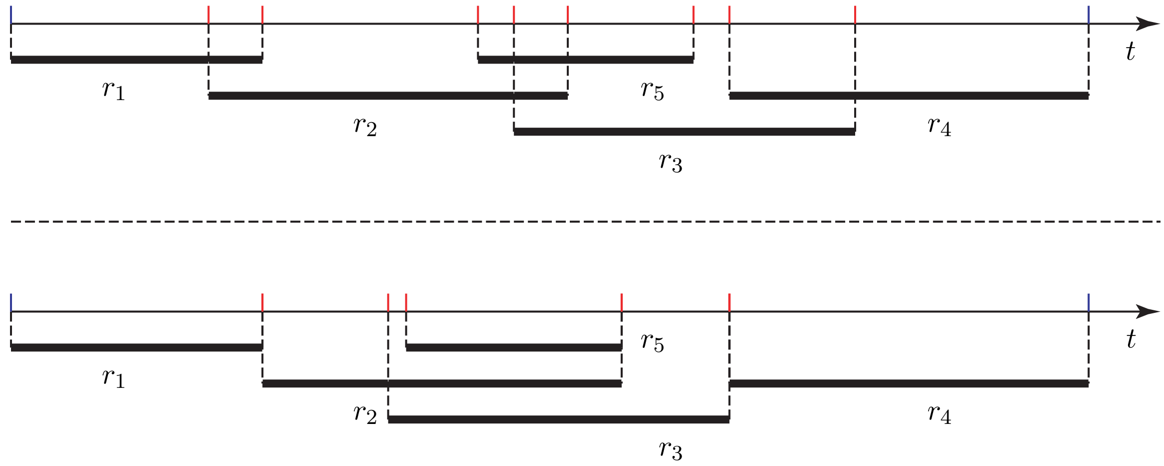

Figure 11: A random schedule (top) and an anchored schedule (bottom). The sequences anchoring the individual maneuvers are: $r_1, r_1 - r_2, r_1 - r_2 - r_5, r_4, r_4 - r_3$.

3. $r_{k_{n_k}} = r_k$.

If an operation is anchored with corresponding sequence $\{r_{k_i}\}_{i=1}^{n_k}$, r_{k_1} is called the *anchor* operation, and r_{k_i}, $1 < i < n_k$ are called $link$ operations. A schedule ψ is called an *anchored schedule* if every operation in the range of ψ is anchored by schedule ψ. Figure 11 illustrates these concepts.

We are now ready to prove some results regarding schedules.

5.2.2 Properties of schedules

We begin by proving a useful lemma. In essence, this lemma states that given an arbitrary superposition of two schedules, we can always shift one of them without increasing the cost so that they have at least one point from their respective Ψ sets matching. The lemma will be important in establishing the optimal way to add a maneuver to an existing schedule, as well as in proving the existence of certain types of optimal schedules later in this section.

Lemma 5.2.1. *Let $\psi^1 : \mathcal{R}^1 \mapsto [0, T_f]$, $\psi^2 : \mathcal{R}^2 \mapsto [0, T_f]$ be two schedules where ψ^1 is fixed. Then given any time t such that*

$$0 \leq t \leq T_f - \max_{r_k \in \mathcal{R}^2} \{\psi^2(r_k) + \delta t_k\}, \tag{88}$$

we can find a time t^* such that:

1. *t^* belongs to the same interval as t.*

2. *There exist $\bar{\tau}_i \in \bar{\Psi}^1$, $\tau_j \in \Psi^2$ such that*

$$\bar{\tau}_i = \tau_j + t^*. \tag{89}$$

3. *The superposition of ψ^2 on ψ^1 at t^* is no more costly than its superposition at t, i.e.:*

$$\mathsf{Tdwn}(\psi^1 + \psi^2_{t^*}) \leq \mathsf{Tdwn}(\psi^1 + \psi^2_t).$$

Proof. We immediately dismiss two trivial cases: the case where condition (89) already holds for t, and the case where

$$t > \max_{r_k \in \mathcal{R}^1} \{\psi^1(r_k) + \delta t_k\},$$

the latter having the obvious solutions $t^* = \max_{r_k \in \mathcal{R}^1} \{\psi^1(r_k) + \delta t_k\}$, since $y = \mathbf{1} \implies \omega = \mathbf{1}$. In the remaining cases, we begin by defining the following two quantities, illustrated in Figure 12:

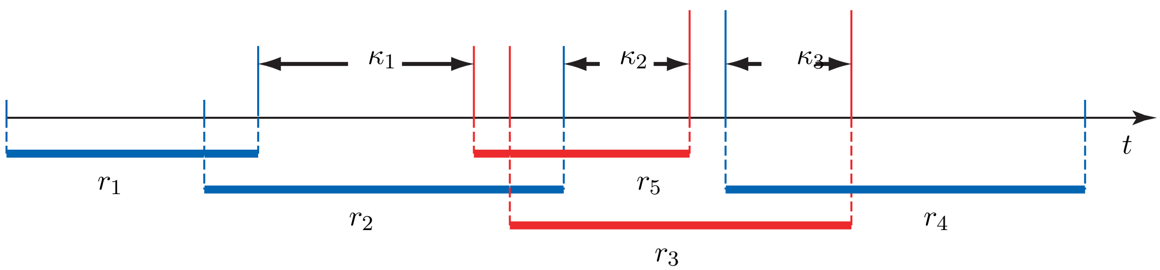

Figure 12: Calculation of δ_1. The blue schedule is ψ^1 and the red schedule is ψ^2_t. We have $\delta_1 = \max\{-\kappa_1, -\kappa_2, -\kappa_3\}$.

$$\delta_1 = \max_{\bar{\tau}_i \in \bar{\Psi}^1, \tau_j \in \Psi^2} \{\bar{\tau}_i - t - \tau_j < 0\} \tag{90}$$

$$\delta_2 = \min_{\bar{\tau}_i \in \bar{\Psi}^1, \tau_j \in \Psi^2} \{\bar{\tau}_i - t - \tau_j > 0\} \tag{91}$$

The two quantities δ_1, δ_2 define the maximum distances schedule ψ_2 can be shifted backward and forward in time, respectively, without changing the order of the $\tau_\ell \in \bar{\Psi}^t$, where $\bar{\Psi}^t$ is the set induced by $\psi^t = \psi^1 + \psi^2_t$. In other words, if $\delta \in [\delta_1, \delta_2]$,

$$\bar{\tau}_i < \tau_j + t \implies \bar{\tau}_i \leq \tau_j + t + \delta$$

and

$$\bar{\tau}_i > \tau_j + t \implies \bar{\tau}_i \geq \tau_j + t + \delta,$$

where $\bar{\tau}_i \in \bar{\Psi}^1, \tau_j \in \Psi^2$. Consequently, for any $\delta \in [\delta_1, \delta_2]$, the vector $x^{t+\delta}$ induced by the schedule $\psi^{t+\delta} = \psi^1 + \psi^2_{t+\delta}$ is the same, with the caveat that if $\delta = \delta_1$ or $\delta = \delta_2$, there will be at least one interval $I_i^{t+\delta}$ such that $\ell(I_i^{t+\delta}) = 0$. Further considering the length of the intervals $I_i^{t+\delta}$, we see that since each $|\bar{\tau}_i^{t+\delta} - \bar{\tau}_i^t| \in \{0, |\delta|\}$ for all $\bar{\tau}_i^{t+\delta} \in \bar{\Psi}^{t+\delta}, \bar{\tau}_i^t \in \bar{\Psi}^t$, we have:

$$\ell(I_i^{t+\delta}) - \ell(I_i^t) \in \{\pm\delta, 0\}. \tag{92}$$

We can then write

$$\begin{aligned}
\mathsf{Tdwn}(\psi^t) - \mathsf{Tdwn}(\psi^{t+\delta}) &= \sum_i x_i^{t+\delta}\ell(I_i^{t+\delta}) - \sum_i x_i^t \ell(I_i^t) \\
&= \sum_i x_i^t[\ell(I_i^{t+\delta}) - \ell(I_i^t)] \\
&= (a-b)|\delta|,
\end{aligned} \tag{93}$$

where a is the number of intervals such that $x_i^t = 1$ and $\ell(I_i^{t+\delta}) - \ell(I_i^t) = \delta$, and b is the number of intervals such that $x_i^t = 1$ and $\ell(I_i^{t+\delta}) - \ell(I_i^t) = -\delta$. Clearly, for $\delta \in [\delta_1, \delta_2]$, since $\mathsf{Tdwn}(\psi^{t+\delta})$ is a continuous piecewise linear function of δ and $\delta_1 < 0, \delta_2 > 0$, we can always ensure that

$$\mathsf{Tdwn}(\psi^t) - \mathsf{Tdwn}(\psi^{t+\delta}) \geq 0 \tag{94}$$

for some $\delta = \delta_{i_{\text{opt}}}$. Moreover, it is clear from the definition of the δ_i that if we move the schedule by a δ_i, we are making it so that for some $\bar{\tau}_i \in \bar{\Psi}^1, \tau_j \in \Psi^2$, we have $\bar{\tau}_i = \tau_j + t + \delta_{i_{\text{opt}}}$, and therefore setting $t^* = t + \delta_{i_{\text{opt}}}$ yields a schedule as required. $\square$

We now state an important theorem that will allow us to optimally superpose two schedules by checking only a finite number of superposition points.

Theorem 5.2.1. *Let $\psi^1 : \mathcal{R}^1 \mapsto [0, T_f]$, $\psi^2 : \mathcal{R}^2 \mapsto [0, T_f]$ be two schedules on $[0, T_f]$ as described in Lemma 5.2.1. Then there exists a time t^* in the interval*

$$0 \leq t^* \leq T_f - \max_{r_k \in \mathcal{R}^2}\{\psi^2(r_k) + \delta t_k\}, \tag{95}$$

such that the following hold:

1. *For any t in the same interval as t^* we have*

$$\mathsf{Tdwn}(\psi^1 + \psi^2_{t^*}) \leq \mathsf{Tdwn}(\psi^1 + \psi^2_t). \tag{96}$$

2. *There exist $\bar{\tau}_i \in \bar{\Psi}^1$, $\tau_j \in \Psi^2$ such that*

$$\bar{\tau}_i = \tau_j + t^*. \tag{97}$$

Proof. This is a direct consequence of Lemma 5.2.1. For every t in the specified interval, there is a time $\bar{t}$ such that

$$\bar{\tau}_i = \tau_j + \bar{t}, \tag{98}$$

and

$$\mathsf{Tdwn}(\psi^1 + \psi^2_{\bar{t}}) \leq \mathsf{Tdwn}(\psi^1 + \psi^2_t) \tag{99}$$

Moreover, for every time t, there is an interval (δ^t_1, δ^t_2) such that for all $t+\delta,\ \delta \in (\delta^t_1, \delta^t_2)$, $\bar{t}$ remains the same. In some cases, $\delta^t_1 = -\infty$, and in others $\delta^t_2 = \infty$, but these two cases cannot arise together. From the construction of the intervals illustrated in Figure 12, it is clear that there is only a finite number of such intervals. Thus, the set of $\bar{t}$ is finite. As such, the set $\{\mathsf{Tdwn}(\psi^1 + \psi^2_{\bar{t}})\}$ is also finite, and must contain its infimum. We denote the corresponding $\bar{t}$ by t^*, and the theorem is proven. □

The following corollary is essential to justify the algorithms presented in the next section, and provides a method for finding an optimal superposition time for a single operation:

Corollary 5.2.1. *Let r be an operation with duration $\delta t < T_f$. Then there exists an optimal superposition of r onto an existing schedule $\psi : \mathcal{R} \mapsto [0, T_f]$ such that either the beginning or the end of the maneuver is a time $t = \tau_i \in \bar{\Psi}$.*

Proof. Follows immediately from Theorem 5.2.1 by defining a schedule $\psi^1 : r \mapsto [0, T_f]$ and optimally superposing it over ψ, which is held fixed. □

Clearly, verification of each point in $\bar{\Psi}$ to find the cheapest is a linear time task for a single operation. We therefore have an efficient way to find where to schedule a new operation so as to minimize downtime. Note that this is the *globally* optimal location, even though we are only looking at a finite number of points on the continuum. We are now ready to state the following theorem, which proves that we can search a discrete, finite space for an optimal solution to the scheduling problem.

Theorem 5.2.2. *For any set of maneuvers $\mathcal{R}$ and any time window $[0, T_f]$, during which operations are allowed to take place, there exists an anchored optimal schedule $\psi^* : \mathcal{R} \mapsto [0, T_f]$.*

Proof. The existence of an optimal schedule follows from the fact that the set $[0, T_f]$ is closed. Suppose, therefore, that we are given a non-anchored optimal schedule $\psi : \mathcal{R} \mapsto [0, T_f]$ such that for some r_k, $\psi(r_k) = 0$. If the schedule started at a later date, we could translate it back without loss of generality. This means that there is at least one anchored maneuver. Consider a sequence $\{r_{k_i}\}_{i=1}^{p}$ of maneuvers such that the following hold:

1. At least one of the relationships below holds for any $1 < i \leq p$:

   (a) $\psi(r_{k_{i-1}}) = \psi(r_{k_i})$.

   (b) $\psi(r_{k_{i-1}}) = \psi(r_{k_i}) + \delta t_{k_i}$

   (c) $\psi(r_{k_{i-1}}) + \delta t_{k_{i-1}} = \psi(r_{k_i})$

   (d) $\psi(r_{k_{i-1}}) + \delta t_{k_{i-1}} = \psi(r_{k_i}) + \delta t_{k_i}$

2. There is no sequence longer than $\{r_{k_i}\}_{i=1}^{p}$ and containing all the elements of $\{r_{k_i}\}_{i=1}^{p}$ that also fulfills Condition 1.

By construction, each of the sequences defined above has a uniquely defined set of satellites and associated starting times under schedule ψ. However, several sequences that fulfill the conditions may be defined over the same set of maneuvers. We only keep one of these, chosen arbitrarily. Denote by Q_i the set of all such sequences of length i, by P^{ij} the jth sequence in Q_i, and by $\mathcal{R}^{ij} \subseteq \mathcal{R}$ the maneuvers contained in P^{ij}. Note that by construction, $i, j \leq |\mathcal{R}|$. We now notice that for each P^{ij}, we can define a subschedule $\psi^{ij} : \mathcal{R}^{ij} \mapsto [0, T_f]$ such that

$$\psi^{ij}(r_k) = \psi(r_k), \quad \forall r_k \in \mathcal{R}^{ij} \tag{100}$$

We can now apply a recursive procedure as follows. Define by i^* the smallest i such that $Q_i \neq \varnothing$. Pick j such that $P^{i^*j} \in Q_{i^*}$, and optimally superpose ψ^{i^*j} over $\psi - \psi^{i^*j}$ using Theorem 5.2.1.

Input: The constraints function ω, the sets $\mathcal{R}$, $\mathcal{Y}_k$ for all k, the final time T_f.
Output: A schedule $\psi : \mathcal{R} \mapsto [t_0, t_f]$ for some $t_0 \geq 0$, $t_f < T_f$
Initialize:
$\psi : \varnothing \mapsto \varnothing$, an empty map.
$\tilde{\mathcal{R}} = \mathcal{R}$, the set of operations to schedule.
$\Psi = \{0, T_f\}$
begin
  while $\tilde{\mathcal{R}} \neq \varnothing$ **do**
    $(t^*, r_{k^*}) \leftarrow \arg\min_{t, t+\delta t_k \in \Psi, r_k \in \tilde{\mathcal{R}}} \{\mathsf{Tdwn}(\psi + \psi^k_{\tau_i})\}$,
      where $\psi^k : r_k \mapsto [0]$, and $0 \leq t < t + \delta t_k \leq T_f$ for all k.
    Set $\psi(r_{k^*}) = t_{i^*}$
    Update Ψ according to its definition.
    $\tilde{\mathcal{R}} \leftarrow \tilde{\mathcal{R}} \backslash \{r_{k^*}\}$
  end
  return *The map* ψ.
end

Algorithm 1: The Greedy Scheduling Algorithm.

This cannot affect the cost, since ψ is already optimal. P^{i^*j} will become a subsequence of a larger sequence fulfilling Conditions 1 and 2. Repeat this procedure recursively on the new schedule until $Q_{i^*} = \varnothing$. Update i^* and repeat until $i^* = |\mathcal{R}|$. When this occurs, all sequences are anchored either to 0 or T_f, thus proving the theorem, since each step keeps the cost the same. □

As will be seen in the following section, this theorem ensures that there is no loss of generality in the scheduling heuristics introduced therein—they searche a space that contains globally optimal solutions.

5.3 Heuristic and Repaired Solutions

We devote this section to presenting and analyzing two algorithms for the quick and intelligent scheduling of refueling maneuvers, as well as to an algorithm that determines, once a schedule is decided upon, if time may be added to some of the maneuvers without affecting downtime. First, we construct an initial solution via a greedy heuristic, using Corollary 5.2.1 as a building block. Then, we apply a repair procedure akin to that used by some neural networks to drive infeasible solutions of CSP towards feasibility [35, 63]. Finally, we try to extend maneuvers backward or forward in time as much as possible without increasing downtime.

5.3.1 The Greedy Heuristic

Algorithm 1, the Greedy Scheduling Algorithm (GSA) applies a greedy search method to the scheduling problem. The idea is that the algorithm will add one maneuver at a time, choosing the maneuver that causes the least *additional* downtime to the existing schedule. From Corollary 5.2.1, we know that an optimal way to add an operation to an existing schedule ψ is to have one of its endpoints match one of the points in the set $\bar{\Psi}$. The algorithm takes advantage of this fact by optimally adding one operation to the schedule with each iteration. It chooses this operation r_{k^*} and its initialization time t^* by selecting the cheapest combination to add to the current schedule, found by exhaustive search. We can thus prove the following:

Proposition 5.3.1. *The Greedy Scheduling Algorithm terminates in $O(n^4)$ where $n = |\mathcal{R}|$, with an anchored schedule ψ.*

Proof. We begin by proving the runtime, assuming that verifying the operability conditions takes constant time. The loop runs n times. Within this loop, setting the value of $\psi(r_{k^*})$ and updating Ψ and $\tilde{\mathcal{R}}$ are constant time operations. The determination of t^* and r_{k^*} is a cubic-time operation. To see this, consider that the assignment is equivalent to the following procedure: for each maneuver yet unscheduled and at each addition point, create a new partition Ψ^1, evaluate the vector x_i^1 at each interval I_i^1, and add the cost. The creation of the partition takes constant time, since it only involves adding one point to Ψ. Each evaluation of an element of x^1 is carried out in constant time by assumption, and the addition is constant time. Thus, for each maneuver, the time spent on evaluating the Tdwn of an addition point is $O(n)$. Since there is a linear number of addition points (at most $n+2$, by construction), finding the optimal addition point is $O(n^2)$. Since this must be carried out for each maneuver, and there are at most n of them, this is also $O(n^3)$. Thus, the determination of t^* and r_{k^*} takes $O(n^3)$ steps. Since this phase is repeated n times, we have a run time of $O(n^4)$.

The fact that the schedule is anchored is easily seen by noting that the first maneuver added is anchored by construction, and each additional maneuver is therefore anchored as well. □

That the output of the algorithm is an anchored schedule shows the importance of Theorem 5.2.2, since without it we would have not guarantee that the GSA does not sacrifice optimality by limiting

Input: The constraints function ω, the sets $\mathcal{R}$, $\mathcal{Y}_k$ for all k, the final time T_f, and an anchored schedule ψ.
Output: A schedule $\psi : \mathcal{R} \mapsto [t_0, t_f]$ for some $t_0 \geq 0$, $t_f < T_f$
Initialize:
$\tilde{\mathcal{R}} = \mathcal{R}$
Ψ as per its definition.
begin
  while $\tilde{\mathcal{R}} \neq \varnothing$ **do**
    $r_{k^*} \leftarrow \arg\max_{r_k \in \tilde{\mathcal{R}}} \{\mathsf{Tdwn}(\psi) - \mathsf{Tdwn}(\psi - \psi^{r_k})\}$,
      where $\psi^{r_k} : r_k \mapsto \{\psi(r_k)\}$.
    $\psi^* = \psi - \psi^{r_{k^*}}$
    $t^* = \arg\min_{t \in \Psi^*} \{\mathsf{Tdwn}(\psi^* + \psi_t^{r_{k^*}})\}$,
      where $0 \leq t < t + \delta t_{k^*} \leq T_f$.
    $\psi(r_{k^*}) \leftarrow t^*$
    Update Ψ according to its definition.
    $\tilde{\mathcal{R}} = \tilde{\mathcal{R}} \backslash \{r_{k^*}\}$
  end
  return *The map* ψ.
end

Algorithm 2: The Repair Algorithm. We assume that the determination of optimal times is done by using Corollary 5.2.1. Note that the algorithm can be repeated any number of times, until the cost is either static or zero.

its search to anchored schedules. This is not in itself a guarantee of optimality, but at least it is a reassuring certificate that we have not excluded optimal solutions with our search method.

We conclude this section by pointing out that while quartic polynomial time is somewhat slow, in the constellation refueling context, this can be sped up by an order of magnitude, resulting in cubic worst-case performance. Indeed, we can achieve this by giving each satellite some individual computing powers so that it can, on its own, find the optimal starting time of a maneuver it is meant to carry out. This will allow the determination of the cheapest maneuver and its initialization time to occur in $O(n^2)$. In addition, note that the real runtime is in fact proportional to $n^4/4$, because the number of points in Ψ increases from 2 to $n+2$ points in increments of 1, resulting in an average number of points of $(n+4)n/(2(n+4))$ or $n/2$, and likewise for the number of maneuvers that must be verified at each step, which decreases from n to 1 in increments of 1.

5.3.2 The Repair Algorithm

The solution obtained from the Greedy Scheduling Algorithm is not guaranteed to be optimal. In fact, no optimality guarantees can be made at all, as far as we currently know. We would therefore want to find a method to improve this solution, as long as it does not substantially worsen the run-time. Such a method is suggested by repair methods to improve infeasible solutions to make them feasible in CSP [35]. The principle, as applied to the scheduling problem, is simple: select the maneuver which incrementally adds the most downtime to the current schedule, and try to find a better placement for it.

Algorithm 2 states this mathematically. This algorithm terminates after trying to place each maneuver once. This is not dependent on the improvement yielded, and so can be repeated until no further improvement is achieved. In direct correspondence with Proposition 5.3.1, we can prove the following:

Proposition 5.3.2. *The Repair Algorithm terminates in* $O(n^3)$.

Proof. All the updating operations occur in constant time. The determination of r_{k^*} requires finding $\mathsf{Tdwn}(\psi - \psi^{r_k})$ for all operations, where $\psi^{r_k} : r_k \mapsto \psi(r_k)$. This requires checking the operability of at most n intervals for each operation. Thus, the total running time for this operation is $O(n^2)$. Likewise, the determination of t^* requires a linear number of times to be checked, each of which requires the operability of a linear number of intervals to be checked, yielding a run-time of $O(n^2)$. One pass of the algorithm then takes $O(n^3)$, since every maneuver will be tested this way. □

The repair algorithm does not return an anchored schedule, as is obvious by considering that the maneuver causing the most downtime may be the anchor maneuver of some other maneuver, or even a link maneuver, and nothing impedes either of them being unique. This is of no concern, since the repair heuristic can only improve the result of the previous heuristic; moreover, the schedule can be re-anchored in polynomial time. Another feature of the repair algorithm is that it may choose to leave the most costly maneuver where it is—it is quite possible that, given the existing schedule, the most costly maneuver is already at its optimum starting time.

Given the run-times of the two algorithms, it is advantageous to include the repair heuristic into any solution of the problem that uses the greedy heuristic, since its run-time is an order of magnitude faster, and it therefore helps at low computational cost: the total run-time of the combined algorithms is still quartic. One drawback of the algorithm, however, is that distribution does not reduce the $O(n^3)$ run-time, because the determination of t is still quadratic.

5.3.3 Reducing Fuel Cost

With the complete formulation of the problem given in the preceding sections, we are now able to take a matching given to us by one of the methods presented in [60, 19, 48, 47] and produce a schedule which hopefully conforms to our requirements in terms of constellation down-time incurred during refueling. The matchings are derived with a time-limit on the duration of the maneuvers, and then a time-limit is imposed on the duration of the refueling process—the time T_f.

We now turn our attention to reducing the fuel cost of the refueling process. As shown in [52], an important limiting factor in the fuel cost of satellite transfers is the time allotted to the maneuver. Given more time, the maneuver can often be carried out using less fuel—and in the case of a constellation where all the satellites are in the same circular orbit, the cost can be asymptotically driven to zero. In previous work on refueling maneuvers, we started out under the assumption that all the maneuvers would be carried out at once, and set a time-limit based on that fact. To obtain the schedule, we extended the time-limit for the refueling process to be carried out, without extending the time allotted to the individual maneuvers—we simply moved maneuvers around in time. We now close the circle.

The idea is to use the schedule, which presumably has desirable, or at least acceptable, effects on the downtime, as a guide to further lower fuel costs without increasing downtime by extending maneuvers that have endpoints inside periods of constellation downtime. The following theorem allows us to do this in a systematic manner:

Theorem 5.3.1. *Suppose we are given a schedule $\psi : \mathcal{R} \mapsto [0, T_f]$ for maneuvers in a constellation with known requirement vector ω. Suppose we are given a time $\bar{\tau}_i \in \bar{\Psi}$ and the sets $\mathcal{R}_i^b, \mathcal{R}_i^e$ of maneuvers beginning and ending, respectively, at $\bar{\tau}_i$. Then the following statements hold:*

1. *If $x_{i-1} = x_i$ and $\mathcal{R}_i^b = \varnothing$, all maneuvers $r_k \in \mathcal{R}_i^e$ can be extended in duration by $\bar{\tau}_{i+1} - \bar{\tau}_i$ without affecting downtime, as long as they maintain their original initialization time.*

2. *If $x_{i-1} = x_i$ and $\mathcal{R}_i^e = \varnothing$, all maneuvers $r_k \in \mathcal{R}_i^b$ can be extended in duration by $\bar{\tau}_i - \bar{\tau}_{i-1}$ without affecting downtime, as long as we set $\psi(r_k) = \bar{\tau}_{i-1}$ for all maneuvers so extended.*

3. *If $x_i = 0$, all maneuvers $r_k \in \mathcal{R}_i^e$ can be extended by $\bar{\tau}_{i+1} - \bar{\tau}_i$ without affecting downtime, as long as $\psi(r_k)$ remains constant.*

4. *If $x_{i-1} = 0$, all maneuvers $r_k \in \mathcal{R}_i^b$ can be extended by $\bar{\tau}_i - \bar{\tau}_{i-1}$ without affecting downtime, as long as we set $\bar{\psi}(r_k) = \bar{\tau}_{i-1}$ for all maneuvers so extended.*

Proof. The proofs of cases 1 and 2 are simple and similar, so we only prove the first case. If there are no maneuvers beginning at τ_i, then all maneuvers that are being carried out during $[\tau_{i-1}, \tau_i]$ must either be in $\mathcal{R}_i^e$ or must end at the earliest at τ_{i+1}. Since $x_{i-1} = x_i$, extending the maneuvers in $\mathcal{R}_i^e$ so that they end at τ_{i+1} can have no effect on downtime, because those maneuvers already in motion during $[\tau_{i-1}, \tau_i)$ and not in $\mathcal{R}_i^e$ will still be in motion during $[\tau_i, \tau_{i+1})$, and thus will leave the conflicts unchanged over that interval, even if the maneuvers are extended.

The proofs of cases 3 and 4 are again very similar, so we again only prove 3. This case is even simpler than the cases above. Here, we simply notice that if the constellation is already down, there is nothing to be lost in adding maneuvers to a given time-increment. Specifically, since $[\tau_i, \tau_{i+1})$ is already downtime, we can simply lengthen all maneuvers that end at time τ_i, i.e. all maneuvers in $\mathcal{R}_i^e$, to end at τ_{i+1}. No additional downtime can ensue. □

The theorem above makes it clear that under certain conditions, we may add time to maneuvers without increasing the downtime of the constellation. As an illustration, Figure 13 illustrates graphically what is involved in the fourth case of the theorem. Therein we see that the last maneuver can be extended significantly, because $x_4 = 0$ and $\mathcal{R}_5^b = r_4$, which will hopefully bring down its cost.

Thus, upon completion of the Greedy Scheduling and Repair algorithms, we can apply Algorithm 3, the Cost Reduction Algorithm, which simply implements verification of the conditions listed in the theorem, to reduce the fuel cost of a schedule. In the case of a circular constellation, this may result in significant fuel savings under the right conditions, but even for a non-circular constellation, fuel savings may be achieved if the transfers must happen quickly, i.e., in about the time required for a zero-revolution transfer. [52]

5.4 Refueling Methodology

Based on our work so far, we reach two conclusions: scheduling may significantly reduce constellation downtime from the worst-case scenario where all maneuvers are carried out simultaneously, and maneuver lengthening may lower the fuel cost of the entire P2P scheme. We therefore propose the following methodology for organizing satellite refueling:

Input: A schedule $\psi : \mathcal{R} \mapsto [0, t_f]$ on $[0, T_f]$
Output: A schedule $\psi : \mathcal{R} \mapsto [0, \bar{t}_f]$ on $[0, T_f]$, and a set $\mathcal{R}$ with modified maneuver durations.
Initialize:
Ψ as per its definition.
x as per its definition.

```
begin
    for i = 2, ..., N_Ψ − 1 do
        if x_{i−1} ≠ x_i then
            if x_i = 1 then
                δt_k ← δt_k + τ_{i+1} − τ_i   ∀r_k ∈ R_i^e
            else
                δt_k ← δt_k + τ_{i−1} − τ_i   ∀r_k ∈ R_i^b
                ψ(r_k) ← τ_{i−1}
            end
        else
            if R_i^e = ∅ then
                δt_k ← δt_k + τ_i − τ_{i−1}   ∀r_k ∈ R_i^b
                ψ(r_k) ← τ_{i−1}
            end
            if R_i^b = ∅ then
                δt_k ← δt_k + τ_{i+1} − τ_i   ∀r_k ∈ R_i^b
            end
        end
    end
    return The map ψ and the set R.
end
```

Algorithm 3: The Cost-Reduction Algorithm.

1. Determine a maximum acceptable downtime, say T_1.

2. Solve the matching problem to achieve the fuel sufficiency objectives subject to the time constraint T_1, as described by any of [60, 19, 48]. The downtime will be less than or equal to T_1.

3. Determine a maximum time of outages, say T_2. This is the time during which it is acceptable for the constellation to suffer downtime—but the limit of total downtime incurred during this period is still T_1.

4. Apply the Greedy Scheduling and Repair Algorithms as described in Section 5.3 to obtain a schedule on $[0, T_2]$.

5. Apply the Cost-Reduction Algorithm to improve total fuel cost.

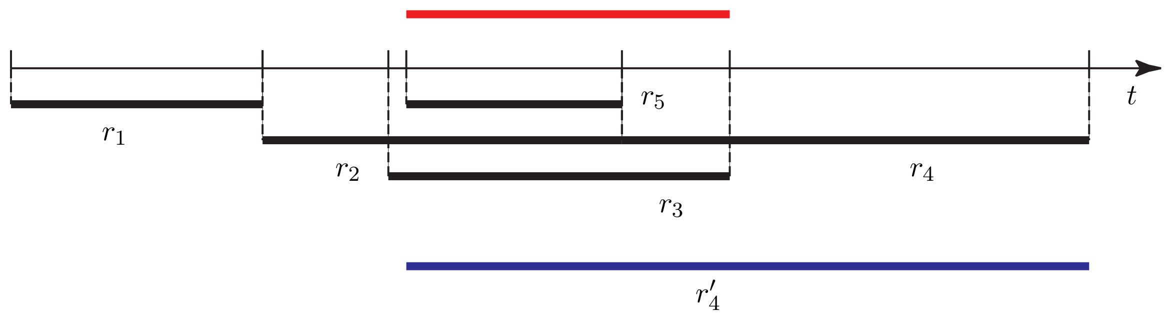

Figure 13: Reducing fuel cost. Suppose that the red line designates the time when the constellation is not operational. Clearly maneuver r_4 can be extended to cover the time that r_4' covers without adversely affecting the downtime.

This methodology is straightforward and allows for an interplay between hard constraints ("The constellation must not be off line more than T_1 units of time.") and malleable constraints ("The down-time may be spread over T_2 units of time."), thus allowing for added flexibility in the maneuvers. For example, if a given T_1 is much greater than the downtime obtained via the schedule, another round of matching and scheduling may be tried with a larger T_1, thus lowering costs across the board (rather than only for those maneuvers which are properly placed in the schedule). Given enough time, this may be repeated until everyone is satisfied with the refueling schedule.

We note here that the Cost-Reduction Algorithm, while useful in theory, might not be used in practice. Indeed, the times t_{ij} (corresponding to δt_k) may be set for operational reasons that are not taken into account in the constellation downtime requirements. As such, it may not be possible to extend maneuver durations. Nevertheless, it is useful to have a consistent way of doing so, if the flexibility exists.

5.5 Numerical Examples

We present two numerical examples to illustrate the method we proposed. In both cases, our starting point will be the matching obtained in [48] for the second example therein, illustrated in Figure 14. This is quite general, as a proper choice of restrictions allows us to model a large variety of situations. In this problem, we assumed a mixed-refueling strategy that left some satellites fuel deficient and others fuel sufficient. In particular, the first ten satellites in the constellation were fuel deficient, while the next ten were fuel sufficient. The matching produced corresponds to the bold lines in

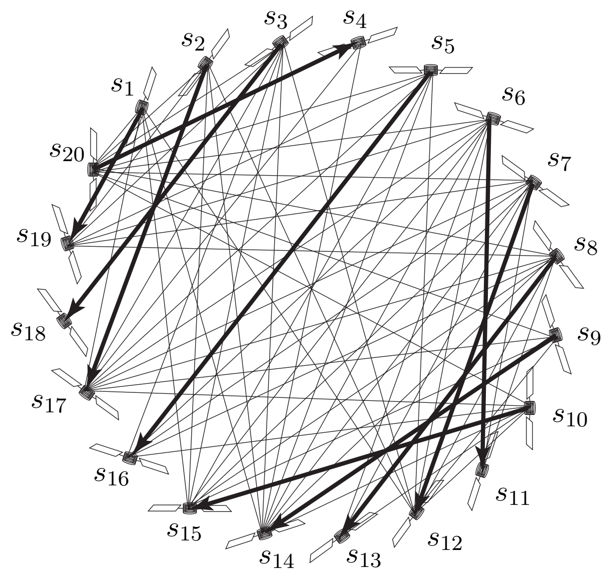

Figure 14: The optimal matching from Chapter 4.

the figure, with the arrows pointing from the active to the passive satellite. Note that the only fuel sufficient satellite to move is satellite s_{20}. While this is interesting in the context of the matching, it is only significant for us in relation to the operability conditions. It is, for example, possible to have operability conditions that are not affected by these maneuvers; the simplest example is to define a skeleton crew with $\gamma = \{11, 12, 13\}$. Since those satellites do not leave their orbits, the refueling maneuver is conducted without affecting operability.

Despite this generality, there are certain limitations to the model. Because we are considering a constellation of satellites sharing a circular orbit, we do not have to take into account differing orbital periods in the model of the in-constellation connectivity requirements, something that is by no means a given. In addition, as we mentioned before, we will not be taking dynamic outside connectivity requirements into account. The principles discussed in this paper readily extend to those situations, but their complexity would distract from the illustration of the basic concepts.

Since we use the same constellation that was studied in [48], we briefly recall that the ΔV required for the transfer was calculated using the phasing approach [33], which also yields a transfer time. We recall from [52] that although we set a limit of $T_1/2$ for the one-way transfer, there may be—and usually is—a coasting time at the end, i.e., a time when the satellite is already in its target slot. In our case, the coasting time is so small for all maneuvers (on the order of 0.05 units of time) that we set all maneuvers to take up T_2 units of time. For the purposes of these numerical examples, we assume that the fuel transfer is instantaneous, and the satellite starts the return trip immediately after the fuel transfer is completed. Table 8 gives the maneuver data.

Table 8: The active satellites and corresponding transfer times for each maneuver. For convenience, the maneuvers are indexed by the active satellites.

Man.	Sat.	δt_k	Man	Sat.	δt_k
r_1	s_1	20	r_7	s_7	20
r_2	s_2	20	r_8	s_8	20
r_3	s_3	20	r_9	s_9	20
r_5	s_5	20	r_{10}	s_{10}	20
r_6	s_6	20	r_{20}	s_{20}	20

Figure 15 shows the requirements of the first example. The different colors signal different skeleton crews. The green lines represent bi-directional communications between two satellites, and the green circles represent the skeleton crew associated with the connectivity requirement of the green communications network. The skeleton crew requirement for the red satellites is 2, that for the blue satellites is 3, that for the green satellites is 7. We note that the red and blue crews are not overlapping along the length of the orbit, i.e., if we were to list all the satellites in order around the orbit, all the satellites from one crew would appear before all the satellites from the other as long as we pick our starting satellite properly.

Table 9 shows the schedule resulting from the calculations carried out by setting $T_2 = 70$. Since the constellation was set in LEO, this is equivalent to about 120 hours, or five days, worth of time. Two facts stand out in this example: there are no situations where we may use the cost-reduction algorithm, and there is no benefit to using the repair algorithm.

A different situation arose when we tried differing maneuver lengths. Specifically, Table 10 shows how we varied the duration of the maneuvers while keeping the same constellation requirements. The results are given in Table 11. This time, the repair algorithm does reduce (to zero) the downtime incurred by the constellation under the GSA schedule, and each maneuver, with the exception of r_{20} can be given additional time while still maintaining the zero cost.

For our next example, we complicate matters by having more elaborate requirements—though they can still be constructed with skeleton and connectivity requirements. Using the same coloring scheme as before, we define overlapping constraints. The structure is easily explained if we note

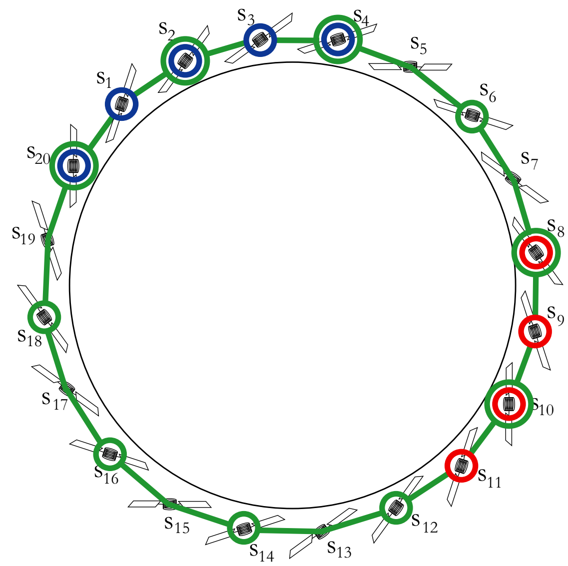

Figure 15: The constellation requirements for Example 1. The red, green and blue circles indicate the sets $\gamma_r, \gamma_g, \gamma_b$ corresponding to the red, green and blue skeleton crew requirements. Here, $p(\gamma_r) = 2, p(\gamma_g) = 7, p(\gamma_b) = 3$. The green skeleton crew requirement is the one associated with the connectivity requirement of the green communications network.

that we have defined twin requirements in addition the the connectivity requirement: first, that the red and blue satellites fulfill their standard skeleton requirement, and second that one of the satellites that is connected to the communications network be in its slot at all times. Figure 16 illustrates the situation, where it is clear that the second requirement amounts to two additional skeleton crews. We ran the problem with both the equal time maneuvers of Table 8 and the varying-time maneuvers of Table 10.

The results for the first case are given in Table 12. Surprisingly, the addition of two new skeleton crew requirements had no effect on the outcome of the algorithms. Based on this, an interesting study could be made of types of requirements that do not alter the results of the heuristics. The results would yield valuable information and allow designers to craft constellations that maximize utility while minimizing interfering requirements—i.e. constellations that can do more while maintaining good refueling possibilities. We also note that, yet again, neither the repair algorithm nor the cost reduction algorithm are useful.

The results of the same example run with the varying maneuver times of Table 10 are given in Table 13. In this case, as in the case of the first set of requirements, we have the repair heuristic and

Table 9: The results of running all three optimization algorithms on the first example with $T_2 = 70$. The downtime is the same for both heuristics.

Maneuver	GSA Initialization	Repaired Initialization	Additional Time
r_1	0	0	0
r_2	0	0	0
r_3	20	30	0
r_5	40	0	0
r_6	0	50	0
r_7	0	50	0
r_8	20	10	0
r_9	20	50	0
r_{10}	50	50	0
r_{20}	40	30	0
Downtime	10	10	

Table 10: The active satellites and corresponding transfer times for each maneuver in the second example. Note the differing δt_k.

Man.	Sat.	δt_k	Man	Sat.	δt_k
r_1	s_1	10	r_7	s_7	20
r_2	s_2	20	r_8	s_8	19
r_3	s_3	15	r_9	s_9	13
r_5	s_5	17	r_{10}	s_{10}	14
r_6	s_6	15	r_{20}	s_{20}	12

the cost reduction algorithm do lower the cost.

In the two constellations that we have presented, the greedy heuristic gives results that cannot be improved upon when the maneuvers are all of equal duration. We do not know at this stage whether or not this is a property related to the equal duration, or if it is due instead to constellation constraints. Several different randomly selected constraint sets were run, and in every case, the GSA returned schedules whose downtime the repair algorithm was unable to improve on. This strongly suggests a correlation, but we have no proof, nor were enough examples run to allow us to make a statement with any level of confidence. Thus, the effect of equal duration maneuvers on the GSA algorithm remains an interesting open question. That the Cost Reduction Algorithm was unable to extend times in the examples of equal duration cited above, on the other hand, seems to be an aberration that had to do with the specific choice of constraints.

Table 11: The results of running all three optimization algorithms on the second example with $T_2 = 70$. The downtime is reduced to nothing with the repair heuristic; moreover, additional time can be given to each maneuver to reduce the fuel cost.

Maneuver	GH Initialization	Repaired Initialization	Possible Additional Time
r_1	0	27	2
r_2	0	7	7
r_3	20	12	5
r_5	0	27	9
r_6	0	12	5
r_7	0	7	7
r_8	47	39	12
r_9	53	57	4
r_{10}	47	39	4
r_{20}	35	27	0
Downtime	8	0	

Table 12: The results of running all three optimization algorithms on the second example with $T_f = 70$. All the results are the same, indicating that the new constraints had no effect on the scheduling problem.

Maneuver	GH Initialization	Repaired Initialization	Additional Time
r_1	0	0	0
r_2	0	0	0
r_3	20	30	0
r_5	40	0	0
r_6	0	50	0
r_7	0	50	0
r_8	20	10	0
r_9	20	50	0
r_{10}	50	50	0
r_{20}	40	30	0
Downtime	10	10	

5.6 Summary

This chapter completes the purpose of this thesis: to provide a unified, coherent approach to satellite refueling. We used the matchings provided by the previous chapter as a starting point to create a schedule that will minimize the downtime incurred by the constellation as a result of the refueling maneuvers that must be executed. While the algorithms that we introduced do not guarantee global optimality, we can guarantee the following:

1. The Greedy Scheduling Algorithm searches a space containing optimal maneuvers.

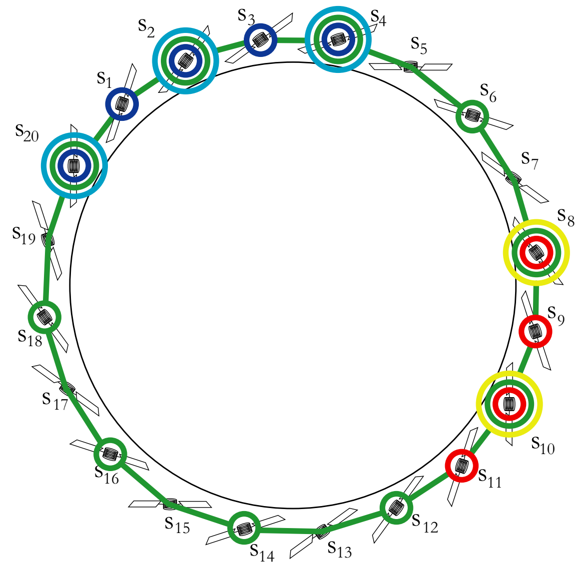

Figure 16: The constellation requirements for Example 2. All is identical to Figure 15, with the addition of the cyan and yellow skeleton crews γ_c and γ_y, with $p(\gamma_c) = p(\gamma_y) = 1$.

2. The Repair Algorithm can modify a schedule output by the GSA to greatly reduce its downtime.

3. Once both algorithms have been run, it may be possible to allow more time for certain maneuvers, thus driving down fuel cost.

This chapter shows that scheduling maneuvers over a period of time longer than the time allotted to each individual maneuver can not only ensure continuous constellation functionality, but also reduce fuel costs. Scheduling should therefore be an integral part of any P2P refueling solution method.

Table 13: The results of running all three optimization algorithms on the fourth example with $T_f = 70$. The repair algorithm is again effective in reducing downtime, and additional time can be allotted to over half the maneuvers.

Maneuver	GH Initialization	Repaired Initialization	Additional Time
r_1	0	7	7
r_2	0	35	0
r_3	20	23	6
r_5	0	38	0
r_6	0	55	0
r_7	0	15	9
r_8	47	17	0
r_9	53	55	2
r_{10}	56	56	1
r_{20}	35	38	5
Downtime	10	1	

CHAPTER VI

CONCLUSIONS AND FUTURE WORK.

This thesis has sought to present a coherent, unified framework to organize peer-to-peer refueling in constellations. The work can be divided into two parts: the matching problem and the scheduling problem. The order of presentation is the order in which a peer-to-peer refueling scenario would be carried out: first, obtain a matching based on the methods of Chapters 3 and 4, and then use the matching as the set of operations to be scheduled as described in Chapter 5. The following results were presented:

1. A novel formulation for the peer-to-peer refueling matching problem. In particular, a notion of fuel sufficiency and fuel deficiency is introduced, founded on factors independent of constellation state. This is in contrast to previous work which defined fuel sufficiency with regards to the constellation fuel average. By defining a fuel sufficiency threshold externally and shifting fuel sufficiency from objective to constraint, it is possible to avoid the two greatest limitations of previous work as follows:

   (a) The new objective function directly minimizes the fuel cost of the refueling maneuvers. This is an improvement over the previous objective function which sought to minimize deviation from the constellation fuel average [60], and incorporated the fuel cost indirectly by considering the average after the maneuvers were carried out. This naturally led to problems with determining the average when different matchings led to different averages. Such issues do not arise in the current formulation.

   (b) The satellites are no longer required to be identical. This presents a generalization of previous work, where averaging fuel content only made sense in the context of identical satellites. While the examples presented had identical satellites and circular co-orbital constellations, the formulation can be expanded to constellations on several different orbits, with several different types of satellites and fuel capacities.

The end-result is that the formulation ends up being a matching on a bipartite graph: an assignment problem. These problems are well-known in the literature, and have many methods for their solution.

2. A demonstration of the use of auction algorithms to solve the peer-to-peer refueling matching problem. Because of their inherently distributed nature, auction algorithms are uniquely suited to solve the asymmetric assignment problem that arises from the new problem formulation. We introduce the algorithms and illustrate several versions, namely:

   (a) The standard, serial version. It is shown to converge quickly to a solution.

   (b) The parallel version. A different solution is obtained, illustrating the fact that the auction algorithm guarantees optimality within bounds. If several matchings have costs within those bounds, the algorithm may return any one of them, depending (in deterministic implementations) on how the algorithm is written.

   (c) The asynchronous version. It is shown that the auction algorithm behaves well even under simulated information loss during the bidding process.

3. A formulation of the scheduling problem for satellite maneuvers in a constellation. The notion of constellation operability was defined in terms of three requirements: outside-world connectivity requirements, skeleton crew requirements, and in-constellation connectivity requirements. These encompass a large number of high-level operational requirements, allowing the constellation designer to account for communications, both within the constellation and with the outside world, as well as for tasks requiring multiple satellites in conjunction. Finally, they have the flexibility to allow constellations with multiple tasks.

4. Two scheduling algorithms to minimize the downtime incurred by the constellation as a result of the maneuvers. The first algorithm, the Greedy Scheduling Algorithm, is shown to search a space guaranteed to contain optimal solutions. The second, the Repair Algorithm, uses notions borrowed from constraint satisfaction programming to improve the results of the GSA. In some cases, the Repair Algorithm greatly improves the results of the GSA, in others, it has no effect. Both algorithms were shown to have polynomial runtimes.

5. An algorithm to add maneuver time to those maneuvers which could be extended without adding downtime to the refueling process. This algorithm extends the maneuvers while maintaining one of their endpoints constant. It may be applied repeatedly until no further extension is possible. This algorithm results in lower fuel costs, which was the objective of the matching problem.

6. A methodology combining all the elements developed in this thesis to approach satellite refueling.

6.1 Future work

Because of the relatively new subject matter this thesis treats, there are a lot of interesting questions. We list a few of them here, though more exist.

6.1.1 Comparison with Single Spacecraft Refueling

As mentioned in Chapter 1, we believe that the methodology presented in this thesis, when combined with external refueling in a mixed refueling strategy, can be cheaper than the methodology of refueling the constellation through some kind of tanker spacecraft that visits every satellite. Nevertheless, it would be good to confirm this numerically. In particular, setting up a fair comparison is a nontrivial problem, since a tanker will presumably refuel all the satellites to capacity, while a mixed refueling strategy will necessarily leave the satellites with less than their full capacity of fuel. The cost analysis could then be carried out either by using the tanker only to fill the satellites to the point that they would be filled at through a mixed refueling strategy, or by comparing the long-term cost over several refueling cycles of both strategies. It is even conceivable that different conclusions will be reached in these studies.

6.1.2 Extension to mid-way rendezvous points.

As seen in Chapters 2 and 3, we based this work on the notion that only one satellite would leave its orbit at one time. This is not necessarily the optimal solution—indeed, if a fuel deficient satellite is too depleted to meet a fuel sufficient satellite, but can meet it halfway cost-wise (i.e., in a different orbit), such a matching might be more economical. However, allowing for such situations would complicate the scheduling problem, as *two* satellites would need to leave their orbit in order for this particular pairing to occur. The algorithms of Chapter 5 are still valid, but the results may be poor. Nevertheless, the topic merits investigation, since it is fuel, not time, that is generally considered the most precious commodity in space.

6.1.3 Investigation of the Effect of Equal-Duration Maneuvers on the GSA

As mentioned in Chapter 5 the GSA seems to return schedules that can not be improved on by the Repair Algorithm when the maneuvers to be scheduled are of equal duration. This could be either an excellent feature of the GSA (namely that it finds optimal solutions in that particular case) or a limitation of the Repair Algorithm (perhaps it is incapable of improving such schedules, even if there are better schedules). It would therefore useful to determine two things: first, whether the pattern observed is general or limited to a special class of constraint structures (inside of which we might unwittingly have chosen the randomly picked examples), and second, if it is indeed a general pattern, whether it occurs because the GSA finds the optimal solution, or because the Repair algorithm is of limited use in the case of equal maneuver durations.

6.1.4 Extension to Fuel-Sufficient Satellites Visiting Several Fuel-Deficient Satellites

As mentioned in Chapters 1 and 5, space maneuvers are risky. Sending a large number of satellites on transfers for refueling purposes is therefore riskier than sending only a few, even if the risk is mitigated by sequencing the maneuvers. Thus it may be of interest to investigate whether reducing the number of satellites that leave their orbits can still lead to fuel sufficient constellations at sufficiently low fuel cost. On the one hand, the more fuel the active satellite gives, the lighter it becomes for its next trip. On the other hand, it was observed in Chapter 4 that most of the time, the active satellite was in fact fuel deficient satellite, which only took enough fuel to return to its original slot and be fuel sufficient. This is because, given equal ΔV costs and identical satellites, the fuel deficient satellite is the one with the least mass, and therefore lighter to move. It is therefore unclear what would happen in a case where the active satellites must be the heavier one. The subject nevertheless merits study.

6.1.5 Extension to Incorporate Risk Tolerance

Continuing with the notion of risk which we have skirted this entire thesis, we may wonder how to take into account the inherent risk of maneuver satellites. If methods of evaluating the risk of a maneuver causing damage to the satellite exist, each maneuver can have a certain risk factor associated with it. Perhaps, however, some satellites will be more important than others. We would therefore like to limit how much they move unless the situation truly warrants it. One way to do this would be to assign a weight to the fuel cost of a rendezvous based on the risk factor and risk tolerance of each satellite. The assignment problem of Chapter 3 would then be modified to reflect the new priorities of the refueling scenario. It would be interesting to investigate as well whether other methods exist which will have an equivalent effect.

6.1.6 Extension to Constellations with Prioritized Operational Requirements

A key assumption in our analysis of the scheduling problem in this work has been that a single operability condition violation resulted in constellation downtime. This is not necessarily realistic. In particular, if a modular constellation exists through which, for example, military surveillance operations are carried out while, in the background, NASA missions exploit the constellations structure to take measurements of earth or the heavens, it would be unlikely that a temporary disruption of the purely scientific studies of NASA would be equally damaging as a similar disruption of the military functions of the constellation. Thus, the question of how to introduce the flexibility to violate only certain constraints presents itself. One way would be to associate a weight with each operability requirement, and try to minimize the weighed sum of the downtimes incurred on each requirement. This approach would, however, require us to either greatly modify the three algorithms of Chapter 5, or perhaps even create new ones, because those algorithms depended on the binary nature of the constellation state to place maneuvers along the time continuum.

6.1.7 Extension to Constellations with Start-Time Restrictions on Maneuvers

Keeping in line with the assumption that certain functions of the constellation are more important than others, another modification that can be introduced is the notion that certain functions are imperative for a certain period of time in the $[0, T_2]$ window. Maneuvers breaking down these functions may not begin prior to a certain time. This restriction can be proposed in two different ways: first as a set of ready-dates on the maneuvers whose active satellites are involved in the function, and second as a prohibition of maneuvers that actively break down the function before the allowed time. The difference is perhaps best illustrated by example. Suppose the active satellites of maneuvers r_1, r_2, and r_3 are all part of a critical function that must be carried out until time t_1. Suppose that if two of the three satellites s_{i_1}, s_{i_2} and s_{i_3} are out of their slots, the function breaks down, i.e. suppose those satellites form a skeleton crew requiring two satellites. In the first version of the constraint, none of the maneuvers could begin before time t_1. In the second, no two of them can be going on at the same time t if $t < t - 1$. The first version is more akin to the ready-dates of the job-shop scheduling problem, while the second can be more easily incorporated and accounted for, we believe, in the framework of algorithms introduced in Chapter 5.

These are a few of the lines of research that are open, and which would build on the work in this thesis. It is clear from them that this thesis is very limited in scope and that realistic application of the results presented herein would require a lot of additional work to determine what the best way to approach a peer-to-peer refueling problem is. Nevertheless, this work presents a self-contained, logical methodology for organizing peer-to-peer refueling maneuvers. We hope it lays enough groundwork to justify further study of this new and fascinating field, a hybrid of aerospace engineering and optimization theory.

REFERENCES

[1] AJMANI, S., LISKOV, B., and SHRIRA, L., "Scheduling and simulation: how to upgrade distributed systems," in *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOSIX)*, May 2003.

[2] ANDREAS, N., "Space-based infrared system (SBIRS) system of systems," in *IEEE Aerospace Conference Proceedings*, vol. 4, (Aspen, CO), pp. 429–438, February 1997.

[3] ANDRINGA, J., "A system study on how to dispose of small satellites," Master's thesis, Massachussetts Institute of Technology, 2001.

[4] ARONOVITCH, L., "Satellite servicing - new capabilities in space," *Satellite Communications*, pp. 36–38, February 1985.

[5] BARTAK, R., "Constraint programming: In pursuit of the Holy Grail," in *WDS99 (Week of Doctoral Students)*, 1999.

[6] BATE, R. R., MUELLER, D. D., and WHITE, J. E., *Fundamentals of Astrodynamics*. New York, NY: Dover Publications, 1971.

[7] BATTIN, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics*. AIAA Education Series, Reston, VA 20191: American Institute of Aeronauctics and Astronautics, 1999.

[8] BERTSEKAS, D. P., "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, no. 1, pp. 7–66, 1992.

[9] BERTSEKAS, D. P. and CASTAÑON, D. A., "Parallel synchronous and asynchronous implementations of the auction algorithm," *Parallel Computing*, vol. 17, pp. 707–732, 1991.

[10] BERTSIMAS, D. and TSITSIKLIS, J. N., *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 2 ed., 1997.

[11] BESTE, D. C., "Design of satellite constellations for optimal continuous coverage," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-14, pp. 466–473, May 1978.

[12] BROWN, C. D., *Spacecrafy Propulsion*. Educational Series, Washington, DC: AIAA, 1996.

[13] BUTLER, A. A., SHARDA, H., and TANIAR, D., "Scheduling outages in distributed environments," in *Advanced Parallel Processing Technologies*, vol. 2 of *Lecture Notes in Computer Science*, (Berlin), pp. 281–291, Springer, 2003.

[14] CHATO, D. J., "Technologies for refueling spacecraft on-orbit," in *AIAA Space 2000 Conference and Exhibition*, AIAA, September 19-21 2000.

[15] CHATO, D. J., "Low gravity issues of deep space refueling," in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, no. AIAA 2005-1148, (Reno, Nevada), AIAA, January 2005.

[16] CHUNG, S.-J., MILLER, D., and DE WECK, O. L., "ARGOS testbed: Study of multidisciplinary challenges of future spaceborne interferometric arrays," *Optical Engineering*, vol. 42, pp. 2156–2167, September 2004.

[17] CLARKE, A. C., "Extra-terrestrial relays: Can rocket stations give world-wide radio coverage?," *Wireless World*, pp. 305–308, October 1945.

[18] DE WECK, O. L., SCIALOM, U., and SIDDIQI, A., "Optimal reconfiguration of satellite constellations with the auction algorithm," in *22nd AIAA Internation Communications Satellite Systems Conference & Exhibit*, (Monterey, CA), May 2004.

[19] DUTTA, A. and TSIOTRAS, P., "Asynchronous optimal mixed p2p satellite refueling strategies," in *Malcolm D. Shuster Astronautics Symposium*, no. AAS-2005-474, (Buffalo, NY), June 13-15 2005.

[20] FRANK, A., "On kuhn's hungarian method - a tribute from hungary.," *Naval Research Logistics*, vol. 52, no. 1, pp. 2–5, 2005.

[21] FRENCH, S., *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Mathematics and Its Applications, Chichester: Ellis Horwood Limited Publishers, 1982.

[22] HICKS, M., MOORE, J., and NETTLES, S., "Dynamic software updating," *ACM SIGPLAN Notices*, vol. 36, pp. 13–23, May 2001.

[23] HOFFMAN, A. and KRUSKAL, J., "Integral boundary points of convex polyhedra," in *Linear Inequalities and Related Systems* (KUHN, H. and TUCKER, A., eds.), ch. Integral boundary points of convex polyhedra, pp. 223–246, Princeton, NJ: Princeton University Press, 1956.

[24] HU, Y. and LI, V. O., "Satellite-based Internet: A tutorial," *IEEE Communications Magazine*, pp. 154–162, March 2001.

[25] HUSSEIN, I. I. and J., S. D., "Interferometric observatories in earth orbit," *Journal of Guidance, Control and Dynamics*, vol. 27, no. 2, pp. 297–301, 2004.

[26] JACKSON, J., "Scheduling a production line to minimize maximum tardiness," management science research project 43, University of California Los Angeles, 1955.

[27] JOHNSTON, M. D. and MINTON, S., "Analyzing a heuristic strategy for contraint-satisfaction scheduling," in *Intelligent Scheduling*, pp. 257–289, San Francisco, CA: Morgan Kaufman Publishers, 1994.

[28] KAMARKAR, N., "A new polynomial time algoirthm for linear programming," *Combinatorica*, vol. 4, no. 3, pp. 373–395, 1984.

[29] KEPLER, J., *Harmonies of the world.* Philadelphia, PA: Running Press, January 2005.

[30] KHACHIYAN, L., "A polynomial algorithm in linear programming," *Soviet Mathematiks Doklady*, vol. 20, pp. 191–194, 1979.

[31] KLEE, V. and MINTY, G., "How good is the simplex algorithm?," in *Inequalities III* (SISHA, O., ed.), pp. 159–175, Academic Press, 1972.

[32] KOHN, D. M., "Providing global broadband internet access using low-earth-orbit satellites," *Computer Networks and ISDN Systems*, vol. 29, pp. 1763–1768, November 1997.

[33] LAMASSOURE, E., “A framework to account for flexibility in modeling the value of on-orbit servicing for space systems,” Master’s thesis, Massachussetts Institute of Technology, Cambridge, MA, 2001.

[34] LANGDON, W. and TRELEAVEN, P., “Scheduling maintenance of electrical power transmission networks using genetic programming,” in *Artificial Intelligence Techniques in Power Systems* (WARWICK, K., EKWE, A., and AGGARWAL, R., eds.), pp. 220–237, IEEE, 1997.

[35] MINTON, S., JOHNSTON, M. D., PHILIPS, A. B., and LAIRD, P., “Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems,” *Artificial Intelligence*, vol. 58, no. 1-3, pp. 161–205, 1992.

[36] NARVAEEZ, P., CLERGET, A., and DABBOUS, W., “Internet routing over leo satellite constellations,” in *Third AMC/IEEE Conference on Satellite-Based Information Services (WOSBIS ’98)*, October 1998.

[37] NEWTON, I., *Philisophiæ Naturalis Principia Mathematica*. Guil, A. and Innys, J., Royal Society Typesetters, 3 ed., March 1726.

[38] NOLET, S., KONG, E., and MILLER, D., “Autonomous docking algorithm development and experimentation using the SPHERES testbed,” in *Proceedings of SPIE: Spacecraft Platforms and Infrastructure* (TCHORYK, P. J. and WRIGHT, M., eds.), vol. 5419, pp. 1–15, SPIE–The International Society for Optical Engineering, August 2004.

[39] PAPADIMITRIOU, C. H. and STEIGLITZ, K., *Combinatorial Opimization: Algorithms and Complexity*. Mineola, NY: Dover Publications, 1998.

[40] PINEDO, M., *Scheduling: Theory, Algorithms and Systems*. Upper Sadle River, NJ: Prentice-Hall, 2002.

[41] PINSON, E., *Scheduling Theory and its Applications*, ch. 13, pp. 277–293. John Wiley & Sons, 1995.

[42] PRUSSING, J., “Geometrical interpretation of the angles α and β in Lambert’s problem.,” *Journal of Guidance, Control and Dynamics*, vol. 2, pp. 442–443, September/October 1979.

[43] PRUSSING, J., *A class of optimal two-impulse rendezvous using multiple revolution Lambert solutions.*, vol. 106, pp. 17–39. San Diego, CA: Univelt, Inc., 2000.

[44] PRUSSING, J. and CONWAY, B. A., *Orbital Mechanics*. Oxford, UK: Oxford University Press, 1993.

[45] RANEY, R. K. and PORTER, D., “WITTEX: An innovative multi-satellite radar altimeter constellation,” in *Report of the High-Resolution Ocean Topography Science Working Group Meeting* (CHELTON, D., ed.), (Cornvallis), Oregon State University, 2001.

[46] ROY, B. and SUSSMAN, B., “Les problèmes d’ordonnancement avec constraintes disjonctives,” tech. rep., SEMA, Paris, France, 1964.

[47] SALAZAR, A. and TSIOTRAS, P., “An auction algorithm for allocating fuel in satellite constellations using peer-to-peer refueling,” in *American Control Conference*, June 2006 (accepted).

[48] SALAZAR, A. and TSIOTRAS, P., "An auction algorithm for optimal satellite refueling," in *Georgia Tech Space Systems Engineering Conference*, no. GT-SSEC.F.5, November 8-10 2005.

[49] SALEH, J., LAMASSOURE, E., HASTINGS, D., and NEWMAN, D., "Flexibility and the value of on-orbit servicing: New customer-centric perspective," *Journal of Spacecraft and Rockets*, vol. 40, no. 2, pp. 279–291, 2003.

[50] SCHARF, D. P., PLOEN, S. R., HADAEGH, F., KEIM, J., and PHAN, L. H., "Guaranteed initialization of distributed spacecraft formations," in *AIAA Guidance, Navigation and Control Conference*, August 2003.

[51] SHEN, H. and TSIOTRAS, P., "Optimal scheduling for servicing multiple satellites in a circular constellation," in *AIAA Guidance, Navigation and Control Conference*, no. 02-4907, (Monterey, CA), August 5-8 2002.

[52] SHEN, H. and TSIOTRAS, P., "Optimal two-impulse rendezvous using multiple revolution lambert's solutions," *Journal of Guidance, Control and Dynamics*, vol. 26, pp. 50–61, January-February 2003.

[53] SHEN, H. and TSIOTRAS, P., "Peer-to-peer refuelling within a satellite constellation part i: The zero-cost rendezvous case," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 4, (Maui, HW, 2003), pp. 4435–4350, 2003.

[54] SHEN, H. and TSIOTRAS, P., "Peer-to-peer refuelling within a satellite constellation part ii: The nonzero-cost rendezvous case," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 4, (Maui HW), pp. 4363–4268, 2003.

[55] SHEN, H., *Optimal Scheduling for Satellite Refueling in Circular Orbits*. PhD thesis, Georgia Institute of Technology, March 2003.

[56] SHEN, H. and TSIOTRAS, P., "Using Battin's method to obtain multiple-revolution Lambert's solutions," in *AAS/AIAA Astrodynamics Specialists Conference*, (Big Sky, MO), AAS/AIAA, August 3-7 2003.

[57] SMITH, S., "Preliminary analysis of the benefits derived to us air force spacecraft from on-orbit refueling," *NASA. Johnson Space Center, Sixth Annual Workshop on Space Operations Applications and Research (SOAR 1992)*, vol. 2, pp. 637–655, February 1992.

[58] SUN, J. and MODIANO, E., "Capacity provisioning and failure recovery for Low Earth Orbit satellite constellations.," *International Journal of Satellite Communications and Networking*, vol. 21, pp. 259–284, June 2003.

[59] TSIOTRAS, P. and DE NAILLY, A., "Comparison between peer-to-peer and single-spacecraft refueling strategies for spacecraft in circular orbits," in *Infotech at Aerospace Conference*, (Crystal City, DC), September 26-29 2005.

[60] TSIOTRAS, P. and SHEN, H., "Peer-to-peer refueling for circular satellite constellations," *Journal of Guidance, Control and Dynamics*, vol. 28, no. 6, pp. 1220–1230, 2005.

[61] VADALI, S. R., SCHAUB, H., and ALFRIEND, K. T., "Initial conditions and fuel-optimal control for formation flying satellites," in *AIAA Guidance, Navigation and Control Conference*, (Portland, OR), August 9-12 1999.

[62] VANDENKERCKHOVE, J. A., “Tankersat - refueling satellites in geosynchronous orbit,” *Astronautics and Aeronautics*, vol. 20, pp. 54–58, September 1982.

[63] WALLACE, R. J. and FREUDER, E. C., “Anytime algorithms for constraint satisfaction and sat problems,” *ACM SIGART Bulletin*, vol. 7, pp. 7–10, April 1996.

[64] WERTZ, J. and BELL, R., “Autonomous rendezvous and docking technologies–status and prospects,” in *Proceedings of SPIE: Space Systems Technology and Operations* (TCHORYK, P. J. and SHOEMAKER, J., eds.), vol. 5088, pp. 20–30, SPIE–The International Society for Optical Engineering, August 2003.

[65] WRIGHT, M. H., “The interior point revolution in optimization: history, recent developments, and lasting consequences,” *Bulletin of the American Mathematical Society*, vol. 42, no. 1, pp. 39–56, 2005.

[66] ZHANG, W. and DIETTERICH, T., “High-performance job-shop scheduling with a time-delta TD(λ) network,” in *Advances in Neural Information Processing Systems*, (Denver, CO), 1996.

[67] ZHANG, W. and DIETTERICH, T., “A reinforcement learning approach to job-shop scheduling,” in *Proceedings of the International Join Conference on Artificial Intelligence*, (Montreal, Quebec), pp. 1114–1120, 1996.