Planning and Scheduling for Earth-Observing Small Satellite Constellations

by

Andrew Kitrell Kennedy

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Space Systems

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

Signature redacted

Author .....
Department of Aeronautics and Astronautics
August 23, 2018

Certified by..... Signature redacted .....
Kerri L. Cahoy
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Signature redacted

Certified by.....
Julie A. Shah
Associate Professor of Aeronautics and Astronautics

Certified by..... Signature redacted .....
William J. Blackwell
Associate Group Leader, MIT Lincoln Laboratory

Signature redacted

Accepted by .....
Hamsa Balakrishnan
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Planning and Scheduling for Earth-Observing Small Satellite Constellations

by

Andrew Kitrell Kennedy

Submitted to the Department of Aeronautics and Astronautics
on August 23, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Space Systems

Abstract

The growth of Earth-observing small satellite constellations requires effective, automated operations management. State-of-the-art techniques must be improved to manage scheduling of observation data collection, data routing through a crosslinked constellation network, and maintenance of limited onboard resources, as well as to enable scaling to hundreds of satellites.

This work has four primary contributions. The first is the development of a hierarchical smallsat constellation planning and scheduling system that addresses data routing and resource management. A centralized ground-based algorithm, the Global Planner, manages the whole constellation, while an onboard algorithm, the Local Planner, replans in real-time to handle urgent, unexpected observations. The second contribution is the development of the software infrastructure for simulating the constellation with high fidelity. The third is the analysis of system performance with a set of representative orbit geometries, ground station networks, and communications contexts. The fourth is the demonstration of routing of urgent observation data.

The Global Planner algorithm demonstrates execution on larger problem sizes than the state-of-the-art, by quickly executing for both long planning horizons (requiring $<$ 1 minute for a 1000 min. horizon) and many satellites ($<$ 30 mins for 100 sats). Representative constellation geometries are simulated and analyzed with a 6U CubeSat bus model, including a 10-sat Sun-synchronous Orbit Ring and a 30-sat Walker Delta constellation. The improvement using crosslinks in addition to downlinks is assessed over a set of metrics including observation data throughput, latency of data delivery to ground, average Age of Information (freshness) of observation data, and freshness of TT&C data. In every case, performance is found to improve when using crosslinks and downlinks versus only using downlinks. Unplanned, urgent observation data is routed effectively by the Local Planner, achieving comparable latency performance with regular observation data (median of 42 minutes versus 38 mins) in a 6-sat simulation.

This work enables efficient scheduling of operations for large, complex smallsat

constellations. Future work is discussed that promises further scalability and schedule quality increases from the algorithm architecture presented.

Thesis Supervisor: Kerri L. Cahoy
Title: Associate Professor of Aeronautics and Astronautics

Committee Member: Julie A. Shah
Title: Associate Professor of Aeronautics and Astronautics

Committee Member: William J. Blackwell
Title: Associate Group Leader, MIT Lincoln Laboratory

Acknowledgments

The author would like to acknowledge several colleagues for their contributions to this work. Emily Clements, for her abundant and profound help while working together on various aspects of constellation and communications link modeling. Much of the algorithm and software development detailed in this thesis drew directly from lessons learned during that work. Patrick Kage, for his copious, stack-traversing knowledge about all things software and his contributions to several components of the CIRCI-NUS code, particularly in overall architecture and visualization. Bobby Holden, for his insights and contributions, particularly for the Local Planner algorithm and Consellation Simulator software. Daniel McGann, for his inputs on implementation for the Consellation Simulator. Warren Grunwald, for contributions in clarifying future work needs.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 Earth-Observing SmallSat Constellations

As detailed in *Achieving Science with CubeSats* [88], small satellite constellation-based Earth Observation (EO) offers measurement advantages, including higher temporal resolution, multi-point instrument coordination, and low-latency data availability. CubeSats [10] and similar small satellite platforms ("smallsats", mass smaller than approximately 50 kg) are increasingly useful for EO applications, primarily because their low cost to develop and launch enables organizations to field many (tens to hundreds) dedicated sensor nodes on-orbit. While the range of different sensor types that CubeSats can feasibly support is more limited than for larger satellites, many sensing payload capabilities are feasible and maturing rapidly [100]. Instruments that can be hosted on a CubeSat include atmospheric sounders (*e.g.* MicroMAS [7], and TROPICS [6] in fig. 1-1), visible imagers [8], and hyperspectral imagers [79], and even potentially Synthetic Aperture Radars [104].

As smallsat payload capabilities mature, their data production rates grow as well, placing more and more importance on the effective management of data across EO constellations [49]. The TROPICS constellation is an example of a relatively low data volume application. Under development at MIT Lincoln Laboratory, the mission aims

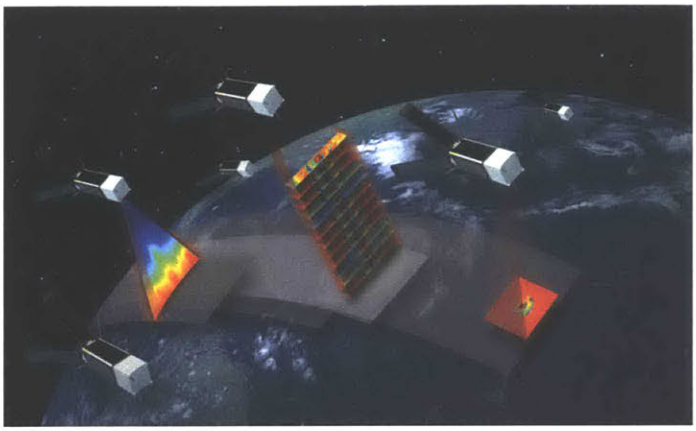


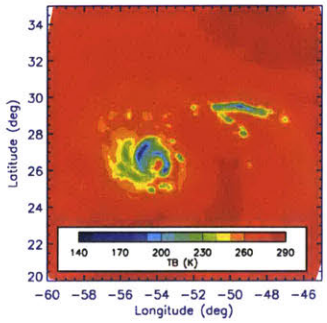
Figure 1-1: Earth weather monitoring with microwave radiometery by the TROPICS constellation, in development [72]

to provide rapidly updated data for weather models using a constellation of six 3U CubeSats with microwave radiometer instruments [6, 72]. The CubeSats continually scan the Earth's atmosphere below, producing data at a rate of roughly 16 kbps, which equates to about 1.5 GB per satellite per day.

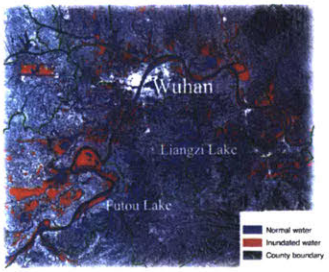
A higher data volume example is Planet Inc.'s flock of Dove satellites that perform moderate resolution visible Earth imaging, with the goal to provide updates of Earth's full surface every day [8, 69]. They accomplish this with a roughly 3U volume optical telescope, with a 3.5 meter ground sample distance from 400 km altitude. Each picture obtained takes about 4 megabytes (MB) and their constellation must downlink roughly 6 terabytes (TB) of land images every day to accomplish their daily full-surface update goal.

An even more data-intensive application is hyperspectral imaging, where the inclusion of many different frequency channels significantly increases output data rate. Mandl *et al.* outlined a CubeSat constellation carrying an instrument similar to the Hyperion hyperspectral imager on the Earth-Observing 1 mission [79]. The CubeSat hyperspectral imager would produce raw data rates in the Gbps range, producing TB of data in the course of a 90 minute orbit, assuming enough power were available to run continuously.

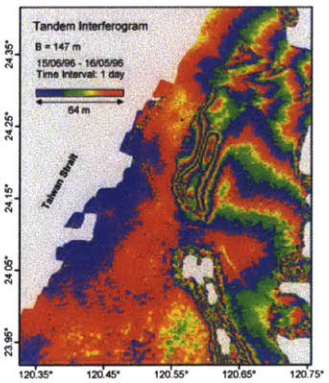
These large data production rates produce not only a large daily data volume to



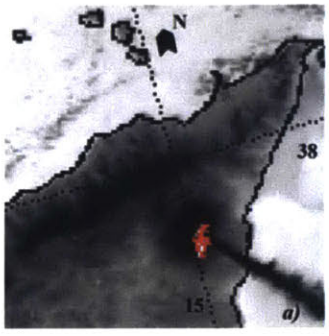
(a) ATMS microwave radiometer (brightness temperature) observations for Hurricane Edouard [6]



(b) Synthetic Aperture Radar (SAR) imagery of flooding in Wuhan, China [121]



(c) Topographic interferogram of ground displacement produced with SAR imagery [75]



(d) AVHRR instrument infrared imagery of Mount Etna erupting [92]

Figure 1-2: Example sensing modalities for small satellite Earth-observation disaster monitoring applications, including extreme weather, flooding, earthquakes, fires, and volcanic eruptions.

downlink to ground, but also large "instantaneous" chunks of data that need to be routed to ground quickly for applications like disaster monitoring. Disaster monitoring applications represent a particularly important use case for large small satellite constellations, because of the many, well-distributed sensor nodes that can be fielded for spotting and reporting changing conditions on the Earth's surface. Some examples of such applications are shown in 1-2. To ensure maximum situational awareness, we would like latency of data delivery for these observations to be nearly instantaneous [9].

The choice of an appropriate orbital geometry for EO constellations has been extensively studied in the literature, with a focus on designing geometries to provide a large percentage of Earth-surface coverage with a minimum number of satellites, and to minimize the revisit times between observations of surface locations [4, 46, 87, 73, 113]. The requirements on revisit time depend on the type of target being observed. Applications with high temporal resolution needs, such as disaster monitoring and meteorology, require average revisit rates ranging from sub-hourly to daily [97, 52].

1.1.2 Smallsat Communications

As data production rates increase, performance for satellites' communications subsystems must improve as well to be able to deliver all of the produced data to ground successfully. Historically, most organizations have flown smallsats with low-rate S-band or UHF radios [68], with data rates up to 3 Mbps. Such a low data rate significantly limits the amount of daily data volume that can be delivered to ground, as illustrated in fig. 1-3.

For this reason, organizations are deploying or developing smallsat radios at the X and Ka-bands and new optical-frequency-based ("lasercom") transmitters to enable higher data rates. Planet has successfully operated X-band radios, with reported downlink peak rates and average rates of 220 Mbps and 160 Mbps, respectfully [28]. Syrlinks and Tethers Unlimited are in various stages of developing commercially-available X band systems [31, 59]. Astro Digital has demonstrated Ka-band data rates of 40 Mbps and expected to achieve 320 Mbps in 2017 [23, 66]. The OCSD

(Optical Communications and Sensor Demonstration) mission from Aerospace Corporation aims to demonstrate lasercom up to 100 Mbps [117, 61], and Sinclair Interplanetary is developing a lasercom system for up to 1 Gbps [102]. MIT is developing several demonstrations, including the Nanosatellite Optical Downlink Experiment (NODE) [20] for LEO lasercom downlinks at 10-70 Mbps as well as full-duplex lasercom crosslinks. These higher data rate communication systems help to more effectively execute observation missions both by providing larger throughput to the ground, and lowering the energy required per bit of useful data delivered to ground. This is shown in fig. 1-3, which indicates that optical (lasercomm) systems in particular enable the operation of very high rate payloads.

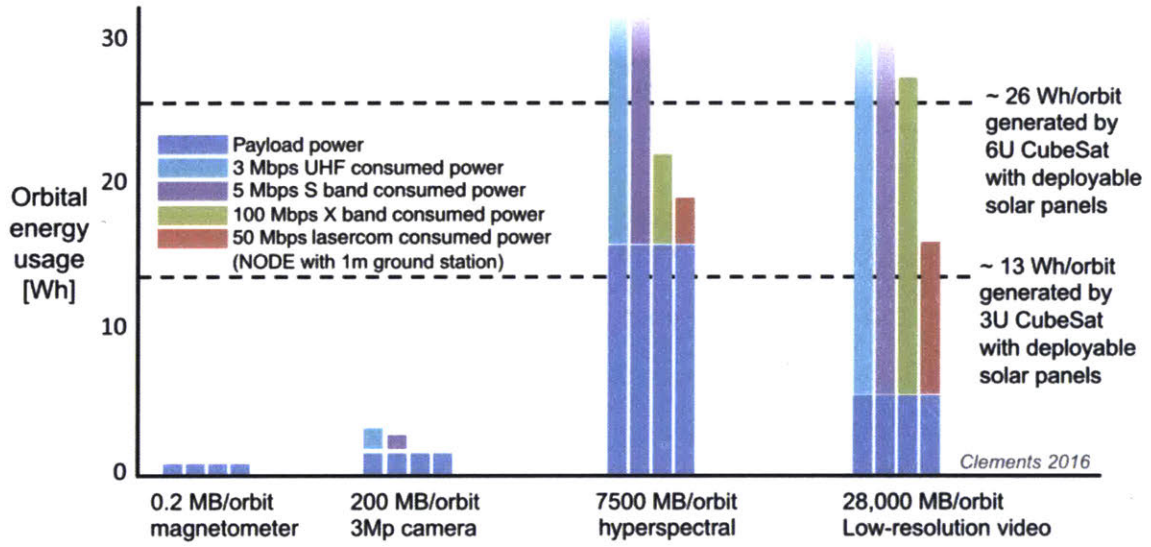


Figure 1-3: Energy usage for both data collection and complete downlink over a set of representative EO CubeSat payloads and communication systems. Notice that more data-hungry payloads exceed power resources for lower-rate communications systems [20].

Ground stations have traditionally been operated by the organization that launched a satellite, usually with one ground station for academic missions [68] and larger in-house networks for commercial organizations such as Planet Inc. [69]. Some new operators intend to offer access to their ground station networks as a service, including Spaceflight Industries [58] and Bridgesat [55], though these are at varying levels of maturity. The use of other, external communication satellites for crosslinking with

smallsats has been demonstrated with the Globalstar constellation, and studied for the Iridium constellation [112, 98, 19]. This capability offers the benefit of extremely low latency commanding and telemetry, but is expensive for bulk data delivery. The use of intra-constellation optical crosslinks for data routing has been studied as well [116].

1.1.3 Planning and Scheduling (P&S) for Constellations

As smallsat constellation sizes scale up, the complexity of efficiently operating the satellites becomes a major concern. Much of this complexity stems from the need to efficiently route large amounts of collected data to a limited set of ground stations, while ensuring communications conflicts do not arise between satellites. This need is complicated by the inherent energy limitations present onboard a smallsat-scale satellite; power usage needs to be carefully planned to maximize both data production and delivery to ground.

Traditional space mission operations architectures that require significant human involvement will not scale well to constellation sizes of tens or hundreds of satellites. Specifically, satellite operations scheduling with a human-in-the-loop planning process becomes so time consuming with more spacecraft that it can be a performance-limiting issue; without a solution, we can expect the number of human operators to scale linearly with the number of spacecraft [101]. For this reason, automated planning and scheduling is a key technology for EO smallsat constellations.

1.2 Literature Review

The work in this thesis focuses on Planning and Scheduling (P&S) for smallsat constellations, including the selection of timing for Earth observation and the routing of collected observation data to ground. Low latency delivery of data from spontaneously appearing, urgent observations is a key capability to be addressed (*e.g.* for disaster monitoring applications). Existing work related to these applications was split into three principal areas, as shown in fig. 1-4: 1) task allocation, planning &

scheduling, 2) network data routing, and 3) onboard replanning.

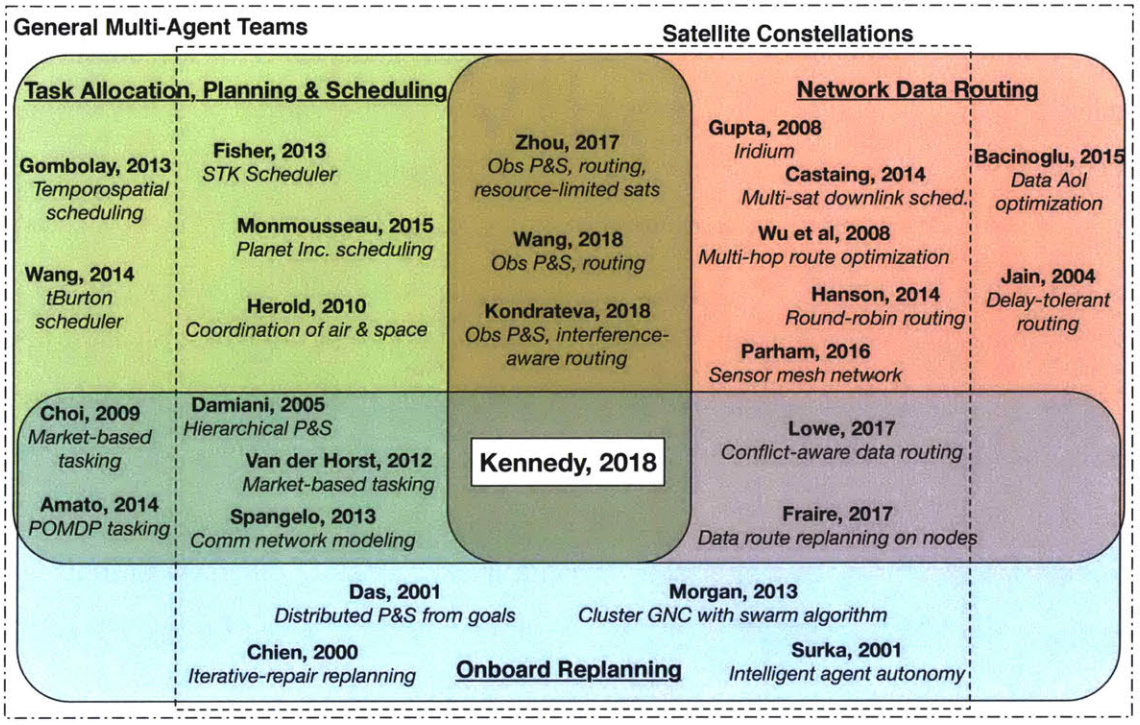


Figure 1-4: Literature review. Assessment focuses on three areas, represented by the three different colors within the Venn diagram. This thesis work lies at the intersection. Note that "Agents" are satellites, robots, or other decision-making nodes.

1.2.1 Task Allocation, Planning And Scheduling

The problem of Earth observation and communication planning has been extensively investigated in the literature, often with a focus on deciding which observation tasks to execute and how to best meet the myriad timing and priority constraints between observations [81]. Complexities are introduced when scheduling constellation operations, due to the interactions in observation and communications timing between spacecraft.

Centralized algorithms offer one avenue for scheduling, in which observation and activity timing for all satellites is solved concurrently. Fisher *et al.* detail the commercially available algorithms in the STK Scheduler tool [34, 33] which plans satellites' activity sequences to meet resource constraints and maximize a figure of merit function.

Operators can define their own scheduling problem instances with custom objective functions. Schedule solutions are determined with two classes of algorithms, "ordered" or sequential assignment algorithms, and neural algorithms that use a competition model. Monmousseau discusses the algorithms used for observation and satellite energy usage scheduling used at Planet Inc., based on Simulated Annealing and Mixed Integer Linear Programming (MILP) [83, 95]. An example schedule artifact is shown in fig. 1-5, illustrating a set of downlinks scheduled over a 24 hour period for multiple spacecraft in their constellation. These algorithms represent the state-of-the-art in planning and scheduling systems applied to operational small satellite constellations; Planet Inc. has demonstrated operations with more than 100 satellites in their constellation [28]. Herold *et al.* discuss a hierarchical system that features a centralized coordination algorithm that interfaces asynchronously with sub-planners on both air and space assets [50]. Schedule solutions are derived using a Mixed Integer Programming (MIP) model and solver. These algorithms efficiently address the problem of observation tasking for satellite networks, but do not deeply consider the routing of data through a constellation network.

There has also been a great deal of work on multi-agent cooperation outside of space applications. Note that "agents" are satellites, robots, or other decision-making nodes. The "tBurton" factored planner detailed by Wang *et al.* plans for interactions between multiple agents by reasoning about the temporal interdependencies of their tasks, and exploiting hierarchy to reduce the planning complexity [114]. Graph decomposition techniques are used to simplify the problem, and heuristic forward search is used to find a feasible solution. Gombolay *et al.* implemented a set of algorithms using a MILP formulation to quickly allocate and schedule sets of tasks to multiple robots on a factory floor, where temporally-evolving spatial constraints are critically important [40]. These approaches can quickly plan for large groups of robots, but lack consideration for the delay-tolerant character of satellite networks.

Other work in multi-agent coordination builds in capabilities for decentralized decision-making, to handle re-prioritization and replanning of tasks in near real time. The market-based task allocation algorithms discussed by Choi *et al.* use interagent

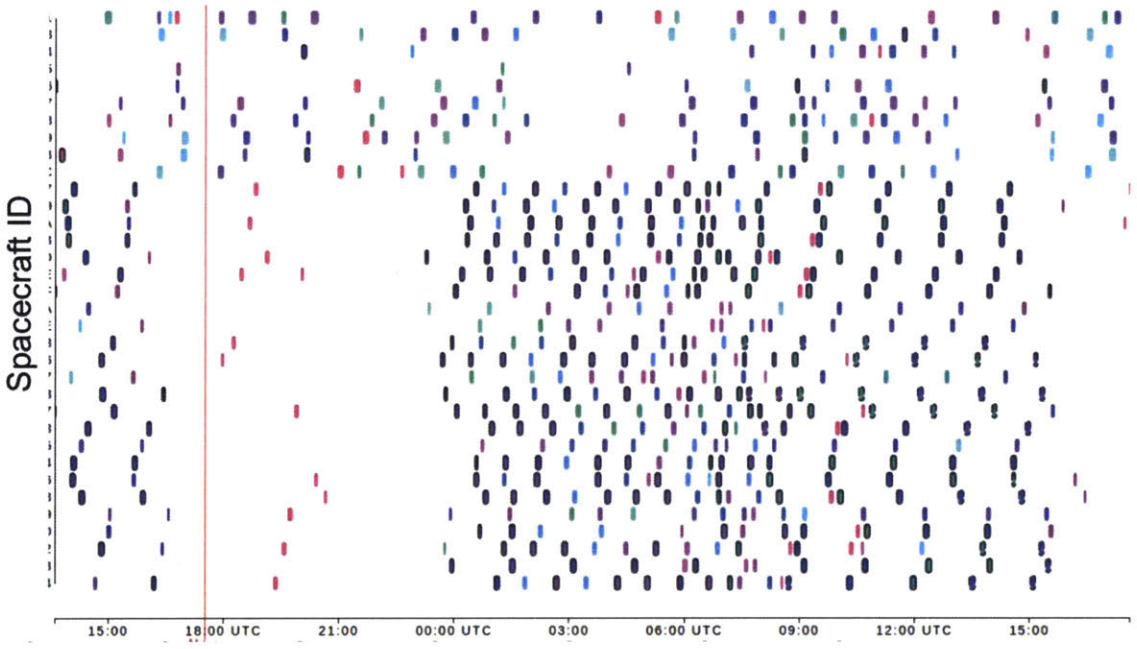


Figure 1-5: Example schedule of ground station overpasses for the satellites in the Planet Inc. constellation. Different colors represent different ground stations. This schedule artifact illustrates the scalability of their scheduling system to many ground stations and satellites [69]

bidding on tasks based on the agents' local valuation of them [18]. The agents come to a consensus on task allocations by propagating messages with information about their decisions through the network. Unfortunately, this consensus mechanism requires significant inter-agent communication, and is not robust in delay-tolerant networks, *i.e.* networks characterized by frequent changes in topology and often sparse links between the agents. The use of Decentralized Partially-Observable Markov Decision Processes (POMDPs) described by Amato *et al.* can be used to find an optimal, decentralized plan across multiple agents by explicitly reasoning about uncertainty in the agents' performance of activities [1, 2]. This technique is very computationally complex however, and does not necessarily scale to tens or hundreds of agents.

Techniques have also been developed for task allocation decision-making onboard satellites to provide additional planning flexibility. Damiani *et al.* discuss a system with satellite onboard activity planning and a centralized ground station system that distributes observation requests to satellites [25]. The centralized element is responsible for keeping track of the globally available tasks, which are distributed to satellites. The satellites then plan to fulfill tasks based on their current state. The approach explicitly considers onboard energy and memory constraints for satellites, and timing constraints across satellites. Dynamic Programming algorithms are used on both the ground and satellites for planning. This hierarchical approach is robust to the limited communications availability in small satellite constellations, but does not sufficiently address the tight coupling introduced by crosslinked data routing. Van der Horst and Noble investigate market-based algorithms to perform task assignment between satellites with intermittent crosslinks [110]. Spangelo derives an analytical modeling framework for space communications networks that hierarchically separates individual satellite responsibilities, and developed a simulation environment based on it [106, 105].

1.2.2 Network Data Routing

Data routing in constellation networks has also been extensively studied and demonstrated in real missions. The Iridium constellation can be considered the canonical

example of large-scale inter-satellite networking, since its introduction in the 1990s (detailed by Gupta [43], and Maine [78]). It features K-band crosslinks between the satellites, and L-band (customer) and Ka-band (operations) downlinks and uplinks. Crosslinks can occur with neighbors back and forward in the same orbit plane, and a neighbor on each closest orbit (4 directions per satellite). Packets routing is scheduled in individual frames, or discrete time steps, for the whole constellation, with the goal to minimize packet route distances (number of hops) from the point of origin at a customer to the destination at a gateway ground station [30]. Analytics are leveraged to predict traffic bottlenecks and route traffic to avoid them. While a successful and long-lived system, it does not take into account the delay tolerant nature of small satellite networks, because Iridium crosslinks operate continuously.

Other work has more directly addressed the limited communications of small satellite networks. Castaing used a MILP formulation to de-conflict and schedule downlinks for multiple satellites communicating with a set of ground stations, while also satisfying resource constraints [12]. Hanson *et al.* discuss the use of short distance crosslinks for the Edison Demonstration of Smallsat Networks (EDSN) mission, with a central satellite collecting data in a round-robin fashion from other nodes in a hub-and-spoke topology [44]. The EDSN mission was lost in a launch anomaly, but a follow-on mission named NODES achieved the sensing and communications goals of EDSN [45]. Parham *et al.* discuss the use of mesh networking for packet forwarding across a small cluster of satellite sensor nodes [91]. Wu *et al.* investigated the optimization of data routes in satellite clusters, applying a Tabu Search algorithm and Genetic algorithm to the minimization of both energy usage and route latency. Though the approaches discussed by Hanson *et al.*, Parham *et al.*, and Wu *et al.* do include data routing over crosslinks given smallsat resource constraints, the first two cannot adapt to changing conditions in the constellation due to the lack of real-time routing decision-making, and the third does not consider the effects of throughput bottlenecks in a larger network.

State-of-the-art algorithms for delay-tolerant networking in small satellite constellations incorporate the consideration of throughput capacities and packet traffic, and

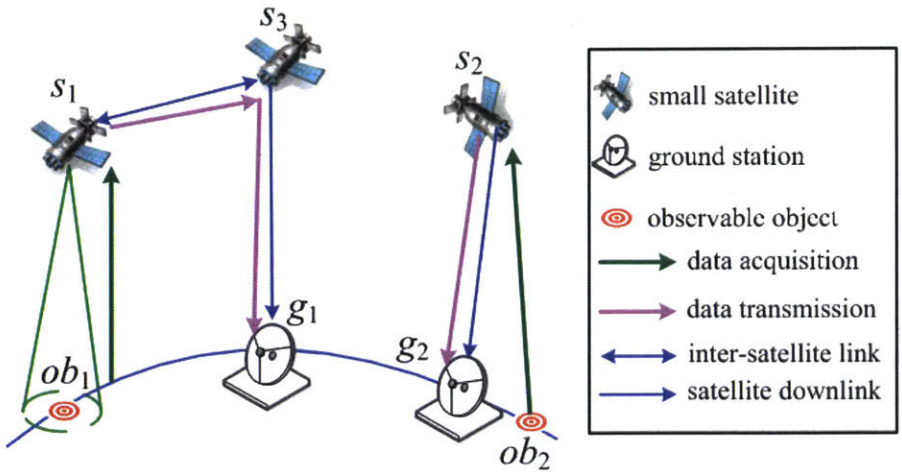


Figure 1-6: Network/operations model used by Zhou *et al.*, featuring observation data collection, crosslinks between satellites, and downlinks to ground stations [124]

explicitly schedule for the completion of observation tasks. Kondrateva *et al.* formulate and solve a MILP scheduling model for observation data routing, incorporating interference constraints between satellites crosslinking and downlinking at the same time [71]. Scheduled throughput results are presented for an 18 satellite constellation over a 100 minute planning window (100 minute planning window confirmed in personal communication with author). Wang *et al.* posit a similar MILP formulation and present results for "guarantee ratios" of observation mission performance and latency requirement fulfillment for as many as six satellites [115]. For their proposed throughput-capacity-aware algorithm AMRS, they demonstrate the delivery of roughly 60% of packets within a 120 minute delay requirement.

Zhou *et al.* present a more robust algorithm that not only considers throughput capacities in traffic, but also onboard energy and data storage constraints [124]. Schedules are created from the solution of a MILP problem that incorporates all of these constraints. Figure 1-6 illustrates the operational model used in their work, and fig. 1-7 illustrates the observation and communications "contact graph" that they model in their formulation. They propose two key algorithms: 1. "Mission Aware Contact Plan" (MACP), an algorithm that solves a MILP formulation to determine optimal throughput in a constellation and 2. "Algorithm based on Conflict Graph"

(ACG), a heuristic algorithm that solves a suboptimal version of the same problem by sequentially assigning data routing decisions over a set of timeslots. They present schedule results that show the delivery of roughly 90% of optimal total constellation data throughput for a six satellite scenario with a planning window of two hours in most cases. This algorithm includes much of the functionality required for effective data routing in a crosslinked smallsat constellation: scheduling of observations, traffic aware data routing, and conformity to energy and data storage limitations onboard. However, their results do not demonstrate scalability to large constellations of tens or hundreds of satellites; only a six satellite scenario was presented. They do not handle the minimization of data routing latency in their model. Throughput results from Zhou *et al.* are used as a basis for comparison in section 3.5.

The algorithms of Kondrateva *et al.*, Wang *et al.*, and Zhou *et al.* assume that all planning is performed in a centralized manner. Other routing algorithms incorporate the ability for satellites to reactively select data routes in real-time. Lowe *et al.* present a network-layer protocol running onboard that uses information about future contacts between nodes to choose least-conflicted routes [76]. Fraire presents an approach based on the delay-tolerant Contact Graph Routing algorithm, a distributed algorithm that chooses routes based on contact plans [35]. The algorithm is augmented with fault models based on mean-time-to-failure for network components, to achieve increased robustness. These algorithms however do not consider bulk data routing, nor onboard resource constraints.

A wide body of literature on data routing exists outside of the satellite community. Some relevant work includes that of Jain *et al.* [60], which contained one of the earliest coherent discussions of delay-tolerant networking, and Bacinoglu *et al.* [3], which examined the maintenance of data freshness (Age of Information, AoI) in sensor networks.

1.2.3 Onboard Replanning

Work on onboard replanning is less well-developed, and has not been extensively fielded in real small satellite missions. Replanning of data routing after un-preplanned

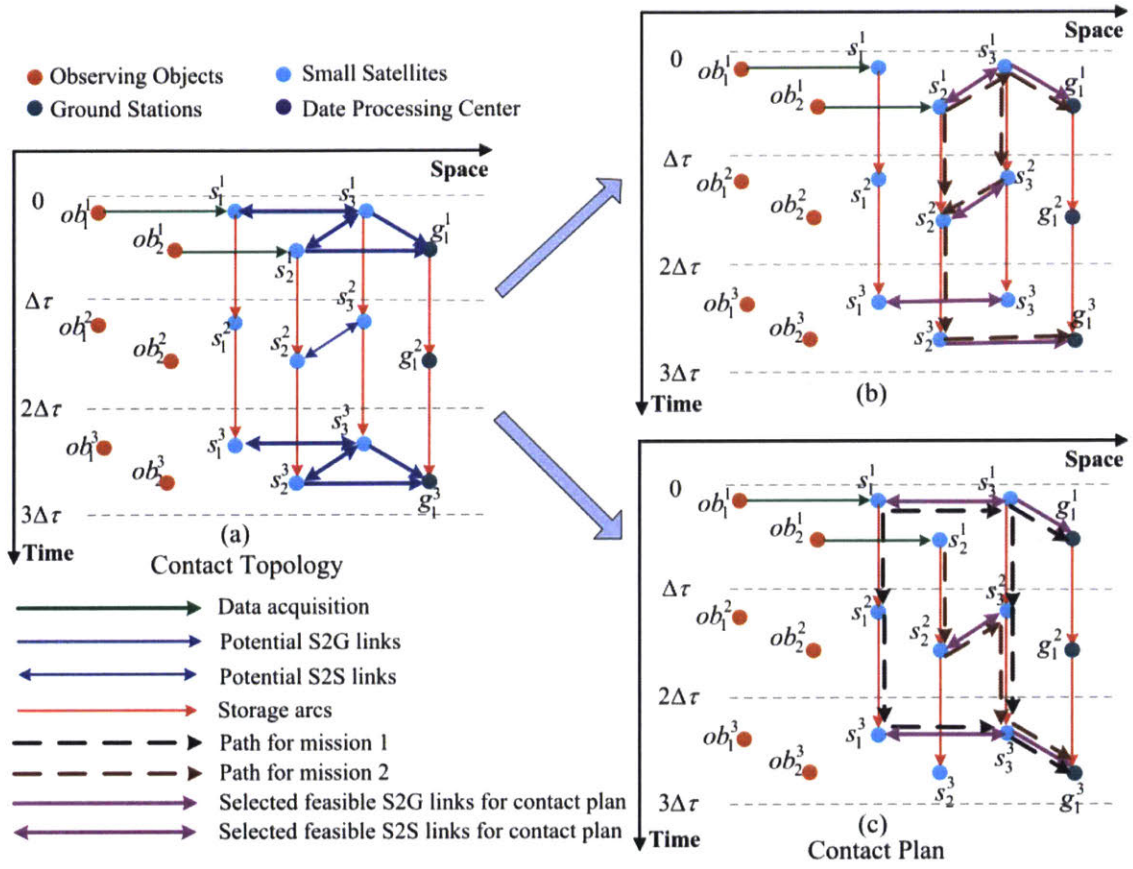


Figure 1-7: The "time-evolving graphs" used for reasoning about network contacts by Zhou *et al.* [124]. Potential contacts are shown on the left in (a) (the "contact graph"), and possible contact schedules are shown on the right in (b) and (c). "mission" is the term used for observation of a given target.

collection of urgent observation data presents a particular challenge for a large constellation. The intermittent nature of communications contacts with the ground and between satellites increases latency of data routing and limits the ability to distribute plan updates through the constellation. To our knowledge, an algorithm for rapid, bulk data routing replanning over intermittent links has not yet been demonstrated in the literature.

Chien *et al.* discuss the CASPER system, which uses an iterative repair algorithm onboard to incrementally make updates until a previously conflicted schedule no longer has conflicts [15, 103, 93, 16]. CASPER has been deployed on the Intelligent Payload Experiment (IPEX) CubeSat [13]. The algorithms do not consider data routing. Das *et al.* as well as Van der Horst investigate task allocation methods that utilize crosslinks between the satellites and require less network connectivity [27, 110]. These approaches help reduce the need for frequent communications with the ground, but sacrifice schedule quality across a widely-distributed constellation because they do not directly attempt to optimize plans for the whole constellation. Surka *et al.* and Schetter *et al.* discuss a multi-satellite cooperative architecture based on "expert agents" running onboard [107, 99]. The system was planned to be deployed on the TechSat-21 mission, augmented by the CASPER system designed by Chien *et al.* [14, 17]. The overall system provides significant operational independence, but requires the satellites be in a highly connected network. Morgan *et al.* detail a decentralized path planning algorithm for swarms of hundreds to thousands of satellites [84]. Though scalable, it does not address the data routing and resource planning needs of EO constellations.

Zheng *et al.* analyze a system with a "mother satellite" responsible for centralized decision making and "daughter satellites" that monitor their health in real-time and replan based on latest state [123]. Planning is performed on daughter satellites with a Genetic Algorithm using a tailored population mutation mechanism, Hybrid Dynamic Mutation. The work considers operations of a closely-coupled cluster of satellites, but does not specifically consider data routing in a delay-tolerant EO network.

1.2.4 Small Satellite Constellation Simulation

Much of the existing work on constellation P&S focuses on the evaluation of either a centralized planner or onboard planner as an isolated algorithm. A common limitation of these evaluations is that the planner is not treated as a constituent component of a larger P&S system featuring both planner types. Also, many analyses focus on the evaluation of schedule quality after a single planner execution, as opposed to assessing constellation performance over a full, extended simulation run with many executions of the underlying planner. Several authors analyze schedule quality from a centralized planner [124, 71, 36, 123, 115] as a single execution. Van der Horst *et al.*, Lowe *et al.*, and Fraire *et al.*, analyze onboard algorithms for task allocation and data routing in constellations without a centralized element [110, 76, 35].

Other approaches incorporate both types of planners, with varying levels of planning decision making. Damiani *et al.* evaluate a P&S system featuring a centralized element which distributes tasks to satellites, which then are reponsible for final task scheduling and execution [25]. Herold *et al.* also assess a hierarchical P&S featuring task distribution to satellites as well as Unmanned Aerial Vehicles [50]. These integrated approaches feature coupling between planners and present results over extended simulation runs, but do not model crosslinks between satellites.

In additional work, software frameworks have been developed for constellation simulation. Fraire *et al.* propose a simulation environment for smallsat delay-tolerant network simulation, DtnSim, that incorporates crosslinks between satellites, onboard planning, and contact access information distribution from a ground station, but no in-depth centralized planning. Surka *et al.* provide a software system for distributed cooperation in satellite clusters, with onboard decision making allocated to purpose-built satellite software modules [107]. Grogan *et al.* develop a simulation environment for federated satellite constellation with crosslinks but highly uncoupled onboard decision making [42].

There is a need for an Earth-observing smallsat constellation simulation environment incorporating crosslinks, hierarchical (centralized,onboard) planning, and

continuous replanning over a long simulation period of interest. In addition, most smallsat P&S work uses software with relatively restricted access outside of academia, *e.g.* AGI's STK for orbit propagation and access period calculation [53], and doesn't provide a capability to visualize scheduling results in 3-dimensional constellation rendering. Such a visualization capability would be helpful for understanding P&S algorithms' operational decisions and debugging.

1.3 Thesis Overview

1.3.1 Research Gaps and Motivation

Previous work has investigated many aspects of the smallsat P&S problem, and has formulated and analyzed algorithms that incorporate several key characteristics: 1) scheduling of observation activities 2) routing of data from observations through a delay-tolerant crosslink and downlink network in the constellation 3) onboard-resource-aware scheduling, including energy and data storage and 4) optimization of data throughput and latency, though not necessarily at the same time. The algorithms in the P&S system presented in this thesis incorporate these characteristics while also demonstrationg execution on larger problem sizes, in terms of both the length of the algorithm planning horizon and the number of satellites in the constellation.

The system also demonstrates the use of an onboard algorithm that can replan data routing to handle urgent, unplanned observations. The onboard algorithm updates the data routing plans already created by a centralized, ground-based algorithm. While previous work has investigated P&S systems with a similar hierarchical structure of both a centralized and an onboard algorithm, the responsibilites of the centralized algorithm do not incorporate the aforementioned four key characteristics. In this work, we demonstrate the execution of an onboard algorithm that operates in a closed loop with such a centralized algorithm. Additionally, we develop a simulation software environment that runs both algorithms in a relevant smallsat EO

constellation context.

We study constellation P&S performance over a larger set of operationally relevant metrics than those investigated so far in the literature, including total network throughput of bulk observation data, latency of observation data delivery to ground, freshness of observation data as well as satellite housekeeping telemetry and ground commands, and onboard resource margin (energy, data storage). Some of these metrics have been investigated separately, for example throughput in Zhou *et al.* [124] and latency in Wang *et al.* [115], but they are all presented in a single unified analysis over a more representative set of constellation simulation cases here.

1.3.2 Research Contributions

There are 4 primary contributions in this thesis:

1. Create the software infrastructure for centralized and onboard P&S algorithm execution and constellation simulation;

2. Develop P&S algorithms for a crosslinked, smallsat EO constellation and demonstrate improved scalability (up to 100 satellites) and commensurate metric performance (90% of optimal constellation data throughput) relative to state-of-the-art;

3. Analyze algorithm performance over representative crosslinked EO constellation geometries and satellite/ground station parameter sets;

4. Demonstrate algorithm handling of urgent ("injected") observation events and investigate their effect on nominal schedule quality.

1.3.3 Thesis Outline

The following chapters present the algorithms and software developed, and the simulations run to assess performance. Chapter 2 discusses the high-level approach for the algorithms, in addition to the simulation cases investigated, the satellite operations model utilized, and metrics for evaluation. Chapter 3 details the formulation

of the centralized Global Planner algorithm and validation results. Chapter 4 details the formulation of the onboard Local Planner algorithm and validation results. Chapter 5 details the constellation simulation framework developed for evaluation of the algorithms. Chapter 6 presents results from algorithm scalability analyses, and constellation simulation cases. Chapter 7 concludes with a summary of findings and suggestions for future work.

Chapter 2

Approach

This chapter details the high-level approach taken for this thesis work. First the architecture for the planning and scheduling system is discussed in section 2.1. The algorithms use a simple, yet representative smallsat operational model, as detailed in section 2.2. Metrics for constellation performance assessment are described in section 2.3. Background is given for the types of algorithms used in planning and scheduling in 2.4, along with details on the primary algorithm type used in this work. The planning and scheduling algorithms are simulated over a set of constellation designs for runtime and performance assessment. The software pipeline used for this simulation is described in section 2.5. The set of simulation cases are enumerated in section 2.6.

2.1 Planning Algorithm Architecture

The concept of operations for the planning and scheduling (P&S) system developed for this thesis work is shown in fig. 2-1. The system is divided into a two-level hierarchy. At level 1 is a global P&S element, the Global Planner (GP), which optimizes observation and communications scheduling across the constellation by reasoning about data production timing and priority as well as data routing through communications links between satellites and ground stations. At Level 2 is the Local Planner (LP) which runs onboard each satellite and reactively replans onboard activities based on

the satellite's latest knowledge of its state as well as any operational changes, such as spontaneously arising observations. Figure 2-1 illustrates the notional scenario where a satellite spots a newly formed forest fire and prioritizes the observation data collected of the fire for immediate downlink, with the lowest latency possible. The first black-outlined arrow illustrates initial schedule creation by the GP, and the second indicates schedule changes made by the LP. The "x" markers are observation targets, red arrows are crosslinks, and blue arrows are downlinks. In this situation it may take too long to inform the GP of the high-priority observation and receive updated plans, potentially tens of minutes or even hours when there is sparse constellation network connectivity, so the LP takes responsibility for replanning.

Two types of communications links are utilized in the constellation: downlinks from satellites to ground stations and crosslinks (or "inter-satellite links") between satellites. Both may be used for bulk (large-volume) observation data routing and exchange of Telemetry, Tracking, and Command (TT&C) updates.

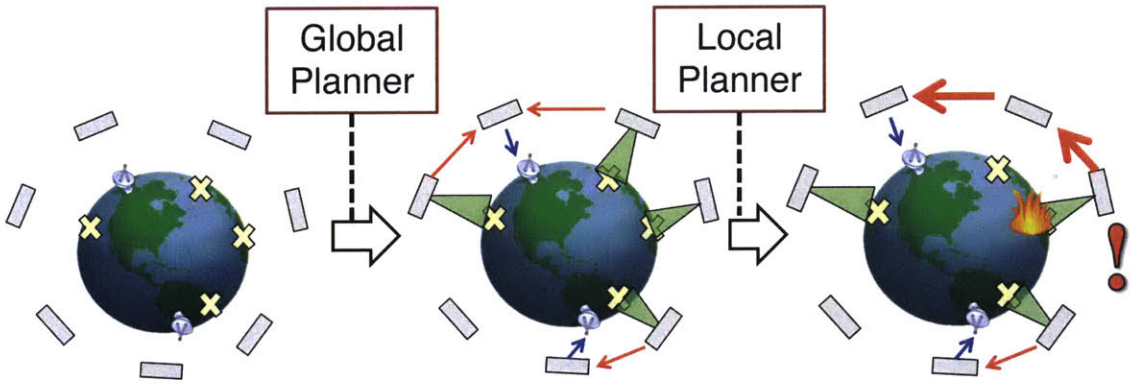


Figure 2-1: Concept of operations for the hierarchical planning and scheduling system developed in this work, including the ground-based Global Planner and the satellite-based Local Planner.

The terms planning and scheduling are often used interchangeably in this thesis. *Planning* generally refers to the process of choosing what activities or jobs to perform, while *Scheduling* generally focuses explicitly on timing selection for performing such activities. The exact division between the terms is not always clear. For the purposes of this work, the terms should be taken to have the same meaning: the selection of

activities to be performed by a satellite and the timing for those activities.

2.1.1 Role of the Global Planner, GP

The GP operates on the ground at a topologically central location to all of the ground stations that communicate with the satellites, effectively giving it a global view of the constellation's operations. This perspective allows the planner to select activity timings for individual satellites that best contribute to the constellation-wide goals of collecting useful observation data and ensuring that data reaches the ground intact and on time. The GP produces nominal activity schedules and data routing instructions that are then distributed to satellites for execution onboard.

This distribution can occur within the constellation itself, via ground station uplinks and crosslinks, or via an external satellite communications network. In general, communications via an external network can be expensive for bulk data routing, but very low latency (26 cents per byte, tens of seconds latency [112]). Distribution via the constellation may take tens of minutes given link availability at the time, and requires a crosslink transceiver if crosslinks are to be used.

The GP selects observation timing based on satellite observation "overpasses" or "accesses", the times when the satellites fly over the observation targets. It selects downlinks based on ground station and satellite overpass geometry. It plans usage of crosslinks between satellites based on satellite-to-satellite visibility, selecting the best ones to most effectively route bulk observation data and Telemetry, Tracking and Command (TT&C) updates through the constellation.

2.1.2 Role of the Local Planner, LP

The LP runs onboard the satellites, updating the plans provided by the GP to best determine observation data routing and activity execution timing in near real-time. The LP monitors new events that have occurred and onboard resources and makes any necessary changes to adhere to the GP's schedule as best as possible while preventing the spacecraft from violating operational constraints. The LP may also change the

selection of observation data to be routed during a given crosslink or downlink activity, to respond to events.

A significant driver for changing data routing selections is the presence of spontaneously arising, or "injected", observations. For this work, we do not detail how such injected observation data is identified, but it could be triggered by a data assessment software module running onboard the satellite in flight software. The LP may choose to route such injected observation data through bandwidth previously reserved for other data by the GP, and will inform other satellites of the updated routing plans.

2.1.3 Rationale For Hierarchical Structure

This hierarchical structure, featuring a centralized, ground-based planner with large decision-making responsibilities and an onboard planner with relatively less responsibility, was chosen due to the large, complex constellations on which it operates and the delay-tolerant characteristic of the constellation network. While smallsats are in general not quite as operationally complex as larger satellites, a significant amount of additional decision-making is added when satellite networking is taken into consideration. In potential future constellations of hundreds of smallsats, a single satellite may have many neighbors with which to choose to crosslink. Yet smallsats are still often very resource-constrained, featuring small batteries and solar panels that severely limit energy availability. This means that the satellites are often unable to maintain continuous operation of their observation payloads or communications subsystems. The inability to maintain continuous operations means that it is important to ensure that satellites collect data and communicate at the best times possible for achieving the overall objective of routing observation data to ground. The use of a centralized planner helps with ensuring that activity schedule timing is consistent across all satellites in the constellation.

Existing work has investigated a similar hierarchical structure [25, 35, 50, 27, 123], with varying degrees of interaction between centralized and onboard algorithms. The algorithms developed in this work feature relatively tight coupling between the two algorithm types, with both using the same "data route" objects to indicate scheduled

activities and data routing decisions to satellites (see the discussion of Data Routes in 3.1.1 for details). The current state-of-the-art in planning and scheduling for smallsat constellations emphasizes centralized planning [83, 124, 71, 34, 36]. Monmousseau details a centralized planner that schedules observations and downlinks for the Planet Inc. constellation [83], and Zhou *et al.* propose an algorithm that additionally includes crosslink scheduling and data routing [124]. Centralized planning can take advantage of the predictability of orbits to create satellite schedules hours to days in advance, and considers inter-satellite constraints all in a single solution process. A centralized planner can use powerful, ground-based computational resources, as opposed to the normally limited onboard computational capabilities.

Another possible architecture is a fully decentralized scheduling system, where satellites have complete autonomy over their activity timing choices. Onboard algorithms have been extensively investigated for data routing in constellations [76, 35], but they generally consider the routing of small packets (*e.g.* internet packets of tens of KB, versus hundreds of MB for bulk data) that do not impose significant scheduling concerns for satellites, and they often do not explicitly consider decision-making for when to execute communications links. Often in decentralized algorithms a mechanism is needed to achieve consensus across multiple agents, such as market-based task allocation schemes [18, 110]. This consensus process can require a large amount of communications overhead due to the many transactions needed between the satellites to settle on a final plan. Overall these considerations make fully decentralized scheduling a poor choice for the constellation bulk data routing problem in this thesis. But it is useful to maintain some level of autonomy in decision-making on board the satellite to handle changing priorities, for which reason the LP is present.

2.2 Satellite Model

The satellite model simulated in this work consists of three main components: 1) operations model, 2) satellite bus model and 3) payload and link models. The operations model specifies what activities the satellites and ground stations can execute

and any inter-activity constraints that must be enforced. The bus model includes the hardware models chosen for the satellite bus, and their onboard resource consumption. The payload and link models prescribe the data rates for observation data collection and communications, as well as constrain the access periods for performing observation and communications activities. The components are detailed in the following sections.

2.2.1 Operations Model

A simple smallsat operational model is used for scheduling of onboard activities. The model is intended to represent the main operational modes of interest for a smallsat in a Low Earth Orbit (LEO), Earth-observing constellation. Figure 2-2 depicts the activities that we model these smallsats performing, including observation of ground targets ("observation" activities or "obs"), transmitting bulk observation data to a ground station ("downlink" activities or "dlnk") and transmitting or receiving bulk observation data to or from a neighboring satellite ("crosslink" activities or "xlnk").

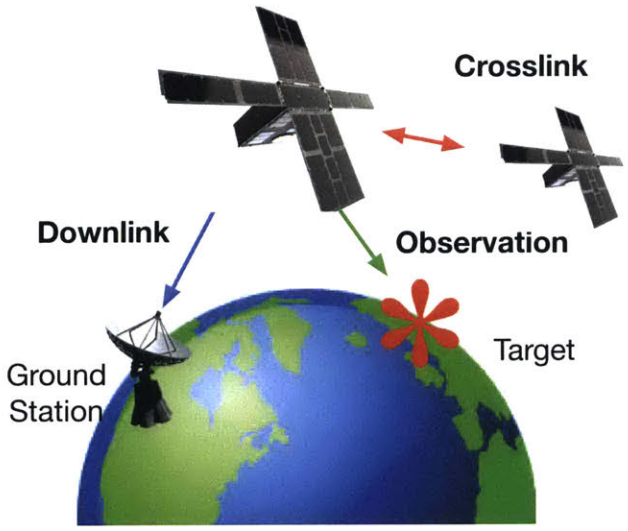


Figure 2-2: Concept of operations for the Earth-observing smallsats simulated in this work. The satellites observe targets on the ground, downlink with ground stations, and crosslink between each other.

Both downlink and crosslink activities are used primarily for routing bulk obser-

vation data, but we also assume that Telemetry, Tracking and Command (TT&C) data is shared during execution of these communications links. For crosslinks, the satellites share their latest knowledge of TT&C data for the entire constellation between each other in both directions. For downlinks, this sharing occurs between a satellite and a ground station. We assume that during a downlink activity an uplink is also available, allowing the ground station to send command data to the satellite, in addition to any other TT&C data the ground station has for the constellation.

We assume in this work that activity times are predictable with sufficient accuracy in advance, as is assumed elsewhere in the literature [36, 124, 71]. In general orbits can be determined well enough that satellite orbit position uncertainty has a negligible effect on onboard planning quality over the short planning horizons (approximately a few hours/orbits) used for the Global Planner (supported by uncertainties found in the literature [21, 94, 41]). The satellites are unable to modify their orbits, and potential collisions between satellites are not accounted for.

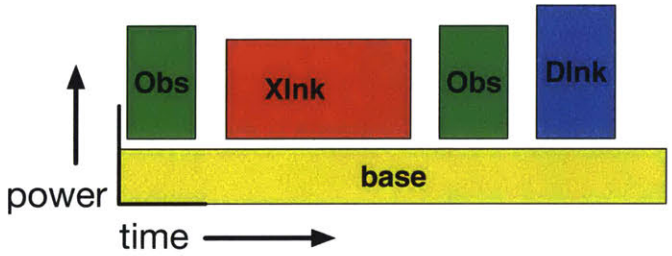


Figure 2-3: Notional activity timeline for a satellite, showing the execution of different onboard activities. The power usage for these activities is added onto a base power consumption.

We model each satellite as having a set of operational power states in which it can be, based on the activities it is performing. We assume that the satellite has some base, idle power consumption. On top of this different onboard devices can be turned on or off, adding a specified amount to the power consumption, as indicated in fig. 2-3. For this work, we assume that a satellite can only perform a single activity at a time. This assumption is based on the fact that smallsats are often very constrained in power, pointing, and thermal management capabilities, usually meaning that they

cannot operate multiple resource-heavy subsystems at a single time. The assumption lowers complexity in the GP and LP algorithms and constellation simulation software, by limiting decision-making to only a single activity at a given point in time. However, we note that allowing multiple activities to execute at a single time is not inconsistent with the algorithm formulations, and would be a useful inclusion in future work.

We assume the satellites produce energy at a constant rate whenever the satellite is not in eclipse, based on an orbit-average input power calculation. We assume that the solar panels are pointed at a fixed angle relative to the sun during this time. Note that this ignores any variation in energy production rate due to a satellite's changing attitude as it performs activities over the orbit. We assume this variation can be accounted for in the orbit-average power input value. We discuss possible improvements to this model as future work (see section 7.2).

Satellite attitude determination and control is handled at a high level in the operational model. We do not directly simulate the pointing of the satellite, but allow for transition time requirements between different activities. These transition times can simulate the need to point the satellite in a different direction, say when transitioning from crosslinking with a neighbor in the same orbit plane to a neighbor in another orbit. They can also be used to model any setup and takedown time required for given activities. The times are set conservatively, overestimating the amount of time a satellite might need between activities. This modeling choice allows us to avoid the complexity of pointing calculations within the P&S algorithms.

Minimum activity durations were enforced for the satellites, as shown in table 2.1. The minimum crosslink and downlink times were selected to ensure sufficient time for any packet retransmission that may be required (though such retransmissions are not directly modeled in this work). The minimum observation time was selected to allow a small variation of observation target geometry for data diversity and to avoid selection of unrealistically small durations for instrument operation (*e.g.* 1 or 2 seconds).

Table 2.1: Activity Minimum Durations

Activity	Minimum Duration (seconds)
Obs	15
Dlnk	60
Xlnk	60

2.2.2 Payload and Link Models

Payload Model

A high data rate, multispectral imager is modeled as the observation payload of interest. Payload parameters are set based on a multispectral imager design discussed by Tsitas and Kingston [108]. The imager features five spectral bands, in optical and near-infrared. This payload was chosen because it models the interesting application case of collecting earth imagery both for routine monitoring of the Earth's surface, and for disaster identification and follow-up. The Tsitas and Kingston design is based on requirements for the RapidEye constellation [109] for regular monitoring applications in agriculture and cartography, and the Disaster Monitoring Constellation [24] for disaster indentification including floods, storms, earthquakes, fires, and pollution. The imager produces data at a 127.5 Mbps compressed data rate, which provides a large amount of observation data to investigate the benefits of bulk data routing through crosslinks. Unless otherwise specified, we model the instrument with a narrow field-of-view, restricting accesses to those times when a satellite is above a 60° elevation angle mask as viewed by the ground target. Observation targets are individual latitude, longitude points on the earth's surface.

Link Models

Downlink and crosslink data rates are calculated based on simple link models for an X-band downlink radio and an optical crosslink transceiver. Parameters for the X-band downlink are selected based on the capabilities described for the Spaceflight

Industries Inc. (Spaceflight Networks, SFN) ground station network and the multiple X-band downlink options available for use with it [58, 31, 54]. We assume that a constant data rate of 80 Mbps (the lowest rate offered by SFN) is available above a 20° elevation mask as seen by the ground station. Note it is optimistic to assume that this data rate will be available for the whole dowlink pass. For the P&S work in this thesis it is not necessary to have high accuracy in this number; rather we just need a representative, high-capacity downlink data rate.

Crosslinks are modeled with an optical transceiver based on the MIT NODE and FLARE designs, with some improvements [67, 20, 85]. Crosslink rates ranged from 73 Mbps at 100 km to 3.2 Mbps at 5000 km. A lookup table of rates for each crosslink range is used, as detailed in appendix B, table B.1. Crosslink accesses are available whenever the line-of-sight vector between two satellites passes above the surface of the Earth. [1]

The data rate parameters for the payload and link models are summarized in table 2.2. A detailed model is not yet used for lower-level details of data handling, including data compression/decompression, error correction, packet drops/resending over links, and packet queuing time. While packet drops and retransmissions will in reality arise on any link, we assume that a model for this effect can be derived and used to lower the link data rate values.

2.2.3 Satellite Bus Model

The satellite bus is modeled as a generic 6U CubeSat [10]. This should allow sufficient space to fit the observation payload (about 3U [108]), optical transceiver (about 1U [85]), and downlink radio [31] and other electronics (1 to 2U). Note that this bus size does not necessarily provide space for terminals or gimbals to suppport concurrent links on multiple faces of the satellite.

Two onboard resources are modeled for the satellite bus, Energy Storage (ES) and

[1] Note that in future this should be adjusted to require crosslinks to pass over most of the Earth's atmosphere as well. In this work, crosslinks passing close to the earth's surface were only of concern for the 6-Sat simulation case, due to its assumption of a constant data crosslink data rate over all inter-satellite distances.

Table 2.2: Payload and Link Data Rate Parameters

Parameter	Design Reference	Value
Observation payload	Tsitas and Kingston [108]	127.5 Mbps above 60° target elevation mask
Downlink	SFN [58, 31, 54]	80 Mbps above 20° ground station elevation mask
Crosslink	MIT NODE, FLARE [67, 20, 85]	73 to 3.2 Mbps, up to 5000 km range, line-of-sight required. See appendix B, table B.1

Data Storage (DS). These correspond to the energy available either from solar panel input or the battery, and storage in mass memory on the satellite. These resources are affected by the performance of activities, and must be maintained within minimum and maximum bounds. A linear model is assumed for energy storage, based on constant energy consumption or production rates in every activity.

Resource usage parameters for the bus are detailed in table 2.3. The MiRaTA CubeSat was used as a reference for base energy consumption, with the value (12.8 W) set at two times MiRaTA's to reflect a larger 6U size here (a simple assumption to provide plenty of margin for 6U bus operations constraints not directly modeled here). The input power consumption for the X-band downlink was set to the largest value identified for the SFN radio options (BitBeam BBSDR, 12.3 W; note that to our knowledge this radio is not available for purchase, but its power consumption is representative). Commercial off-the-shelf parts from Clydespace and Delkin were assumed for energy storage/production and data storage. Efficiency factors were included for charging and discharging; these were multiplied by the power production/consumption values for the given activities.

Table 2.3: Satellite Bus Model Parameters

Parameter	Design Reference	Value
Power Usage	Base: MiRaTA CubeSat (x2) [64]	12.8 W
	Obs: Multispectral Imager [108]	10 W
	Dlnk: SFN [58, 31, 54]	12.3 W
	Xlnk, Tx: MIT NODE, FLARE [67, 20, 85]	12 W
	Xlnk, Rx: Zhou *et al.* [124]	5 W
Solar Charging	Clydespace 6U bus [77]	24 W orbit-average power
Energy Storage	Clydespace 6U bus [77]	40 Wh max, 60% discharge allowed (min 24 Wh)
Charge/ Discharge Efficiency	Tsitas and Kingston [108]	Charge: 0.7; Discharge: 0.9
Data Storage	Delkin industrial-grade SD card [29]	max 4 GB (32 Gb), min 0 GB

2.3 Metrics

Five principal metrics are used for assessing Constellation performance, as summarized in table 2.4.

2.3.1 Metric 1: Total Observation Data Throughput

The first metric is determined by summing up the data downlinked from executed observations over the course of the full simulation scenario, then dividing by the total potential downlinkable data volume. Potential data volume may be calculated in different ways, including: 1) an optimal scheduled value determined by the optimal version of the GP algorithm and 2) the sum of all observation window (overpass) throughput capacities, without scheduling. The second case represents an optimistic value, and constitutes a performance bound for the optimal value. Also, for context, occasionally throughput is presented as a raw value, not divided by the potential data

Table 2.4: Metrics for Constellation Performance Assessment

Metric Type	#	Metric	Goal Value
Observation Data	1	Total data throughput	$\geq$ 90% of potential
	2	Latency of executed observations	$\leq$ 10 minutes
	3	Average AoI of observation targets	$\leq$ 1 hour
Satellite Health	4	Average AoI of satellite command and telemetry TT&C data	$\leq$ 0.5 hour
	5	Average satellite energy and data margin	$\geq$ 50%

volume.

This metric assesses how well the constellation network performs at executing observation activities and routing the data through the network. In general, data throughput will be less than the sum of all observation window capacities due to schedule quality trade-offs made in the algorithms to reduce computation time. Previous work by Zhou *et al.* [124] in delay-tolerant networking for small satellite constellations has achieved about 90% of optimal throughput for a non-optimal P&S algorithm.

2.3.2 Metric 2: Observation Data Latency

This metric assesses how quickly data can be available on ground after collection. It is calculated as the difference in absolute times of the downlink of a given slice of observation data and the collection of that data during an observation window. For disaster monitoring purposes, we would like latency to be as small as possible, with "near instantaneous availability" [9]. For meteorological applications, latency requirements can be as small as 6 minutes for Global Numerical Weather Prediction (NWP) models [90]. A goal of 10 minutes for all executed observation activities was chosen.

Previous operational SmallSat constellations have generally only used downlinks,

limiting their ability to achieve low delivery latencies. The small satellite Disaster Monitoring Constellation was operated with a requirement of $\leq$ 90 minutes in the early 2000s [24]. The CubeSat constellation of Planet Inc. has a large 11-site ground station network [69] that can help lower latency by providing good downlink access diversity, but coverage gaps still can result in latencies of multiple hours.

We assume in this work that delivery latency is determined based on the downlink of an initial small amount of data from an observation window, rather than all of the data from the window, to reflect the fact that often a "snapshot" of the data is sufficient. This initial small amount is set to 100 Mb, unless otherwise indicated. This metric is referred to throughout this thesis as "obs(ervation) initial latency".

2.3.3 Metric 3: Average Age of Information (AoI) of Observation Targets

This metric measures the average freshness for data collected from a given observation target. AoI measures essentially the same thing as an average target revisit time, but eliminates some of the statistical problems introduced by using a simple average (*e.g.* clumping of revisits causes a simple average to be lower than expected [73]). AoI captures the freshness of data, that is, how long it has been since the data was updated. The metric originated in the queuing theory field [63, 3].

Average AoI is calculated as an average of the AoI values for all points within a measurement window of interest. In this work we calculate two versions of this metric, the average AoI at data collection ("at collection"), and the average AoI at data downlink ("with routing"). The latter incorporates the effects of routing through the network on data freshness, whereas the first does not.

A notional plot of AoI, with routing, for an observation target o is shown in fig. 2-4. The $t_{o,i}$ values are observation execution times and the $t_{d,i}$ values are the downlink times of the data collected in the observations. Notice that AoI is not reset to zero when a downlink occurs, but rather to the current age of the data that was downlinked.

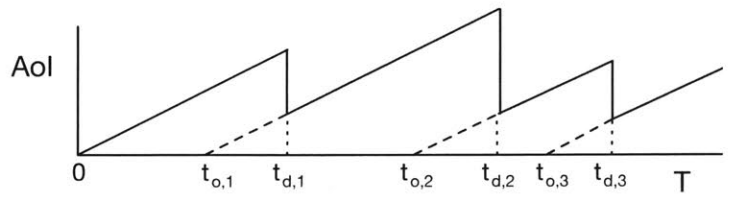


Figure 2-4: Notional plot of Age of Information (AoI) as function of time (T). Note the slope of the AoI curve is 1 at most points. Average AoI is a time average of a given AoI curve over a period interest (*e.g.* from 0 to T).

Average AoI for target o is calculated as:

$$\overline{AoI} = \sum_{i}^{M} [\frac{(t_{d,i} - t_{o,i-1})^2 - (t_{d,i-1} - t_{o,i-1})^2}{2}]/T \tag{2.1}$$

Where T is the metric measurement window and M is the number of downlinks of distinct observation data slices that occurred.

We want average AoI for any given observation target to be as low as possible. Global NWP applications can require sub-hour values [90]. For this work, we assume a goal of 1 hour, which covers many of the requirements for Global Numerical Weather Prediction applications. Note that this value is highly dependent on the constellation orbit design, which dictates when satellites pass over observation targets.

Note that AoI is a useful metric in addition to observation latency. If we only optimize for latency in P&S algorithms, we may end up with a case where only the observation data that can be *quickly* downlinked gets collected, and other observation opportunities with useful data get ignored. The AoI metric ensures that observation exeuction is evenly spread out; it adresses the issue of how often data should be collected, whereas latency adresses how quickly that data should be delivered to ground.

2.3.4 Metric 4: Average Age of Information (AoI) of Command and Telemetry Data

This metric is also calculated as an average AoI value, this time averaged for a single satellite instead of an observation target. The metric has two components, one for

command data and another for telemetry data (collectively, TT&C). Each measures the average freshness, for a given satellite, of the command data it receives from ground stations and telemetry data received by ground stations from the satellite. AoI for command data is updated whenever the satellite receives new data from any ground station, and AoI for telemetry data is updated when any ground station receives new data from the satellite. TT&C data can propagate through the network via crosslinks.

We want the average AoI for both to be as low as possible: for telemetry, to maximize operators' situational awareness of the constellation, and for commands, to minimize the amount of time needed to wait before a satellite can act on updated plans from the ground. The goal for both of these was chosen as 0.5 hours, half the average AoI goal for observation data average AoI. This ensures that on average, the ground is able to hear from and send updates to satellites as frequently as a given observation target is visited, which allows close to real-time decision-making about observation prioritization.

2.3.5 Metric 5: Average Satellite Energy and Data Margin

This metric is the average energy and data margin for a given satellite. The margin values are calculated as the difference between the current value for a resource and the most constraining limit on the resource (*i.e.* minimum energy storage, and maximum data storage) divided by the total range (max minus min) allowed for that resource. Note that in general, the minimum energy storage value is not set to full discharge of the battery, but rather to a desired depth-of-discharge level. The margin values are then averaged for a given satllite over all time points of interest. This essentially measures how close a satellite is to operating at its resource limits; a lot of margin means the satellite is not very constrained, and very little margin means that it is. The goal for these metrics was chosen at 50% or better so that the satellites are kept well away from resource limits on average.

2.4 Planning and Scheduling Algorithm Background

The GP and LP algorithms solve the high-level problem of planning and scheduling for the constellation, and replanning on an individual satellite, respectively. These high-level algorithms construct a problem model instance from a set of inputs and use lower-level optimization algorithms for solving the problem; that is, picking the activities and times to execute in order to achieve good metric performance.

The lower-level algorithms used are from the combinatorial optimization problem class [32], which involve selecting an optimal solution from a finite number of alternatives, specified as decision variables. The solution is optimal with respect to a given objective function. The problems often incorporate a large set of constraints to model the trade-offs between alternatives. The algorithms are broadly applicable to real-world problems in many domains, including routing, scheduling, production planning, decision making, transportation (air, rail, trucking, shipping), energy (electrical power, petroleum, natural gas), and telecommunications [32]. Difficulties arise, however, due to the large growth in computation time as problem instances grow large. Many combinatorial optimization problems are in the NP-complete (non-polynomial) class, with solve times growing exponentially with problem size.

Because of this computational complexity, many approaches have been developed for solving these problems over the years, including three broad classes of algorithms [32]:

1. **Exact Approaches**: search for the exact optimal solution to a problem, generally by dividing it into smaller subproblems and solving them. Algorithms include Branch & Bound and Dynamic Programming;

2. **Approximation Approaches**: search for a suboptimal solution within a bounded error of the optimal solution. Algorithms include greedy, sequential, local search, and random algorithms;

3. **Heuristic/Metaheuristic Approaches**: search for a sub-optimal solution without a known error bound, terminate when solution quality improvement

slows to an acceptable level. Algorithms include Simulated Annealing, Tabu Search, Evolutionary Algorithms, and GRASP (Greedy Randomized Adaptive Search Procedures).

Exact approaches are able to find the optimal solution, but may be intractably slow for large problem instances. Approximation approaches aim to solve a simpler problem with a known error bound relative to the optimal solution. A function specifying this error bound must be determined as well to accurately characterize the quality of their output. Heuristic methods have been developed extensively in recent years. By sacrificing a guarantee of optimality [70], they are often able to make large problem instances tractable.

Many of these algorithms have been applied to the multi-satellite scheduling problem. The Mixed-Integer Linear Programming (MILP) exact approach is often used [71, 124, 36, 83, 105, 12, 37] because the activity time scheduling and data route selection sub-problems often feature linear objective functions and constraints, or can be closely approximated by them. Other exact approaches can be applied, including Constraint Programming [37] and Dynamic Programming [25]. Applications of heuristic approaches to this problem include Genetic Algorithms [122, 123, 118, 62], Simulated Annealing [83, 119], and Tabu Search [119, 111, 120].

In this work, the MILP approach is extensively used due to its guarantee of optimality when run to completion. More details on the application of MILP to the GP and LP algorithms are provided in their respective chapters (3, 4), and background for this approach is provided in 2.4.1.

2.4.1 Mixed-Integer Linear Programming (MILP) Background

The underlying technique for MILP is Linear Programs, which are frequently used to model problems that are largely linear in nature, or can be cast as a linear model. They have been extensively studied and documented in the literature since their introduction by George Danzig in the 1940's [5, 26, 86, 96]. A basic Linear Program utilizes a linear objective function and a set of linear inequalities forming a convex

region. The canonical linear program is formulated as:

$$Maximize : z = \mathbf{c}^T\mathbf{x} \tag{2.2}$$

$$Subject\ to : \mathbf{Ax} \leq \mathbf{b} \tag{2.3}$$

$$\mathbf{x} \geq \mathbf{0} \tag{2.4}$$

where Equation 2.2 is the objective function. The vector $\mathbf{x}$ is a set of continuous-valued decision variables constrained by Equations 2.3 and 2.4, which define a convex region. Variable $\mathbf{c}$ is a set of costs for $\mathbf{x}$. Variable $\mathbf{x}$ can represent many types of decisions: for example, the start and end times for a set of activities or varying amounts of materials to choose from several stockpiles. The flexibility of this formulation is key to its widespread adoption in industry and academia.

A MILP problem is a variant on a linear program that constrains some of the decision variables to be integers. Integer variables are often useful for representing disjunctive conditions in the problem; for example, requiring certain activities to be performed before others, or specifying different resource stockpiles in different situations. In general, the solution of a MILP is much more computationally intensive than a Linear Program due to the discrete quality imposed by the integer variables.

One technique for solving MILP problems known as the Branch-and-Bound method [32, 57], involves searching through a tree of nodes defined by specific values taken on by the integer variables within the MILP. Linear Programs with relaxations on the integrality constraints are used to guide the search. Incumbent, feasible solutions (*i.e.* those that have an assignment for all the integer variables) and best bounding objective values are remembered as the algorithm searches through the tree, defining an "optimality gap". New feasible solutions found can be discarded if they are no better than an incumbent solution. The optimality gap can be used for determining how close a feasible solution is to optimality; when the gap reaches a small enough value, *i.e.* a solution that is close enough to the bound, the algorithm terminates and returns the solution. For our purposes, this solution is defined as "optimal".

Many techniques have been created over the years for speeding up the Branch and Bound algorithm. For the Gurobi commercial MILP solver [56], these include 1) Presolve, which tightens and simplifies the MILP problem in advance of solution, through size reductions such as elimination of redundant constraints, 2) Cutting Planes, which progressively cuts solutions out of the search space as more is learned about the optimum values of the decision variables, 3) Heuristics, a collection of techniques used to speed up search (say by "nudging" a non-integer solution to an integer one), and 4) Parallel processing [57]. Not all MILP solver software packages incorporate these techniques.

For this work, the commercial MILP solvers Gurobi [56] and CPLEX [51] were used for MILP solution due to their generally quick solution speeds [82].

2.5 CIRCINUS Simulation Pipeline

The performance of the GP and LP algorithms are assessed in simulation, for a desired constellation scenario. Several steps are required in the process of setting up the simulation scenario, simulating constellation operations, and understanding the output of the simulation. These steps are the basis of the CIRCINUS simulation pipeline, illustrated in fig. 2-5. The CIRCINUS acronym stands for "Constellation Investigation Repository with Communications, Inter-agent Networking, Uncertainty, and Scheduling", which describes the software's intended use in a wide array of constellation-related investigations. The pipeline was implemented mostly in Python, with certain routines in Mathworks' MATLAB language [2]. We intend to release this software open-source, refer to 7.1.1 for details.

The first module of the pipeline is **Orbit Propagation**. This module propagates the orbits of the satellites in the constellation, with parameters specified by an input file. The module consists of two parts. The first is a simple, low-fidelity orbit propagator to give the position of every satellite as a function of time through the

[2]The discussion here reflects commit ed989d954456128ab728d2c7004f30223c838641 of the top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

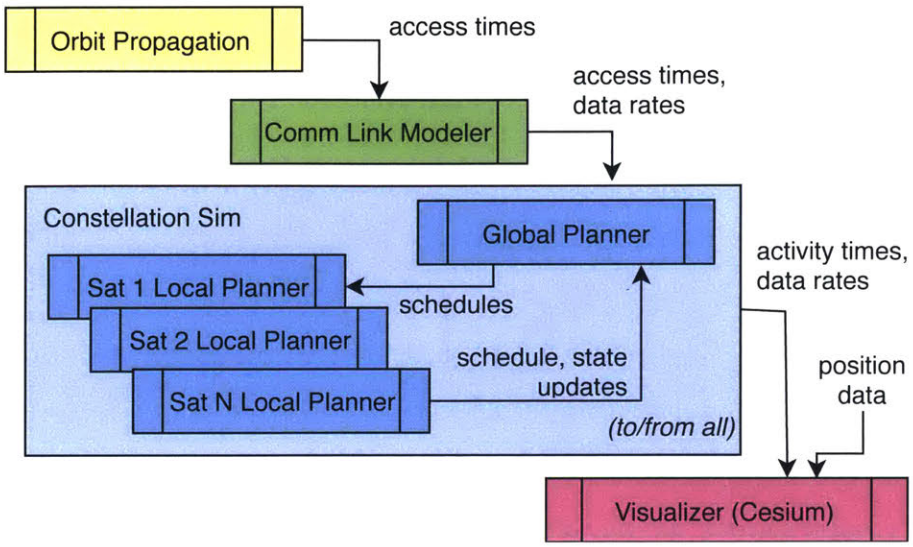


Figure 2-5: The CIRCINUS simulation software pipeline, with all of the code modules involved in constellation simulation.

simulation period of interest. We use an open source tool, PROPAT, that considers only the effects of Earth's gravitational field with the flattening of the poles [11], which is sufficiently accurate for the purposes of the high-level operations scheduling with short time horizons (approximately a few hours) performed for this work. The centralized Ground Planner algorithm is rerun at a regular interval, allowing it to use updated orbit ephemerides on subsequent runs (higher fidelity propagators could be integrated in the future) [3]. The second part of **Orbit Propagation** passes the satellite position data through a series of routines to determine access windows for ground target observations, downlinks to ground stations, and crosslinks to other satellites, as well as sun-satellite eclipse timing.

The second module, **Comm Link Modeler**, calculates data rates for downlinks and crosslinks based upon geometric data obtained from Orbit Propagation. These data rates can be determined either from a communications link model explicitly incorporating signal degration effects (*e.g.* signal degradation from passing through

[3]PROPAT orbit position solutions were found to diverge significantly from the J2 Perturbation propagator in AGI's STK (with a position displacement of about 1000km after 24 hours of simulation), but cumulative access time for a set of 7 ground stations over the same period with both PROPAT and J2 Perturbation agreed to within 30 seconds, or about 5% in most cases (because all satellites are propagated in the same manner)

Earth's atmosphere) or from an input lookup table which matches data rate with link distance (for simpler analyses). Note that these data rates are calculated in advance before simulating constellation operations. Currently no capability exists for reactively changing data rate within the GP and LP algorithms. Such a capability may be incorporated in the future, but for this thesis the maximum achievable data rate is used at any given time.

The third module, **Constellation Simulator**, simulates the constellation. It runs the GP, passes the schedule outputs from it to the LP instances for every satellite, and passes the LP outputs back to the GP if needed. Satellites, ground stations, and an overall ground station network are simulated as individual agents (simulation entities which may perform activities and run planning algorithms). This simulator module enables constellation performance to be assessed with representative simulation agents, as opposed to relying on the outputs of the GP and LP alone as final performance results. This module is discussed in detail in chapter 5.

The final module, **Visualizer**, displays output from the GP and LP algorithms and the constellation simulation in a 3D visualization. It uses a custom Python library that interfaces directly with AGI's Cesium, the open source Earth and space visualization engine [22]. An example of the visualizer running is shown in fig. 2-6. The visualizer was an invaluable tool for validation of results, because it enables the user to directly see the activities performed by satellites as opposed to attempting to read this information from data structures within Matlab or Python code. Note that the Cesium tool handles rendering and defines an API for providing input data, but all orbit position calculations as well as custom visualization elements (*e.g.* downlink/crosslinks lines and timing, data storage bars) were added as part of this work.

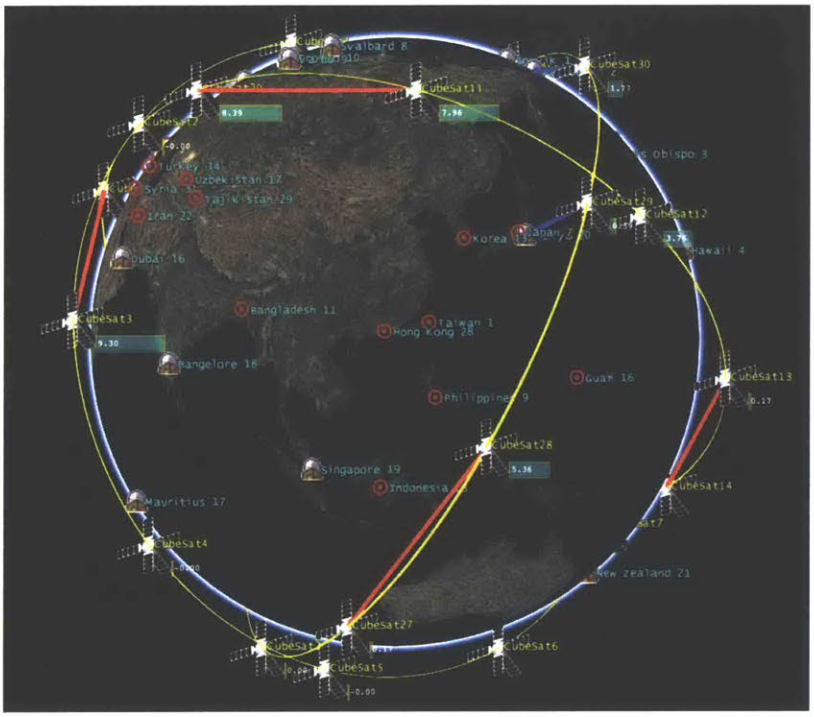


Figure 2-6: Image from the Cesium-based visualization front-end. Smallsats are shown along their orbits (yellow lines) with changing stored data volume (cyan horizontal bar), downlinking to ground stations (blue lines), and crosslinking with each other (red lines).

2.6 Simulation Cases

Three simulation cases were run for assessing planning and scheduling performance. Each case includes a constellation orbit geometry, a set of ground stations, and a set of observation targets. The main details of the simulation cases are summarized in table 2.5, with additional parameters detailed in appendix A.1 and appendix A.2. The constellations are illustrated in fig. 2-8 and fig. 2-8. The 6-Sat and SSO Ring cases each utilize Sun-Synchronous Orbits (SSO), which are useful for achieving good Earth-surface coverage with a small constellation and good lighting conditions for observations.

Table 2.5: Summary of Simulation Cases

Case Name	**Orbit Design**	**Observation Targets**	**Ground Stations**
6-Sat	6 satellites, 2 SSO planes (3 sats per plane)	5 targets	4 stations
SSO Ring	10 satellites, 1 SSO plane	40 targets, tropical land area	3 stations, polar
Walker	30 sats, 3 plane Walker Delta pattern (30° inclination)	40 targets, tropical land area	9 stations, well-distributed

The first case, **6-Sat**, reflects a simulation case designed and evaluated in the literature by Zhou *et al.* [124]. This case is used for assessing algorithm run time, and comparing performance to the state-of-the-art in the literature in small satellite constellation planning and scheduling (*i.e.* The algorithms in Zhou *et al.*). This case is additionally used for validating algorithm execution in other contexts, due to its relative small size and quick execution times compared to the other sim cases (6 satellites versus 10 for SSO Ring and 30 for Walker).

The second case, **SSO Ring**, reflects a small constellation that might be feasible for technology demonstrations or operational applications in the near future. In principle, the 10 satellites in the constellation could be deployed from a single launch

vehicle and distributed over the orbit using differential drag or small thrusters [39, 38, 125]. The three ground stations for the constellation are located near the Earth's poles, at sites that are already in use or are planned for use [58]. These three polar ground stations were chosen to be representative of the typical downlink availability a constellation operator might have. The observation targets are spread out over the dry landmass within the Earth's tropics, representing an area that is interesting for meteorological science applications [6].

The third case, **Walker**, represents an ideal constellation geometry and network for observation of the tropical target set. Thirty satellites are arrayed in a 30°:30/3/1 Walker Delta configuration, with the Walker Delta formation chosen to provide geometric diversity of observation and downlink overpasses for the satellites within the constellation [113]. This constellation also provides good inter-orbit crosslink connectivity, increasing overall network connectivity. It is expected that this constellation will perform better in terms of latency for routing observation data. The ground stations for this case are selected to provide increased access for downlinks. They are a subset of the ground station network planned by a commercial operator, Spaceflight Industries Inc. [58].

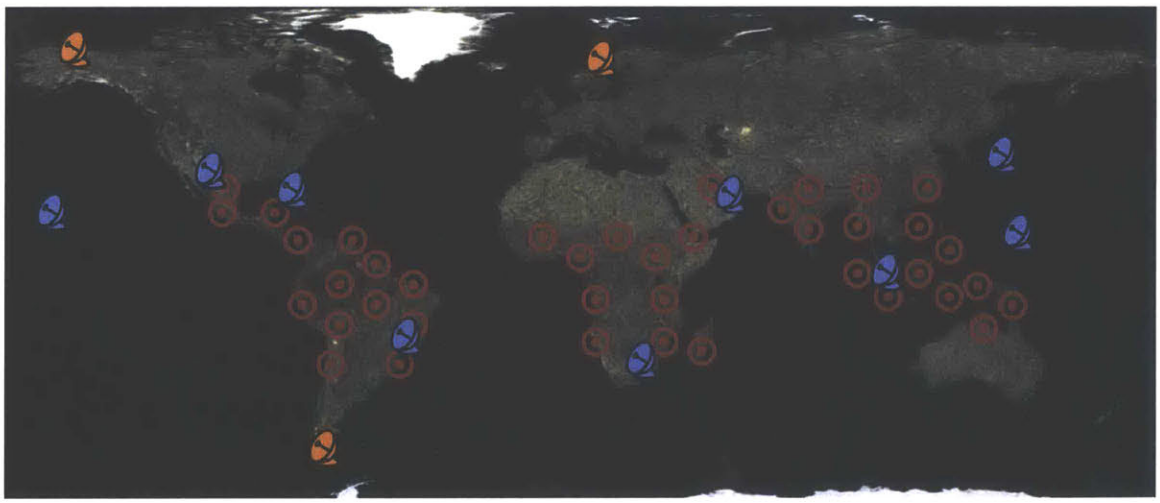
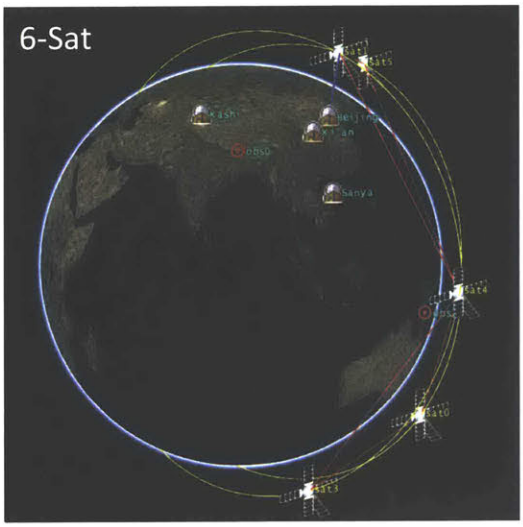
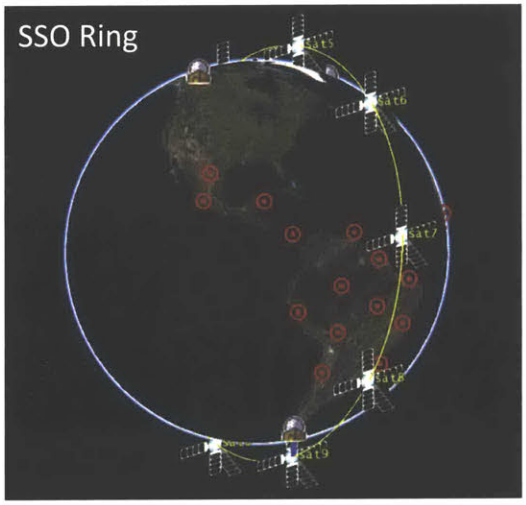


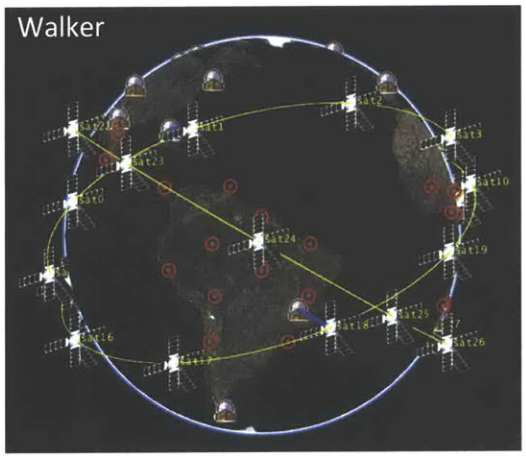
Figure 2-7: Targets and ground stations for SSO Ring and Walker. SSO Ring ground stations (3x) are in orange, Walker ground stations (9x) are in blue. There are 40 observation targets.



(a) "6-Sat" geometry



(b) "SSO Ring" geometry



(c) "Walker" geometry

Figure 2-8: Constellation orbit geometries analyzed in simulation cases

Chapter 3

Global Planner

The purpose of this chapter is to explain the formulation for the Global Planner (GP) algorithm. First the details of the problem model for the GP are introduced in section 3.1.1. Then, two versions of the algorithm are presented, an optimal version, GP-Optimal, in section 3.2 and a fast version, GP-Fast, in section 3.3). Basic algorithm validation results are presented in the respective sections. Limitations of the GP algorithms are discussed briefly in section 3.4. End-to-end validation results for the GP relative to the state-of-the-art are presented in section 3.5. Finally, sensitivity results for varying GP objective weightings are presented in section 3.6.

3.1 Global Planner Overview and Problem Model

3.1.1 Overview

The GP has the responsibility to determine activity execution schedules for the satellites and ground stations. It is assumed to be running on the ground, in a centralized location from the perspective of the constellation's network topology. A high level view of the GP architecture is shown in fig. 3-1. The two versions of the GP are shown here. GP-Optimal is formulated to provide an optimal scheduling solution for the problem model assumed in this thesis work. GP-Fast is intended to solve the same problem much more quickly, at a minor cost of schedule quality relative

to the optimal solution. It comprises two stages that separate the problem out into the creation of data routes through the constellation, **Route Selection** (RS), and the de-confliction and scheduling of those routes, **Activity Scheduling** (AS). The creation of data routes is in turn divided into two steps, **Route Construction** (RS1) and **Route Downselection** (RS2).

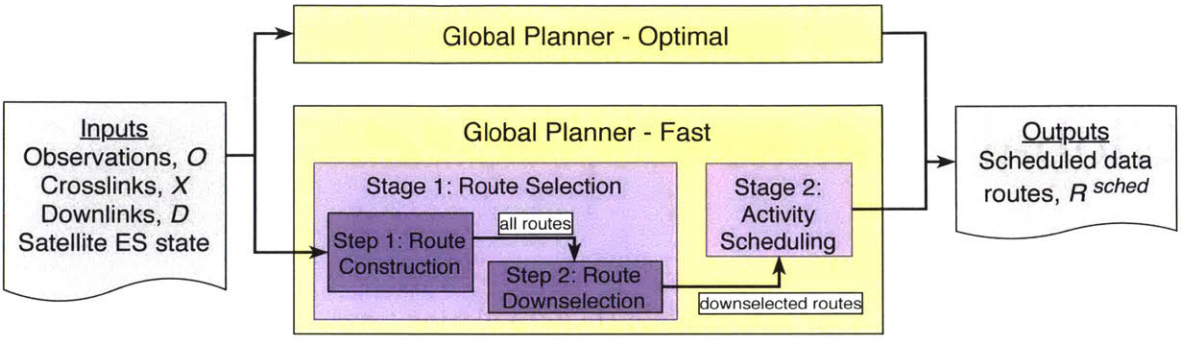


Figure 3-1: Overview of the GP-Optimal and GP-Fast algorithms.

In the rest of the section, details of the problem model for the GP are discussed, including the underlying solver technology used (section 3.1.2), input activity windows used in scheduling (section 3.1.3) and the output data route objects produced by the GP (section 3.1.4). Further details relating to the planning window, satellite resources, and notation are explained in the remaining subsections.

3.1.2 GP Solver Algorithm Selection

The high-level GP algorithms approach encompasses a lower level solver algorithm to perform the mechanics of the scheduling process. The following two paragraphs give context for these mechanics.

The GP algorithms use the same basic problem model as the algorithms presented by Zhou *et al.* [124]. The GP incorporates the same constraints, including limited throughput through link activities, data flow balance in and out of satellites, limited onboard energy and data storage, and link interference between satellites and ground stations (*e.g.* only two satellites can crosslink at a time). As discussed in section 1.2.1, the Zhou *et al.* algorithms are unique in incorporating both data and energy storage

constraints while jointly scheduling observations and data routing, as opposed to other approaches that do not consider both constraints [71, 36]. Because of the broad similarity in problem model, the Zhou *et al.* algorithms are the primary point of comparison for validation of GP results, as detailed in section 3.5. The Zhou *et al.* algorithms do not consider observation data latency, so we focus on comparison of total throughput. All of these smallsat constellation models are linear in both constraints and objective function, with disjunctive constraints for certain decisions [124, 71, 36].

The GP algorithms are broadly similar to the scheduling algorithm used for the largest operational smallsat constellation to date, the one deployed by Planet Inc. [83, 95, 69, 28], in that they both account for observation scheduling, limited on-board energy, limited throughput capacity on downlinks, and downlink de-confliction across satellites. Crosslinks are not incorporated in the Planet scheduling system. Additional scheduling constraints are present within the Planet model, such as differentiation between low-speed and high-speed downlinks and higher-fidelity battery modeling (including constraints to restrain satellites with weak batteries). Due to the preponderance of more general-purpose smallsat models in the literature that do not incorporate such implementation-specific constraints [124, 71, 36, 76, 105, 12, 115], they are not incorporated in the GP model. However because of the similarity in principle operational constraints, the GP is considered to be representative of the state-of-the-art in operational small satellite constellations when run with only downlinks.

For schedule solution, a Mixed-Integer Linear Programming (MILP) model and solver were used in both the GP-Optimal and GP-Fast algorithm variants, fitting the linear nature of the problem model and the need for disjunctive constraints (see 2.4.1 for more detail on MILP). GP-Optimal is intended to provide a benchmark for optimal performance in the model in this work, hence an exact combinatorial optimization algorithm is useful because it guarantees the optimal solution will be found if given sufficient computation time.

GP-Fast incorporates both a heuristic algorithm in the Route Selection stage and a

MILP-based exact algorithm in the Activity Scheduling stage, to reduce computation time by lowering the size of the problem solved in AS. Other approaches do not separate out the planning and scheduling process into multiple stages, but rather apply a heuristic algorithm to a full scheduling problem in a single stage. Approaches used include simulated annealing [83], Genetic Algorithms [122], and Tabu Search [119, 111]. For GP-Fast, the use of another heuristic algorithm in the AS could speed up the solution process further. However, Route Selection already introduces sub-optimality to the scheduling solution by choosing a small number of data routes, so it is desirable to achieve an exact solution in AS to avoid sacrificing more schedule quality. In practice, it was found that the MILP solution process in AS was not a significant bottleneck for GP-Fast in most of the problem instances tested (see section 6.1.2 for details).

Note that the GP objective function avoids inclusion of nonlinear objectives (*e.g.* the average data delivery latency metric), by using linear terms as a proxy for such metrics (*e.g.* latency reward factor terms, inequalities 3.25 and 3.53). These proxies are formulated to reward schedules that provide good performance for non-linear metrics.

3.1.3 Activity Windows

For the GP problem model, there are access "periods" or "windows" during which a satellite can perform an activity, *i.e.* an observation, a crosslink, or a downlink. The periods when these activities can be performed are based on orbital geometry, and dictated by when a satellite passes over an observation target, has access for a crosslink to another satellite, or passes over a ground station for a downlink.

Activity access windows are generally broken down into smaller periods of time, or "slices", that represent a single choice for the satellite of whether to perform an activity or not. For example, if satellite m has a neighbor satellite n in the same orbit plane that it can continually crosslink with due to favorable geometry, that single continuous access is broken down into a series of several-minute-long activity slices that each represent a distinct time period during which a unique decision can

be made for m and n to crosslink. This granularization serves to provide a diverse set of opportunities for routing data. Note that the duration of these slices need not be fully utilized; the scheduling process determines how much of a slice to use. The terms "activity window", "activity", and "window" are all used to refer to these granularized activity slices. There are several important activity window sets input to the GP:

1. Observations, "obs": O
2. Downlinks, "dlnks": D
3. Crosslinks, "xlnks": X
4. All observations, downlinks, and crosslinks: $A \mid A = O \cup D \cup X$

In the GP scheduling output, an activity is assigned a scheduling utilization fraction, $0 \leq x_a \leq 1.0$, which represents the fractional utilization of potential data volume throughput (the "capacity") for an activity window slice; it essentially represents "how much" of an activity gets scheduled. The throughput for an activity is determined by multiplying the data rate at every point during the activity by a small Δt. This rate is either for data transmission or data reception, as appropriate. For observations and downlinks, it is always receive and transmit, respectively. For crosslinks, it could be either transmit or receive. A single satellite is only allowed to perform one activity at a given time, hence the need to de-conflict the timing of activities. This is performed in a pair-wise fashion across all activity windows by choosing the scheduled start and end time of activity m to avoid conflicts with any other activity n. For any activity a, we have:

$$t'_{a,s}, t'_{a,e} = t_{a,c} \mp \frac{c_a x_a}{2\bar{r}_a} \tag{3.1}$$

Where $t_{a,c}$ is the center time of the activity (calculated as the average of the orginal start and end of an activity), $t'_{a,s}$ and $t'_{a,e}$ are the scheduled start and end of the activity, c_a is the throughput of the activity at full utilization, and $\bar{r}_a$ is the

average data rate of the activity. The original start and end times are the times of the original activity window slice before scheduling. One can think of the time scheduling decision for an activity as a dilation around the activity's center time: we symmetrically choose a scheduled start and end time around the center time based on the utilization. This precludes us from choosing to schedule, say, only the first minute of a two-minute-long activity, so a sufficiently fine granularity should be used for the activity lengths. A fine granularity (*e.g.* under a minute) helps to avoid situations where inter-activity temporal overlap conflicts cause many activities not to be schedulable, but also significantly increases computational complexity by adding to the number of activity execution decisions.

We assume a direct correlation between data volume usage and time usage for a given activity; *e.g.* If 50% of the potential duration of an observation window is used, then we collect 50% of the potential data volume that could be collected. In general, the transmit and receive data rates seen during a link window are not constant and could vary from 0 to a much larger maximum over the course of a downlink window from horizon to horizon. The use of the average data rate $\bar{r}_a$ for the activity takes this into account. Activity windows are created such that the maximum data rate for a given window will be in the middle, around the center time. The use of the average data rate to directly connect time and data volume utilization is in general a conservative assumption, because the majority of data volume is transferred in the middle of the window.

In the global planner algorithm formulations, activities are de-conflicted with each other by constraining the scheduled start and end times of a given activity based on all other scheduled activities that it temporally overlaps with. A key point is that this is only performed for activity pairs where both activities are scheduled. If an activity is not selected to be performed, then it exerts no timing constraints on other activities. Scheduled activities are also constrained to meet a minimum time duration.

3.1.4 Data Routes

The Data Route (DR) is the fundamental planning object used within the GP, Local Planner (LP), and Constellation Simulator algorithms and software. It represents a potential or executed path taken by data from an observation o to a downlink d. A DR is constructed from a time-ordered list of all the activity windows used to route the data, in the form $[o, x_0, x_1, ..., x_n, d]$ where each x_k represents a crosslink used in the route. There may be no crosslinks present if the collecting satellite delivers the data directly to one of its own downlink activities. By definition there is only one observation window in the route at the beginning and one downlink window in the route at the end. The basic structure of a DR is shown in fig. 3-2. In this case there are three crosslinks, between a set of example satellites.

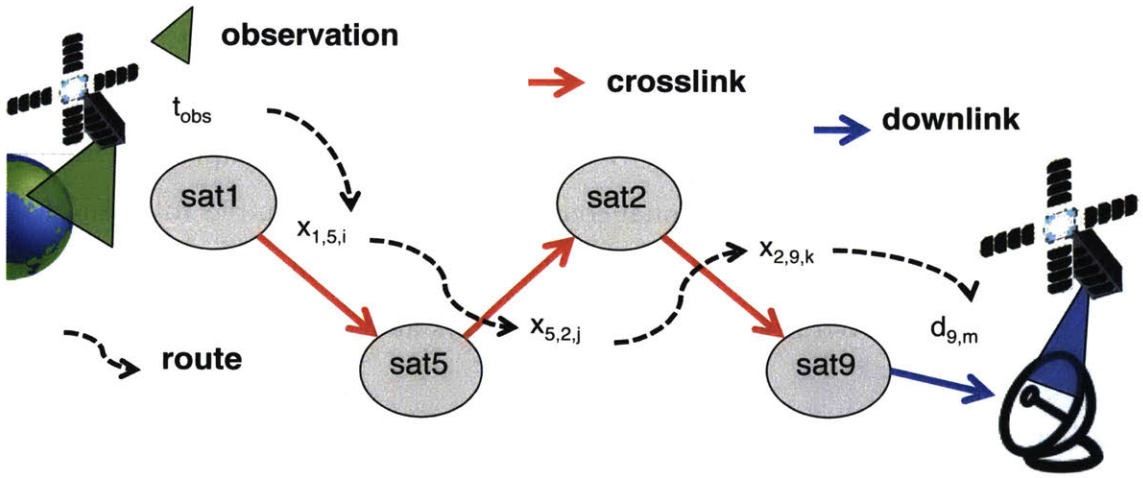


Figure 3-2: Depiction of the basic structure of a Data Route (DR). A DR consists of an observation, zero or more crosslinks, and a final downlink.

Data Route Definitions and Basic Constraints

A "scheduled" route is one that has been scheduled by the GP or LP but has not yet been executed: the current simulation time has not yet reached the start of the first window within the route. An "executed" route is one for which all of the windows have been fully executed: the current simulation time has passed the end of the last window the route. A "partially executed" route is in between: some of the windows

have been executed. A "constructed" route is one that has not yet been scheduled and may still be in conflict with other routes in both activity window timing and capacity assignment.

DRs are required to enforce temporal consistency along their length: each window must precede its successor window in absolute time, with a sufficient time duration allowed between successive windows to meet any transition time requirements. For scheduled and executed routes, this implies that the end time for every activity window in the route, $t_{a,e}$ must precede the start time $t_{a,s}$ for the following window in the route. Note that this particular requirement is a modeling choice to ensure the integrity of bookkeeping for data volume as it moves through the constellation. We know that for any scheduled DR, the data volume to be transmitted in a given activity window will arrive at the appropriate satellite at the needed time as long as all activity windows before it are executed successfully. We do not need to impose any further constraints, within the context of a single data route, to ensure that data volume is accounted for appropriately. We could model DRs with temporal overlap allowed between activities, but additional constraints would be required to ensure that when executing a given activity, all of the data volume that it expects to transmit has already been received from the previous windows along the route.

Data Route Capacity

DRs have a specified data volume capacity, and a scheduled data volume usage. Before being scheduled, a DR's capacity may be as large as the minimum activity capacity along the route. For example, if a route starts at an observation window that could potentially provide 100 Mb, progresses through a crosslink that could route 10 Mb, and finishes at a downlink that could deliver 50 Mb to ground, the DR's capacity may be up to 10 Mb. Route capacity is signified by c_r. After scheduling, a smaller data volume may be scheduled than the actual route capacity. The algorithms in this thesis represent this as a utilization factor x_r multiplied by the capacity of the route, giving a scheduled data volume of $v_r = c_r x_r$ for the route. A DR inherits the linear relationship between data volume and duration present within the activities within

the route. If x_r fraction of the route is utilized, then that same x_r determines how much data volume the route uses from every activity, and in turn exerts a minimum bound on the amount of time that each activity must be executed. If the utilization for every window along the route is x_a, then the relation $c_a x_a \geq c_r x_r$ must hold. The inequality is required because there can be multiple routes passing through a given window.

A data route must meet a minimum data volume requirement, $MinDV$, in order to "execute" an observation successfully and to count towards the latency reward term in the objective function for the GP algorithms. This number is the same as the amount of data required to deliver a "snapshot" of the observation, discussed in section 2.3.2.

There is occasionally a need to split a route, allowing some data volume to have parallel paths through activities. In the most extreme case a route could consist of an observation window, two parallel sub-routes of crosslinks with no shared windows, and a final shared downlink window (the observation and downlink must be the same per the DR definition). Such a route is referred to as a Data Multi Route (DMR). A similar relation holds for utilization of a DMR, in that we can again constrain the utilization for every activity within the route to be greater than or equal to the utilized data volume from the route. This is possible because we enforce that there is no overlapping throughput present in the DMR; even if multiple of its parallel paths pass through a single activity, they each must claim mutually exclusive partitions of throughput from the activity. This allows us to treat both DRs and DMRs as a single object abstraction.

3.1.5 Planning Window

The GP algorithms are run over a finite planning window considering all activities from an initial time to a time horizon (t_h), the "planning window", for scheduling. Three different horizons can be set for the GP, one for observations, one for crosslinks, and one for downlinks. This allows searching for downlinks far in the future while removing the decision-making complexity introduced by considering additional obser-

vations and crosslinks up to that downlink horizon.

Due to the computational complexity of the GP algorithms, in general it is infeasible to extend the time horizon arbitrarily far into the future. The GP is rerun at a later point in time to extend plans further, usually at a replan time sooner than the planning horizon so that the constellation is never left without plans. This planning approach is referred to as "receding horizon" planning [80].

3.1.6 Satellite Resources

As discussed in section 2.2.3, two onboard satellite resources are explicitly considered in the GP algorithms: 1. Energy storage (ES) and 2. Data storage (DS). Note data storage is the current amount of bulk observation data stored on the satellite. We do not consider the storage cost of Telemetry, Tracking, and Command (TT&C) data, because this is in general orders of magnitude lower in volume than observation data.

3.1.7 Notation

In the equations defined within this text, a contiguous sequence of "for all" ($\forall$) expressions is meant to imply the order in which equations are generated. That is, the sequence $\forall(o \in O, s \in S), \forall t \in T_o$ means "for all (o,s) tuples in the Cartesian Product of O and S, and subsequently for every o,s tuple a new tuple o,s,t for every t within T_o (where the subscript indexes by o). The Cartesian Product of two sets O and S, $O \times S$, is the sets of tuples (or pairs) for every element of O matched with every element of S). This order is important because often the inner "for all" statements are only valid in the context of an outer "for all" (*e.g.* T_o).

3.2 GP-Optimal Algorithm

The GP-Optimal algorithm [1] is formulated as a Mixed-Integer Linear Program (MILP), reflecting the linear nature of the underlying problem model: all data volumes, activity utilizations, and satellite resource states are modeled with linear relationships. This algorithm serves as a benchmark for the GP-Fast algorithm. Because it is solved entirely as a single MILP, and because its formulation was not designed to limit computation time, it does not have the same scalability as the GP-Fast algorithm. Nonetheless, for small constellations it is useful for determining what level of performance is desirable from GP-Fast.

The GP-Optimal algorithm returns optimal schedules, to within the optimality gap tolerance for the MILP solver (generally set at 1% or 0.1%). These schedules are optimal for the particular problem model used here, with the following important conditions:

1. Bulk data is routed through pre-computed activity windows with capacities c_a and absolute start, center, and end times $t_{a,s}, t_{a,c}, t_{a,e}$;

2. Scheduled activities must remain centered at their original absolute center times;

3. Data becomes available to transmit in an activity n only after the end of the activity m that received the data;

4. For any two activities $m, n \mid t_{m,c} \leq t_{n,c}$, required transition time $\tau_{m,n}^{transition}$ must be allowed;

5. Activities consume energy at fixed rates, and a linear model is used for satellite energy storage;

6. Energy usage for an activity may be spread out over the activity's original start/end time (as opposed to the final scheduled times);

[1] The discussion here reflects commit 476f089e4fb882f44d423837725d028103c1ddac of the CIRCINUS Global Planner repository at `https://github.mit.edu/star-lab/circinus_global_planner`

7. Activity capacities are fixed and not subject to change due to environmental uncertainty (*e.g.* changes in downlink data rate due to atmospheric effects)

Given that these conditions hold, the GP-Optimal algorithm returns an optimal schedule solution (within the allowed optimality gap).

3.2.1 Formulation

Decision Variables

The decision variables used in the optimal formulation are shown in definitions 3.2 to 3.8. v_a is the data volume used for activity a. I_a is an indicator variable which is set high (to 1) if a is scheduled for execution. $v_{l,o}$ is the data volume to be routed on link l (crosslink or downlink) for observation o. These variables are generated for links across all satellites which occur after o (the set O_l). $v_{s,o,t}$ is the data volume present for o on satellite s at time t. These variables are only generated for time points which occur after o (set T_o). $I_{o,d}$ is set high to indicate that downlink d completes the minimum data volume routing requirement for o, for all downlinks across all satellites that follow o (the set D_o). f_o is the latency reward factor for o. $e_{s,t}$ is the energy storage for s at t. T^{rsrc} is the set of times at which we choose to enforce resource constraints.

$$v_a \geq 0 \quad \forall a \in A \tag{3.2}$$

$$I_a \in \{0,1\} \quad \forall a \in A \tag{3.3}$$

$$v_{l,o} \geq 0 \quad \forall l \in L, \forall o \in O_l \tag{3.4}$$

$$v_{s,o,t} \geq 0 \quad \forall o \in O, \forall s \in S, \forall t \in T_o \tag{3.5}$$

$$I_{o,d} \in \{0,1\} \quad \forall o \in O, \forall d \in D_o \tag{3.6}$$

$$0 \leq f_o \leq 1 \quad \forall o \in O \tag{3.7}$$

$$e_{s,t} \geq 0 \quad \forall s \in S, \forall t \in T^{rsrc} \tag{3.8}$$

Constraints

The first set of constraints, inequalities 3.9 to 3.16, enforces the general shape of the formulation. They ensure that data volume from observations is accounted for appropriately as it is routed across activity windows within the planning window. The main points they enforce are:

1. Data volume transmitted or received in an activity is less than or equal to that activity's capacity;
2. Data volume transmitted in an activity is less than or equal to the data volume available on the satellite at the time of that activity;
3. All observation data volume that is sent through a downlink is propagated through crosslinks, in correct temporal order, to arrive at that downlink (or the satellite performing the downlink collected the observation data volume itself);
4. If any data volume from an activity is used, then the indicator variable for that activity is set high.

$$v_a \leq c_a \quad \forall a \in A \tag{3.9}$$

$$I_a \geq \frac{v_a}{c_a} \quad \forall a \in A \tag{3.10}$$

$$v_l \geq \sum_{o \in O_l} (v_{l,o}) \quad \forall l \in L \tag{3.11}$$

$$v_{l,o} \leq v_o \quad \forall l \in L, \forall o \in O_l \tag{3.12}$$

$$v_o = \sum_{d \in D_o} (v_{d,o}) \quad \forall o \in O \tag{3.13}$$

$$v_{s,o,t} = \sum_{\tau \in T | \tau < t} \left(\sum_{l \in L_{o,\tau}} (\mu_l^s \cdot v_{l,o}) \right) + v_o^s \quad \forall o \in O, \forall s \in S, \forall t \in T_o \tag{3.14}$$

$$\sum_{l \in L_{o,t}^-} (v_{l,o}) \leq v_{s,o,t} \quad \forall o \in O, \forall t \in T_o \tag{3.15}$$

$$\sum_{\tau \in T | \tau \leq t_d} \left(\sum_{l \in D_{o,\tau}} (v_{l,o}) \right) \geq MinDV \cdot I_{o,d} \quad \forall o \in O, \forall d \in D_o \tag{3.16}$$

Inequality 3.9 enforces that the data volume used from a given activity is less than the capacity for that activity (this includes observations). Ineq. 3.10 forces an activity indicator to be set high if any data volume is used. Ineq. 3.11 restricts the total observation data volume used for a link activity, for all potential observations O_l, by the data volume utilization of the activity. Ineq. 3.12 restricts the amount of data volume sent on any one link for observation o. Ineq. 3.13 enforces that the data volume used for a given observation is equal to the sum of the data volume allocated for it to every potential downlink.

Ineq. 3.14 enforces that the data volume for observation o on satellite s at time t is equal to the sum of incoming data volume from crosslinks and observation o minus outgoing data volume from crosslinks and downlinks. Note that o only provides incoming data volume on the satellite that collects o (hence the s superscript). $L_{o,\tau}$ is the set of both transmit and receive links available at time τ. The μ_l^s term is $+1$ if link l is receiving for satellite s, and -1 if transmitting. The inner summation sums over all links occurring at time τ, where τ is accumulated over the entire time span from the end of the observation to time of interest t. This is the constraint that is the most significant driver of complexity in this formulation, as it encompasses all satellites, all observations, and all time points. It ensures that every observation has a chance to be routed to every subsequent downlink, and thus provides optimality, but a steep complexity penalty is paid due the generation of inequalities over all subscripts s, o, t. Ineq. 3.15 complements equation 3.14 by restricting the available data volume that can be routed for o on link l by the $v_{s,o,t}$ variable. $L^-_{o,\tau}$ is similar to $L_{o,\tau}$, except is restricted to only transmitting links.

Ineq. 3.16 constrains the latency reward for an observation-downlink pair. The activity indicator for downlink d can only be high when the data volume for o from that downlink, added on to the data volume for o from all preceding downlinks, reaches the minimum data volume requirement. The inner summation sums across all downlinks at time τ, and the outer summation varies τ from the time of the observation to the time of downlink d. What this effectively means is that a downlink can only be counted as "completing" the data volume delivery requirement for o if

$MinDV$ data volume has been routed by the time it is finished.

A note on timing is in order. T_o is the set of all link activity center times from the center time of the observation onwards to the end of the planning window. This means that this formulation will consider routing data between two windows m and n that are overlapping, if the center time of m follows n. This is significantly less restrictive than only allowing routing between two windows if they are completely un-overlapped (that is, in original window start, end times).

The deconfliction of undesired activity overlap and activity minimum duration is enforced by the next set of constraints.

$$I_m + I_n \leq 1 \quad \forall m, n \in A \mid t_{n,c} - t_{m,c} < \tau_{m,n}^{transition} \tag{3.17}$$

$$t_{n,c} - \frac{c_n x_n}{2\bar{r}_n} - t_{m,c} - \frac{c_m x_m}{2\bar{r}_n} \geq \tau_{m,n}^{transition} \quad \forall m, n \in A \mid t_{n,s} - t_{m,e} < \tau_{m,n}^{transition} \tag{3.18}$$

$$\frac{c_a x_a}{\bar{r}_a} \geq \tau_a^{min} \cdot I_a \quad \forall a \in A \tag{3.19}$$

Inequality 3.17 allows only one activity to be scheduled if the center times of the two activities are not far enough apart to allow the required transition time $\tau_{m,n}^{transition}$ between those two times. Ineq. 3.18 imposes a less stringent constraint, requiring the scheduled start time of activity n, $t'_{n,s} = t_{n,c} - \frac{c_n x_n}{2\bar{r}_n}$, to follow the scheduled end time of activity m by at least the required transition time. Ineq. 3.19 forces an activity to be scheduled for at least a minimum amount of time.

The next set of inequalities enforces energy and data storage constraints.

$$\sum_{o \in O}(v_{s,o,t}) \leq d_s^{max} \quad \forall o \in O, \forall s \in S, \forall t \in T_o \tag{3.20}$$

$$e_s^{min} \leq e_{s,t} \leq e_s^{max} \quad \forall s \in S, \forall t \in T^{rsrc} \tag{3.21}$$

$$e_{s,t=0} = e_s^{start} \quad \forall s \in S \tag{3.22}$$

$$e_{s,t+1} \leq e_{s,t} + (\dot{e}_{s,t}^{charging} + \dot{e}_s^{base} + \dot{e}_a x_a) \cdot \Delta t \quad \forall s \in S, \forall t \in T^{rsrc}, \forall a \in A_t \tag{3.23}$$

$$e_{s,t+1} \geq e_{s,t} + (\dot{e}_s^{base} + \dot{e}_a x_a) \cdot \Delta t \quad \forall s \in S, \forall t \in T^{rsrc}, \forall a \in A_t \tag{3.24}$$

Ineq. 3.20 bounds maximum data storage on a satellite. Eqs. 3.21 through Ineq. 3.24 constrain stored energy onboard satellite s at time t, $e_{s,t}$. The first two constrain the minimum and maximum value of energy storage, and constrain the initial bound to be equal to the current state of the satellite, at time $t = 0$. The second two constrain energy change from time step the time step. In Eqs. 3.23 and 3.24 we factor in any gain in energy storage due to charging from the solar panels ($\dot{e}_s^{charging}$) (dot notation signifies time derivative), any loss from base energy usage present on the satellite at all times ($\dot{e}_s^{base}$), and any loss from activity execution ($\dot{e}_a$). These terms are positive or negative, as appropriate. T^{rsrc} in general uses a different, higher level of granularity (smaller timestep) from activity durations. Δt is the time difference between each successive time point in T^{rsrc}.

We assume that charging is only available when the satellite is not in eclipse; the t subscript accounts for this time dependence. The base usage is assumed to be constant for a given satellite. The activity energy usage is not indexed by time t due to the fact that our activity window definition assumes the activity starts and ends at a defined time, known *a priori*. The activity's energy usage is only included for those time steps during which the activity can potentially be executed (accounted for by the set A_t, the activities executable at timepoint t). At first glance, it may seem like A_t is not known *a priori*, because the scheduled start and end times of a given activity are decided as part of the optimization. This difficulty is resolved by the use of the x_a activity utilization factor: we include the activity's energy usage for all potential time steps during which it could be executed (from original start time to original end time) and discount that energy usage at every time step by the fractional execution of the activity. Effectively we spread the activity's energy usage out over a longer time period to avoid having to include extra complexity in the formulation to deal with the true scheduled times of the activity. While this is not a conservative assumption, because it can spread out spikes in energy usage that could potentially push $e_{s,t}$ below the minimum bound, we assume that activities are sufficiently granularized in duration that the negative effect (if any) is small.

The final set of constraints equations, 3.25, enforce appropriate accounting for

latency rewards for observations.

$$
\begin{aligned}
f_o &\leq 0 + M^{lat} \cdot \sum_{k \in \{1,2,3,\ldots|D_o|\}} (I_{o,d_k}) \\
f_o &\leq \phi_{o,d_1} + M^{lat} \cdot \sum_{k \in \{2,3,\ldots|D_o|\}} (I_{o,d_k}) \\
&\ldots \\
f_o &\leq \phi_{o,d_{|D_o|}} + M^{lat} \cdot \sum_{k \in \{|D_o|\}} (I_{o,d_k})
\end{aligned}
\tag{3.25}
$$

Recall that f_o is the latency reward factor for a given observation. M^{lat} is a large value, following the "big M" approach [5]. If M^{lat} is multiplied by zero, the constraint has an effect; if it is multiplied by any other number (which in this case will be a positive integer), the constraint is effectively disabled. Here, one equation is generated for every d_k in D_o, the set of all downlinks following observation o. D_o is sorted in decreasing downlink absolute time, hence the k subscript enumerates the downlinks in decreasing latency for o. The $0 \leq \phi_{o,d_k} \leq 1.0$ constant term is the reward factor for a given downlink, determined by its latency for o. Latency is calculated as the difference between the downlink absolute time and the observation absolute time (*e.g.* center time). The reward factor for a given downlink is calculated as:

$$
\phi_{o,d_k} = \max\left(\frac{t_{d_k} - t_o}{\min\limits_{d_k \in D_o} (t_{d_k} - t_o)}, \phi_{d_k}^{min}\right) \tag{3.26}
$$

Where $\phi_{d_k}^{min}$ is 1.0 if the latency for d_k is less than a configurable minimum latency value, to avoid over-emphasizing the reward for very short latencies. The shortest latency downlink (or downlinks) has a 1.0 reward factor and longer downlinks have rewards that decrease linearly with increasing latency. Overall, Eq. 3.25 enforces a staircase pattern for f_o's value as earlier downlinks are chosen, by constraining f_o to be less than or equal to the reward factor for the earliest scheduled downlink that completes the minimum data volume delivery requirement for o (the I_{o,d_k} term). This incentivizes the optimization process to choose earlier downlinks for every observation.

Objectives

There are three objective terms implemented in the current version of the GP-Optimal algorithm, as detailed in equations 3.27 to 3.29. The first objective term maximizes the total data volume routed to ground from all observations. The second objective term maximizes the latency rewards summed over all observations. The third objective term maximizes average energy margin over all satellites, where energy margin for s is defined as the difference between energy storage at t and the minimum allowed for s, divided by the difference between maximum and minimum for s. The third objective term maximizes average margin because it can trade margin at one timepoint for margin at another.

$$\Omega_1 = \sum_{o \in O}(v_o) \tag{3.27}$$

$$\Omega_2 = \sum_{o \in O}(f_o) \tag{3.28}$$

$$\Omega_3 = \sum_{t \in T^{rsrc}, s \in S} \frac{e_{s,t} - e_s^{min}}{e_s^{max} - e_s^{min}} \tag{3.29}$$

These objective terms are each normalized to a maximum value of 1.0, weighted by individual weighting terms $w_i \in \mathbb{R}$, and summed together to produce a composite objective score Ω:

$$\Omega = w_1 \frac{\Omega_1}{\nu_1} + w_2 \frac{\Omega_2}{\nu_2} + w_3 \frac{\Omega_3}{\nu_3} \tag{3.30}$$

The normalization factors ν_i are:

1. The sum of all observation window throughput capacities: $\nu_1 = \sum_{o \in O}(c_o)$;
2. The number of observations: $\nu_2 = \mid O \mid$;
3. The number of resource timepoints times satellites: $\nu_3 = \mid T^{rsrc} \mid \cdot \mid S \mid$.

The weighting terms are specified by the user. They may all be set to 1.0 to equally weight the terms, or be tuned in order to achieve a desired prioritization of

objectives. The sensitivity of objective performance to changing weighting terms is investigated in section 3.6. For a given problem instance, it may be the case that it is impossible to schedule a solution with a normalized score of 1.0 for one or more of these objective terms. In practice this is not problematic because it is not necessary to achieve the theoretical maximum for each objective, only to balance the objectives appropriately against each other.

3.2.2 GP-Optimal Validation

The GP-Optimal algorithm was run over a two hour planning window (from 2016-02-14T16:00:00 to 2016-02-14T18:00:00) for the 6-Sat scenario, with parameters as specified in A.1 [2]. Figures 3-3 to 3-5 show the schedule results from this run. Table 3.2 in section 3.3.3 summarizes the numerical results (grouped with results from GP-Fast to avoid redundancy).

Figure 3-3 shows the executed activities for the run. All of the potential data volume from the observation windows within the planning window is scheduled for delivery to downlinks successfully. The labels indicate, for each observation, the scheduled data volume and the potential data volume. Note that many crosslink windows are executed, which is a result of the fact that the crosslink data rate (10 Mbps) is much lower than the observation collection rate (50 Mbps), and satellites 0 and 3 do not have a downlink opportunity of their own within the planning window. Inter-activity transition times are also set to 0s here, so no attitude slew requirements between crosslinks are enforced.

Figure 3-4 shows scheduled energy usage. It can be seen that energy storage dips during eclipses due to the lack of sunlight available for charging, and at various points the line changes slope due to activities being executed. The slope changes align with the activities in fig. 3-3, as expected. Figure 3-5 shows scheduled data storage on the satellites, again aligning with the times when data is received and transmitted during activities. The blocky shape of these curves is due to the fact that data routes are

[2]Results were obtained with commit e6ffe2be1305e372d22a591874b6048b53110970 of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

considered to store their full data volume between the start of their first activity on a given satellite, and the end of their last activity on a given satellite. The scheduler does not consider incremental collection of data packets over the course of a given link activity, it simply considers that data volume to be present on the satellite at the start of the activity. This is symmetric, with the data volume being assumed present on both satellites on either end of a crosslink. While this is a conservative assumption in the scheduler, in this particular case it is evident from the data storage plots that it does not impact performance to a large degree due to the large amount of margin available at most points in time.

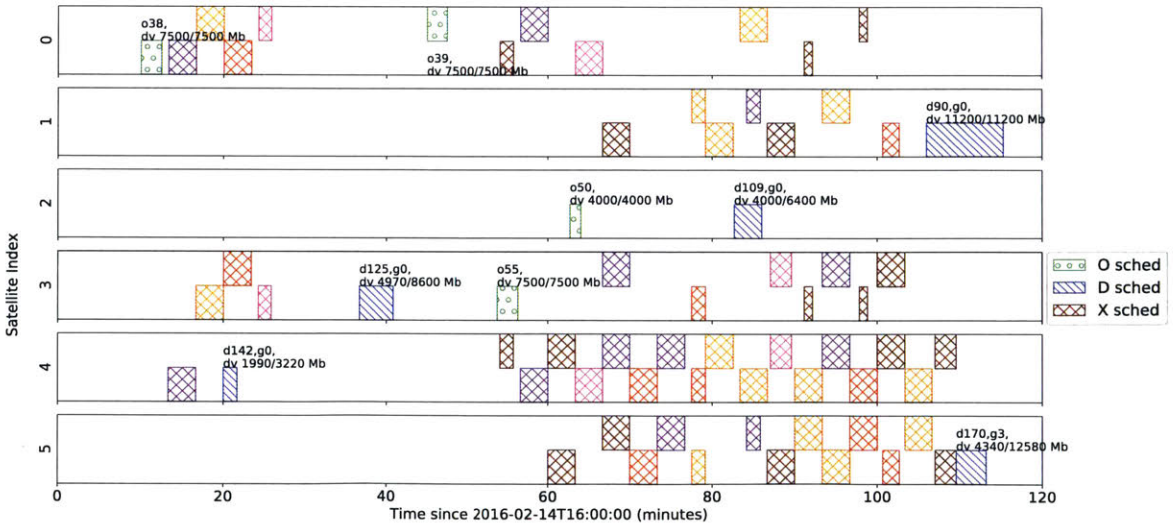


Figure 3-3: GP-Optimal scheduled activities over 2 hour planning window for 6-Sat scenario. Legend entries are matched by pattern to executed windows. The varied colors for crosslinks indicate different data routes passing through them (note some ambiguity may be present due to limited color choice allowance). Data volume usage for individual observation windows is indicated in text (scheduled/capacity). Windows are staggered vertically for a given satellite for legibility.

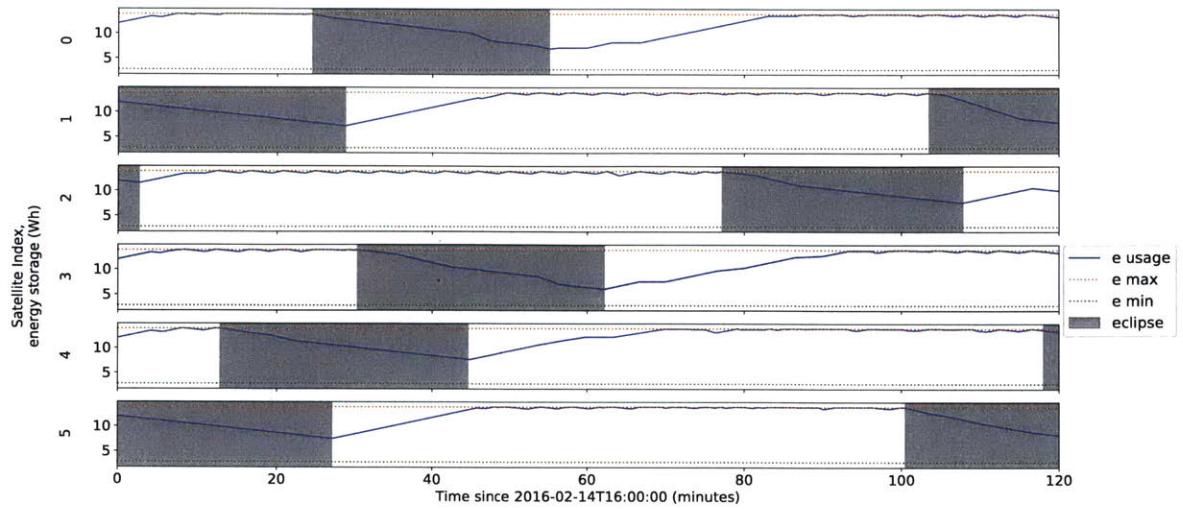


Figure 3-4: GP-Optimal scheduled satellite energy storage over 2 hour planning window for 6-Sat scenario. Same parameters as fig. 3-3.

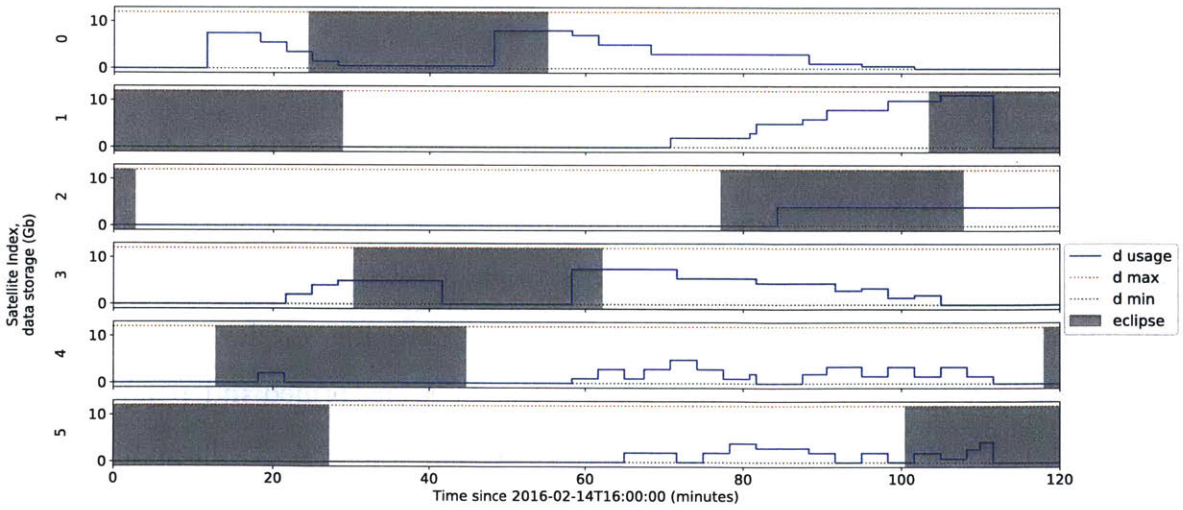


Figure 3-5: GP-Optimal scheduled satellite data storage over 2 hour planning window for 6-Sat scenario. Same parameters as fig. 3-3.

3.3 GP-Fast Algorithm

The Global Planner Fast algorithm [3], or "GP-Fast" was developed to significantly reduce the amount of time required to compute an acceptable schedule solution. It reduces the complexity of the GP-Optimal algorithm by pre-constructing a set of promising data routes before attempting to schedule them in the MILP solver. This removes the responsibility for enforcing route temporal constraints that is present within the GP-Optimal MILP formulation, effectively removing a large portion of the search space from the optimization.

3.3.1 GP-Fast Stage 1: Route Selection

The first stage of GP-Fast constructs a set of routes that are provided to the optimization solver. This stage is broken down into two steps, with the following responsibilities:

1. **Route Construction**: creates an initial, large set of routes from the input activity windows;
2. **Route Downselection**: selects a smaller set of routes from the initial set in order to reduce complexity in the Activity Scheduling stage.

Step 1: Route Construction (RS1)

Route Construction Overview

The Route Construction algorithm (algorithm 1) is run for a single observation window, o. It constructs a set of data routes originating at that observation window and ending at all possible downlinks d within the planning window from t_0 to t_h. One of the key benefits of this algorithm is that it is performed independently for every observation, allowing it to be run in parallel over a potentially large number of observation windows.

[3]The discussion here reflects commit 476f089e4fb882f44d423837725d028103c1ddac of the CIRCINUS Global Planner repository at `https://github.mit.edu/star-lab/circinus_global_planner`

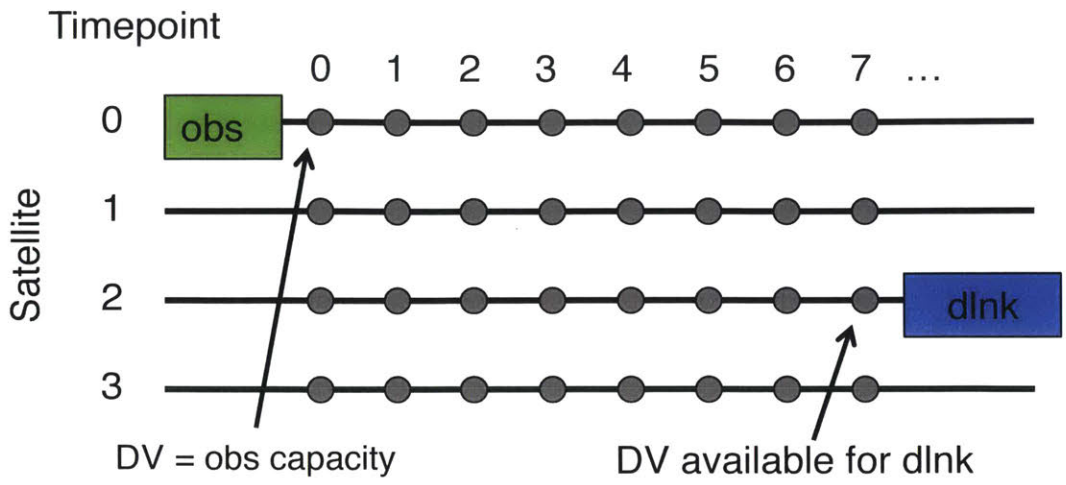


Figure 3-6: Depiction of the search space for the Route Construction algorithm. Each gray point is a (satellite,time) tuple at which the algorithm seeks to maximize deliverable data volume.

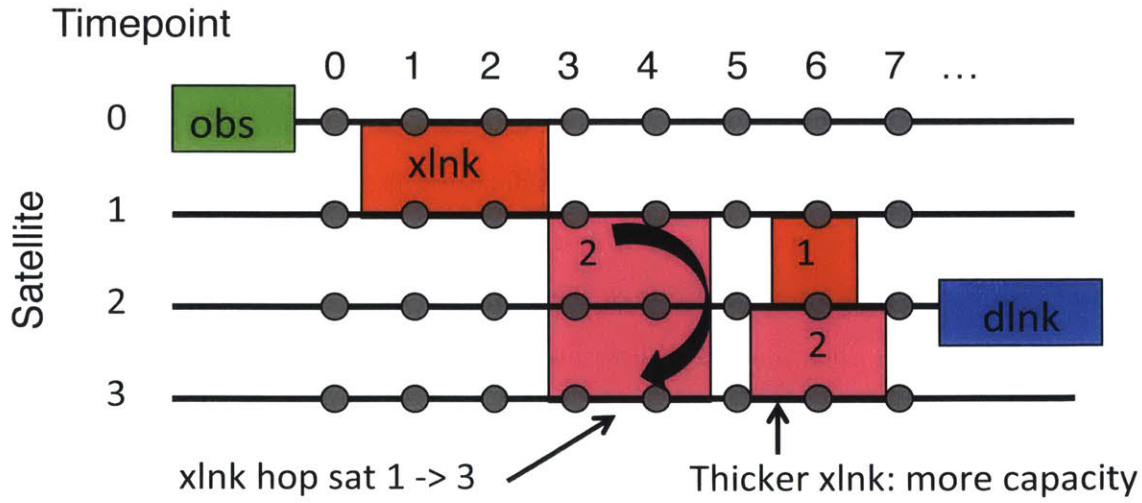


Figure 3-7: Depiction of two potential data routes constructed by Route Construction algorithm. Route 2 conflicts with route 1 around timepoint 6, so the algorithm will choose 2 due to its larger throughput.

The process is illustrated in figs. 3-6 and 3-7. Figure 3-6 shows the search space for the algorithm, which can be thought of as a matrix of satellite indices s on one axis and timepoints t up to the time horizon. The goal of the algorithm is to determine the set of constructed data routes which maximize the amount of data volume from o

that is present on s at the time point t immediately preceding each possible downlink d for s. This is deemed to be the amount of potential deliverable data volume for d, and a set of complete DRs is output for each d that accounts for the data in this deliverable amount. The activities in these DRs are o, any crosslinks that were used to deliver this potential data volume to d, and d itself.

Figure 3-7 gives further context by showing two possible DRs to deliver data volume from an observation obs to a downlink dlnk. Both routes include the first crosslink window xlnk from satellite 0 to 1. Route 1 also includes the crosslink labeled "1" and route 2 includes the two crosslinks labeled "2". Both routes deliver some data volume from obs to timepoint 7 on satellite 2. This particular example illustrates a case where two overlapping crosslinks are incompatible with each other; if we assume that satellite 2 may not crosslink with both satellite 1 and 3 at the same time, then only one of the routes can be executed. Here route 2 delivers more data volume to the downlink (as indicated by its thicker crosslink windows), so it would be output from the "Route Construction" step and route 1 would be discarded.

Route Construction Initialization

The Route Construction algorithm works forward from the first time point on every satellite (t_0, the observation time) and progressively attempts to determine at each (s,t) tuple the maximum amount of data volume from o that can be delivered. This is referred to as the "potential downlink capacity" for (s,t). Inputs to the algorithm are t_0, t_h, the time delta between timepoints Δt, the sets of downlink and crosslink windows to consider, d^{input} and x^{input} respectively, the set of all satellite indices S, and the index of the observing satellite s^{obs} (line 1 of algorithm 1).

The algorithm starts at t_0, constructing an initial "proto"-DR on the observing satellite s^{obs} containing only the observation window with a data volume equal to o's full capacity times a multiplicative factor, $m_o > 1$ (line 4). Restricting potential downlink capacity to the observation's capacity tends to limit the algorithm to only picking early crosslinks (*i.e.* immediately following o), so m_o encourages picking additional, later crosslinks. Satellite s^{obs} is marked as having a potential downlink capacity equal to this data volume. This "marking" is stored as an object called a

Algorithm 1 The RS Step 1: Route Construction Algorithm (RS1)

```
1: procedure RouteConstruction(t_0,t_h,Δt,o,d^input,x^input,S,s^obs)
2:     A ← FilterActs(t_0, t_h, d^input, x^input)
3:     RR, RR^dlnk ← ∅                                   ▷ RR := route record
4:     DR^init ← CreateDR(o)
5:     RR^init ← CreateRouteRecord(t_0, s^obs, DR^init)        ▷ RR^init dv = c_o m_o
6:     T ← Range(t_0, t_h, Δt)                          ▷ timepoints between t_0, t_h
7:     for t ∈ T do
8:         for s ∈ S do
9:             a ← GetActsAtTime(A, t)
10:            XlnkRtsCandidates ← ∅
11:            for x ∈ GetXlnks(a) do
12:                RR_s ← GetBestRR(RR, t, x, s)
13:                RR_i ← GetBestRR(RR, t, x, GetPartner(x, s))
14:                XlnkAvailRts ← GetDeconflictedRts(x, RR_s, RR_i)
15:                XlnkRtsCandidates ∪ CreateListObj(XlnkAvailRts)
16:            end for
17:            BestXlnkRtsCandidate ← ArgMax_dv(XlnkRtsCandidates)
18:            if not BestXlnkRtsCandidate = ∅ then
19:                RR^new ← GetMatchingRR(BestXlnkRtsCandidate, s)
20:                                    ▷ (above) recover the RR_s from line 12
21:                AddRoutes(RR^new, BestXlnkRtsCandidate)
22:            else
23:                RR^new ← GetRR(RR, t − 1, s)
24:            end if
25:            RR ∪ RR^new
26:            for d ∈ GetDlnks(a) do
27:                RR_s ← GetBestRR(RR, t, d, s)
28:                if GetDV(RR_s) = 0 then
29:                    continue                          ▷ next for loop iter (next d)
30:                end if
31:                Rts ← GetRoutes(RR_s)
32:                DlnkRts ← GetCapacRoutes(Rts)        ▷ all routes up to d's dv
33:                DlnkRtsUpdate ← Append(DlnkRts, d)
34:                RR^new ← CreateRouteRecord(t, s, DlnkRtsUpdate)
35:                        ▷ (above) RR^new dv = sum DR dv ∀DR ∈ DlnkRtsUpdate
36:                RR^dlnk ∪ RR^new
37:            end for
38:        end for
39:    end for
40:    return R^RS1 ← GetRoutes(RR) for RR in RR^dlnk
41: end procedure
```

Route Record, which stores the set of proto-DRs used to deliver a certain data volume to an (s,t) point. All other satellites are marked as having zero potential downlink capacity at t_0. The algorithm advances through timesteps on line 7.

Route Construction Crosslink Candidate Enumeration

The following procedure applies after the algorithm advances to any next time step t. It loops over all satellite indices s, for each one checking if there are any crosslink windows $x_{i->s}$ which have ended in the time between t and $t-1$ for which s is a receiving satellite (indicated with the x variables, line 11). For each of these crosslinks, if the transmitting satellite i has any useful data volume from o, then that x is stored off as a potential crosslink for s to choose to execute at this time step (line 15). The GETBESTRR routine (lines 12,13) is used to find the best Route Record (set of proto-DRs) on each satellite, accounting for any required transition time between x and activities in the proto-DRs. In the current version of RS1, the best choice is the most recent preceding Route Record that does not overlap with x (with transition time included). The best Route Record may be found an arbitrary number of time steps m into the past (*e.g.* at $(i,t-m)$), to account for varying crosslink length and activity overlap.

For every potential crosslink, the data routes ending at $(i,t-m)$ (the route record RR_i at $(i,t-m)$), are de-conflicted with any data routes present at $(s,t-m)$, RR_s (line 14). This deconfliction procedure attempts to maximize the total data volume deliverable to (s,t) by varying the data volume assignments to each DR within RR_i. The deconfliction is performed because in general we may not want to double count data volume; *e.g.* f satellite i is the observing satellite, it could be the case that s has already chosen to perform several crosslinks with i already and its potential downlink capacity is already equal to the available observation data volume. In this case we may decide that the crosslink doesn't need to be executed. The deconfliction algorithm itself is formulated and solved as a linear program in which the decision variables are the allocated data volumes for all routes r in RR_i, and the constraints derive from capacities of all of the activities present within the routes in both RR_s and RR_i. The data volumes for all routes r passing through a given activity must be

less than or equal to the capacity of that activity. The objective function is the total sum of all data volumes for all r. After solving this deconfliction step, we have a final set of potential routes $XlnkAvailRts$ that represent the possible downlink capacity if the given crosslink is chosen.

Route Construction Crosslink Candidate Selection

Next the best crosslink is chosen from all potential crosslinks $x_{i->s}$ for (s,t). The best crosslink is the one that delivers the most additional data volume to s, thus increasing its potential downlink capacity the most. This is determined as:

$$v_{s,t} = \max(\max_{i}(v_{s,t-m} + c_{x_{i->s}}), v_{s,t-1}) \tag{3.31}$$

Where $v_{s,t}$ is the new potential downlink capacity and $c_{x_{i->s}}$ is the capacity of $x_{i->s}$. So if any crosslinks are available and can deliver data volume, the one that delivers the most will be chosen. If none are present, then the new potential downlink capacity at t is the same as the value at the previous time step $t-1$. The new data routes that reflect this crosslink choice for s at t are stored in a Route Record RR^{new}, which is added to all the records **RR** for consideration at future timesteps. This whole candidate selection procedure is captured in lines 17 to 25.

Route Construction Finalization

Finally, every time the algorithm finds a downlink window d that starts between $t-1$ and t, it creates a set of final, constructed DRs that end at d. It takes every DR from the best Route Record found prior to the downlink (line 27) and selects a subset that sum up to capacity of d (line 32). The d window is appended to the end, creating the final data routes, $DlnkRtsUpdate$, line 33). This set of DRs is added to the full set of output DRs for the Route Construction step, R^{RS1}.

The current version of the Route Construction algorithm focuses on minimizing latency for observations, by ensuring that for any given observation the earliest possible downlinks can be found that can deliver data to ground for that observation. This is why an emphasis is placed on choosing routes with early crosslinks in the algorithm. In practice, this approach can be too rigid for dealing with large transition

time requirements between activities, because it might not provide enough temporal diversity in the set of routes for any given observation. Improvement of this construction process is an item for future work.

Step 2: Route Downselection (RS2)

The second step of the route construction process selects a small number of routes to pass to the Activity Scheduling stage by significantly pruning the potentially large number of routes from step one, R^{RS1}. This process is illustrated in figs. 3-8 and 3-9. Figure 3-8 shows a notional set of output routes from step one. Every observation has one or more routes to every subsequent downlink within the planning window that is reachable via crosslinks over the constellation network. In the general case many of these routes will be redundant, having one or more of the same windows as other routes, and many of them will deliver an undesirably small amount of data volume or have too long of latency (time difference between the observation and downlink) to be useful. Figure 3-9 shows a downselected set of routes, in which the routes to lower-latency downlinks and downlinks with higher capacities are chosen.

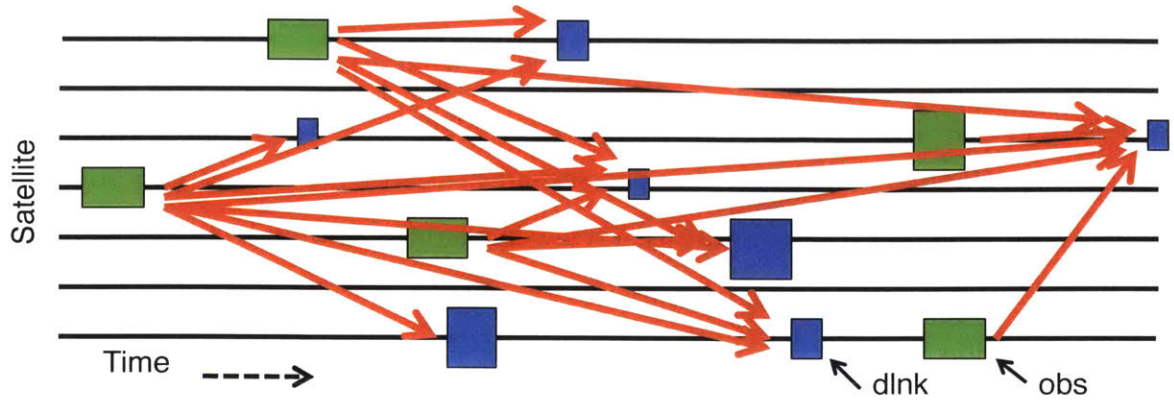


Figure 3-8: Notional depiction of the data routes output by the Route Construction algorithm. Every observation attempts to route data to every subsequent downlink, producing a large set of potential data routes. Red arrows represent paths taken for potential routes.

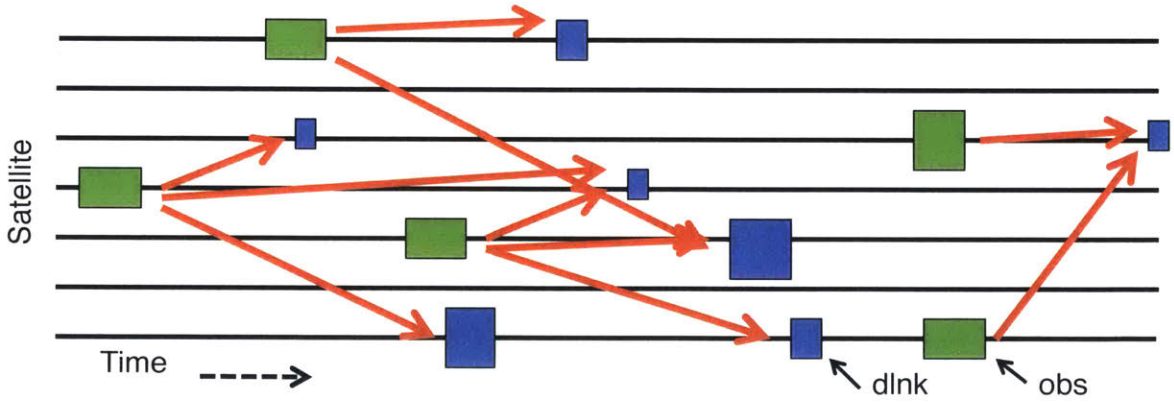


Figure 3-9: Notional depiction of the data routes output in GP-Fast after the Route Downselection step. For every observation, a smaller set of potential data routes has been selected based upon their throughput, latency, and overlap with other routes.

A simple procedure is used to perform this downselection. For every observation, sets of routes are chosen using three heuristic algorithms:

1. Highest data volume (ρ_1^{RS2})
2. Lowest latency (ρ_2^{RS2})
3. Least overlap (ρ_3^{RS2})

The ρ_i^{RS2} values specify how many routes are chosen for each algorithm. The first heuristic incentivizes choosing routes that deliver a large amount of data volume. This is important because it is often desired to maximize the amount of total data volume throughput in the network. The second heuristic incentivizes choosing low latency routes in order to provide at least a few low latency downlink choices for every observation. The routes for these heuristics are determined in a similar manner: for each observation routes are sorted in decreasing data volume order or increasing latency order. Then the first $\rho_{1/2}^{RS2}$ routes are chosen from the sorted list.

The third heuristic algorithm is slightly more complicated. The algorithm iterates through all observations from the latest observation to the earliest observation, picking one route for each observation. It repeats this iteration ρ_3^{RS2} times. For each observation at each iteration, the picked route is is the one with the most "available data volume" over all of the routes still available to be picked for o. The available data volume metric for route r is determined in the following manner. At the start of execution, the third heuristic algorithm creates a mapping of activity window IDs to available data volume numbers (window IDs are unique for every activity window). These available data volumes are initially populated with the capacity of each window. Every time a route is picked, all of the windows in the picked route have their capacities subtracted from the available capacity for their corresponding window ID. The available data volume for r is then the minimum available data volume across all windows in the route. The available data volume for a given activity in the mapping is allowed to go negative, so as to increasingly dis-incentivize the selection of a route including that window. Finally, the route with the highest available data volume number at each iteration is chosen.

The third heuristic is intended to provide a set of fallback routes that can be scheduled for an observation if the lowest latency and highest data volume routes are for some reason not easy or impossible to schedule. This can come up, for example, when multiple observations occur within a short period of time (either on a single satellite or multiple satellites) and all of their best routes rely on a single satellite for downlink. This case can be particularly important when certain satellites are unable

to downlink within the planning window, and need to offload all of their observation data to other satellites for downlink. The heuristic attempts to to distribute the usage of activity windows across the satellites in order to reduce the overall effect of temporal and data volume constraints for scheduling within the activity scheduling stage. Note that this heuristic becomes computationally expensive over a very large set of routes, because the "most available data volume" calculation needs to check all activities within all routes for each observation at each iteration. Nonetheless, the method has been shown to work when scheduling for a 100 satellite constellation, as discussed in section 6.1.2.

3.3.2 GP-Fast Stage 2: Activity Scheduling

Decision Variables

The decision variables for the GP-Fast formulation are shown in definitions 3.32 to 3.39. x_a is the fractional utilization for activity a, which specifies both the scheduled duration and the scheduled data volume for a. x_r is the fractional utilization for route r, which similarly specifies the amount of data volume used out of the route's potential throughput, and also fixes the scheduled times for all activities within the route. I_a is an indicator variable which is set high (to 1) if a is scheduled for execution. I_r is another indicator variable, which is set high if a route has sufficient scheduled data volume to meet the minimum data volume downlink requirement ($MinDV$) for the observation to which it corresponds. x_r^{reward} is also a fractional utilization term, but is specifically included in the objectives in order to reward the selection of existing routes $r \in R^{existing}$. Existing routes are those that have already been scheduled by the GP on a previous run. We want to explicitly consider them and reward selection of them in subsequent runs of the GP in order to prevent the GP from vacillating back and forth between scheduled route possibilities. The f_o and $e_{s,t}$ are similar to those in the GP-optimal formulation in section 3.2.1; the first is the latency reward factor for observation o and the second is the energy storage on satellite s at time t. The final variable $d_{s,t}$ is an explicit variable for data storage. This explicit variable

differs from GP-Optimal, which used the $v_{s,o,t}$ term to double as a data storage value.

$$0 \leq x_a \leq 1 \quad \forall a \in A \tag{3.32}$$

$$0 \leq x_r \leq 1 \quad \forall r \in R \tag{3.33}$$

$$I_a \in \{0,1\} \quad \forall a \in A \tag{3.34}$$

$$I_r \in \{0,1\} \quad \forall r \in R \tag{3.35}$$

$$x_r^{reward} \geq 0 \quad \forall r \in R^{existing} \tag{3.36}$$

$$f_o \geq 0 \quad \forall o \in O \tag{3.37}$$

$$e_{s,t} \geq 0 \quad \forall s \in S, \forall t \in T^{rsrc} \tag{3.38}$$

$$d_{s,t} \geq 0 \quad \forall s \in S, \forall t \in T^{rsrc} \tag{3.39}$$

Constraints

The first set of inequalities, 3.40 to 3.43, form the core of the GP-Fast formulation:

$$c_a x_a - \sum_{r \in R_a} (c_r x_r) \geq 0 \quad \forall a \in A, \forall r \in R_a \tag{3.40}$$

$$I_a \geq x_a \quad \forall a \in A \tag{3.41}$$

$$c_r x_r \geq MinDV \cdot I_r \quad \forall r \in R \tag{3.42}$$

$$I_a \geq I_r \quad \forall a \in A, \forall r \in R_a \tag{3.43}$$

Inequality 3.40 constrains the cumulative data volume usage for all routes r passing data through activity a to be less than or equal to the throughput used in a. The set R_a is all routes passing through a, and is constructed in advance for every activity. This only includes activities present in the routes R, which is generally a much smaller set than all activities within the overall planning window. Ineq. 3.40 is the linchpin of this formulation, restricting the data volume usage for any route r by the throughput available in all activities along its path, and de-conflicting data volume assignments

for routes that overlap in a given window. Ineq. 3.41 constrains the activity indicator to be set high if there is any utilization of a. Ineq. 3.42 allows the route indicator to only be high if the minimum routed data volume requirement is met. Ineq. 3.43 is not strictly necessary, as it is implied by Ineqs. 3.40 and 3.41. However, it is included to help speed up the MILP solver by explicitly relating binary variables together, which can be used to cut down the binary variable search tree used in the Branch and Cut algorithm.

The next set of inequalities, 3.44 to 3.46, is exactly the same as those in the GP-optimal formulation (section 3.2.1). They enforce the required transition times between activities and minimum duration times for activities:

$$I_m + I_n \leq 1 \quad \forall m, n \in A \mid t_{n,c} - t_{m,c} < \tau_{m,n}^{transition} \tag{3.44}$$

$$t_{n,c} - \frac{c_n x_n}{2\bar{r}_n} - t_{m,c} - \frac{c_m x_m}{2\bar{r}_n} \geq \tau_{m,n}^{transition} \quad \forall m, n \in A \mid t_{n,s} - t_{m,e} < \tau_{m,n}^{transition} \tag{3.45}$$

$$\frac{c_a x_a}{\bar{r}_a} \geq \tau_a^{min} \cdot I_a \quad \forall a \in A \tag{3.46}$$

The next set of inequalities, 3.47 to 3.52, constrain resource usage for the satellites:

$$e_s^{min} \leq e_{s,t} \leq e_s^{max} \quad \forall s \in S, \forall t \in T^{rsrc} \tag{3.47}$$

$$e_{s,0} = e_s^{start} \quad \forall s \in S \tag{3.48}$$

$$e_{s,t+1} \leq e_{s,t} + (\dot{e}_s^{charging} + \dot{e}_s^{base} + \dot{e}_a x_a) \cdot \Delta t \quad \forall s \in S, \forall t \in T^{rsrc}, \forall a \in A_t \tag{3.49}$$

$$e_{s,t+1} \geq e_{s,t} + (\dot{e}_s^{base} + \dot{e}_a x_a) \cdot \Delta t \quad \forall s \in S, \forall t \in T^{rsrc}, \forall a \in A_t \tag{3.50}$$

$$d_{s,t} = \sum_{r \in R_{s,t}^{store}} (c_r x_r) \quad \forall s \in S, \forall t \in T^{rsrc} \tag{3.51}$$

$$d_{s,t} \leq d_s^{max} \quad \forall s \in S, \forall t \in T^{rsrc} \tag{3.52}$$

Ineqs. 3.47 to 3.50 are the same as those from the GP-optimal formulation (section 3.2.1, repeated here for completeness). They relate energy usage to activity timing.

The following two inequalities, 3.51 and 3.52, are new. They constrain the data storage (buffered data) onboard satellite s at time t. Equality 3.51 fixes the the value of the data storage term to be equal to the sum of the data volume stored by all routes which are "passing through" s at t: $R^{store}_{s,t}$. This set of routes is precomputed before constructing these constraints. This is possible because we know all of the activities that are present along a route, so we know which satellites a route must store data volume on along its path. For example, if route r includes a crosslink from satellite 5 to 2 ending at $t_1 = 30$ minutes and a crosslink from sat 2 to 15 starting at $t_2 = 45$ minutes, we know that r will need to buffer $c_r x_r$ amount of data volume on satellite 2 for all time points between t_1 and t_2.

The next set of inequalities, ineqs. 3.53, are again similar to those in the GP-optimal formulation (section 3.2.1), and fix the values of the observation latency reward terms. r_k is the kth route in the set R_o, which is sorted in decreasing route latency for observation o. The ϕ_{o,r_k} terms are computed in a similar manner to ϕ_{o,d_k} in Eq. 3.26, with the latency term calculated as the difference between the downlink time in the observation time in route r.

$$\begin{aligned} f_o &\leq 0 + M^{lat} \cdot \sum_{k \in \{1,2,3...|R_o|\}} (I_{r_k}) \\ f_o &\leq \phi_{r_1} + M^{lat} \cdot \sum_{k \in \{2,3...|R_o|\}} (I_{r_k}) \\ &... \\ f_o &\leq \phi_{r_{|R_o|}} + M^{lat} \cdot \sum_{k \in \{|R_o|\}} (I_{r_k}) \end{aligned} \tag{3.53}$$

The final set of inequalities, 3.54 to 3.56, allow the current run of the GP-Fast algorithm to include consideration for existing routes from a previous run of GP-Fast:

$$x_r \leq x_r^{existing} \quad \forall r \in R^{fixed} \tag{3.54}$$

$$x_r^{reward} \leq x_r \quad \forall r \in R^{existing} \tag{3.55}$$

$$x_r^{reward} \leq x_r^{existing} \quad \forall r \in R^{existing} \tag{3.56}$$

$$\tag{3.57}$$

These routes, $R^{existing}$, are similar to new routes constructed in GP-Fast (in Route Selection), except that they could include activity windows that occur before the planning window for the current run of GP-Fast starts. These activities are considered to be "fixed"; that is, the utilization for them cannot be increased because it would represent the introduction of new timing constraints that were not included in the original schedule. This means that the utilization for the routes including these fixed windows, $r \in R^{fixed}$ cannot be increased beyond their current value. Ineq. 3.54 captures this constraint, forcing the utilization in the current schedule to be less than or equal to the utilization in the previous schedule, $x_r^{existing}$. We must also consider the fact that new routes could be introduced in this run of GP-Fast that are essentially equivalent to ones in $R^{existing}$, and could be chosen by the optimization process. It would not be good to allow the global planner to continually switch the activities it has assigned for execution to satellites, because this could end up breaking apart routes midway through execution. For this reason a reward factor is assigned for existing routes, x_r^{reward}, which is essentially equivalent to the utilization for the existing routes. It is constrained to be less than or equal to the utilization in the current schedule, x_r, as well as the utilization from the previous schedule $x_r^{existing}$. Both constraints 3.55 and 3.56 are necessary in order to prevent the optimization from simply maximizing the utilization of existing routes (*i.e.* setting each to 1.0); no additional reward is given for scheduling more data volume from an existing route than was scheduled previously.

Objectives

The objective terms for GP-Fast are implemented similarly to GP-Optimal (in section 3.2.1), with a notable additional term to handle reward for existing routes. The objective terms are shown in equations 3.58 to 3.61. The first maximizes total data volume (throughput) delivered from all observations. The second maximizes the latency rewards summed over all observations. The third maximizes average energy margin over all satellites. The fourth maximizes reward for existing routes, attempting to send as much data volume along the routes as was previously scheduled. Note that the fourth objective term was not included in GP-Optimal in its current version simply because it was not run in the full constellation simulation, with a receding horizon planning process.

$$\Omega_1 = \sum_{r \in R}(c_r x_r) \tag{3.58}$$

$$\Omega_2 = \sum_{o \in O}(f_o) \tag{3.59}$$

$$\Omega_3 = \sum_{t \in T^{rsrc}, s \in S} \frac{e_{s,t} - e_s^{min}}{e_s^{max} - e_s^{min}} \tag{3.60}$$

$$\Omega_4 = \sum_{r \in R^{existing}} (x_r^{reward}) \tag{3.61}$$

The objective terms are each normalized to a maximum value of 1.0, weighted by individual weighting terms ($w_i \in \mathbb{R}$, specified as inputs), and summed together to produce a composite objective score Ω:

$$\Omega = \sum_{1:4}(w_i \frac{\Omega_i}{\nu_i}) \tag{3.62}$$

The normalization factors ν_i are:

1. The sum of all observation window capacities: $\nu_1 = \sum_{o \in O}(c_o)$

2. The number of observations: $\nu_2 = \mid O \mid$

3. The number of resource timepoints times number of satellites: $\nu_3 = \mid T^{rsrc} \mid \cdot \mid S \mid$

4. The sum of existing route utilization terms: $\nu_4 = \sum_{r \in R^{existing}} (x_r^{existing})$

3.3.3 GP-Fast Validation

Route Selection Validation

The route selection stage was validated by comparing the output routes from RS1 (Route Construction) to the output routes from RS2 (Route Downselection), as summarized in table 3.1 [4]. Results were examined from two simulation cases, 6-Sat and SSO Ring, with simulation parameters summarized in appendices A.1 and A.2. Both were run with a planning window of 95 minutes for observations and crosslinks and 760 minutes for downlinks. The numbers of routes in the downselection stage were: $\rho_1^{RS2} = 6$ (highest data volume), $\rho_2^{RS2} = 6$ (lowest latency), and $\rho_3^{RS2} = 30$ (least overlap). Multiple simulations were run with different setting for $\rho_{1/2/3}^{RS2}$, and this combination was found to deliver good performance in reasonable runtime.

We see that for both sim cases, downselection significantly reduces the number of routes that will be input to Activity Scheduling. The average number of routes per observation was decreased to about the full number of routes specified for downselection (42), though slightly less because some observations had fewer routes available to downselect. Potential throughput and initial observation latency are determined here by 1) summing observation window data volume over all route capacities (bounded by the available capacity of each obs) and 2) taking the minimum latency over all routes for a given obs, then averaging over all obs windows. These values were the same for RS1 and RS2 for both sim cases; assuming that all the routes from both RS1 and RS2 are fully schedulable in AS (not true in general), the two sets of routes would provide equal schedule quality.

The final metrics measure the amount of inter-route overlap present. The metric

[4]Results were obtained with commit 52f8fb2051673b4834c8a96afb3645e43b339f7d of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

Table 3.1: Comparison of Output Routes Quality For Route Selection Steps

Metric	6-Sat		SSO Ring	
	RS1 Output	**RS2 Output**	**RS1 Output**	**RS2 Output**
Number of routes	635	222	1505	298
Average num routes per obs	106	37	167	33
Potential total throughput (Gb)	30.0	30.0	79.1	79.1
Potential average obs initial latency (minutes)	19.4	19.4	25.73	25.73
Average percent unoverlapped routes	6.2 %	13.2 %	23.5 %	39.15 %
Average percent unoverlapped routes, with xlnk	0 %	4.4 %	0 %	13.2 %

"percent unoverlapped routes" is the percentage of routes for each observation that have no activity window overlaps with routes from other observations; the averaged version of the metric is averaged across all observations. The metric "average percent unoverlapped routes, with xlnk" is the same calculation, but only considers routes that contain a crosslink. These metrics essentially measure how much the routes are conflicted between each other. A higher average percent suggests that the routes are easier to schedule. The "with xlnk" version focuses specifically on routes with crosslinks, *i.e.* those that tend to have lower latency. We see that in both cases these overlap metrics are reduced in the routes output from RS2, suggesting that the downselection algorithm chooses routes with fewer conflicts successfully. Note that these last metrics are not necessarily that useful for larger constellations, where the chance of inter-route overlap increases simply because there are more observations for which to pick routes.

Activity Scheduling Validation

Figure 3-10 shows the scheduled windows for the GP-Fast algorithm in the same 2 hour planning window as for the previous validation results for GP-Optimal in section 3.2.2 (from 2016-02-14T16:00:00 to 2016-02-14T18:00:00 for the 6-Sat scenario, with parameters as specified in A.1) [5]. It shows slightly different scheduled windows from the optimal algorithm, but the same data volume throughput and latency performance.

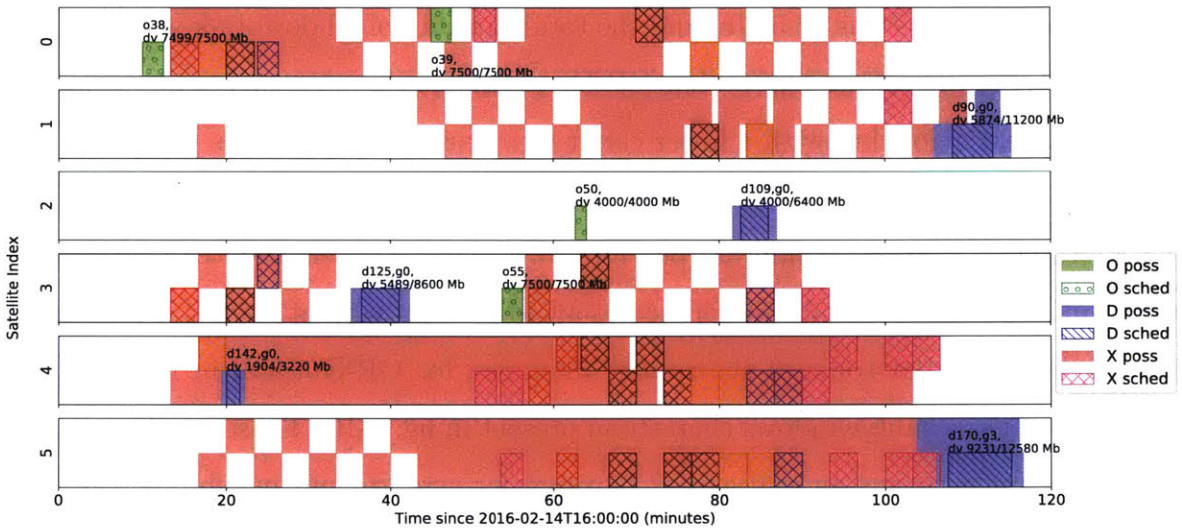


Figure 3-10: Plot of scheduled activity windows output by GP-Fast for 6-Sat scenario, 2 hour planning window. Legend entries are matched by pattern to executed windows. Windows are staggered vertically for a given satellite for legibility.

Table 3.2 compares GP-Fast performance to GP-Optimal. The first metric is the number of observations scheduled, the second is the total data volume throughput scheduled across all observations, the third is the average latency to deliver the first 100 Mb of each observation ($MinDV$), the fourth is the satellite-average energy margin as averaged across all satellites, and the fifth is the count of routes scheduled by AS. Standard deviation is indicated with the "$\pm$" terms.

We see that GP-Fast performs as well as the optimal case in throughput and latency. Note that for this scenario, the total collectible data volume for all four

[5]Results were obtained with commit e6ffe2be1305e372d22a591874b6048b53110970 of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

Table 3.2: Comparison of Metric Performance for GP Algorithms

Item	GP-Optimal	GP-Fast
Number obs scheduled	4	4
Total throughput (Gb)	26.5	26.5
Average obs initial latency (mins)	37.63 (±22.9)	37.63 (±22.9)
Average energy margin (%)	82.7 (±2.1)	81.8 (±3.0)
Number of routes scheduled	26	14

observation windows was 26.5 Gb, and the total capacity of all downlink windows was 60.4 Gb. One point of interest is that GP-Fast achieves the same metric performance as the optimal algorithm with a lower count of routes scheduled. This is because the GP-Fast algorithm tends to favor larger throughput routes relative to GP-Optimal. GP-Optimal is able to consider all possible ways that an observation can arrive at a downlink, and has no inbuilt preference to avoid low throughput routes. This can be seen in the fact that some of the routes scheduled for GP-Optimal in fig. 3-3 have much shorter crosslink windows than those present in fig. 3-10. These results validate that GP-Fast performs as well as GP-Optimal for a representative simulation case; further validation result are presented in section 3.5. Note that GP-Fast executed within 10 seconds on a late 2013 Macbook Pro with a 2 GHz Intel Core i7 processor and 8 GB of RAM.

3.4 Global Planner Limitations

The GP algorithms assume a linear problem model, in a similar manner to the models used in multiple state-of-the-art algorithms [124, 71, 83]. This naturally limits the type of scheduling problems that can be solved to those that have linear constraints and objectives. Others have investigated non-linear problem models that can feature more general-purpose constraints and objectives, *e.g.* Zheng *et al.* [122]. Such models could directly incorporate objectives terms involving averages (such as average latency or average AoI), as opposed to the linear terms used here that simply push the solver

towards a solution that has good performance for such metrics (see the GP-Optimal objectives in section 3.2.1). A non-linear model capability would be more straightforward for users, because there would be no adaptation process necessary for such terms.

As detailed in the GP-Fast algorithm formulation, the Route Construction step creates a set of data routes for each observation, deconflicting routes between each along the way in order to determine the most data volume that can arrive at a given downlink. As formulated, this procedure is inherently reductive: choices are made about which routes to send through which crosslinks. For this reason GP-Fast cannot be expected to return an optimal solution, no matter how much parameter tuning or extra runtime is given.

3.5 GP End-to-End Validation

For end-to-end validation of the schedules produced by the Global Planner, the GP algorithms are compared to the state-of-the-art algorithms proposed by Zhou *et al.* [124]. The constellation simulation case in their work was reproduced as accurately as possible, as the 6-Sat case in this work (see appendix A.1 for details). Certain parameters from the scenario were left unspecified (start time of planning window, right ascension of the ascending note of orbits, elevation masks of observations and downlinks), so representative parameters were selected to produce roughly the same throughput results, about 60 Mb of total throughput over a two hour planning window (results from an additional set of parameters are presented as well).

3.5.1 GP Throughput Performance Checks

GP throughput results were validated in the 6-Sat scenario with two sets of elevation masks for observation targets and ground stations [6]. Output schedules were produced for a set of 4 test case periods, summarized in table 3.3. Each test case period specifies

[6]Results were obtained with commit c7fd7faa0549d26f031bddbfe238cca3a7fede4a of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

the planning window length for all three activity types, observations, downlinks, and crosslinks.

Table 3.3: GP Throughput Test Cases, on 2016-02-14

Test Case	Period
1	4:00 to 6:00 UTC
2	7:00 to 9:00 UTC
3	16:00 to 18:00 UTC
4	18:40 to 20:40 UTC

Throughput results are summarized in table 3.4 and table 3.5. The total throughput is the sum of data volume over all scheduled data routes output by either GP-Optimal or GP-Fast. The number of observations scheduled is the number of observations that successfully downlink 100 Mb. We see that in almost all cases GP-Fast throughput performance is at least 90% of optimal. This is similar to the performance results presented by Zhou *et al.* for their fast heuristic algorithm (ACG, "algorithm based on conflict graph", referred to here as "Zhou-Fast"), which showed about 90% of optimal throughput performance in most cases with a two hour planning window (as seen in figs. 3-11 to 3-13). It should be noted that the Zhou *et al.* throughput results factored in weightings (≤ 1.0) for individual observation targets, whereas the GP algorithms weight all observation targets equally. By dividing the fast algorithm results by the optimal results in both cases, we can make a fair comparison between the two.

The GP-Fast algorithm performs slightly worse with the wider elevation masks (30° for obs, 0° for downlink), achieving only 85% of optimal in one case. This is due to the fact that the wider elevation masks lead to longer access periods and activity windows for both observations and crosslinks, leading to more overlap between the two types. Downlink windows were allowed to be as long as 20 mins before being cut into multiple activity window slices, which allowed some of the windows to fully overlap with (and entirely contain) observation windows. This can have the effect of preventing an observation from being scheduled, if the downlink is needed for

Table 3.4: Observation Activity and Throughput Scheduling Results for 6-Sat Sim Case with 30° Obs Elevation Mask and 0° Downlink Elevation Mask

Test Case	Number of Obs Scheduled		Total Througput (Gb)		
	GP-Optimal	GP-Fast	GP-Optimal	GP-Fast	% of Opt.
1	8	7	72.6	71.5	98.5
2	9	7	65.3	59.2	90.7
3	9	6	50.9	50.7	99.5
4	11	9	70.5	60.0	85.1

Table 3.5: Observation Activity and Throughput Scheduling Results for 6-Sat Sim Case with 60° Obs Elevation Mask and 10° Downlink Elevation Mask

Test Case	Number of Obs Scheduled		Total Througput (Gb)		
	GP-Optimal	GP-Fast	GP-Optimal	GP-Fast	% of Opt.
1	6	6	29.9	29.1	97.2
2	7	7	36.8	34.4	93.6
3	4	4	26.5	26.0	98.2
4	5	5	27.4	27.4	100

delivering data volume from a previous observation. This occurs as a result of the GP-Fast Activity Scheduling data storage constraints (inequalities 3.38 and 3.39), which require the data volume for a given data route to be considered as present on a satellite for the entire original duration of any activity along the data route (not the scheduled duration). This constraint is not present in GP-Optimal, which is thus able to schedule both the observation and the downlink in such a case. This overlap loss effect reduces greatly as the size of the windows are shortened, as evidenced by the results from shorter windows in table 3.5.

3.5.2 GP Throughput Sensitivity Comparison

Similar to the results in Zhou *et al.*, the throughput performance of GP-Fast was also calculated with changing planning window size, energy collection rate, and onboard data storage (buffer) size [7]. Results from these runs are shown in figs. 3-11 to 3-13.

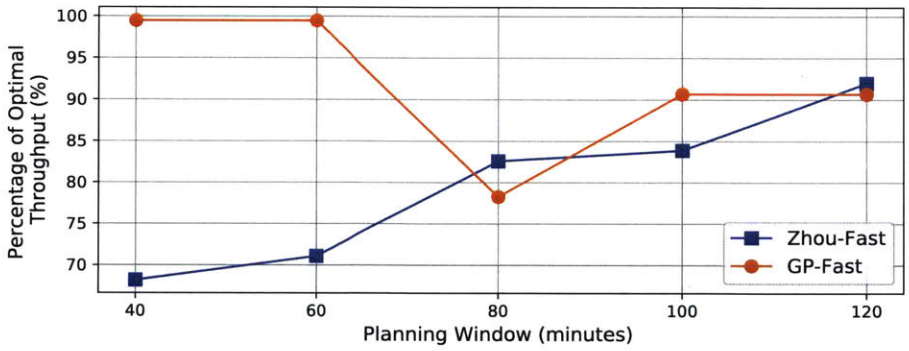


Figure 3-11: Variation of throughput performance for the GP-Fast and Zhou-Fast (ACG) algorithms with changing planning window size. Run with 6-Sat sim case with parameters specified in appendix A.1. 30° obs elevation mask and 0° downlink elevation mask used. The large dip in GP-Fast performance at 80 minutes is due to overlap of observation and downlink windows.

In all cases, the 30° obs elevation mask and 0° downlink elevation mask were used. We see again from the figures that GP-Fast performance is close to Zhou-Fast performance (about 90% of optimal throughput) in most cases. The dip in performance

[7]Results were obtained with commit c7fd7faa0549d26f031bddbfe238cca3a7fede4a of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

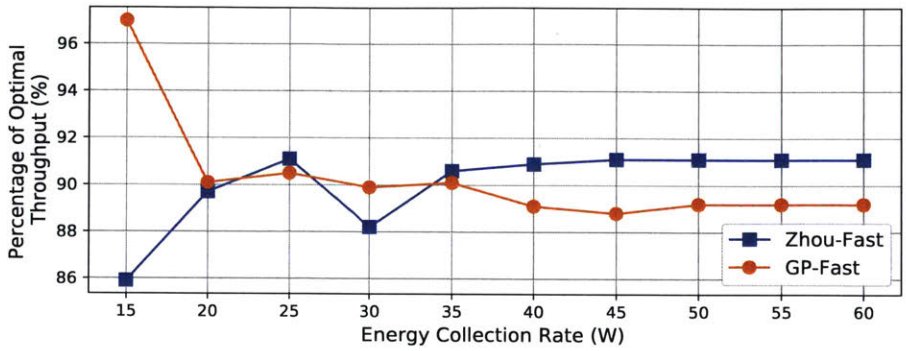


Figure 3-12: Variation of throughput performance for the GP-Fast and Zhou-Fast (ACG) algorithms with changing energy collection rate. Same simulation parameters used as for fig. 3-11.

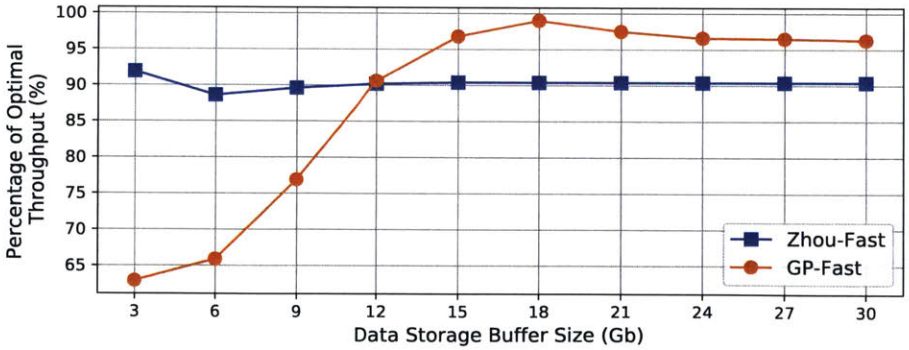


Figure 3-13: Variation of throughput performance for the GP-Fast and Zhou-Fast (ACG) algorithms with changing data storage buffer size. Same simulation parameters used as for fig. 3-11.

at the 80 minute planning window in 3-11 is due to the planning horizon stretching to include a new observation window that GP-Fast is incapable of scheduling due to the previously described difficulty with observation-downlink overlaps. Once the planning window stretches to 100 and 120 minutes, the un-schedulable observation represents less of the total potential throughput. A similar situation arises for GP-Fast with low data storage buffer size (3 to 9 Gb, fig. 3-13), as there is significantly less data volume overall that gets collected by the constellation so small, downlink-overlapping observation windows represent more of the overall throughput amount. This problem is rectified as the buffer size grows. No overlap concerns arise with low energy collection rate, however.

Additional runtime comparisons to the Zhou *et al.* results are presented in section 6.1.1.

3.6 GP Schedule Quality Sensitivity

We examine the sensitivity of GP-Fast output schedule quality to varying objective function weighting factors, *i.e.* the w_i terms discussed in section 3.3.2. First we look at the effect of varying observation data throughput and latency weightings, then the effect of varying throughput and energy margin weightings, and finally the effect of varying the weighting for existing routes.

Sensitivity results were examined with the 6-Sat simulation case, with parameters values summarized in appendix A.1, except for different parameter values specifically mentioned here and for each sensitivity analysis. To make the 6-Sat scenario more resource intensive, the observation execution data volume requirement was set to 1000 Mb, the "Xlnk, Tx" and "Xlnk, Rx" power consumption were set to 30 W, and solar charging was set to a constant 15 W in sunlight. These settings mean that the satellites are more energy constrained than the base scenario, and require more crosslink time usage to achieve low latency for observation data. The GP-Fast algorithm is run multiple times with different objective function weightings for each sensitivity analysis.

Varying Throughput and Latency Weightings

Figure 3-14 shows the variation of total observation data throughput (also referred to as "DV") and observation initial latency with varying throughput and latency weightings [8]. Here, total observation data throughput is presented as a percentage of possible throughput, where "possible" is determined by summing the data volumes for every observation within the 95 minute planning window. This performance is not necessarily achievable in reality due to downlink conflicts, but provides an upper bound. The latency weighting (w_2) is varied from 0 to 10 in steps of 1, and the throughput weighting (w_1) is varied from 10 to 0 in the opposite direction. The other weights, w_3 and w_4, are kept at 0 to isolate the effects of changing prioritization of throughput and latency. Table 3.6 shows the values for every weighting ("weight") combination. P_{25}, P_{50}, and P_{75} indicate the 25th percentile, median, and the 75th percentile values, respectively.

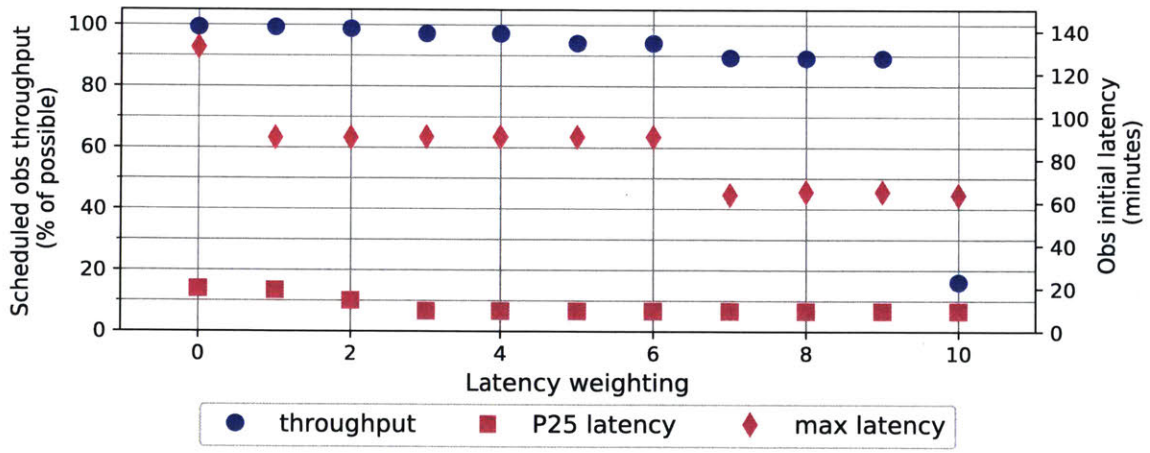


Figure 3-14: Variation of total observation throughput and initial latency performance with changing throughput (w_1) and latency (w_2) weightings (only w_2 weighting is indicated here). 25th percentile and maximum latency are presented here because they exhibited the most change. All obs window capacities are summed to determine "possible" value.

We see that 100% of possible throughput gets scheduled when the throughput

[8]Results were obtained with commit a24a4657bc3f2957bdfd2673c7b302f8753ca1ca of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

weighting is 10, and all other weights are 0. This drops off slowly to 89.1% at $w_2 = 9$ ($w_1 = 1$) and then precipitously drops to 16.1% at $w_2 = 10$, when there is no objective reward given for observation throughput. Latency performance improves (the value reduces) as w_2 increases, but not in the same steady fashion as throughput performance. There are discrete steps in performance, which results from the fact that these high-level metrics don't capture all of the detail of the latency value distribution. As the latency weight increases, individual observation windows are assigned to lower-latency data routes, which may not be reflected here. Overall, we see a steady, slow change in performance in the expected direction as the weightings change. The P_{25} and maximum values for latency are shown because the other latency metrics did not change much, as seen in table 3.6.

Table 3.6: Total Observation Throughput (DV) and Initial Observation Latency Metrics with Varying w_1 and w_2 Values

DV Weight, w_1 **Latency Weight, w_2**	10 0	9 1	8 2	7 3	6 4	5 5	4 6	3 7	2 8	1 9	0 10
DV:											
% of Possible	99.3	99.3	98.7	97.2	97.2	94.1	94.1	89.4	89.1	89.1	16.1
Latency (mins):											
Minimum	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8
P_{25}	19.9	19.2	14.3	9.4	9.4	9.4	9.4	9.4	9.4	9.4	9.4
P_{50}	40.8	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0
P_{75}	108	40.6	40.6	40.6	40.6	40.6	40.6	40.6	40.6	40.6	40.4
Maximum	132	90.2	90.2	90.6	90.6	90.6	90.6	63.8	65.2	65.2	63.8

The shallowness of the latency curve in fig. 3-14 illustrates the fact that for this particular scenario, the achievement of good throughput is not impacted to a large degree by achieving good latency performance. This makes sense in the simulation context. In practice, the minimum observation data volume requirement (1000 Mb) is low enough such that a single data route through a single crosslink window originating on the observation data collector satellite is sufficient to deliver the data volume. This means that a single satellite does not have to use a large number of crosslinks (and

thus a large amount of energy) to achieve low latency, and it remains able to downlink the remainder of the data later. If the minimum data volume requirement were set larger, generally a single data route would not be able to meet this requirement. This situation does work with the latency constraints as currently formulated in GP-Fast (inequalities 3.53). It would be useful in future to extend these constraints to allow latency reward factors to be determined from multiple data routes.

Varying Throughput and Energy Margin Weightings

Figure 3-15 shows the variation of total observation data throughput (also, "DV") and median satellite-average energy margin with varying throughput (w_1) and energy margin (w_3) weightings [9]. The weightings are varied at the same time, as shown in table 3.7. The other weightings, w_2 and w_4, are fixed at zero. Again, we see a gradual decline in throughput performance as its weight decreases. Energy margin increases slightly more over all weighting values, ranging from 47% to 79.7% at the highest weight. It is unable to completely bottom out at 0% when its weighting is zero because the satellites still must keep their energy storage within the minimum and maximum balance, which requires recharging during non-eclipse periods. The middle point with equal weightings for each appears to performe fairly well for each case, with about 97% for throughput and 70% for energy margin.

Varying Existing Routes Weighting

For this analysis, first GP-Fast was run with a short time horizon (210 minutes for observations, crosslinks, and 840 minutes for downlinks) for equal objective function weightings, and then run again for a longer time horizon (360 minutes for observations, crosslinks, and 840 minutes for downlinks) while including the existing routes from the first run [10]. This gave GP-Fast new observation and crosslink windows to use for routing, thus incentivizing schedule changes, but also gave it an incumbent schedule.

[9]Results were obtained with commit a24a4657bc3f2957bdfd2673c7b302f8753ca1ca of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

[10]Results were obtained with commit 4be1b4cb2f83336b7cf8981e804dc30b28e8bdd5 of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

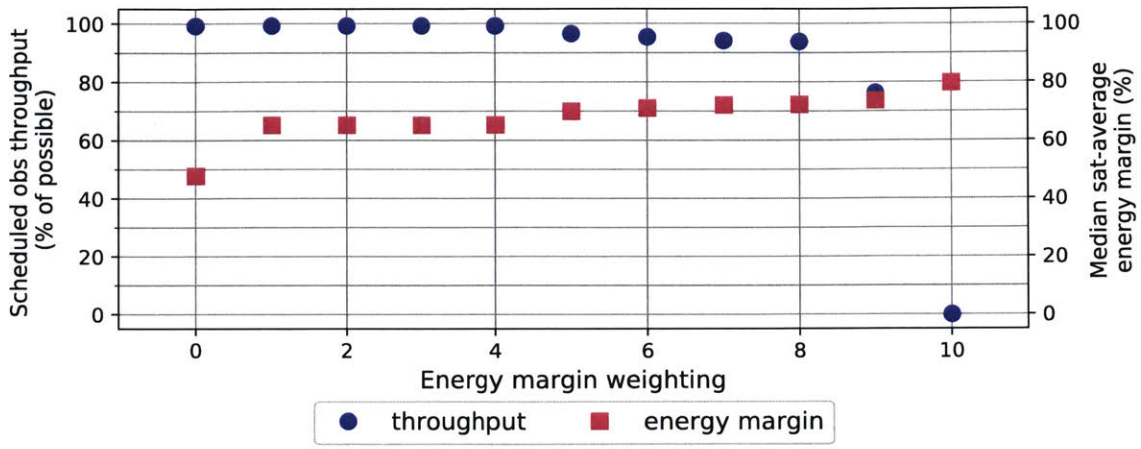


Figure 3-15: Variation of total observation throughput and energy margin performance with changing throughput (w_1) and energy margin (w_3) weightings (only w_3 weighting is indicated here). All obs window capacities are summed to determine "possible" value.

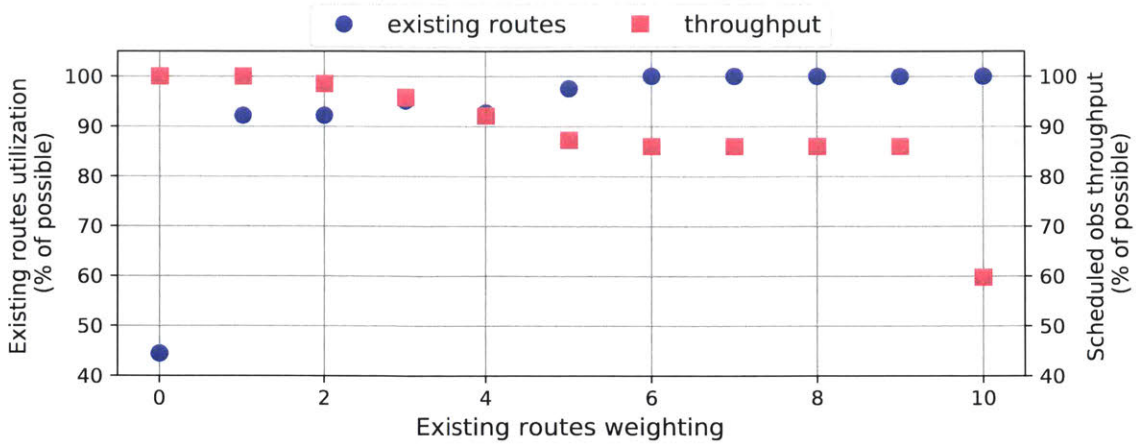
Table 3.7: Total Observation Throughput (DV) and Satellite-Average Energy Margin Metrics with Varying w_1 and w_3 Values

DV Weight, w_1	10	9	8	7	6	5	4	3	2	1	0
Energy Weight, w_3	0	1	2	3	4	5	6	7	8	9	10
DV:											
% of Possible	99.3	99.3	99.3	99.3	99.3	96.6	95.4	94.2	93.9	76.3	0.0
ES Margin (%):											
Minimum	37.3	54.8	54.7	54.9	54.7	55.1	55.1	55.1	55.4	71.1	78.7
Median	47.7	65.2	65.2	65.2	65.3	69.9	71.0	72.0	72.2	73.7	79.7
Maximum	50.0	77.5	77.5	77.5	77.5	77.5	78.6	80.2	80.0	80.3	81.1

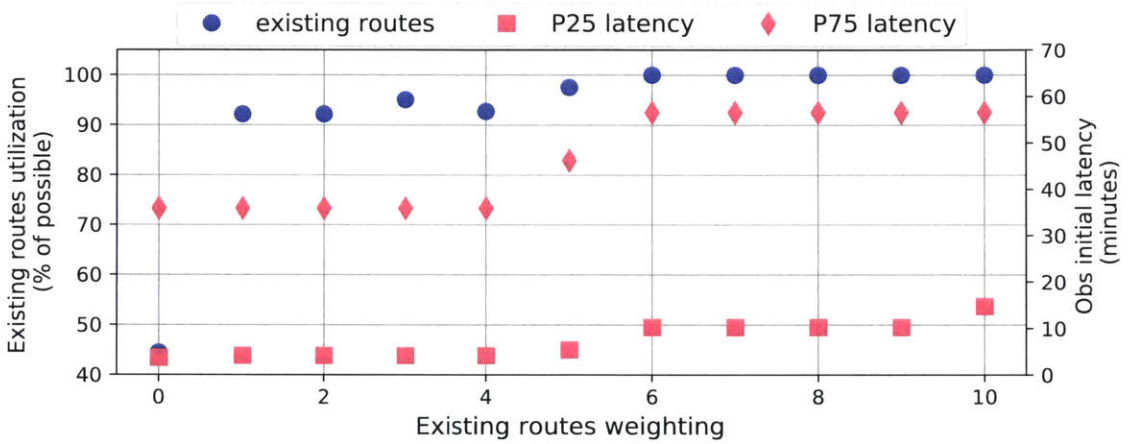
The route utilization numbers from the first run were summed together to determinate a "possible" value for existing route utilization, and this is compared to the existing route utilization actually scheduled on the second run.

Figure 3-16 shows the variation of throughput, latency, and energy margin metrics versus existing routes utilization with varying existing routes weight (w_4) and varying, but equal, other ($w_1 = w_2 = w_3$) weightings. The other weightings are kept equal to equally reward these metrics, while trading off against choosing existing routes. The weightings are varied at the same time, as shown in the results summary in table 3.8.

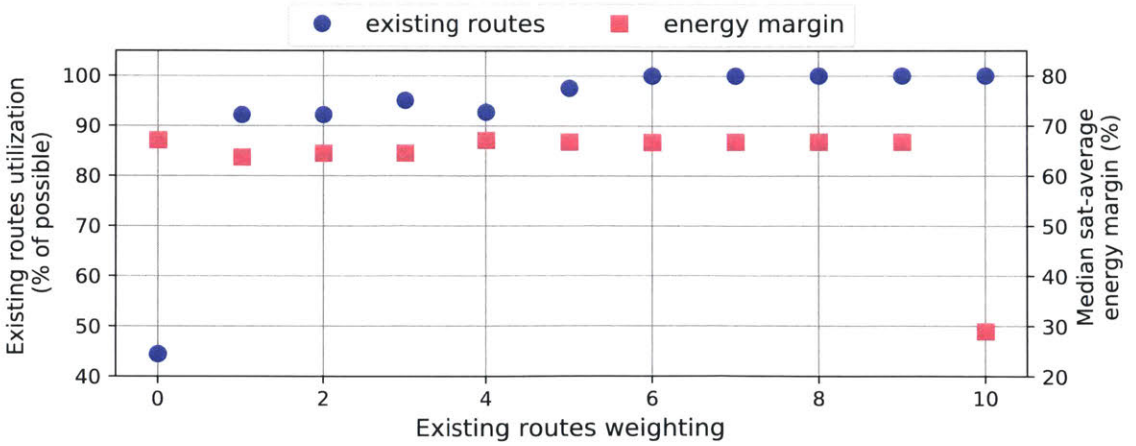
We see that as the existing routes weight is increased relative to the other weights, more of the possible existing routes utilization get scheduled. When the majority of objective reward is concentrated in existing routes ($w_4 = 6$), 100% of the existing utilization gets scheduled, as expected. The low value for existing routes utilization at $w_4 = 0$, 44.5%, shows that without an explicit reward for these routes, GP-Fast is not guaranteed to pick them. In this particular case, because the global planner was run from the same initial time (with a different time horizon), Route Selection constructs essentially duplicate routes to the existing ones, and the MILP solver in GP-Fast Activity Scheduling may choose the duplicate copies depending what path the solver takes through the search space. These findings show that an existing routes term in the objective function is effective at preventing full schedule changes every time the Global Planner runs.



(a) Throughput versus existing routes performance



(b) Latency versus existing routes performance



(c) Energy margin versus existing routes performance

Figure 3-16: Variation of throughput, latency, and energy margin metrics with changing existing routes weighting.

Table 3.8: Existing Routes Utilization, Observation Data Throughput (DV), Observation Initial Latency, And Satellite-average Energy Margin Metrics (ES Margin) with Varying w_1, w_2, w_3, and w_4 Values

Other Weights, $w_1 = w_2 = w_3$	3.33	3.00	2.67	2.33	2.00	1.67	1.33	1.00	0.67	0.33	0
Existing Routes Weight, w_4	0	1	2	3	4	5	6	7	8	9	10
Existing Utilization:											
% of Possible DV:	44.5	92.2	92.2	95.1	92.7	97.5	100	100	100	100	100
% of Possible Latency (mins):	100	100	98.5	95.8	92.1	87.3	86.0	86.0	86.0	86.0	59.8
Minimum	2.2	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	10.3
P_{25}	3.7	4.2	4.2	4.2	4.2	5.4	10.3	10.3	10.3	10.3	14.8
P_{50}	14.8	14.8	14.8	14.8	14.8	14.9	15.1	15.1	15.1	15.1	15.1
P_{75}	35.9	35.9	35.9	35.9	35.9	46.2	56.6	56.6	56.6	56.6	56.6
Maximum	84.9	84.9	84.9	84.9	84.9	84.9	84.9	84.9	84.9	84.9	84.7
ES Margin (%):											
Minimum	44.1	44.1	44.1	44.1	44.1	44.1	44.1	44.1	44.1	44.1	24.4
Median	67.1	63.7	64.5	64.5	67.1	66.8	66.7	66.8	66.8	66.8	29.0
Maximum	71.1	68.2	68.2	69.6	69.5	69.6	69.6	69.6	69.6	69.7	30.3

Chapter 4

Local Planner

We first discuss the purpose of the Local Planner (LP) algorithm in section 4.1.1, then the algorithm formulation in section 4.2, and finally present validation results for the LP in section 4.3.

4.1 Local Planner Overview

4.1.1 Purpose

The LP provides an onboard re-planning capability for satellites. It is intended for use in response to changing conditions onboard the satellite for which a quick plan or schedule change is needed. It is not effective to solely rely on GP replans because of the relative inaccessibility of the GP through the constellation network. Use cases for the LP include:

1. Handling routing of urgent, "injected" observations to downlinks;

2. Adjusting timing for the satellite's scheduled activity windows due to differences in current resource state from the GP's expectation at original scheduling time;

3. Modification of data routes to deal with one-off cases of unsuccessfully executed activity windows ;

4. Response to long-duration faults, including continuous unresponsiveness or loss of a satellite.

Only the first use case is demonstrated here, per the research contributions enumerated in section 1.3.2. The current LP algorithm implementation could handle the other use cases as well, though due to its limited ability to coordinate with other satellites would likely significantly reduce global schedule quality.

Injected observations

In this work, we are particularly interested in new observation events that arise onboard in an unexpected manner, which will be referred to as "injected observations" or "urgent observations". An example of this might be if a new observation window needs to be executed because a scanning imager on the satllite has identified a prime opportunity for an opportunistic observation of a target on the ground. This type of spontaneously arising event generally merits quick response; in this case, we want to route the data from this injected observation to ground as quickly as possible to provide maximum situational awareness for ground operators. The LP provides the satellite the capability to make decisions about what other data to de-prioritize in order to deliver this new, urgent data to ground effectively. An example case for unsuccessfully executed activity windows would be when one satellite attempts to transmit to another satellite during a crosslink but the receive end doesn't answer, perhaps due to a mis-executed schedule, or a module in onboard software hanging. In this case, we might send the data that was originally intended to travel over that crosslink through another crosslink.

Motivation

The LP is intended to make quick data routing decisions onboard based on the latest updated state for a given satellite. Its decision making is purposefully restricted to the scope of a single satellite's activity execution and the data to route through those activities, to allow it to execute much faster than the Global Planner. This reduces

the need to run the GP frequently, because we can trust that the LP will be able to respond to quickly new circumstances effectively. In the same way, it allows us to give the GP more time to come up with better quality schedule solutions for the constellation as a whole.

The LP is tightly coupled with the GP, relying on the data routing decisions made by the GP as a template for making its own routing decisions. Other work has featured more decision making and automony in the onboard planning process. Van der Horst *et al.* discuss auction-based algorithms for satellite-cluster task allocation [110] in which satellites bid for the tasks they want to perform, with no guidance from a centralized algorithm. Damiani *et al.* discuss a hierarchical systems with high-level goal distribution and onboard goal refinement into detailed tasks using Dynamic Programming [25]. In this case, the onboard planning process has more responsibility for choosing how to split up and schedule timing for tasks. Zheng *et al.* discuss a system where one "mother" satellite performs general planning for a set of "daughter" satellites, which then execute plans and report updates to their health status to the mother satellite [123]. A Genetic Algorithm is used on the mother satellites for planning, and the system can respond to emerging changes in real-time. Morgan *et al.* investigated decentralized swarm protocols for distributed guidance and control of satellites, with each satellite performing simple actions based on its postion relative to its direct neighboring satellites [84].

These methods give the satellites themselves significant autonomy in making decisions about tasking and replanning. The LP algorithm in this work does not feature as much autonomy, largely because it does not include a mechanism for coordinating plan changes across satellites. For this work, we focus on allowing the LP algorithm to make small changes to data routes already planned by the GP, to increase the responsiveness of the constellation.

4.1.2 Local Planner Algorithm Description

Concept of Operations

The basic principles of the LP algorithm are illustrated in fig. 4-1. The LP operates on a planning window from the current time on the satellite to a configurable time horizon, t_h. It it takes as an input the list of currently planned DRs and essentially breaks them apart in the middle, turning them into inflows and outflows that bring data to or send data out from the LP's satellite, respectively. Currently stored data on the satellite and any newly injected observations are are added as inflows. Then the LP finds an optimal matching of inflows to outflows given its objective function. Figure 4-1 illustrates how the matching works. In this case there is a pre-existing observation window, "obs1", that was scheduled by the GP to be routed through downlink "dlnk". An inflow crosslink "xlnk1" was scheduled to outflow through "xlnk2". Existing stored data was also scheduled to be routed. However, the LP determines that the injected observation "obs2" should replace most or all of the data volume from xlnk1 being routed through xlnk2, in order to achieve a low latency downlink on a nearby neighbor satellite.

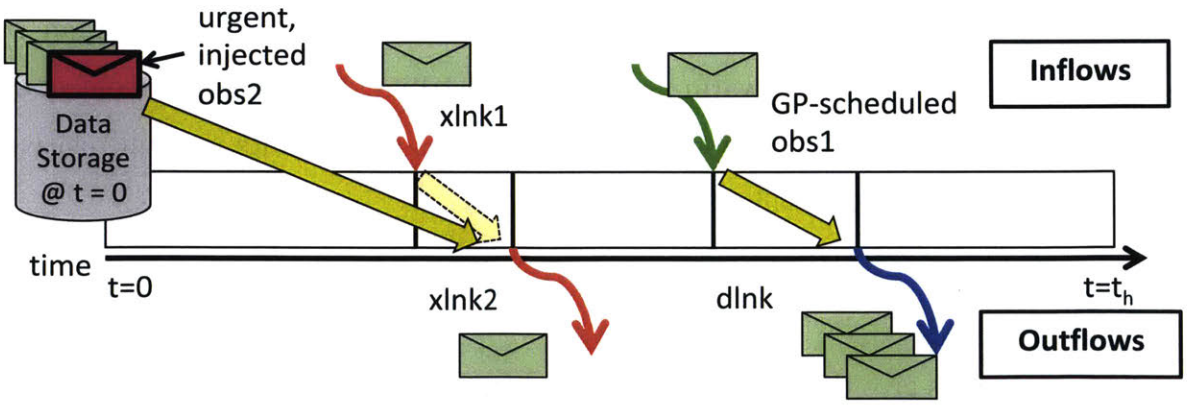


Figure 4-1: Illustration of Local Planner's inflow to outflow matching procedure, over a planning window from 0 to t_h. In this case, one GP-planned data route is preserved (obs1 to dlnk) and another (xlnk1 to xlnk2) is replaced with a route for injected data collected before LP execution (obs2 to xlnk2).

Planning Window

The LP operates on a defined planning window, in a similar manner to the GP. Unlike the GP, which can have different planning window settings for each type of activity, only one planning window is specified for the LP. It can be set as large as desired to search for potential outflows. Usually it is set to the same length as the crosslink planning window for the GP, in order to examine all routes created by the GP containing crosslinks (which are generally the lower latency route options).

Processing Input Data Routes

The LP breaks apart input data routes (DRs) in the following manner for a satellite s: for a planned DR r consisting of activity windows $w_1, w_2, ..., w_N$, any window w_e (e for "entering") that has s as a receiver is deemed the source of an "inflow" and any window w_g (g for "going") that has s as a transmitter is deemed the source of an "outflow". Then the DR is (temporarily, for the LP's planning process) split in two, with all windows up to and including w_e constituting an inflow "data route segment", e, and all windows including and subsequent to w_g constituting an outflow data route segment, g. If a DR only has scheduled windows after the end of the time horizon, that DR is not included in the inflows and outflows. If a DR is in the middle of execution from s's perspective and is currently storing data on s, then the stored data constitutes another inflow with an inflow data route segment equal to the windows that have been executed thus far for the data to arrive on s. Any DRs which have an inflow within the current planning window for the LP but an outflow past the planning window are not considered.

Synthesis of Output Data Routes

The goal of the LP's scheduling algorithm is to perform an optimal matching of inflow data volumes to outflow data volumes, per a set of objective terms (detailed in section 4.2). The LP may split inflows over multiple outflows, and vice versa. Each possible matching of inflow to outflow is termed a unified flow, u. A matching is only

possible when the outflow completely temporally follows the inflow, *i.e.* w_g follows w_e. The resulting unified flows either represent the original routes that the LP broke apart (possibly with a different, lower data volume than previously), or entirely new DRs. In the case of an entirely new DR, the activity windows from the inflow are pre-pended to the windows from the outflow, creating a new DR object that meets the requirements of the Data Route definition (see section 3.1.4).

The LP calculates route utilization fractions for all the resulting scheduled DRs. For existing, GP-created DRs, these utilizations are calculated as LP-scheduled data volume for the route divided by the original data volume for the route determined by the GP. New, LP-created DRs have a utilization of 1.0 by definition. This tracking of utilization fractions, as opposed to modifying the route itself, is a simulation bookkeeping convenience; it enables the routes created by the planners to be tracked as unique objects across multiple planner executions, as opposed to creating entirely new objects every time the LP runs. Note that if the LP chooses to create a new match of inflow to an outflow, it must create a new DR object to track this because a DR has a specified observation window at the beginning, and therefore routes a unique slice of data through the constellation.

The use of existing, GP-created DRs as a planning resource simplifies the formulation of the LP. They represent a reservation of throughput data volume that all satellites along the route have agreed to execute. Thus a satellite may make decisions about what data to send through its outflow DRs with complete autonomy, because it knows that the other satellites will honor the execution of those DRs (in the nominal case). This can be thought of as a "single fault tolerance" to plan changes: if only one satellite decides to make changes in data sent along a DR, then as long as other satellites' LPs are not making plan changes as well, the first satellite can be confident the data will be delivered.

LP Limitations

The current version of the LP is inherently limited to scheduling throughput that has already been reserved for satellites by the GP. It may not introduce new activities

or lengthen scheduled activity times from the GP's plans. This prevents the LP from introducing new time constraints and schedule obligations that were not already present, and avoids the need for tight planning coupling between LP instances running on multiple satellites.

If LP could introduce new schedule obligations, some form of consensus would be needed across the constellation. For example, if satellite s_1 wants to take more downlink time with a ground station that was already scheduled to talk with another satellite s_2, the temporal overlap would need to be de-conflicted. Also if a satellite wants to introduce a new crosslink, it would need agreement from the crosslink partner satellites to execute the activity. For the current work, it is sufficient for the LP to take data volume from GP-created DRs for the purpose of routing injected observations. The implementation of a mechanism for consensus on plan changes is suggested for future work (see section 7.2).

Note that the LP gives no guarantee of maintaining global schedule optimality (from the perspective of the GP, given that all information about satellite state and injected observations is available to it) with its onboard decision-making. This could potentially lead to large reductions in schedule quality, however the LP includes tunable objective function terms that favor maintaining existing routes.

4.1.3 LP Solver Algorithm Selection

Examples of other work in onboard planning feature an iterative repair or local search algorithm for making changes to existing plans [15], a Genetic Algorithm for onboard replanning to deal with emerging changes in real-time [123], decentralized swarm protocols for distributed coordination [84], auction-based algorithms for satellite-cluster task allocation [110], and hierarchical systems with high-level goal distribution and onboard goal refinement into tasks using Dynamic Programming [25]. These approaches add scalability and decentralized coordination to onboard decision-making.

The LP problem presented here only considers decision-making from the small-scale perspective of a single satellite, and does not need to provide consensus across satellites. The size of the LP problem is much smaller compared to the GP, and

also uses GP-provided plans for *a priori* adherence to schedule constraints on other satellites. For these reasons, problem scalability was not a large factor in selection of the underlying solver algorithm here.

Due to the linear satellite operations model and the need for disjunctive constraints, a Mixed-Integer Linear Program (MILP) formulation is a natural approach. It gives a guarantee of an optimal solution as long as sufficient time is provided for execution, versus heuristic algorithms that may not find it. In this case, the limited problem complexity means that there are not significant scalability concerns, and the LP can be directly formulated and solved as a MILP (as opposed to GP-Fast, which features a complexity-reducing stage before MILP problem solution). Similar to the GP, the LP objective function avoids inclusion of nonlinear objectives (*e.g.* average data delivery latency), by using linear terms as a proxy for such metrics.

4.2 LP Formulation

The following sections explain the decision variables, constraints, and objective function for the MILP formulation used by the LP. [1]

Decision Variables

The decision variables for the LP are shown in definitions 4.1 to 4.7. Variable x_p is the utilization fraction for a given inflow or outflow, known as a "partial flow", p. In a similar manner to route and activity window utilization fractions, this variable may be multiplied by the capacity of an inflow $e \in E$ ("entering") or outflow $g \in G$ ("going") to determine the utilized data volume for that flow. v_u is the data volume utilization for a unified flow $u \in U$. A u is constructed for every feasible unified flow, *i.e.* a feasible inflow-outflow matching. x_a is the utilization for activity a. I_p is an indicator variable for p, indicating that the partial flow meets minimum data volume requirements. I_u indicates that u meets minimum data volume requirements.

[1] The discussion here reflects commit cf3ba21fe66ea2ccc991d26928bb1e1579ae7463 of the Local Planner code in the CIRCINUS Constellation Simulator repository at `https://github.mit.edu/star-lab/circinus_sim`

I_a indicates that activity a has a utilization fraction greater than 0. f_o is the latency reward factor for $o \in O^{injected}$, where $O^{injected}$ is the set of injected observations. This variable is used to incentivize the choice of low latency unified flows for o.

$$0 \leq x_p \leq 1 \quad \forall p \in E \cup G \tag{4.1}$$

$$v_u \geq 0 \quad \forall u \in U \tag{4.2}$$

$$0 \leq x_a \leq 1 \quad \forall a \in A \tag{4.3}$$

$$I_p \in \{0, 1\} \quad \forall p \in E \cup G \tag{4.4}$$

$$I_u \in \{0, 1\} \quad \forall u \in U \tag{4.5}$$

$$I_a \in \{0, 1\} \quad \forall a \in A \tag{4.6}$$

$$f_o \geq 0 \quad \forall o \in O^{injected} \tag{4.7}$$

$$\tag{4.8}$$

Constraints

The first set of constraints, inequalities 4.9 to 4.11, connect data volume utilizations for partial flows to the data volume used for activities. Ineq. 4.9 constrains the sum of the data volumes for all inflows e passing through a, the set E_a, to be less than or equal to the activity's data volume. Ineq. 4.10 is similar, but operates on outflows through a, G_a. Terms c_e, c_g, and c_a represent the throughput capacities of inflows, outflows, and activities, respectively. In ineq. 4.11, the activity indicator I_a is set high if any of the the data volume/time of a is utilized.

$$c_a x_a - \sum_{e \in E_a} (c_e x_e) \geq 0 \quad \forall a \in A \tag{4.9}$$

$$c_a x_a - \sum_{g \in G_a} (c_g x_g) \geq 0 \quad \forall a \in A \tag{4.10}$$

$$I_a \geq x_a \quad \forall a \in A \tag{4.11}$$

The next set of constraints, ineqs. 4.12 to 4.15, connect partial flows to unified flows. U_e is the set of all unified flows created for a given inflow (*i.e.* one for every feasible outflow for that inflow). U_g is similar, the set of unified flows created for a given outflow. Ineq. 4.12 constrains the distribution of data volume from a given inflow to all of its matching unified flows, making them sum to the inflow's utilized data volume. Ineq. 4.13 is similar, but for outflows. Ineq. 4.14 only allows an indicator to go high if the unified flow uses at least $MinDV$. Ineq. 4.15 allows the indicator variable for inflow e to go high only if at least one unified flow including e is used.

$$\sum_{u \in U_e} (v_u) = c_e x_e \quad \forall e \in E \tag{4.12}$$

$$\sum_{u \in U_g} (v_u) = c_g x_g \quad \forall g \in G \tag{4.13}$$

$$v_u \geq MinDV \cdot I_u \quad \forall u \in U \tag{4.14}$$

$$\sum_{u \in U_e} (I_u) \geq \cdot I_e \quad \forall e \in E \tag{4.15}$$

The next constraint is similar to the GP constraints of ineq. 3.19 and ineq. 3.46. It enforces that an activity must meet a minimum time requirement in order for it to be allowed to execute to any degree at all. Note that activity time overlap constraints are not enforced in the LP because it is assumed that these constraints were already handled by the GP, and the LP is not able to extend the execution times of activities longer than what was scheduled by the GP. Though there is no upper bound on x_a enforced explicitly in the LP formulation, the final activity utilizations are determined based on the scheduled data volumes for the routes passing through them after the LP runs. Activity utilization is constrained to be the minimum value that achieves the throughput required for the data volume in all the routes. Because the LP is not able to introduce new throughput that was not already present in the inflow and outflow capacity terms, this means that activity utilizations can only decrease after a run of the LP.

$$\frac{c_a x_a}{\bar{r}_a} \geq \tau_a^{min} \cdot I_a \quad \forall a \in A \tag{4.16}$$

The final set of constraints limit latency reward factors in a similar manner to the GP (ineq. 3.25, ineq. 3.53). One equation is generated for every u_k in U_o, the sorted unified flows that contain observation o. U_o is sorted in decreasing latency for the unified flow, hence the k subscript enumerates the flows in decreasing latency for o.

$$\begin{aligned} f_o &\leq 0 + M^{lat} \cdot \sum_{k \in \{1,2,3,...|U_o|\}} (I_{u_k}) \\ f_o &\leq \phi_{u_1} + M^{lat} \cdot \sum_{k \in \{2,3,...|U_o|\}} (I_{u_k}) \\ &\cdots \\ f_o &\leq \phi_{u_{|U_o|}} + M^{lat} \cdot \sum_{k \in \{|U_o|\}} (I_{u_k}) \end{aligned} \tag{4.17}$$

The $0 \leq \phi_{u_k} \leq 1.0$ constant term is the reward factor for a given unified flow, determined by its latency for o. The latency of a unified flow is calculated in a similar manner to data routes, the absolute time (*e.g.* center time) of the downlink minus the absolute time of the observation. The reward factor for a given unified flow is calculated in a similar manner to equation 3.26. It is equal to 1.0 if u_k is the minimum latency unified flow for o (or has a smaller latency then a minimum cutoff), and decreases linearly with increasing latency. Inequalities 4.17 incentivize the choice of lower latency unified flows for a given observation.

Objectives

There are 6 objectives implemented in the current version of the LP algorithm, as detailed in equations 4.18 to 4.23. The first objective maximizes the total data volume scheduled for all unified flows. The second maximizes total data volume for existing unified flows, $U^{existing}$, which are the flows that preserve an existing match of an inflow with an outflow. The third maximizes total data volume for all injected unified flows,

$U^{injected}$. It may at first seem unexpected to have separate data volume summations for these three different items, but they allow us to put different weighting factors on each of the types. The fourth objective term again rewards existing routes, this time using the indicator variable. This allows putting a different preference on meeting the minimum data volume delivery requirements than for total data volume. So we could, say, allow some existing scheduled data volume to be sacrificed in order to make room for an injected observation, but still incentivize the LP to maintain minimum data volume for the existing routes. A similar rationale underlies objective term five, which rewards meeting minimum data volume requirements as opposed to total data volume for injected inflows. Finally, objective term 6 sums the latency reward terms for the injected observations.

$$\Omega_1 = \sum_{u \in U}(v_u) \tag{4.18}$$

$$\Omega_2 = \sum_{u \in U^{existing}}(v_u) \tag{4.19}$$

$$\Omega_3 = \sum_{u \in U^{injected}}(v_u) \tag{4.20}$$

$$\Omega_4 = \sum_{u \in U^{existing}}(I_u) \tag{4.21}$$

$$\Omega_5 = \sum_{e \in E^{injected}}(I_e) \tag{4.22}$$

$$\Omega_6 = \sum_{o \in O^{injected}}(f_o) \tag{4.23}$$

These objectives are each normalized to a maximum value of 1.0, weighted by individual weighting terms ($w_i \in \mathbb{R}$), and summed together to produce a composite objective score Ω:

$$\Omega = \sum_{1:6}(w_i \frac{\Omega_i}{\nu_i}) \tag{4.24}$$

The normalization factors ν_i are:

1. The total capacity available to be routed: $\nu_1 = \sum_{u \in U}(c_u) - \delta^{doubled}$
2. The total capacity of existing unified flows: $\nu_2 = \sum_{u \in U^{existing}}(c_u)$
3. The total capacity of injected inflows: $\nu_3 = \sum_{e \in E^{injected}}(c_e)$
4. The number of existing unified flows: $\nu_4 = \mid U^{existing} \mid$
5. The number of injected inflows: $\nu_5 = \mid E^{injected} \mid$
6. The number of injected observations: $\nu_6 = \mid O^{injected} \mid$

Where c_u is the potential throughput of unified flow u. This is determined as the minimum of the inflow and outflow capacities for u. For the ν_1 term, a subtraction $\delta^{doubled}$ is necessary for any "double-booked" inflow or outflow capacity, otherwise the term would make it seem like much more data volume was available than is actually the case.

Note that in general not all of these objective terms are used; some of them may be left with zero weight $w_i = 0$ to ignore the reward for a certain term. They are all described here for completeness. In the validation discussion in section 4.3, we examine two sets of weightings that prioritize routing of injected obs data volume in different ways.

4.3 Validation

For performance validation, the LP was run in multiple instances in the 6-Sat constellation sim case [2]. Details for this case can be found in appendix A.1. The constellation was simulated with the Constellation Simulator software (see chapter 5), with injected observations included. The LP was executed after injected observation data was collected by the respective satellite. Four instances of LP execution were chosen from the sim run as test cases to illustrate the decision-making performed by the algorithm. The test cases are detailed in table 4.1.

[2]Results were obtained with commit b9dbdee7155194bccad261c3a9f87ff0b99b5fc9 ("latency emphasis") and 2f6b7271e25a74d0226de9c07ce885cc34f278a7 ("DV emphasis") of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

Table 4.1: Injected Observations Used in Local Planner Test Cases

Test Case	Sim Run Time	Injected Obs Indices
1	2016-02-14T04:17:10	6
2	2016-02-14T05:03:10	28
3	2016-02-14T09:55:50	0, 27
4	2016-02-14T14:00:30	28, 34

In cases 1 and 2, a single injected observation was input to the LP. In the other two cases, two injected observations were present. The "sim run time" is the time within the simulation at which the LP was run. More details on the injected observation windows are found in appendix A.4. Note that observations 28 and 34 are separated by about nine hours; in this case, the earlier observation was scheduled for routing by the LP but some remaining data volume was left onboard until the later injected observation occurred.

The inputs to the LP in each test case are shown in table 4.2. The number of data routes is the number of pre-existing routes that the LP can divide into inflows and outflows. The total throughput is the sum of scheduled data volumes over all these routes. The minimum time to downlink is similar in function to route latency, but is calculated as the minimum over all routes of the difference between the current time at LP execution and the downlink time of the route. This serves as a "temporally localized" interpretation of the latency that a route can provide as an outflow, *i.e.* if an injected observation is routed through the outflow then this would be the remaining time to delivery for that data. We use this metric to verify that low-latency outflows are chosen for injected observations.

Two different sets of representative objective weights were investigated for LP execution:

1. "latency emphasis": $w_1 = 1$, $w_2 = 1$, $w_3 = 0$, $w_4 = 1$, $w_5 = 0$, $w_6 = 5$

2. "DV emphasis": $w_1 = 1$, $w_2 = 1$, $w_3 = 5$, $w_4 = 1$, $w_5 = 0$, $w_6 = 1$

Table 4.2: Data Routes Input to the Local Planner for Validation Test Cases

Test Case	Number of Data Routes	Total Throughput (Mb)	Minimum Time to Downlink (mins)
1	9	10800	47.9
2	2	4500	19.2
3	7	13900	42.3
4	7	14500	140.5

The first puts a heavy emphasis on low latency for injected observation execution ($w_6 = 5$), little emphasis on bulk data volume for injected observations ($w_5 = 0$), and tries to preserve existing routes and maximize both regular and injected data volume as much as possible ($w_1 = 1$, $w_2 = 1$, $w_4 = 1$). The second set has the same base preference for existing routes, but heavily favors routing bulk injected data volume ($w_3 = 5$) and puts relatively less emphasis on injected latency.

Results from running the LP with the first and second set of weights are shown in table 4.3 and table 4.4. Note a small amount of roundoff error is present in some cases (due to the allowance of slightly more data volume in inflow routes to avoid them being considered under the 100 Mb limit from numerical roundoff elsewhere in the sim).

Table 4.3: Data Routes Output from the Local Planner for Validation Test Cases, "latency emphasis" weightings

Test Case	Number of Data Routes		Total Througput (Mb)		Minimum Time to Downlink (mins)	
	Regular	Injected	Regular	Injected	Regular	Injected
1	8	1	10698	100	47.9	47.9
2	2	1	4400	100	19.2	19.2
3	7	2	13708	200	42.3	42.3
4	6	2	14300	200	140.5	140.5

For the first weightings, we see that the LP routes only as much data volume as is necessary to meet the minimum data volume requirements for downlink (100 Mb).

Table 4.4: Data Routes Output from the Local Planner for Validation Test Cases, "DV emphasis" weightings

Test Case	Number of Data Routes		Total Througput (Mb)		Minimum Time to Downlink (mins)	
	Regular	Injected	Regular	Injected	Regular	Injected
1	9	2	10505	295	47.9	47.9
2	2	1	4200	300	19.2	19.2
3	5	2	13402	500	42.3	42.3
4	7	3	14004	500	140.5	247.3

This amount is subtracted from the data volume that was present in the input data routes, validating that total data volume is conserved. We see that the minimum time to downlink for the injected observations matches the minimum times for the input routes in all test cases. This shows that although the LP steals data volume from the existing routes to route this unplanned, injected data volume, it does so gracefully, preserving throughput on existing routes generally.

The second set of weightings shows the effect of increased emphasis on injected data volume. For test cases 1 and 2, it routes roughly the whole 300 Mb collected from both observation windows. For test cases 3 and 4, less data volume was available from the injected observations initially - in both cases, 100 Mb had already been routed for one observation, so only 500 Mb is available to be routed. The LP successfully routes this remaining volume. This points to one limitation of the current LP implementation, that it does not explicitly account for the fact that a previously routed injected observation might have already delivered data volume to ground and met the goal for low latency. The minimum time the downlink for the injected observations matches that for the first set of weightings except for in test case 4, where the weighted reward for preserving existing routes and existing data volume overcame the relatively low weighted reward for injected latency.

Note that the LP ran in under 10 seconds in all trials on a machine, "Kalamity", with an Intel Core i7-7700K @ 4.20 GHz processor, 64 GB RAM, and Windows 10 64-bit. The Gurobi commercial MILP solver version 8.0.0 was used [56].

These results verify that the LP successfully matches inflows to outflows to produce plans for injected observations in real-time. Results from a full 24 hour simulation run with injected observations are presented in section 6.3.

Chapter 5

Constellation Simulator

A custom simulation software package, the Constellation Simulator or CSim, was developed to execute the GP and LP algorithms on small satellite constellations of arbitrary size. The motivating factors for developing this custom simulation include:

1. The need to assess the long-term effectiveness (after many receding-horizon planning cycles) of the GP and LP algorithms executing in closed-loop fashion within a relevant constellation environment;

2. The need for a tool that can simulate the small satellite model and custom planning information sharing mechanisms used in this thesis;

3. The desire to integrate the LP algorithm in a satellite simulation that can be expanded for future autonomy work;

4. The intent to release the software to the wider community as an open-source tool for constellation modeling and simulation

5.1 Architecture Overview

The high-level architecture of CSim is shown in fig. 5-1. There are three main types of self-contained planning entities or "agents" present within the sim: 1. Satellites, 2. Ground stations (GSs), and 3. The Ground Station Network (GSN). Each of these

agents executes its own internal decision-making logic about when to create new plans and how to execute plans. A description of plans and planning information (PI) is given in section 5.1.1, followed by a discussion of the roles played by simulation agents in section 5.1.2, and a discussion of communications in the simulator in section 5.1.3.

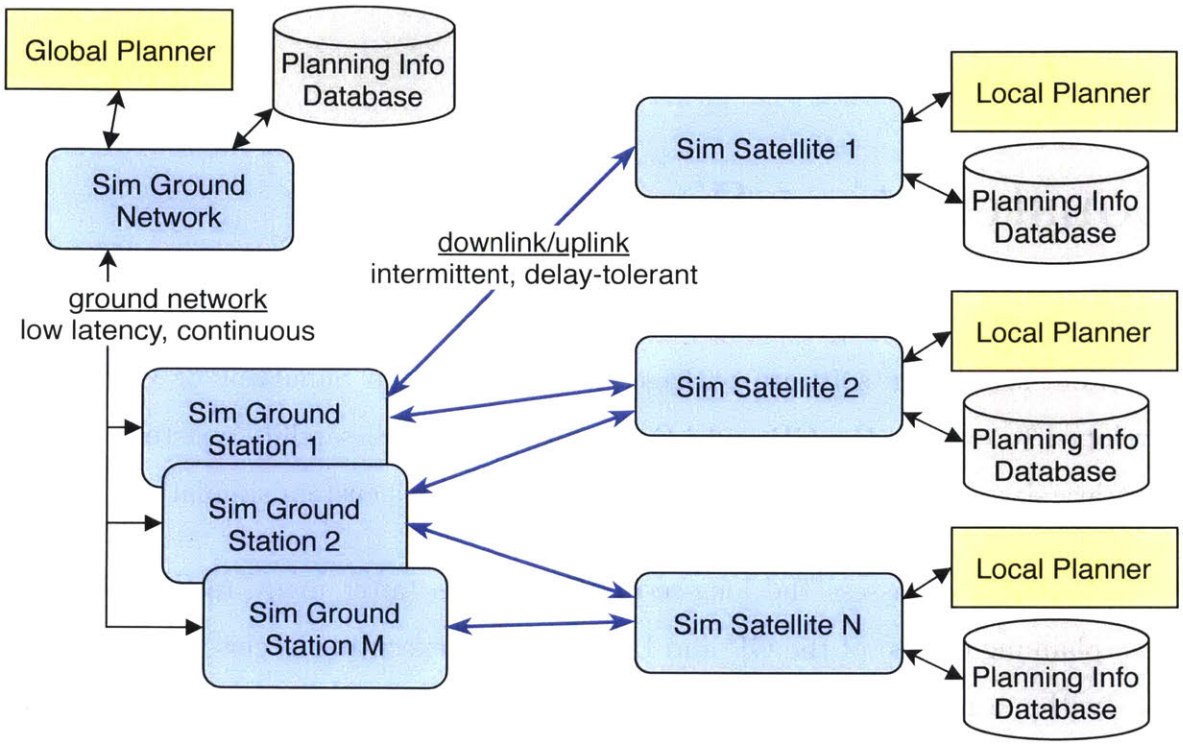


Figure 5-1: High-level overview of Constellation Simulator. Blue boxes are agents. Black arrows are the ground network and onboard satellite messaging, and blue arrows are down/uplinks.

5.1.1 Planning Information

In the context of CSim, "plan" and "planning information" (PI) are taken to mean all of the information used by agents for both deriving and executing activity schedules. This includes all of the items enumerated in table 5.1. As the simulation progresses, agents exchange PI with each other through communications links in the network. The agents store their own copy of PI within a local PI database ("planning info DB", PI DB), assumed to exist in Random Access Memory (RAM) or mass data storage onboard the satellites and in local computer facilities for the ground stations and

ground network.

Table 5.1: Planning Information used in Constellation Simulator

Item	Description
Sim Route Containers (SRC)	Contains a DR produced by either the GP or LP. Stores additional information needed for modifying or executing the DR, including scheduled utilization, creation time, last update time, and ID of the agent that originally created the DR
Satellite state	The latest known satellite energy storage and data storage state
TT&C update times	The last time that TT&C data was received from each other agent in the simulation

DRs are encapsulated within Sim Route Containers, which serve as a convenient means for tracking the data route along with any other information relevant to it. Because the GP and LP include existing DRs in their planning and scheduling process and can make changes to those DRs, SRCs track the latest utilization value and update time of the underlying DR. All SRCs are uniquely indexed across the entire simulation to ensure that plans are correctly tracked by all agents. When PI is exchanged between agents, each agent both adds any new SRCs from the other agent that it has not yet seen, and updates any SRCs that it already knows about if and only if the other agent has a more recent update time for the SRC. In this way, an agent is able to make changes to plans and then distribute the knowledge of that plan change through the constellation network.

Satellite state including energy and data storage is also contained in the PI DB. The GP uses the most up-to-date information in its PI DB about satellites' states whenever it runs. Finally, the satellites also track the last time they received a TT&C data update from another sim agent; *i.e.* the last time they "heard from" another agent. An agent, agent m, does not need to directly execute a link with another agent, agent n, in order to update this TT&C data; it is possible for these updates to propagate through other links executed by multiple agents before they reach agent m. As every agent along the way updates its own PI database, it will refresh its latest update time for agent n, thereby gradually propagating the TT&C data from agent

m to n. These updates are said to propagate by "flooding" through the constellation network.

5.1.2 Sim Agent Responsibilities

There are two types of agents that may create new plans: the GSN and satellites. The GSN is responsible for running the GP, while the satellites are each responsible for running their own instance of the LP. No internal state is kept within the GP and LP modules themselves; all such state is handled by the agents and preprocessed, if necessary, for input to the GP/LP. The GP/LP simply see an interface from which they accept their required inputs and to which they provide their required outputs. GSs do not create their own plans, rather plans are produced for them by the GSN. The general execution flow for planning from the perspective of the GP is shown in fig. 5-2. The GP is run in receding horizon fashion, starting from an initial planning window t_0 to t_h, and replanning later with a new planning window from $t_0 + \Delta t_{sim}$ to $t_h + \Delta t_{sim}$. After executing, the GP stores its output DRs in its PI DB, and exchanges the PI with other agents.

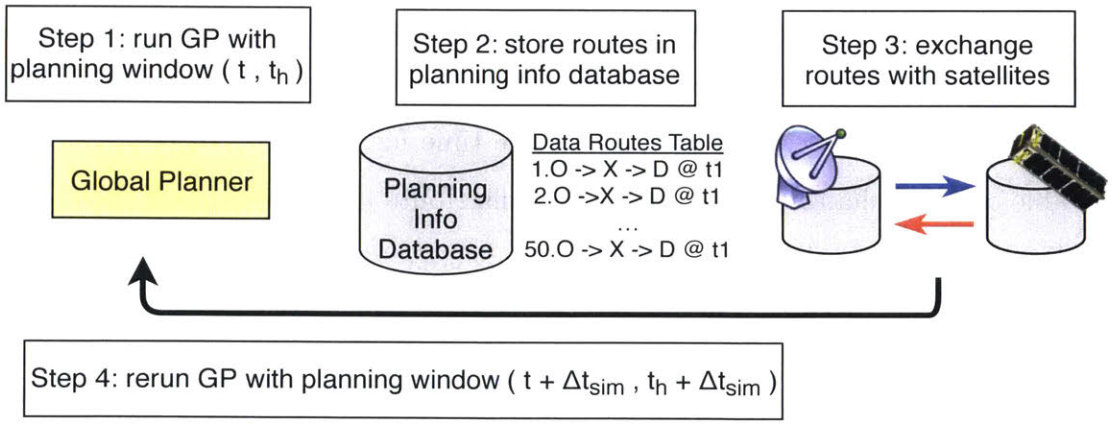


Figure 5-2: Illustration of receding horizon planning for Global Planner.

There are two types of agents that may execute plans: satellites and GSs. For satellites, this involves the execution of all of the activity windows present within the DRs of which the satellite has knowledge. GSs, are responsible for executing the downlink windows present within the plans produced by the GSN. Satellites collect

bulk observation data during observation activities, exchange data during crosslinks, and send data to ground during downlinks. The exact data volumes that are exchanged are specified by the DRs within the PI DB for the satellite.

5.1.3 Communications in the Simulation

The link between the GSN and GSs is modeled as a zero latency, continuous wide-area network connection. Through this network, all GSs are immediately able to receive updated PI from the GSN, and can immediately exchange between each other any updated PI received from satellites. This assumption of zero latency continuous availability in the ground network is acceptable because the GSs only need to exchange small amounts of PI periodically. The bulk observation data that arrives from downlinks can be collected in centralized data storage at a slower pace.

The downlinks and uplinks between ground stations and satellites take place during the downlink windows specified in scheduled DRs. We assume that PI can be sent from a GS to a satellite during any of these downlink windows, over a low data rate, low power consumption uplink. PI is also propagated between satellites when crosslinks are executed. This is referred to as intra-constellation PI propagation.

External Communications Backbone

The constellation could also use an external communications backbone for routing of both PI and bulk observation data. Such a capability could be provided, for example, by a two-way radio link with the Globalstar or Iridium low-Earth orbit satellite communications constellations, as demonstrated on previous CubeSat missions [112, 98]. For the Globalstar case, Voss *et al.* say that data can be sent from satellite to ground at about 0.26 cents per byte in the cheapest case. Sending 100 KB per satellite per day for a 30 satellite constellation for a full year, the total cost would be about $3000. Scaling up to 100 MB, this would approach $3 million. In this particular case, an external communications constellation could be economical for sending low-volume PI to ground, but less so for bulk data routing.

In the results for this work, we assume that PI is shared via an external constellation, in addition to over intra-constellation links. Immediately after execution of the GP, output data routes are sent from the GSN to the satellites, and state information is returned from the satellites to the GSN. This assumption is made due to a current limitation in the GP algorithm; the GP does not explicitly optimize routing for maximizing PI freshness across the constellation. If PI is only be propagated over infrequent down/uplinks and crosslinks planned for bulk data routing, satellites can fall out of sync with the GP by learning about planned data routes too late to actually execute them. Moreover, if satellites have PI from vastly different update times, conflicts or dropouts might occur for crosslink and downlink activities. The addition of a GP capability for optimizing PI distribution is an item for future work. Note that PI updates from the external communications backbone are assumed not to update TT&C data, meaning the TT&C freshness metrics measure the effects of propagation through intra-constellation links alone.

5.2 Global Planner and Local Planner Interaction

Figure 5-3 highlights the flow of PI updates for DRs between the GP and the LP instances running on the satellites. Here data routes are represented in their raw form rather than the Sim Route Container encapsulation used in the real sim, to make the illustration cleaner. The DRs (SRCs) can be thought of as entries in a table within the PI DB.

A notional example of DRs being propagated through the constellation and updated is shown. At time t_1, the GP creates an initial list of DRs, with a last update time tag of t_1 on all of them. These are distributed to the satellites, and in the general case where they are not propagated through a communications backbone but rather through the constellation network, they arrive at individual satellites with some delay. At time t_2, satellite Y has incorporated them into its PI DB and also run its LP, which created its own DR, LP Y.1. Note that some data routes (1 to 4) are old - they have already been fully executed, or at least all of the windows along the route are

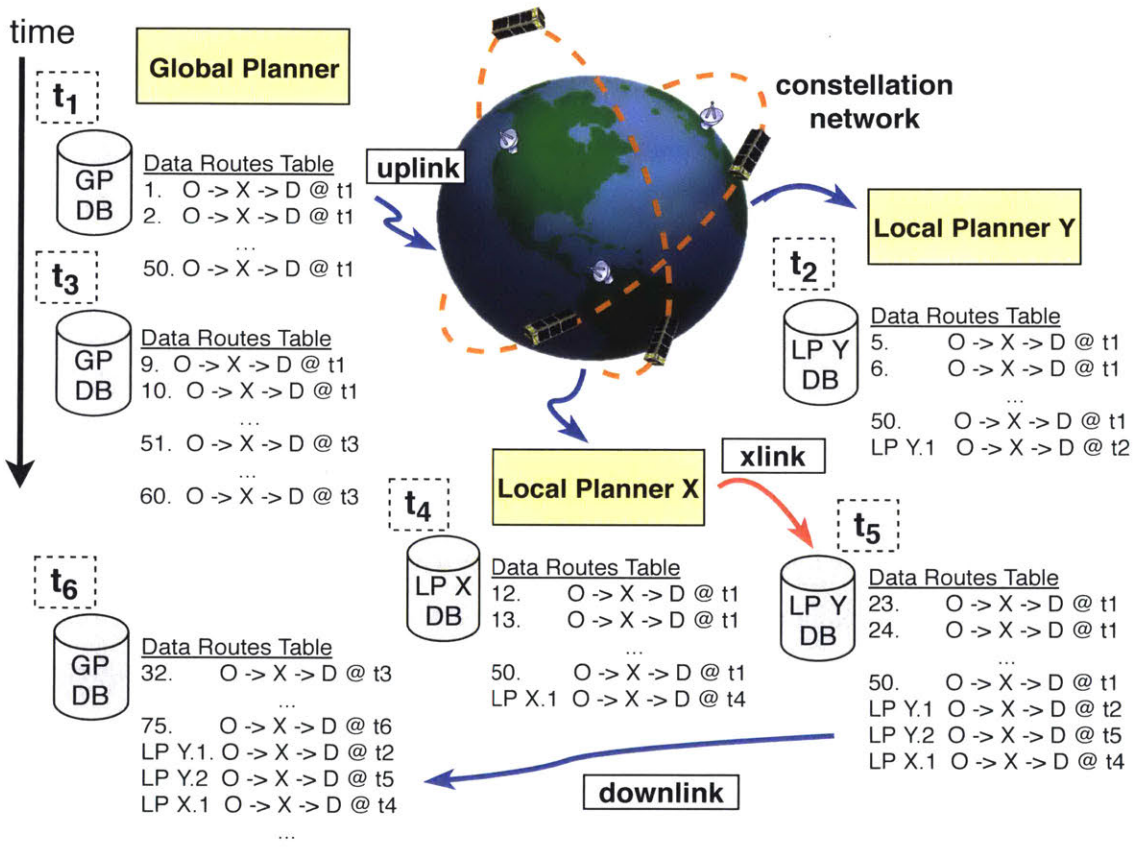


Figure 5-3: Diagram of Data Route distribution and update between GP and LPs through the constellation network.

in the past at time t_2 - and are not shown in the table. Meanwhile, the GP has run again on the ground at time t_3 and added new routes to its table. At t_4, the LP has run on satellite X and produced its own data route. At time t_5, PI has propagated from satellite X to satellite Y, either via a direct crosslink between satellites, multiple crosslink hops between the satellites (or even potentially via a downlink to a ground station and then a subsequent uplink to satellite Y, though that case is not shown here). All routes have been merged into satellite Y's PI DB, with a multitude of last update times on the routes. Note that though it is not indicated here, the GP and LP are both also capable of updating the utilization for existing routes, which would show up as a change in the update time and the utilization value for the route. Finally, we see at time t_6 that the global planner has received updates from both of the local planners.

In the current version of the GP and LP, the planners are each equally authorized to make changes to existing data routes. The inclusion of rewards for maintaining existing routes dis-incentivizes them from completely changing the input plans they receive, which helps both in tracking plans across the simulation, and comparing planned routes versus executed routes at the end of the simulation. For this reason, plan changes tend to be small tweaks to the utilization numbers for existing routes when they show up.

The GP is run at a regular interval in the simulation, usually for at least an orbit period and preferably more. The LP is run when conditions arise that merit running it, for example the occurrence of an injected observation.

The simulation keeps track of a global time for all of the agents. Whenever the GP and LP are run, their actual execution time on the computer used to run the simulation is independent of this simulation global time. For this reason, a configurable wait time is built-in before updated plans are made available to planning agents after it chooses to run the GP/LP.

Role of the LP with an External Communications Backbone

Because we assume an external communications constellation for the results in this work, the importance of the LP is slightly diminished from the case where all updates solely travel through intra-constellation links. Indeed, if we assume the external constellation is always available for sharing of PI, then it would seem like the GP could simply take on the role of making decisions about data routing, rather than needing the LP.

The LP still plays an important role however, because it decomposes the decision making about injected data routing into a much smaller problem that is quickly solvable. The GP-Fast algorithm can take a large amount of time to run for a large, complex constellation: as discussion in section 6.1.2, the execution times for a 100 satellite constellation on a moderately capable workstation computer was about 4000 seconds, or 1600 with increased parallelization). This means that even with a continuous connection for all satllites to the ground system running the GP, updated decisions about injected data routing may not be available soon enough to meet the latency metric goal of under 10 minutes. The LP runs in a much shorter time, under 10 seconds in general (see section 4.3), so it is able to make updated data routes available almost instantaneously.

5.3 Software Implementation

The details of the software implementation for CSim are shown in fig. 5-4 [1]. The same sim agents from fig. 5-1 are shown here, the satellite, ground station (GS), and ground station network (GSN) agents. The "agent components" are software modules running on each agent. The "data storage" blocks are bulk storage for PI, observation data packets, and recorded state information for the agents (*e.g.* energy storage history, executed activities). The yellow planning components are the GP and LP and wrapper interfaces around the planners that facilitate input and output

[1]The discussion here reflects commit 536b9edb2b3383242f78860db346ca8ab35b23b3 of the CIRCINUS Constellation Simulator repository at `https://github.mit.edu/star-lab/circinus_sim`

with them. The simulation is executed in a high-level loop that steps through a series of time steps from the beginning of the simulation scenario to the end of the scenario. For each agent, an activity execution step is first executed, followed by a state update step that changes the values of internal state to reflect the results of activity execution. Replanning occurs during the state update step.

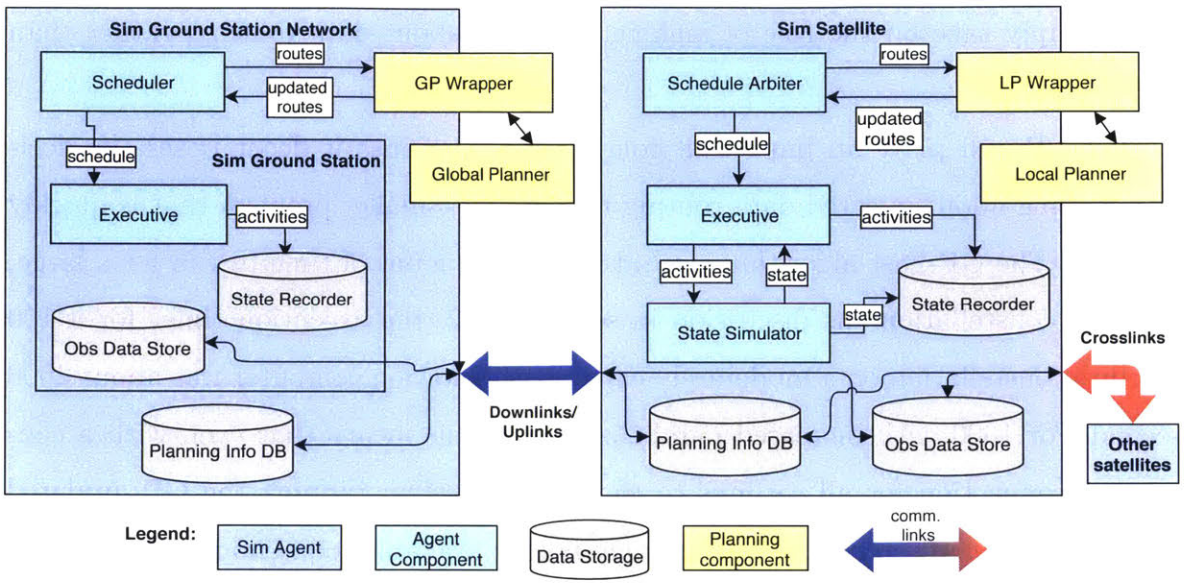


Figure 5-4: Block diagram of software implementation for Constellation Sim. Black arrows show the flow of information between components.

Simulation Agents

Each agent is stored as its own object, where "object" is in the Object-Oriented Programming (OOP) sense where all of the data and the functions that act on that data for a given agent are stored in the same coherent module. This helps to keep the interfaces between agents clearly defined and prevents inadvertent propagation of information (*e.g.* Through shared memory addresses). This approach also lends itself to adaptation for a hardware-in-the-loop-implementation, wherein the sim agents can be replaced by hardware running a software stack to implement the agent's functionality.

Communications links between the agents are simulated through simple interface function calls. For example, a satellite s_1 trying to transmit over a crosslink to

another satellite s_2 will first make the decision by itself to transmit based on its understanding of planned crosslinks and then poll s_2 for availability to receive. s_2 decides whether or not to receive data based upon its own understanding of planned crosslinks and expectation for throughput in the time slice of interest. While this functionality is currently implemented as direct function calls between the agents in a single runtime environment, an interface layer could be placed between the agents to simulate the quality of the link. This interface layer could even be run on another machine remotely accessible via an external network (note that this is referring to the real-life network on which the machines are running, not the simulated constellation network). This design for adaptability to a hardware-in-the-loop implementation is intended to support future adaptation to a full-up constellation autonomy software stacks.

Agent Components

The scheduler agent components, the "Scheduler" and "Schedule Arbiter" on the GSN and satellite agents, respectively, are responsible for turning Sim Route Container entries in the PI database (and their underlying DRs) into time-ordered lists of activities to be executed on the satellites and ground stations. They also manage replanning intervals, deciding when the LP and GP need to be executed.

The Executive components on the ground stations and satellites are responsible for executing the activities in these lists. The Executive keeps its own internal state that maintains a list of current activities that are being executed. While only single activities are allowed to be executed at a given time in this work, the Executive is adaptable for multiple activity execution in the future. The Executive progressively executes activities as time steps elapse. It keeps track of how much data volume has been transmitted or received for a given activity based upon successful exchange with other agents (that is, for downlinks and crosslinks). This delegation of activity execution bookkeeping to a single place in the Executive helps to ensure the integrity of the overall simulation by reducing interface complexity between the various software modules.

The State Simulator component on the satellite is present for simulating resource states. Currently it tracks energy and data storage states. Energy state is propagated by polling the Executive for the currently executing activities, factoring in their energy usage for the given time step, and then including any charging and base consumption energy terms. The State Simulator does not currently track attitude for satellites, but is intended to be a natural place for this capability to be implemented in the future. The Executive polls the State Simulator at each time step to check that state is nominal and activity execution can proceed as expected.

Implementation Details

The CSim software [2] was developed implemented in the Python programming language, and tested extensively on Python 3.5. The commercial MILP solvers Gurobi [56] and CPLEX [51] were wrapped within the GP and LP modules for scheduling solution. The Pyomo optimization modeling framework [47, 48] was used for specifying the MILP models, and interfacing with the commercial solvers. The CSim software is capable of simulating an arbitrary number of agents in any constellation configuration. The simulation loop and multi-agent framework was implemented from scratch in the Python code.

Note that the Python code makes extensive use of unordered mapping types, which can cause small changes in the schedule decisions made by the Global Planner for the same set of inputs (primarily due to choosing constructing slightly different routes in the Route Selection stage). The observed effect of this is small however: changes on the order of a 3% points of potential observation data throughput were seen for a 24 hour sim run with the 6-Sat constellation.

[2] Describes git commit 536b9edb of https://github.mit.edu/star-lab/circinus_sim; see further details on open-source availability in section 7.1.1

5.4 Constellation Simulator Validation

To validate the CSim software, a 24 hour sim was run for the 6-Sat scenario (discussed in section 2.6) [3]. The sim parameters used are detailed in appendix A.1. For this particular run, the GP time horizon for downlink windows is set the same as for observation and crosslink windows: 210 minutes, about 2 orbits. No injected events were present, in order to assess nominal performance. Quantitative results from this run are detailed in table 5.2, where metrics are calculated with the full set of planned data routes (sim route containers) from the full 24 hours are presented in column two and metrics calculated with the executed routes that data packets actually followed are presented in column three. The metrics are as described in section 2.3, and the values in parentheses are the standard deviations in those values.

We see immediately that the results from the executed data routes exactly match the planned data routes. GP-Fast was run eight times over the course of the 24 hours, producing 69 distinct data route objects, and only modifying a single data route. 28 observations are executed, with an average initial delivery latency of the first 100 Mb of each observation of 27.2 minutes, under a half hour. Average observation AoI is fairly large at about five hours. Satellite command and telemetry AoI are significantly lower than observation AoI, due to the benefits of TT&C data propagating through the network. Finally energy and data storage margin are kept high, at roughly 90% each.

For the single modified data route mentioned, the utilization of the route was slightly lowered, from 86.8% to 81.5%, and this slight reduction in data volume was made and communicated to the satellites sufficiently in advance to allow the satellite executives concerned with executing it to make the change without orphaned/dropped data volume. This illustrates how the GP is able to make updates as it gains additional information about activity window choices when running in a receding horizon planning loop.

The exact match between planned and executed routes over multiple runs of the

[3]Results were obtained with commit 640b814571166692d3c7ef03fc05d097edaab24b of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

Table 5.2: Summary of Metric Output for 24h, 6-Sat Constellation Sim

Item	Planned	Executed
Number obs windows executed	28	28
Total throughput (Gb)	159.4	159.4
Average obs initial latency (mins)	27.2 (±25.5)	27.2 (±25.5)
Average obs target AoI (hours)	5.05 (±1.75)	5.05 (±1.75)
Average sat command TT&C AoI (hours)	2.68 (±0.26)	2.68 (±0.26)
Average sat telemetry TT&C AoI (hours)	2.46 (±0.31)	2.46 (±0.31)
Average energy margin (%)	87.9 (±0.80)	87.9 (±0.80)
Average data margin (%)	92.4 (±3.07)	92.4 (±3.07)

GP in a single scenario validate that the CSim software correctly executes the intent of the GP. The satellite agents in CSim independently keep track of satellite state, and must verify at every time step that continued plan execution is consistent with current state. This shows that state and the simulator matches that expected by the GP, at least in the normal case.

Figures 5-5 to 5-12 graphically display the results from this sim run. Figure 5-5 shows the activities executed over the course of the sim run. The data volumes collected for the observation windows are indicated in the labels, showing that all observation data is collected successfully (there are small round off errors in some cases, these are inconsequential). The executed and planned windows lie exactly on top of each other, which is as expected. Note the tight clustering of crosslink windows around the 13 hour mark. This is due to satellite 0 having to shift all of its data volume to other satellites for downlink. Satellites four and five do not have observations of their own during this time so they are able to service the downlink needs of sat 0. This shift of data is needed because the planning window is only about 3.5 hours (210 minutes), and does not reach out to the downlink present at about the 19 hour mark on satellite 0. This demonstrates again the ability of the GP to effectively share communications resources across the constellation.

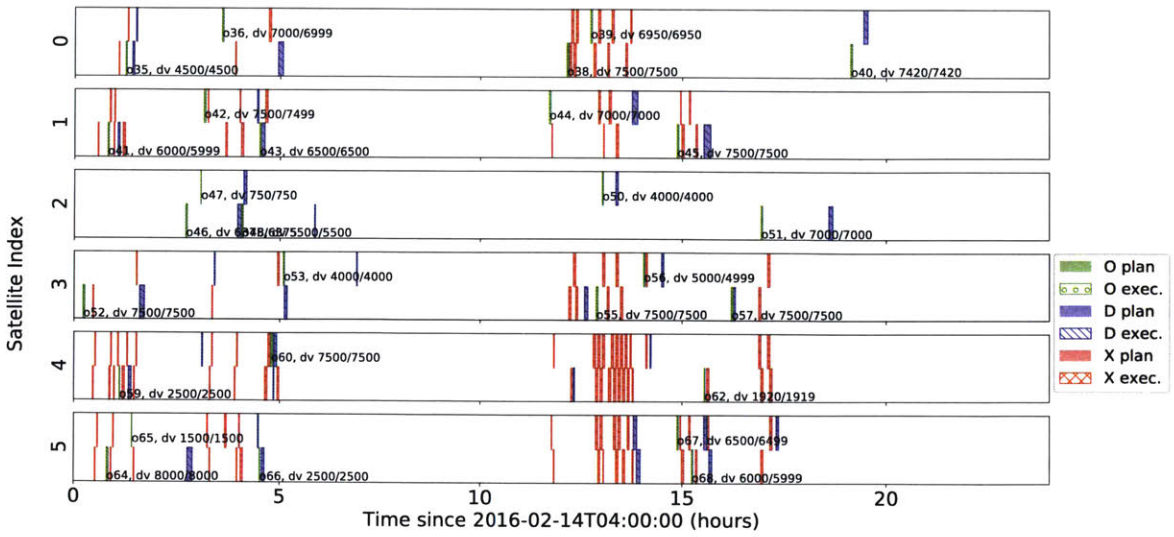


Figure 5-5: Planned and executed activities over 24 hour CSim run for 6-Sat scenario. Executed activities exactly matched those planned by GP-Fast. Windows are staggered vertically on a given sat for legibility.

Figures 5-6 and 5-7 show data and energy storage utilization in the sim run. We see that the large rises and falls of the peaks in data storage line up with observations and downlinks in fig. 5-5. At a couple points, data storage maxes out. We verified that no substantial amount (i.e., > 1 Mb) of data is dropped during this time and all planned data is accounted for at the ground stations at the end of the simulation. Note that this does not necessarily mean the full data volume capacity of each observation window was, only that the data scheduled for collection was routed successfully.

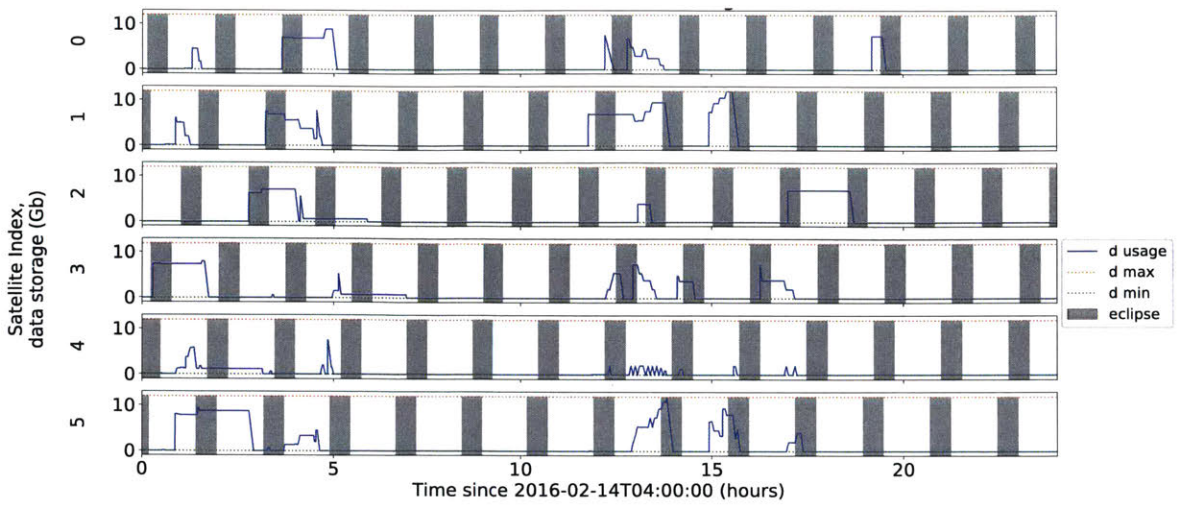


Figure 5-6: Data storage utilization over 24 hour CSim run for 6-Sat scenario. Data storage limits are 0 and 12 Gb.

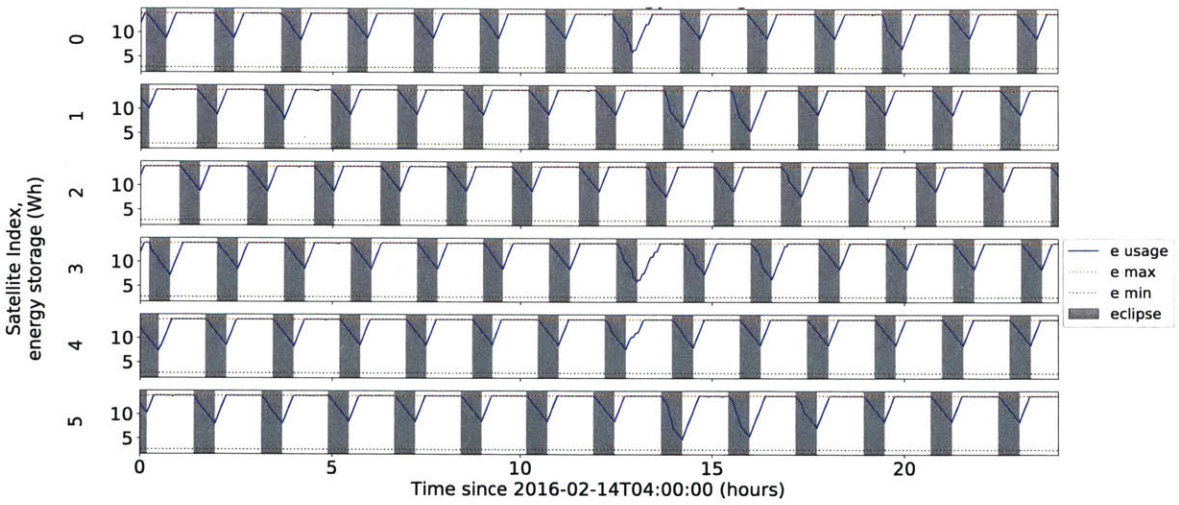


Figure 5-7: Energy storage utilization over 24 hour CSim run for 6-Sat scenario. Energy storage limits are 2.78 and 13.89 Wh.

Note the less blocky nature of the data storage curves in this plot versus fig. 3-5, because of the fact that in the simulation data is transmitted or received in time-step-sized chunks, making for a much smoother curve. We see in fig. 5-7 that the satellites are under-constrained in energy storage. The regular dips are caused by eclipses, and recovery to full storage is quick afterwards. The periods of relatively deeper dips for satellites 0,1,3, and 5 are caused by increased communications activity that aligns

with eclipse times.

Figure 5-8 shows the continuous AoI curves for the five observation targets in the scenario. AoI is assumed to start out at zero at the beginning of the scenario. There are long periods when the satellites do not have overpasses of the targets, and AoI grows to large values (up to about 19 hours for target 3). This shows that this small constellation is not well suited for frequent (roughly hourly) revisit of the targets. There are periods when AoI is kept low for an extended time, due to multiple observation windows for the same target executed across different satellites. At first it seems inconsistent that certain targets appear to have many more observation windows executed than others, for example target 4 has 10 observation windows executed, whereas target 3 has only one window executed. This is because of the planning window used for the simulation; in certain cases no downlinks are available within the planning window (recall this is limited to 210 minutes for all window types in this particular run), and the GP is unable to construct any routes at all for the observation windows during that time. This performance could be improved by extending the planning window for the downlinks to be longer.

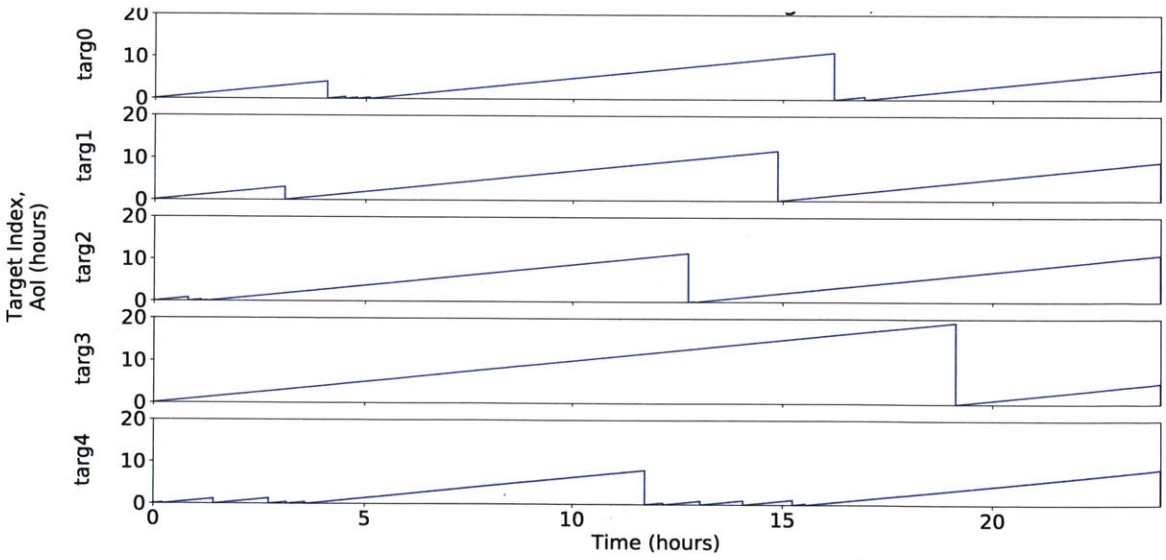


Figure 5-8: Observation target AoI curves over 24 hour CSim run for 6-Sat scenario. Large AoI drops occur when observation data is downlinked. Observation data (initial) execution requirement is 100 Mb. AoI is refreshed to 0 at observation execution ("at collection").

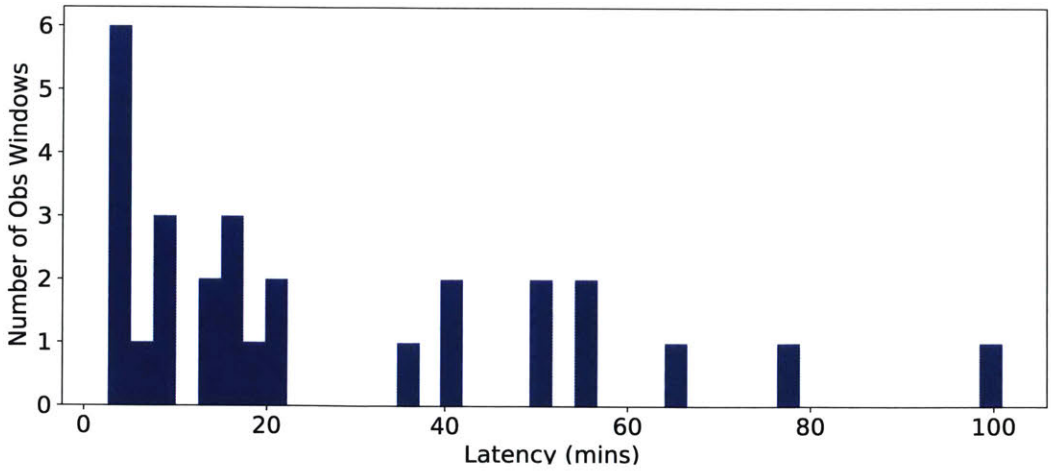


Figure 5-9: Histogram of observation window initial latency over 24 hour CSim run for 6-Sat scenario. Observation data (initial) execution requirement is 100 Mb. Most observations achieve a latency under 20 minutes.

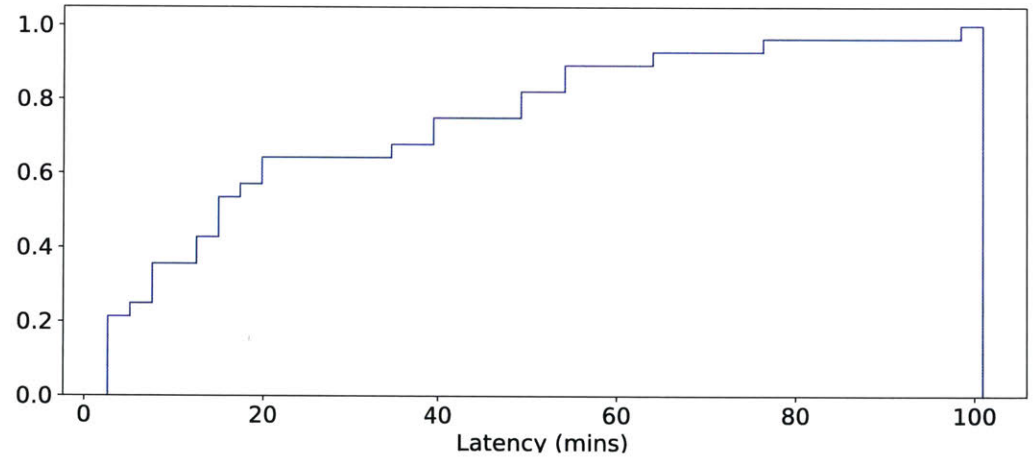


Figure 5-10: CDF of observation window initial latency over 24 hour CSim run for 6-Sat scenario. Vertical axis is the fraction of observation windows. Observation data (initial) execution requirement is 100 Mb.

Figure 5-9 presents a histogram of the initial latency for the 28 observation windows executed. As a reminder, the initial latency requirement for an observation window is that at least 100 Mb of that observation get downlinked by a given time. We see a widespread of different latencies, with the long tail reaching all the way to 100 minutes for one of the observation windows, close to a full orbit. This long tail is present because certain observation windows occurred during a relatively sparse

period of ground station overpasses for the constellation. In this instance the collecting satellite ends up downlinking the data itself, because it cannot crosslink to an earlier downlink window. This situation could be averted by having more geometric diversity in the constellation, not just two SSO planes that are fairly close together. However, the concentration of low latencies at about 20 minutes and under shows that in general the constellation is able to route data with low latency effectively. Figure 5-10 presents a cumulative distribution function of the same latencies, showing the fraction of windows that are routed at or below a given latency on the x-axis. Sixty percent of all the 28 observation windows are downlinked within 20 minutes.

The final two plots, figs. 5-11 and 5-12 show the AoI for command distribution to satellites and telemetry collection from satellites. We see that average AoI generally tracks the level of communications activity on the satellites; in the beginning of the sim and around the 15 hour mark AoI is kept fairly low, under about two hours. This is as expected, because when more communications activities are happening, more TT&C updates are distributed to the network. Recall that the execution of downlinks directly update the AoI for both command and telemetry to 0, and the execution of crosslinks updates individual AoI values on each satellite to fresher of the values on both the receiving and transmitting satellite. For example, the large drop to 0 in telemetry AoI for satellites 0,1,3,4, and 5 around the 13 hour mark occurred because of a single downlink performed by satellite 4, where satellite 4 had just recently received TT&C updates from the other satellites in the time before the downlink. Satellite 1 did not receive a refresh because of its relative isolation in the constellation (it performs no crosslinks at all over the duration of the 24 hours, which helps to test that such a case is handled properly in the simulation).

A more staggered effect is seen on command AoI around the 13 hour mark because the satellites have to wait for updated TT&C to propagate through the constellation to them. In general, this asymmetry is present between command and telemetry AoI when plans are created with the current version of the GP, because the GP tends to perform crosslinks right before the execution of a downlink in order to achieve low latency on routes. This tends to compress crosslink connectivity in the telemetry

update direction to happen immediately before a downlink. However, there is no forcing function for this to happen with command update in the current version of the GP, because it does not directly optimize for command and telemetry AoI.

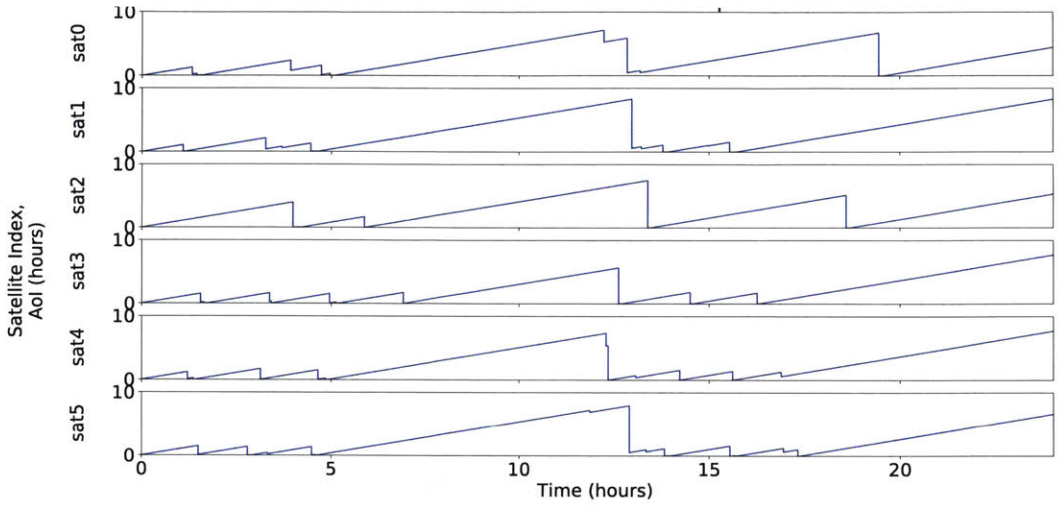


Figure 5-11: Satellite command TT&C data AoI over 24 hour CSim run for 6-Sat scenario. Large AoI drops occur when updated data arrives on satellites from any ground station.

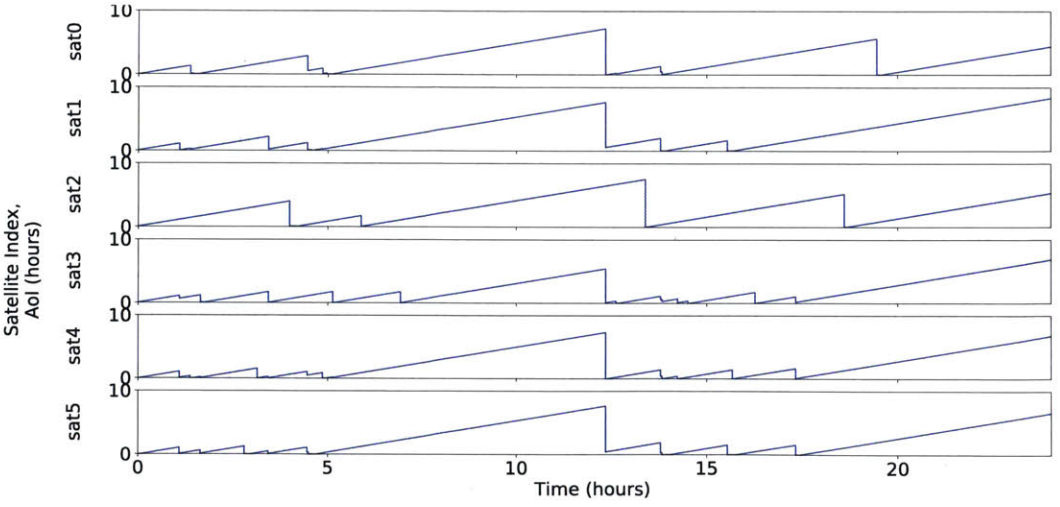


Figure 5-12: Satellite telemetry TT&C data AoI over 24 hour CSim run for 6-Sat scenario. Large AoI drops occur when updated data arrives at any ground station from a given satellite.

Chapter 6

Algorithm Performance And Constellation Study Results

We examine the performance of the algorithms presented in this thesis in three main analyses:

1. Scalability of the Global Planner (GP) algorithm compared to the state-of-the-art;
2. Performance of the integrated planning and scheduling system in representative, simulated constellation scenarios;
3. Routing of urgent, injected observation data.

These analyses directly correspond to research contributions 2 through 4 outlined in section 1.3.2.

Results for GP scalability are presented in section 6.1, results for the constellation simulation scenarios are presented in section 6.2, and injected observation results are presented in section 6.3.

6.1 Global Planner Scalability Results

We examine how well the GP algorithm scales with increasing problem size and complexity. We first assess the growth in algorithm run time with increasing planning

window length. Longer planning windows introduce more activity windows as choices for data routing, thus growing the problem size. Next we assess growth in runtime as the number of satellites in the constellation increases. We demonstrate successful execution of GP-Fast for both longer planning windows and a larger number of satellites than the state-of-the-art.

6.1.1 Runtime Variation with Planning Window Length

Figure 6-1 and table 6.1 show runtime for the GP-Fast and GP-Optimal algorithms [1]. Results were obtained using the 6-Sat scenario detailed in section 2.6 and appendix A.1, starting from the initial time of the scenario. The planning window length is varied from 80 to 1000 minutes for both GP algorithms, and total execution runtime is measured. RS2 settings were $\rho_1^{RS2} = 6$, $\rho_2^{RS2} = 6$, $\rho_3^{RS2} = 30$, which was found to deliver good schedule quality in a reasonable runtime. The runtimes presented here are the results from a single run of the relevant GP algorithm at the given planning window length. It was confirmed that runtime did not significantly change between runs with the same planning window length. The algorithms were run on a machine, "Kalamity", with an Intel Core i7-7700K @ 4.20 GHz processor, 64 GB RAM, and Windows 10 64-bit. The Gurobi commercial MILP solver version 8.0.0 is used [56].

We see that the runtime for GP-Fast increases much more slowly than GP-Optimal. While it is not quite linear, it avoids the huge increase in compute time required by GP-Optimal at the 1000 minute planning window mark. This highlights the key benefit of the GP-Fast algorithm: it reduces computational complexity by sifting out a large number of redundant data route choices at the Route Selection (RS) stage, and only inputs a small subset of routes to the MILP solver in the Activity Scheduling stage. Because a large combinatorial penalty is paid for increasing the number of decisions to make in the MILP solution process (generally a non-polynomial, essentially exponential, increase in complexity with growing problem size

[1] Results were obtained with commit b468c2a4ea896576ad9c530f6c8ed2abffe100a3 of CIRCINUS Global Planner repository at `https://github.mit.edu/star-lab/circinus_global_planner`

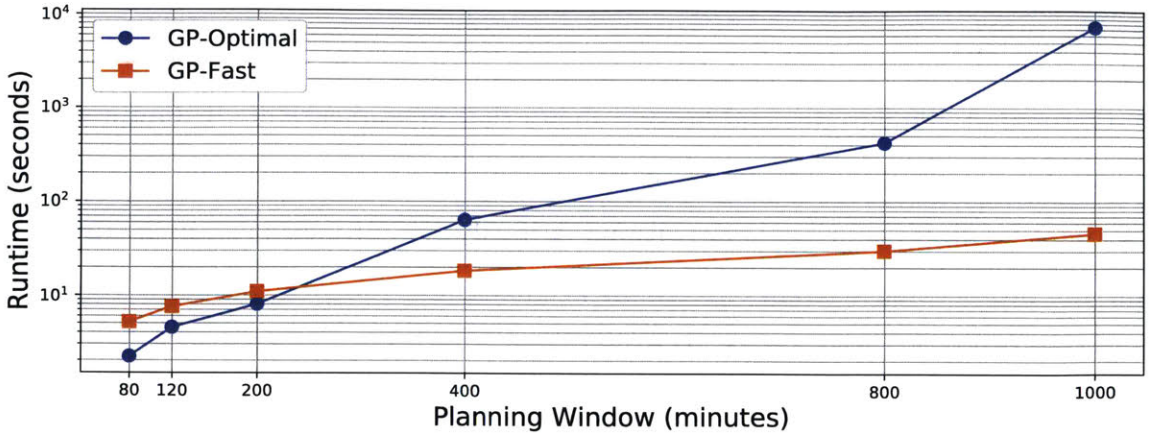


Figure 6-1: Runtime comparison of GP algorithms. Run with the 6-Sat constellation, with parameters specified in appendix A.1. The Gurobi solver was run to an optimality gap of 1% in all cases.

[32]), this cuts out a lot of the complexity of the scheduling problem.

Table 6.1: Runtime with Varying Planning Window Length for GP Algorithms in 6-Sat Sim Case

Plan Window (minutes)	**GP-Optimal Time (seconds)**	**GP-Fast Time (seconds)**
80	2.2	5.2
120	4.5	7.6
200	8.0	10.9
400	63.4	18.3
800	424	30.1
1000	7386	46.5

For GP-Optimal, total runtime includes the time to construct the MILP problem model in Python with the Pyomo mathematical modeling framework, the time to run the MILP solver, and a small amount of time spent in post-processing the solved problem to extract a set of data routes. For GP-Fast, the time includes running both the Route Selection, constructing the MILP problem model for Activity Scheduling (also in Pyomo), running the solver, and minimal postprocessing time to check schedule integrity. GP-Fast requires more setup in general, in particular when it creates

parallel threads for executing Route Selection. For this reason GP-Fast actually runs slower than GP-Optimal with small planning windows. The much faster growth of GP-Optimal at 400 minutes and beyond shows that fundamentally GP-Optimal performs more computations to arrive at its schedule solutions. Note that the current code implementation for both algorithms is highly un-optimized, as it is currently built for prototyping usability (it could be made faster by, say, using a compiled language like C++ as opposed to Python).

For the state-of-the-art algorithms presented by Zhou *et al.* with a planning window of 120 minutes in the same 6-Sat simulation case, a runtime of roughly 40,000 seconds was required for their optimal algorithm (MACP), and roughly 200 seconds for their heuristic algorithm (ACG, "Zhou-Fast"). This is compared to 7400 and 47 seconds for GP-Optimal and GP-Fast. A different solver was used for the Zhou *et al.* runtime results, which the authors refer to as "the free-software LPsolver (version 5.5.2.0 for windows 64 bits)". No citation is provided, but this most likely refers to the lp_solve MILP solver [89] due to the similarity of version numbers. While the GP algorithms have not been run with lp_solve, it is expected that the Gurobi solver will be able to solve the GP problem instances faster due to the techniques it incorporates to speed up solution of the Branch-and-Bound algorithm (see section 2.4.1).

The use of different solvers effectively means that we cannot unambiguously compare the run time for the Zhou *et al.* algorithm with the GP algorithms, based only on the results presented by Zhou *et al.*. However, the Zhou *et al.* algorithms do not incorporate minimization of latency, which simplifies the problem model for the Zhou-Fast algorithm by allowing it to operate on a timeslot-by-timeslot basis.

We also verified that throughput and latency performance for GP-Fast were of good quality relative to GP-Optimal for changing planning window size. Figure 6-2 shows values for total throughput and average latency as a percentage of optimal over the same planning window sizes as in fig. 6-1. The values in the plot are summarized in table 6.2. Total throughput is the summation of data volume downlinked for all scheduled observations, and average latency is calculated as an average over the same scheduled observations.

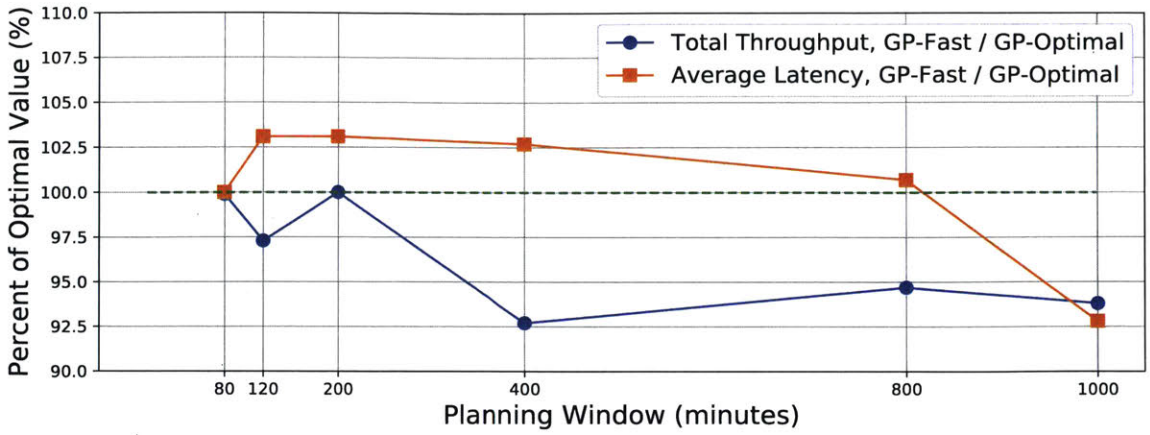


Figure 6-2: Variation of GP-Fast schedule quality with planning window links. Run with the 6-Sat constellation, with parameters specified in appendix A.1.

We see that total throughput stays better than 92% of optimal and average latency stays within 8 percentage points of optimal for all planning window sizes. Performance for throughput jumps when progressing from 120 to 200 to 400 minutes due to additional downlink windows being included in the planning horizon. These downlinks are clumped temporally for the 6-Sat case, because all of the ground stations happen to be in China. This causes many more downlink possibilities to be added in distinct periods of time as the planning window lengthens, as opposed to gradually over time, with the result that there is a discrete, jump-like character to the changes in schedule quality. We see that average latency performance is slightly better (lower) than optimal for the 1000 minute planning window, due to a trade-off that the GP-Fast algorithm made between throughput and latency (slightly better latency for slightly less throughput). This highlights the potential sensitivity of schedule quality to objective function weighting factors; however in this case both throughput and latency are close to optimal so the trade-off is not cause for worry.

These results show that GP-Fast scales more effectively than GP-Optimal, while sacrificing little of the schedule quality delivered by GP-Optimal.

Table 6.2: Variation of GP-Fast Schedule Quality with Planning Window Length in 6-Sat Sim Case

Plan Window (mins)	Throughput (Gb)			Ave. Latency (mins)		
	GP-Opt	GP-Fast	Fast/Opt	GP-Opt	GP-Fast	Fast/Opt
80	12.0	12.0	99.9 %	27.6	27.6	100.0 %
120	29.4	28.6	97.3 %	19.4	20.0	103.1 %
200	30.0	30.0	100.0 %	19.4	20.0	103.1 %
400	82.0	76.0	92.7 %	21.9	22.5	102.7 %
800	93.8	88.8	94.7 %	60.1	60.5	100.7 %
1000	163.8	153.6	93.8 %	62.2	57.7	92.8 %

6.1.2 Runtime Variation With Number of Satellites

GP-Fast execution time was also investigated with a changing number of satellites in the constellation [2]. Figure 6-3 presents results for GP-Fast runtime as the number of satellites changes from 10 to 100. The constellation geometries present are summarized in appendix A.3. There are 40 observation targets and 7 ground stations. The planning window length was 120 minutes in all cases. RS2 settings were $\rho_1^{RS2} = 6$, $\rho_2^{RS2} = 6$, $\rho_3^{RS2} = 10$, to reduce runtime for larger constellation sizes. No GP-Optimal results are presented because the optimal run time was observed to grow too large for these problems (for 18 satellites with the Gurobi MILP solver, no solution could be found within 3 hours of runtime). GP-Fast was run on the same machine as in section 6.1.1, also with the Gurobi solver for Activity Scheduling (AS). Runtimes are presented from a single run, but it was confirmed that the time did not significantly change between runs with the same number of satellites. The Gurobi solver was run to an optimality gap of 1% or a timeout of 1000 seconds, whichever came first. Note that for the 100 and 140 satellite cases, optimality gaps of only 8.27% and 9.88% were achieved; the gap was less than 1% for all other cases.

The curve labeled "regular (8 cores)" shows the runtime found when execution of

[2]Results were obtained with commit ead12723d3ef77df7fe0b74bfff83118034e4ff1 of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

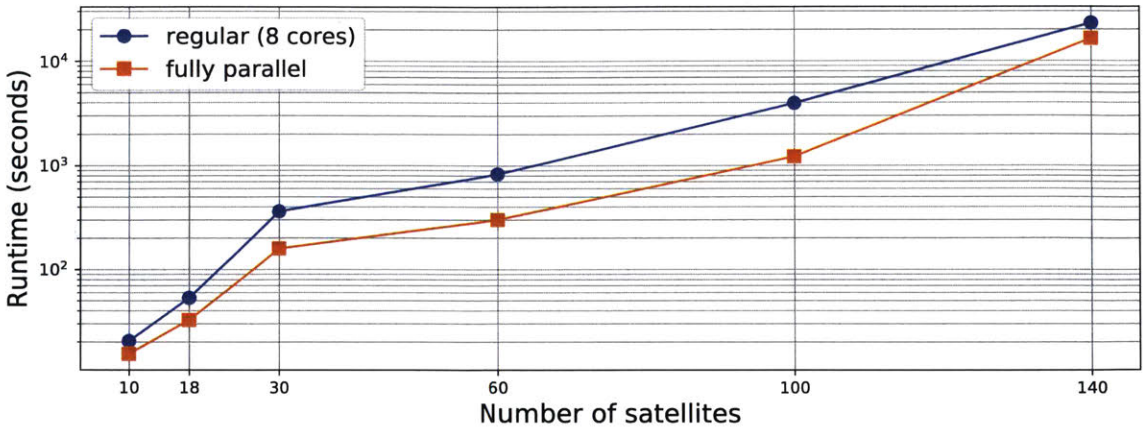


Figure 6-3: GP-Fast runtime variation with the number of satellites and constellation geometry. Planning window size was fixed at 2 hours in every case. Other parameters are specified in appendix A.3. The "fully parallel" represents the potential best case runtime with moderate scale parallelization. The Gurobi solver was run to an optimality gap of 1% or a timeout of 1000 seconds, whichever came first.

route selection step 1 (RS1, Route Construction) was run in parallel over 8 cores. Because RS S1 can be run independently for any given observation window, this parallelization could theoretically be extended to a large number of cores, given that the required computing resources are available contemporaneously. The second curve, labeled "fully parallel", presents the calculated run time given an unlimited number of cores, hence making the run time for all of RS1 equal to the maximum run time for a single observation. For the largest, 140-satellite problem solved here, 241 observation windows were searched for RS1, so about that many parallel cores would be needed. Note that this represents an ideal case; in reality, there would be extra time required for setting up these parallel cores, passing data to them, and receiving data from them. We assume here that this fixed overhead is negligible.

The real runtimes, with 8 cores, are shown in table 6.3, which splits out the runtime for route selection "RS Step 1 Time", from the full runtime "Full GP-Fast Time". The number of observation windows run for RS1 is also indicated. These runtime values are used to determine the maximum route selection time per observation in table 6.4. Assuming that RS1 does indeed finish when the slowest-solved observation finishes, then we simply add the time taken by RS2 ("route downselection") and Activity

Scheduling (AS) to get the fully parallel runtime.

Table 6.3: Runtime for Varying Number of Satellites, for Constellation Sim Cases from Figure 6-3

Number of Sats	Num RS Obs Searched	RS Step 1 Time (seconds)	Full GP-Fast Time (seconds)	AS MILP Optimality Gap
10	14	8	20	0.30 %
18	69	31	53	0.92 %
30	119	253	361	0.94 %
60	102	737	824	0.86 %
100	175	3166	3932	8.27 %
140	241	7467	22928	9.88 %

Table 6.4: Potential Minimum Runtime with Full Parallelization

Num Sats	Max RS Step 1 Time/Obs (seconds)	RS Step 2 Time (seconds)	AS Time (seconds)	Fully Parallel GP-Fast Time (seconds)
10	2.8	0.1	12.5	15.5
18	10.2	2.4	20.1	32.7
30	51.1	14.9	92.6	159
60	215	14.8	71.8	301
100	446	92	674	1212
140	832	273	15187	16292

We see that the fully parallel runtime cores for a 100 satellite constellation reaches about 1200 seconds, under 0.5 hours. Note that RS stage as a whole could likely lose significant runtime through optimization of the GP-Fast codebase. As noted, the current, Python-based implementation was built as a prototype for usability, not for maximum performance.

The runtime for 60 satellites does not increase as much as would be expected from 30 satellites. This is due to the fact that the 60 satellite constellation has an inclination of 60°, as opposed to the 30° for 30 satellites. At 60°, the constellation

has relatively fewer access accesses to the tropical observation set than the 30° constellation does.

Though RS2 runtimes are relatively short in these examples, the runtime can grow very large as 1) more input routes are provided by RS1 to RS2 for downselection, and 2) the number of routes to select for each observation grows larger (the $\rho^{RS2}_{1/2/3}$ parameters, see the RS2 description in section 3.3.1). For example, RS2 executed in 14.9 seconds for the 30 satellite case here, with a 2 hour planning window, 40 observation targets, 7 ground stations, and $\rho^{RS2}_1 = 6$, $\rho^{RS2}_2 = 6$, $\rho^{RS2}_3 = 10$. RS2 received 152044 routes from RS1 (for 119 unique observation windows), and downselected to 2315 routes. With the same 30 satellite geometry, a slightly different planning window (95 minute time horizon for observations and crosslinks, 12 hours for downlinks [3]), 40 observation targets, 9 ground stations, and $\rho^{RS2}_1 = 6$, $\rho^{RS2}_2 = 6$, $\rho^{RS2}_3 = 10$, RS2 executed in 3045 seconds, about 50 minutes. In this case, RS2 received 366721 routes from RS1 (for 95 unique observation windows), and downselected to 3527 routes. The second case had many more input routes from which to downselect (due to additional downlink windows to use for routing), and many more output routes to select for the "least overlap heuristic" (ρ^{RS2}_3). This illustrates the computational complexity of the "least overlap" heuristic. The least overlapped route is determined by checking the data volume availability for every single window within every single route that could be picked, for the observation of interest at each iteration. It would be useful to devise a better approach for route downselection in future work to alleviate this bottleneck. A possible approach is outlined in section 7.2.

Figure 6-4 and table 6.5 present results for the schedule quality produced by GP-Fast over the variation in number of satellites. Here results are compared to the potential throughput and latency metrics achievable from all of the routes output by RS2, Route Downselection. This potential number is not actually scheduled, so it does not represent an optimal value, but does give a best performance bound for the optimal value of each metric. We see that schedule quality degrades significantly for

[3] only one downlink window was allowed to be used for route construction after the end of the obs/crosslink horizon, to limit complexity.

the largest constellations, reaching about 50% of potential throughput and 170% of potential latency for 140 satellites. Also, the 30 satellite case shows only about 78% of throughput and 127% of potential latency (it performs worse than 60 satellites due the its lower inclination orbits - its tighter coverage for the set of tropical observation targets- and thus increased routing complexity).

This degradation stems from two factors: 1) the selection of fewer routes at the downselection stage than for the changing planning window length analysis, with $\rho_3^{RS2} = 10$ instead of $\rho_3^{RS2} = 30$ (the "least overlap" heuristic) and 2) the fact that for the large constellations, the MILP solver retains a roughly 10% optimality gap, as shown in table 6.3. The use of fewer downselected routes means there are fewer options for routing data through the constellation, generally leading to a poorer quality schedule. It was found that running with $\rho_3^{RS2} = 30$ simply took too long for the large constellations. The retention of a large (about 10%) optimality gap means that the solver has not fully solved the problem; schedule quality could still be improved if more runtime were allowed. In practice, the solver often does not improve schedule quality quickly after 1000s: for 100 sats, an 8.27% gap was achieved at a 1000s timeout, which was lowered to 3.1% at 18000. This represents an increase in total throughput by about 4% and a reduction of average observation initial latency from about 10.68 to 8.3 minutes.

Table 6.5: Comparison of GP-Fast Schedule Results with Potential Schedule Quality from RS2 Downselected Routes, for Constellation Sim Cases from Figure 6-4

Num Sats	10	18	30	60	100	140
RS2 Potential Throughput (Tb)	0.108	0.595	1.062	0.782	1.499	2.076
AS Scheduled Throughput (Tb)	0.093	0.503	0.829	0.702	0.993	1.093
Percent of Potential %	86.2%	84.5%	78.1%	89.8%	66.3%	52.6%
RS2 Potential Average Obs Initial Latency (mins)	4.67	8.78	5.93	5.99	7.08	6.63
AS Scheduled Average Obs Initial Latency (mins)	5.10	9.68	7.54	7.02	10.68	11.29
Percent of Potential %	109.2%	110.2%	127.2%	117.2%	150.9%	170.3%

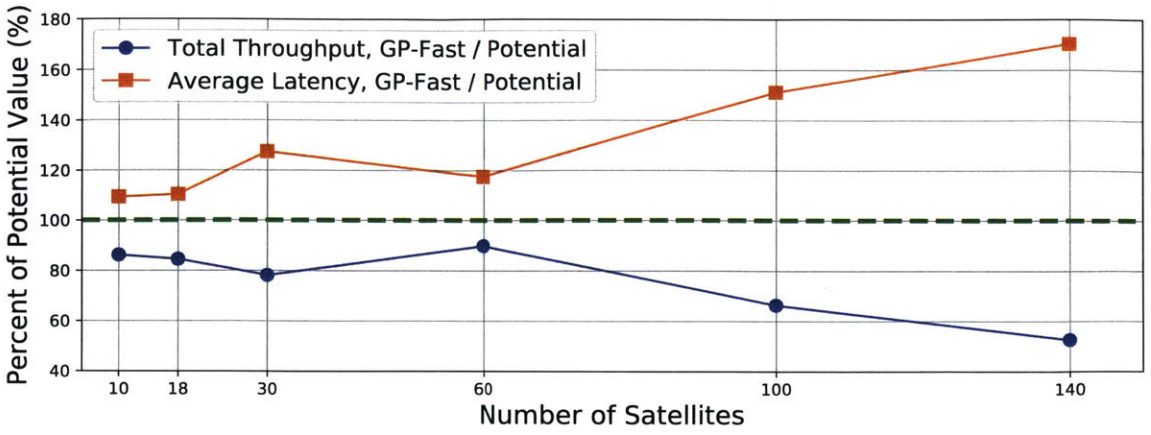


Figure 6-4: Variation of GP-Fast schedule quality, compared to potential quality, with changing number of satellites and constellation geometry. Planning window size was fixed at 2 hours in every case. Other parameters are specified in appendix A.3. Potential quality is determined based on unscheduled RS2 output routes. The Gurobi solver was run to an optimality gap of 1% or a timeout of 1000 seconds, whichever came first.

These results show that the GP-Fast algorithm can execute for larger constellation sizes than the state-of-the-art, though not necessarily within 90% of optimal latency in all cases (assuming the potential values shown here are close to the optimal values). Zhou *et al.* only demonstrated planning for a six satellite constellation [124] over a 120 minute window (about 200 seconds runtime with their heuristic algorithm), and Kondrateva *et al.* demonstrated planning for an 18 satellite constellation over a 100 minute window (83 seconds runtime with their LP2 sub-optimal algorithm). Based on the results in section 6.1.1, we expect that GP-Fast will be able to achieve better quality schedules (within about 10 percentage points of optimal values) with a better selection of input routes to the Activity Scheduling stage. It would not help to simply input more routes, because the MILP solver may run too slow; rather a more effective set of input routes is desired. A suggested approach for achieving this is outlined as an item for future work (see section 7.2).

6.2 Constellation Simulation Studies Results

We examine planning and scheduling performance for the two constellation simulation cases detailed in section 2.6, SSO Ring (10 satellites) and Walker (30 satellites) [4]. Parameters for these constellations are detailed in appendix A.2. The constellations are simulated over a 24 hour period in three different communications contexts:

1. Downlinks only, "Dlnk Only"
2. Downlinks + Crosslinks, "Dlnk + Xlnk"
3. Downlinks + Crosslinks - constrained, "Dlnk + Xlnk, constrained"

The first context simulates the current state-of-the-art in operational CubeSat constellations, which feature only downlinks to ground stations, *e.g.* Planet, Inc. [69, 28]. The second simulates the baseline downlink and crosslink constellation studied in this work, one which allows satellites to only perform one activity at a given time, but requires no additional transition time between activities, *e.g.* a satellite s_1 can crosslink with another one s_2 within its same orbit plane and afterwards immediately start crosslinking with another, s_3, in a completely different orbit plane.

The third context simulates a constellation with downlinks and crosslinks, but with constrained timing for inter-activity transitions. This context models a more operationally realistic smallsat bus design, which is only capable of transmitting in a single direction within its own orbit plane (in the velocity or anti-velocity direction), must perform long attitude slews in order to crosslink to another plane, and must slew between observation and crosslink activities (because the observation payload and crosslink transceiver are assumed to be mounted 90°apart). The required transition times for this communications context is summarized in table 6.6. For this context, satellites were only allowed to transmit in the velocity direction.

Results are presented for simulations for the SSO Ring and Walker cases, over the relevant communications contexts. We examine the following metrics in order:

[4]Results were obtained with commit d1fa26250ee579a97415449084e80e4269f34277 of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

Table 6.6: Required inter-activity transition times for "Dlnk + Xlnk, constrained" context (in seconds)

Transition Description	**Transition Time Req. (seconds)**
Obs to/from Xlnk	180
Xlnks: Intra-orbit to/from inter-orbit	180
Xlnks: Inter-orbit to inter-orbit, same xlnk sat	0
Xlnks: Inter-orbit to inter-orbit, same orbit, different xlnk sat	180
Xlnks: Inter-orbit to inter-orbit, different orbit	360

1) observation data throughput and energy/data margin, 2) observation latency, 3) observation Age of Information (AoI), and 4) TT&C Age of Information (AoI). Refer to section 2.3 for more detail on the metrics. We examine the amount of downlink and crosslink capacity that is utilized by the constellations, in section 6.2.5.

6.2.1 Observation Data Throughput Comparison

Figure 6-5, table 6.7, and table 6.8 summarize the observation data throughput results for the constellations over the first two communications contexts. The "potential observation data throughput" is the minimum of the total data volume that could be collected from observation access periods (*i.e.* the times when satellites are flying over observation targets multiplied by the payload data rate, 127.5 Mbps), and the total data volume that could be downlinked (the capacities for all downlink windows, summed). The observation data throughput executed is the actual amount of observation data that was downlinked by the constellation in the simulation case. Note that the potential data throughput is not the same as the optimal data throughput in 6.1.1; this "potential throughput" is not necessarily executable even in the optimal case due to the various operational constraints on the constellation (*e.g.* too little

data storage). However, this potential number gives an upper bound and a rough idea of how much data could be executed.

We see that the Dlnk + Xlnk context executes more data volume than the Dlnk Only context, and significantly more for the Walker constellation: 70.5% versus 43.6% (Dlnk + Xlnk vs. Dlnk Only). The Walker constellation also has a much larger potential (and executed) total throughput value than SSO Ring. The fits with the larger number of available observation windows in the Walker constellation, as shown in table 6.8. In terms of the number of observation windows executed, the difference is not as stark between Dlnk Only and Dlnk + Xlnk.

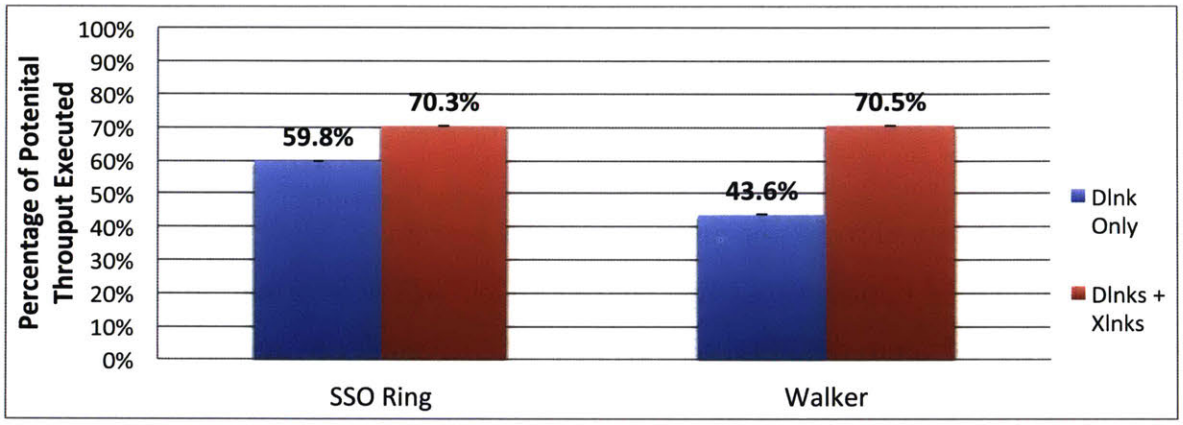


Figure 6-5: Percentage of potential observation data delivered to ground in constellation sim cases. Sim run for 24 hours, with GP-Fast, 40 observation targets, 4 GB limit on data storage. See appendix A.2 for other parameters.

Table 6.7: Total Observation Data Throughput for Constellation Sim Cases from Figure 6-5

Metric	SSO Ring		Walker	
	Dlnk Only	Dlnk+Xlnk	Dlnk Only	Dlnk+Xlnk
Potential throughput (Tb)	1.83	1.83	14.41	14.41
Throughput executed (Tb)	1.09	1.28	6.29	10.16
Percent of potential	59.77 %	70.32 %	43.62 %	70.49 %

Table 6.8: Number of Executed Observation Windows for Constellation Sim Cases from Figure 6-5

Metric	SSO Ring		Walker	
	Dlnk Only	**Dlnk+Xlnk**	**Dlnk Only**	**Dlnk+Xlnk**
Potential obs windows count	202	202	1516	1516
Obs executed count	173	191	1213	1374
Percent executed	85.64 %	94.55 %	80.01 %	90.63 %

Table 6.9 sheds more light on why the Dlnk + Xlnk context performs so much better in terms of data volume than does Dlnk Only. This table presents statistics for the satellite-average resource margins (percent of the resource remaining) over the simulation, where "satellite-average" means the average value of a metric over the full simulation scenario for a single satellite. While energy margin does not vary much between the communications contexts, data margin differs significantly. The median value for sat-average data margin is only 31.8% for Walker, Dlnk Only, versus 56.2% with Dlnk + Xlnk. Also the minimum for Dlnk Only, 17.8% is very low. The low amount of executed data volume for the Dlnk Only case is due to limited data storage causing the satellites to cease collecting additional observation data.

Table 6.9: Satellite-average Resource Margins for Constellation Sim Cases from Figure 6-5

Average Resource Margin Metric	SSO Ring		Walker	
	Dlnk Only	**Dlnk+Xlnk**	**Dlnk Only**	**Dlnk+Xlnk**
Energy, Median	86.5 %	84.9 %	86.5 %	84.5 %
Data, Min	30.1 %	52.0 %	17.8 %	42.5 %
Data, Median	49.0 %	60.8 %	31.8 %	56.2 %
Data, Max	58.0 %	76.2 %	44.8 %	66.2 %

This is apparent in figures 6-6 and 6-7, which show the evolution of data storage

for Walker Dlnk Only and Dlnk + Xlnk. The satellites in Dlnk Only max out their data storage roughly 7 hours into the simulation and, while they reduce it later, they generally retain a high value for storage. In the Dlnk + Xlnk however, data storage is much lower overall, due to the fact that satellites are able to offload the data they have collected through crosslinks. This offloading enables more data to move through the network, moving it away from sources (executed observation windows) to sinks (executed downlink windows) across satellites. In the Dlnk Only case, every satellite is responsible for handling its own sources and sinks, and the downlinks are simply not frequent enough to service the observations.

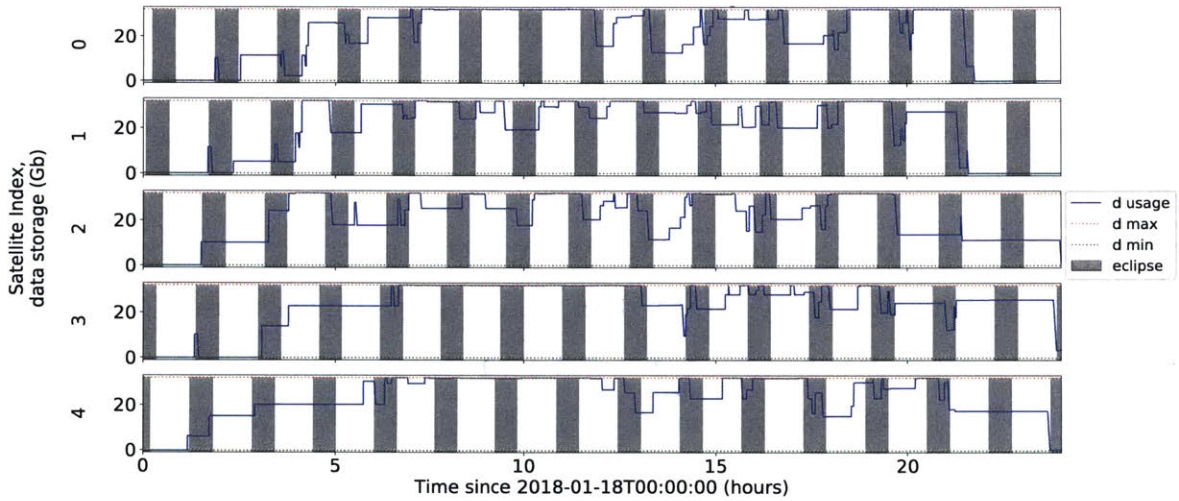


Figure 6-6: Data storage utilization for Walker, Dlnk Only sim case. Utilization is high (>75%) or maxed out for much of the run. Data storage limit is 4 GB, see appendix A.2 for other parameters.

This data storage limitation would have been less of a problem had a larger number been chosen for data storage capabilities on the satellites. A conservative 4 GB (32 Gb) was assumed for the satellite bus design in these constellations. However, this example serves as a good illustration of the effectiveness of the Global Planner. It was able to automatically determine the best way to maximize data volume collected by offloading data from satellites in the Dlnk + Xlnk case. The algorithm could perform similarly for energy storage.

An additional simulation case was run with increased data storage, 32 GB (256

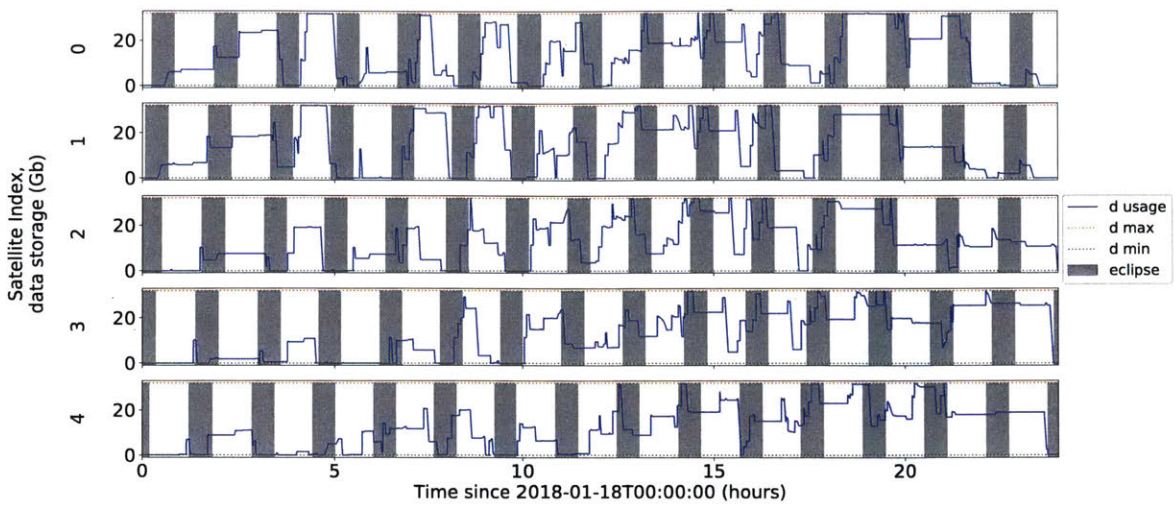


Figure 6-7: Data storage utilization for Walker, Dlnk + Xlnk sim case. Utilization is much lower in general than the Dlnk Only case. Data storage limit is 4 GB.

Gb). Here we found that throughput performance was good the SSO Ring sim case, at about 95% of potential throughput for both contexts (Dlnk Only and Dlnk + Xlnks). Slightly more executed data volume was seen with crosslinks, at 1.54 Tb versus 1.46 Tb for Dlnk Only. This shows that crosslinks still did have a positive effect on delivering data volume to downlinks. Initial obserservation latency performance (discussed further in section 6.2.2) was still much better in the Dlnk + Xlnks case, with a median value of 12.5 minutes versus 61.5 minutes for Dlnk Only.

6.2.2 Observation Data Downlink Latency Comparison

Figure 6-8 and table 6.10 summarize initial observation data downlink latency performance for the simulation cases. The "initial latency" metric in this case is the latency for the delivery of the first 100 Mb of data from each observation window (recall that latency is calculated as the difference in absolute time of the downlink and observation windows for a given data route). For each executed observation, all corresponding data routes are collected and the earliest route to deliver the minimum data requirement is determined. The P_{25} and P_{75} values denote the 25th and 75th percentiles, respectively.

In fig. 6-8, we see that median latency is lower for the Dlnk + Xlnk cases, particu-

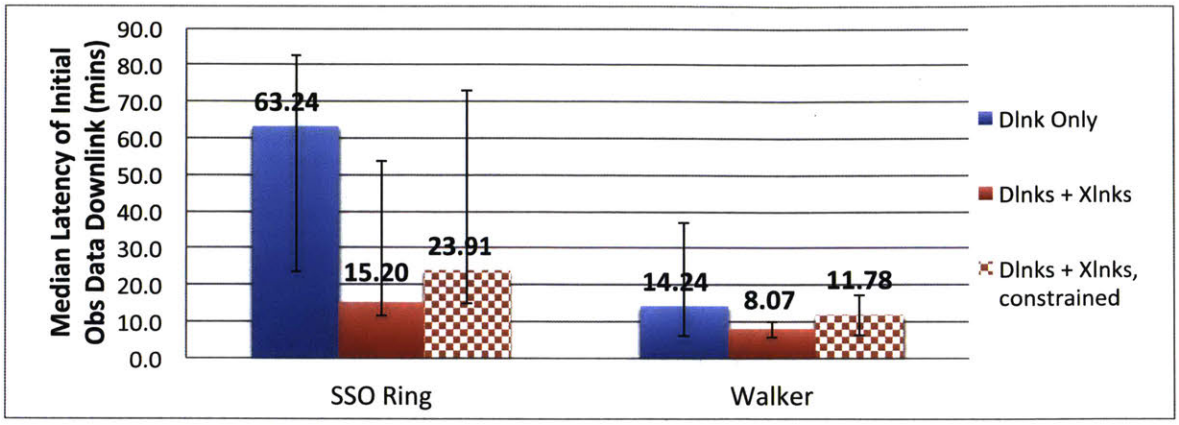


Figure 6-8: Median latency of initial observation data delivery, across all executed observation windows. Error bars indicate the 25th and 75th percentile values for each case. Sim run for 24 hours, with GP-Fast, 40 observation targets. See appendix A.2 for other parameters.

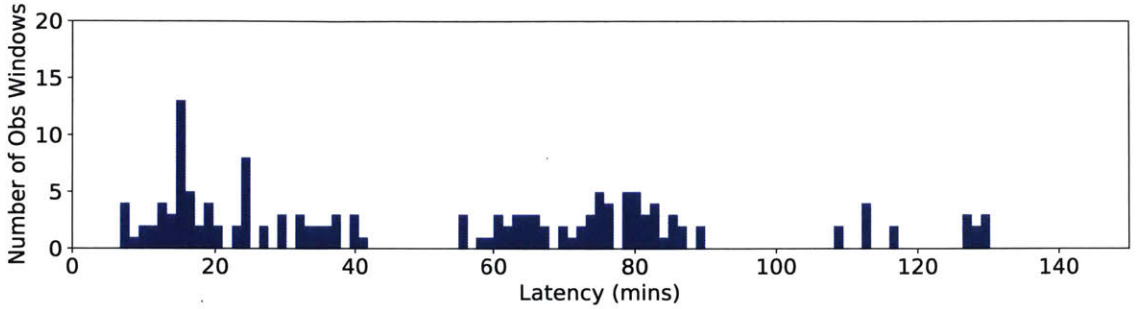
Table 6.10: Summary of Initial Observation Data Downlink Latency Results

Latency Metric (mins)	**SSO Ring**			**Walker**		
	Dlnk Only	**Dlnk+ Xlnk**	**Dlnk+Xlnk Constr.**	**Dlnk Only**	**Dlnk+ Xlnk**	**Dlnk+Xlnk Constr.**
Min	6.66	3.73	5.74	0.66	0.82	0.66
P_{25}	23.49	11.57	14.91	5.91	5.74	6.33
Median	63.24	15.2	23.91	14.24	8.07	11.78
P_{75}	82.66	53.91	72.91	36.9	9.91	17.08
Max	656.9	339.99	358.16	598.41	201.99	251.66

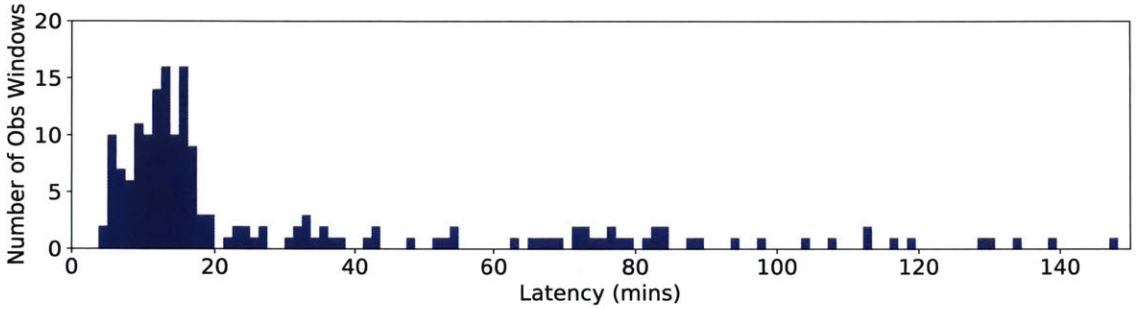
larly for SSO Ring. Table 6.10 shows that while median value is significantly lower for SSO Ring, the difference or spread between the P_{25} and P_{75} values (the inter-quartile range) is not lowered quite as much (from about 59 minutes to about 42 mins for regular xlnks and 58 mins for constrained xlnks). The maximum latency values are cut in half from Dlnk Only. For Walker, the reduction in inter-quartile range is more significant in both crosslink cases (from about 31 minutes to about 4 mins for regular xlnks and 11 mins for constrained xlnks).

These results show that the use of crosslinks, even when constrained in directionality and by inter-activity transition time requirements, can significantly reduce both the median and spread of latency for initial observation downlink. The reduction in median value is more pronounced for the SSO Ring because of the more limited downlink access time relative to the Walker constellation (because SSO Ring ground stations are only at the poles). The plots in figs. 6-9 to 6-12 provide more insight into this behavior, showing histograms and CDF distributions for the latencies over all executed observation windows. The histograms reveal that the use of cross-links significantly reduces the "tail" part of the latency distribution. For SSO, we see a first cluster of latencies around 20 minutes and the second cluster around about 80 minutes. This is due to the placement of ground stations at the poles; the satellites often collect data and then have to wait until they pass over ground stations at the poles again to downlink. The orbit period for the constellation is about 95 minutes. The cluster is not exactly at the orbit period because there are ground stations at both poles, and observation data is collected between -30°and positive 30°latitude for these simulation cases. We still see some amount of this clustering behavior with the crosslinked cases, but it is much smaller. With Walker, the ground stations are distributed fairly evenly between -30°and positive 30°latitude, so clustering is not seen.

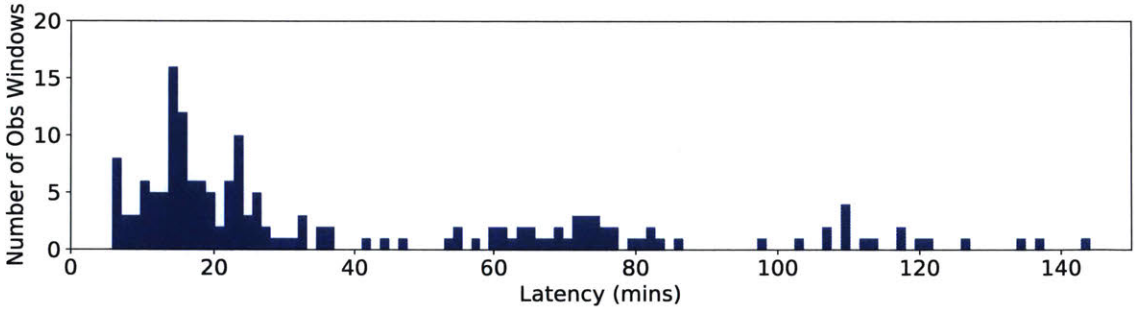
The CDF plots in figs. 6-11 and 6-12 highlight the effectiveness of cross-links for shifting the latency distribution. For both SSO and Walker, the curves for both Dlnk + Xlnk contexts are significantly higher than that for Dlnk Only. This is particularly true for SSO Ring, where 60% of observations are delivered within about 15 minutes



(a) Dlnk Only

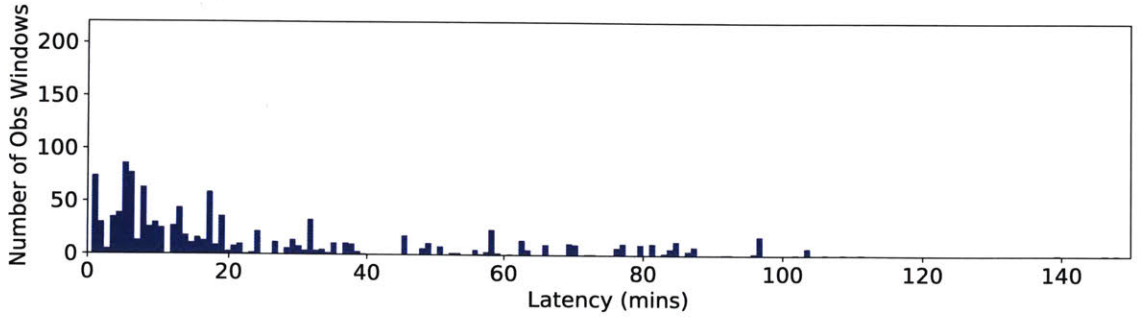


(b) Dlnk + Xlnk

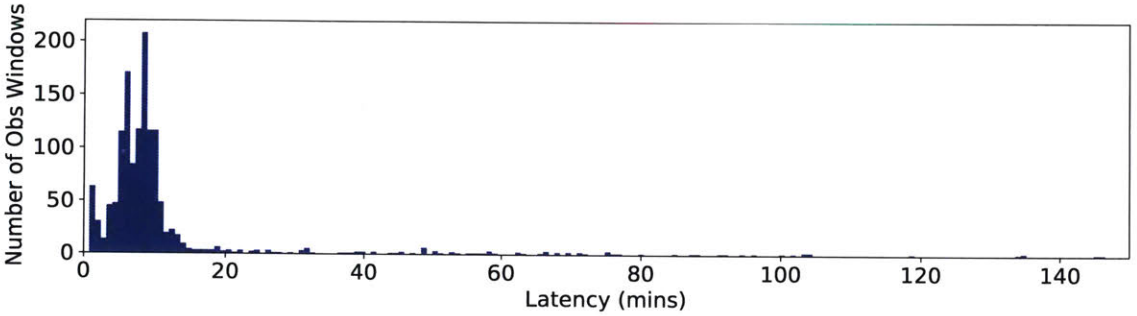


(c) Dlnk + Xlnk, constrained

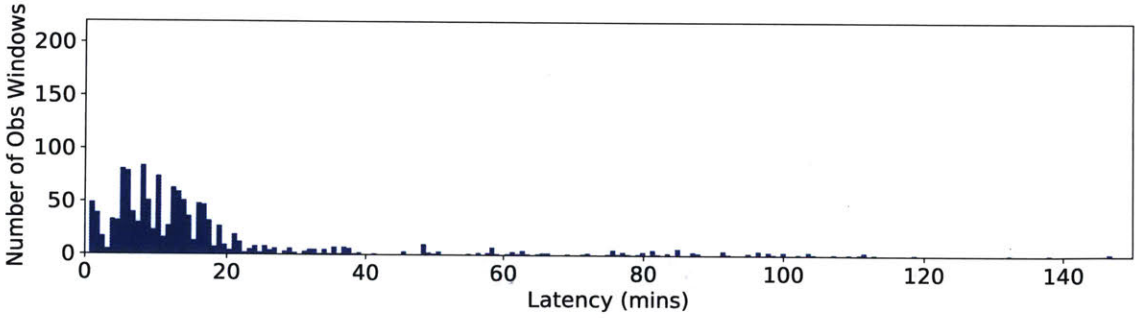
Figure 6-9: Histograms of initial observation execution latency for the three SSO Ring communications contexts. Sim run for 24 hours, with GP-Fast, 40 observation targets. See appendix A.2 for other parameters. The latency distribution is concentrated at lower values with crosslink usage.



(a) Dlnk Only



(b) Dlnk + Xlnk



(c) Dlnk + Xlnk, constrained

Figure 6-10: Histograms of initial observation execution latency for the three Walker communications contexts. Sim run for 24 hours, with GP-Fast, 40 observation targets. See appendix A.2 for other parameters. Similar to SSO Ring, latency distribution is concentrated at lower values with crosslink usage.

for Dlnk + Xlnk, and within about 70 minutes for Dlnk Only.

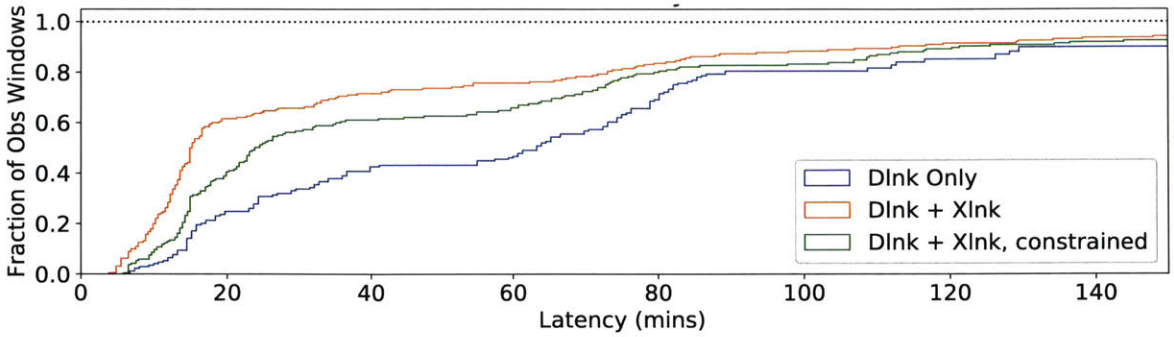


Figure 6-11: Cumulative distribution function of initial executed observation latency for SSO Ring sim case, in all communications contexts. Sim run for 24 hours, with GP-Fast, 40 observation targets. See appendix A.2 for other parameters.

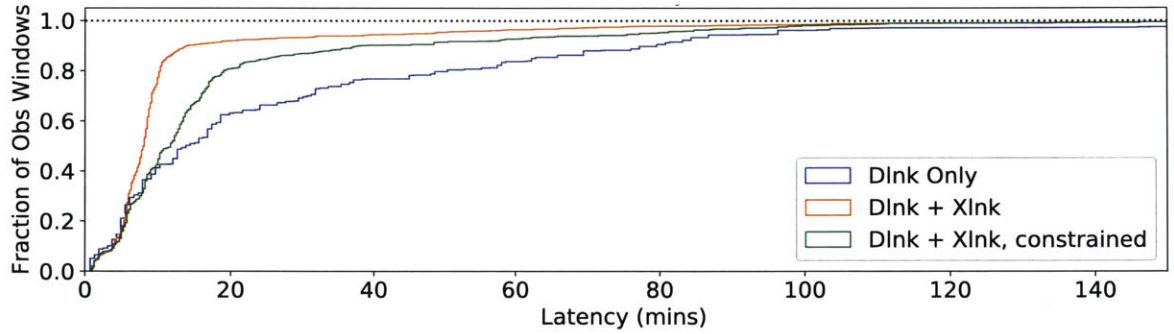


Figure 6-12: Cumulative distribution function of initial executed observation latency for Walker sim case, in all communications contexts.

6.2.3 Observation Age of Information (AoI) Comparison

Figure 6-13 and tables 6.11 and 6.12 present results for observation-target-average AoI for the simulation cases. The "observation-target-average" qualifier signifies that AoI is averaged for a given observation target, before statistics are calculated across all observation targets. The "at collection" metric is the average AoI determined by setting an observation target's AoI to 0 when an observation window for that target is executed. The "with routing" metric only refreshes target AoI when observation data is downlinked, at which time the age is set to the time since that data was collected

at the observation window execution. The "with routing" metric includes the effects of network latency on average AoI.

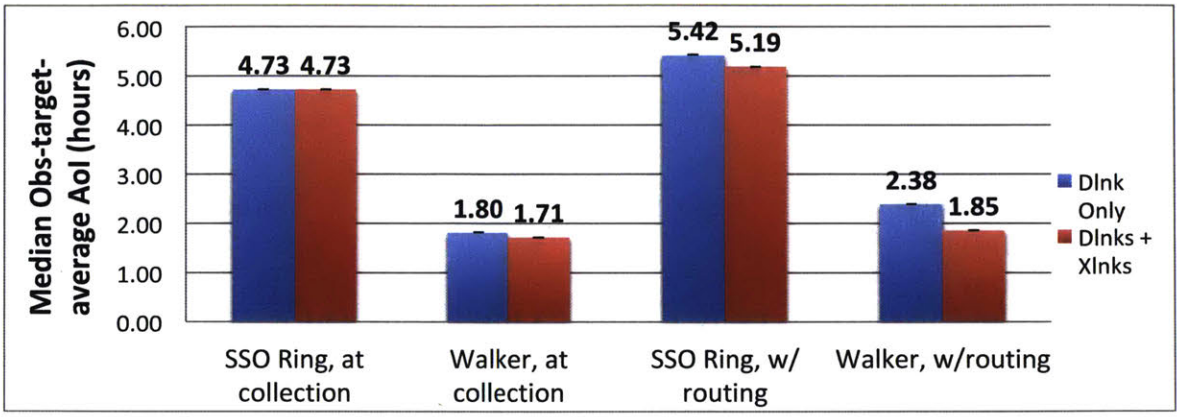


Figure 6-13: Median of target-average AoI values over all observation targets. Sim run for 24 hours, with GP-Fast, 40 observation targets. See appendix A.2 for other parameters. "at collection" updates AoI to 0 at obs execution, "w/ routing" updates at data downlink to current age of obs data. Crosslink usage lower average AoI, though AoI is bounded by constellation geometry and availability of obs overpasses.

Median average AoI is significantly lower for the Walker constellation, as is expected due to the larger number of satellites and the 30°inclination orbits for this constellation. We would at first expect the median values to be the same for Dlnk Only and Dlnk + Xlnk in the "at collection" metric, because routing is not involved in calculating the average AoI in this case. This is indeed true for Dlnk Only, with the median value of 4.73 hours in both communications contexts. But the median value is slightly lower for Walker (1.71 versus 1.80 hours) because the Walker constellation is able to execute more observation windows when crosslinks are used, as discussed in section 6.2.1. For each constellation, the "with routing" metric is higher than "at collection", as expected. However, the change is smaller when crosslinks are used. For example, for Walker the median value increases from 1.80 to 2.38 hours in the Dlnk Only context, but from 1.71 to 1.85 hours in the Dlnk + Xlnk context. This shows that crosslinks can be used effectively to keep average AoI lower when considering the effects of routing.

Note that the changes in these average AoI values across the simulation cases

Table 6.11: Summary of Average AoI Results, At Collection, for Constellation Sim Cases from Figure 6-13

Target-Average AoI Metric (hours)	**SSO Ring**		**Walker**	
	Dlnk Only	**Dlnk+Xlnk**	**Dlnk Only**	**Dlnk+Xlnk**
Min	4.29	4.29	0.61	0.61
Median	4.73	4.73	1.80	1.71
Max	6.02	5.60	3.10	2.99

Table 6.12: Summary of Average AoI Results, with Routing, for Constellation Sim Cases from Figure 6-13

Target-Average AoI Metric (hours)	**SSO Ring**		**Walker**	
	Dlnk Only	**Dlnk+Xlnk**	**Dlnk Only**	**Dlnk+Xlnk**
Min	4.89	4.46	0.70	0.74
Median	5.42	5.19	2.38	1.85
Max	11.21	7.19	3.56	3.07

are relatively small compared to those seen for latency. This is due to the inherent base value for average AoI that is fixed by the constellation geometry and its resultant access periods for the observation targets. This base value is the "at collection" metric. This value is highly dependent on constellation design, and a full investigation of the span of distribution of possible values will require simulation of likely many additional constellation geometries. This is left to future work. However we might expect to see much more pronounced differences in average AoI for a very large constellation (say, hundreds of satellites) with few ground stations, because the observation targets would be observed frequently, but data delivery would be much less frequent. in such a case, crosslinks could be critical for lowering average AoI for increasing data refresh rate for the observation targets.

6.2.4 Satellite TT&C Age of Information (AoI) Comparison

Figures 6-14 and 6-15 and tables 6.13 and 6.14 summarize the average AoI results for satellite commanding and telemetry (TT&C) data. In this case, average AoI was determined for every satellite, and then statistics were calculated across all satellites. Recall that satellite command AoI is reset to 0 whenever an update from any ground station arrives at a satellite, and satellite telemetry AoI is reset to 0 whenever an update from a given satellite arrives at any ground station. These updates propagate through the constellation network in a "flooding" procedure, see 5.1.1 for more details.

Figures 6-14 and 6-15 present the maximum satellite-average TT&C AoI values, because this was considered to be the most important metric for constellation operators; it tells them the maximum expected time between command or telemetry updates. Results are also presented for the Dlnk + Xlnk, constrained context because this context exerts additional constraints that affect communications availability.

For both SSO Ring and Walker, crosslinks lower average AoI to a large degree. For example, maximum command average AoI is decreased from 1.80 to 1.07 hours for SSO Ring and from 1.04 to 0.41 hours for Walker. The Dlnk + Xlnk, constrained context does not perform significantly worse than Dlnk + Xlnk, and still shows a large performance gain over Dlnk Only.

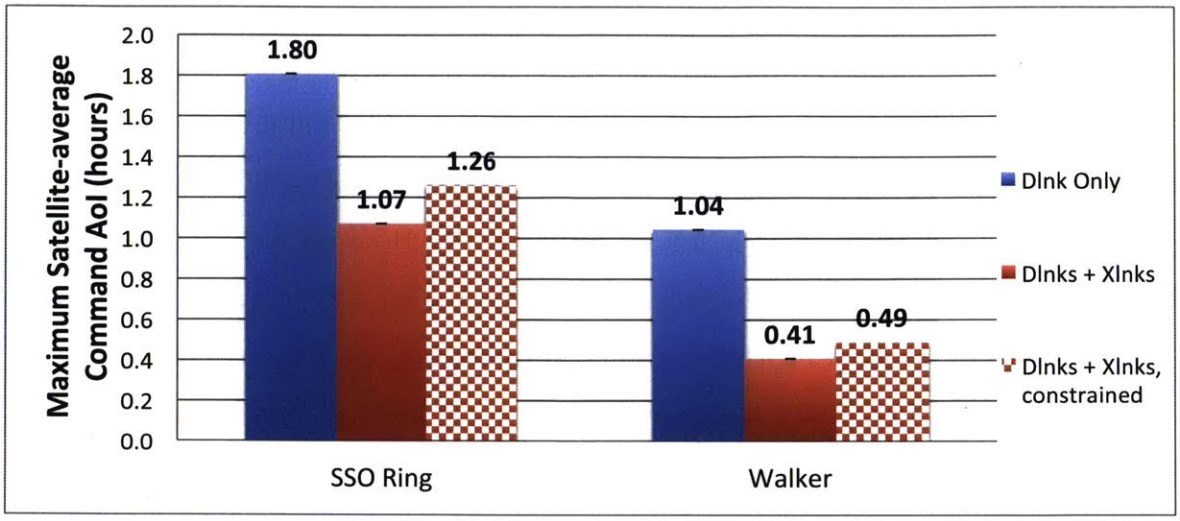


Figure 6-14: Maximum of satellite-average command AoI values over all satellites. Sim run for 24 hours, with GP-Fast, and 9 ground stations. See appendix A.2 for other parameters. Walker has 30 satellites in 3 orbit planes versus 10 in one plane for SSO Ring, resulting in lower average AoI for Walker.

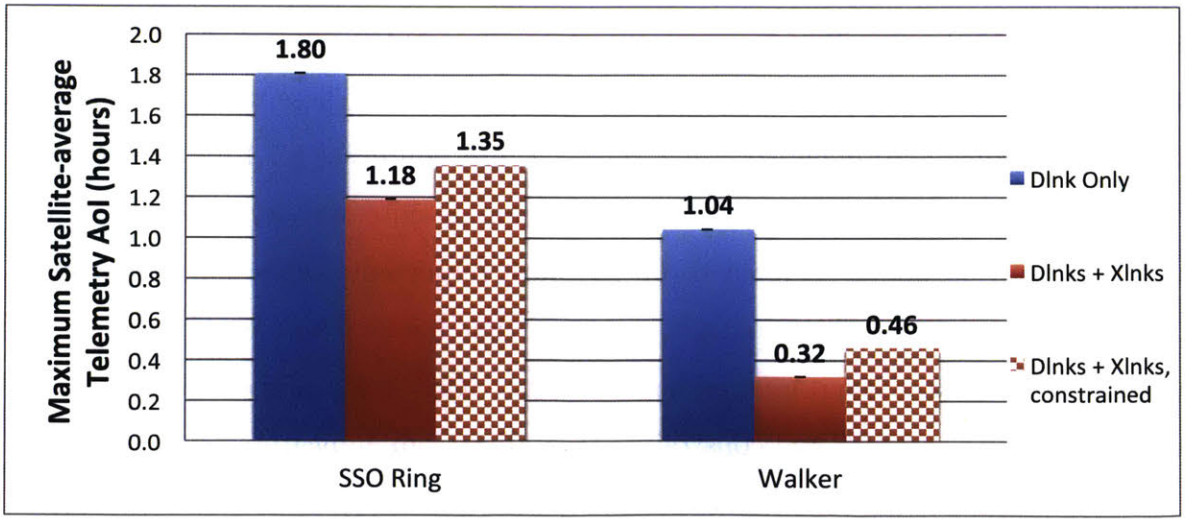


Figure 6-15: Maximum of satellite-average telemetry AoI values over all satellites, with same parameters as fig. 6-14.

Table 6.13: Summary of Satellite TT&C Results from Figure 6-14: Satellite-average Command AoI

Satellite-Average Command AoI Metric (hours)	**SSO Ring**			**Walker**		
	Dlnk Only	**Dlnk+ Xlnk**	**Dlnk+Xlnk Constr.**	**Dlnk Only**	**Dlnk+ Xlnk**	**Dlnk+Xlnk Constr.**
Min	1.17	0.79	0.89	0.50	0.25	0.36
Median	1.29	0.90	1.04	0.59	0.31	0.40
Max	1.80	1.07	1.26	1.04	0.41	0.49

Table 6.14: Summary of Satellite TT&C Results from Figure 6-15: Satellite-average Telemetry AoI

Satellite-Average Telemetry AoI Metric (hours)	**SSO Ring**			**Walker**		
	Dlnk Only	**Dlnk+ Xlnk**	**Dlnk+Xlnk Constr.**	**Dlnk Only**	**Dlnk+ Xlnk**	**Dlnk+Xlnk Constr.**
Min	1.17	0.70	0.69	0.50	0.23	0.29
Median	1.29	0.82	0.89	0.59	0.27	0.34
Max	1.80	1.18	1.35	1.04	0.32	0.46

With Dlnk Only, the requirement for TT&C average AoI, <0.5 hours, is not met. With cross-links, the requirement is met for the Walker constellation when using cross-links. Overall, Walker has lower TT&C AoI than SSO Ring in both communications contexts due to the presence of significantly more downlink access time. SSO Ring is much more limited in performance by the fact that there are "blackout periods" during the 24 hour simulation when the orbit passes between the ground stations at the Earth's poles and no ground station overpasses occur. This causes the AoI values for both telemetry and command to grow large for a time before they get reset when the orbit again passes over ground stations. This problem is not present for the Walker constellation due to the generally high availability of ground station accesses with its low inclination orbits. This finding points to the need to select orbits and ground stations with crosslink communications access in mind; crosslinks are able to lower TT&C significantly when downlink accesses are available, but are of little help when not available.

No objective function term was included in GP-Fast to explicit prioritize lower TT&C AoI. These results simply present the average AoI values that resulted from opportunistically utilizing the links that were scheduled for bulk data routing. If the links could be rewarded for their use in propagating TT&C updates, it is expected that even better AoI values could be achieved in the crosslinked communications contexts. Note that this would likely require additional modeling work to include low-throughput data routes that are primarily for propagating low bandwidth TT&C data. This is included as an item for future work in section 7.2.

6.2.5 Executed Link Capacity Comparison

As an additional metric, we present the amount capacity utilized for both downlinks and crosslinks for the two sim cases and the Dlnk Only and Dlnk + Xlnk communications contexts.

Figure 6-16 and 6-17 as well as table 6.15 and table 6.16 summarize these results. The total available downlink and crosslink capacity is not deconflicted in these numbers; *i.e.*, some of this capacity might actually be overlapped and impossible to fully

schedule. These numbers do give a good estimate though of the potential available throughput for both types of links.

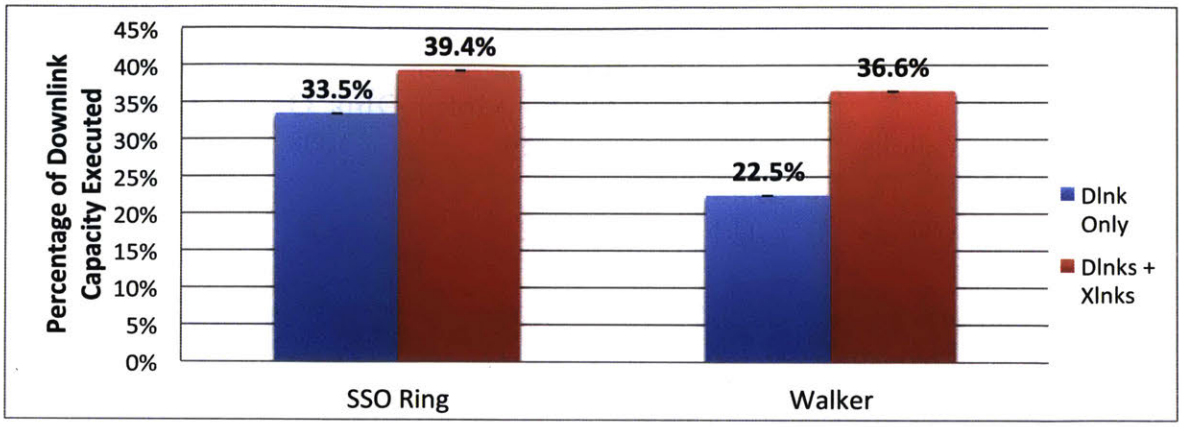


Figure 6-16: Percentage of downlink data volume capacity utilized in constellation sim cases. Sim run for 24 hours, with GP-Fast, 9 ground stations, 40 observation targets. See appendix A.2 for other parameters.

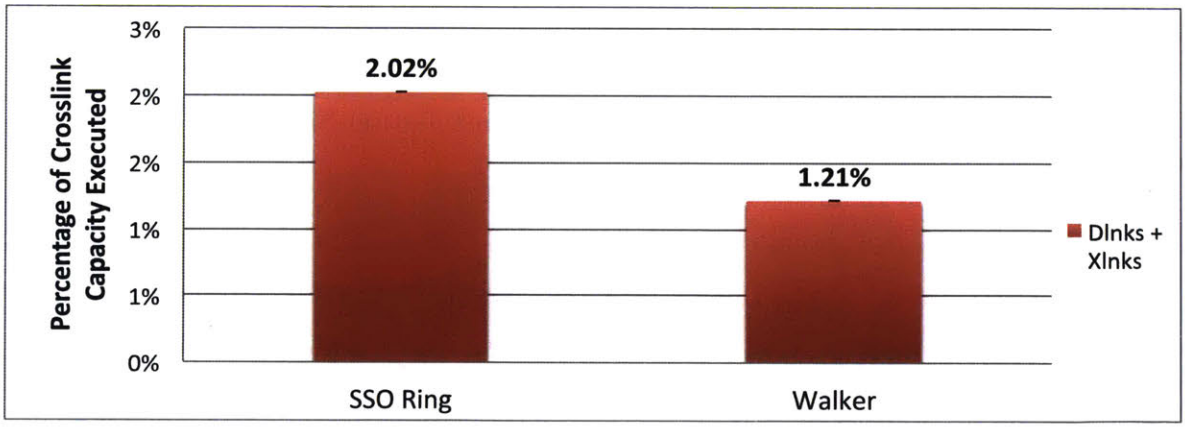


Figure 6-17: Percentage of crosslink data volume capacity utilized in constellation sim cases. Sim run for 24 hours, with GP-Fast, 40 observation targets. See appendix A.2 for other parameters.

We see that under 50% of the available downlink capacity is executed, in all sim cases and communications contexts. Very little of the available crosslink capactity is executed, less than about 2% in both cases. These results show that there is likely significantly more room for observation data collection and routing in these constellations. The large amount of unused crosslink capacity suggests that the xlnk

Table 6.15: Downlink Activity Utilization for Constellation Sim Cases from Figure 6-16

Metric	SSO Ring		Walker	
	Dlnk Only	**Dlnk+Xlnk**	**Dlnk Only**	**Dlnk+Xlnk**
Number of downlinks available	160	160	1208	1208
Number of downlinks executed	112	124	851	1036
Percent of available executed	70.0%	77.5%	70.4%	85.8%
Total downlink capacity (Tb)	3.26	3.26	27.73	27.73
Downlink throughput executed (Tb)	1.09	1.29	6.25	10.14
Percent of available executed	33.5%	39.4%	22.5%	36.6%

Table 6.16: Crosslink Activity Utilization for Constellation Sim Cases from Figure 6-17

Metric	SSO Ring		Walker	
	Dlnk Only	**Dlnk+Xlnk**	**Dlnk Only**	**Dlnk+Xlnk**
Number of crosslinks available	14400	14400	107788	107788
Number of crosslinks executed	0	583	0	2017
Percent of available executed	0.00%	4.05%	0.00%	1.87%
Total crosslink capacity (Tb)	5.54	5.54	64.87	64.87
Crosslink throughput executed (Tb)	0.00	0.11	0.00	0.79
Percent of available executed	0.00%	2.02%	0.00%	1.21%

system is actually overly performant for these constellations, and similar operational performance may be achievable with a less capable transceiver. Note that these metrics do not capture the fact that most crosslink usage is concentrated immediately after observations are performed, to achieve low-latency data delivery.

6.3 Algorithm Performance with Injected Observation Events

In the final simulation analysis, we study algorithm and constellation performance when injected or "urgent" observations are present [5]. The 6-Sat constellation design was used for this study, simulated for 24 hours. The constellation is described in section 2.6 and sim parameters are specified in appendix A.1. A baseline simulation case, "No Injects", was run with no injected observations present. A second simulation case, "Injects", was run with a randomly-distributed set of injected observation windows present, as detailed in table A.17. The satellites had no prior knowledge of injected observations, they simply executed the observation activities when prompted and ran the Local Planner afterwards to determine how best to route the data collected from these observations. In at least one case, an injected observation conflicted with and was allowed to pre-empt a pre-existing planned activity for a satellite. The activity was simply canceled and ceased to execute. The collected data volume for the injected observations was set low (300 Mb, versus the usual observation capacity of several Gb or more) so that the satellites would not be overly perturbed by a large amount of excess unexpected data that does not contribute to meeting the 100 Mb initial "snapshot" requirement for a given observation window (see 2.3.2 for additional context).

Table 6.17 summarizes the initial observation data delivery latency statistics for executed observations in the simulations, with the same 100 Mb minimum data vol-

[5]Results were obtained with commit 3409a7c10454f6d025d4692a1b314024dfd43f7e ("No Injects" case) and f4341a8ea62cfbc26a753f788b6c7df965cdb370 ("Injects" case) of top-level CIRCINUS repository at `https://github.mit.edu/star-lab/CIRCINUS`

Table 6.17: Summary of Initial Observation Data Downlink Latency for No Injects and Injects Sim Cases

Latency Metric (mins)	No Injects Case	Injects Case	
	Pre-planned Obs	Pre-planned Obs	Injected Obs
Min	2.67	2.83	7.29
P_{25}	9.15	14.86	18.64
Median	19.95	37.99	42.87
P_{75}	55.82	77.32	85.72
Max	319.99	428.42	414.94

ume requirement as in fig. 6-8. For No Injects, only pre-planned observations were present. For Injects, both the pre-existing pre-planned observations and the injected observations were present. In both No Injects and Injects, 32 pre-planned observations were executed, with a total delivered data volume of 165 and 157 Gb for the two cases, respectively. In Injects, 21 of the 28 possible injected observations were executed for a total data volume of 4.6 Gb. This shows that the execution and routing of the injected windows did not have a significant effect on pre-planned observation routing performance. Note that when they were generated, no attempt was made to ensure that the injected observation activities would actually be deliverable. Some of them took place at the end of the 24 hours when no subsequent down links were available, and thus the observations were not deliverable. It was confirmed that in Injects the satellites did not exhibit significantly less average data margin than the No Injects case (median satellite-average data margin decreased from 69% to 67%).

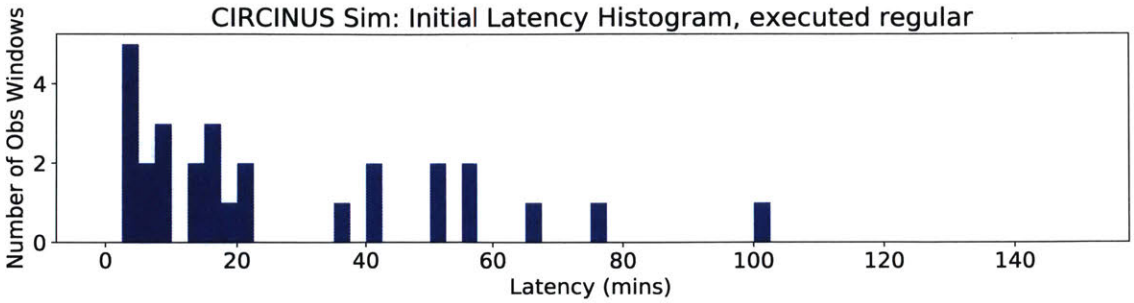
We see from table 6.17 that the injected observations negatively affected latency for pre-planned observations, increasing the median latency from 19.95 to 37.99 minutes, because they took over some of the routes that had been planned for the pre-planned observations. Injected observations appear to perform about on par with pre-planned observations though, which is expected. The histograms in fig. 6-18 detail the distributions of latencies for the different types of observations. We see that the distribution of pre-planned observations has been slightly flattened and spread

out in the Injects case. Injected observations tended to cluster at lower latencies with a few higher latencies spread beyond. Note that latencies longer than 150 minutes are not displayed here.

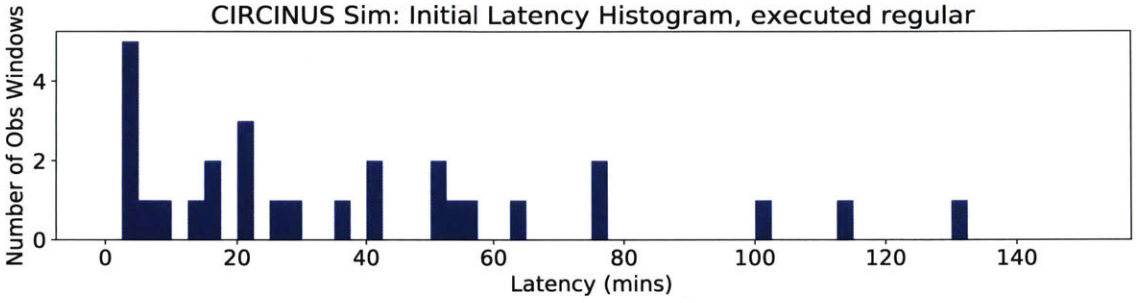
Figure 6-19 shows the cumulative distribution functions for the different observation window types. Again, we see that the distribution of pre-planned observations shifted slightly to the right, but injected observations perform on par with pre-planned observations.

These results show that the Local Planner is able to effectively utilize pre-existing data routes planned by the GP for pre-planned observations and use them to route urgent, injected observation data. Currently, the LP is limited to existing data routes and cannot create new data routes of its own. If the LP could create its own data routes, then performance for injected observations could be improved even more. This is included as a suggestion for future work in section 7.2. Importantly, servicing of injected observation data did not significantly hinder routing of pre-planned observations in terms of both total throughput and latency.

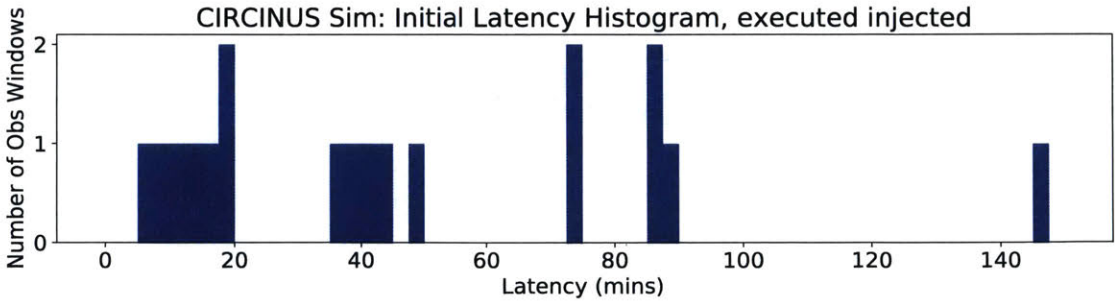
Also recall that these results assumed the presence of an external communications backbone (see section 5.1.3). While we do not investigate a completely independent constellation with planning information sharing exclusively over intra-constellation links, these results demonstrate that with sufficiently fresh planning information from the GP, the LP makes routing decisions effectively.



(a) No Injects, Executed Pre-planned Observations



(b) Injects, Executed Pre-planned Observations



(c) Injects, Executed Injected Observations

Figure 6-18: Histograms of initial observation execution latency for No Injects and Injects sim cases from Table 6.17. Each sim run for 6-Sat case for 24 hours with GP-Fast. 32 observation windows were present in No Injects case, and 28 for Injects. See appendix A.1 for all 6-Sat parameters, and A.4 for all injected obs parameters.

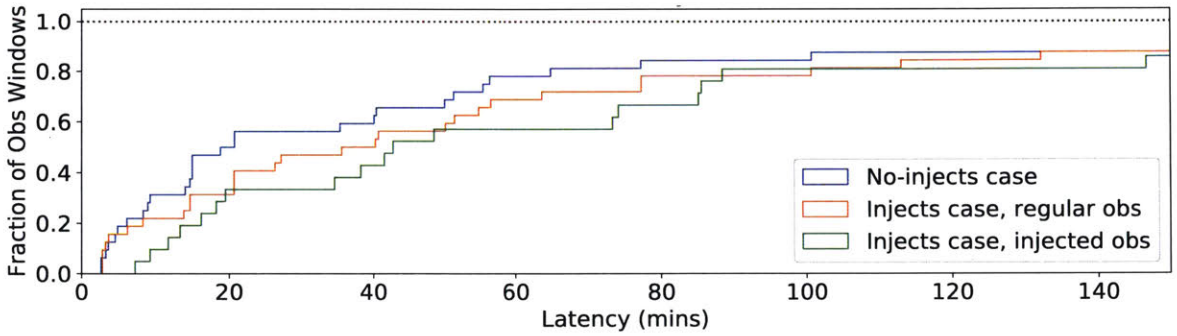


Figure 6-19: Cumulative distribution function of initial executed observation latency for No Injects and Injects sim cases from Table 6.17. Latency is displayed only from 0 to 150 minutes for clarity. Note that the distribution of injected obs latency is close to that for pre-planned, "regular" obs.

Chapter 7

Conclusion

7.1 Summary of results

Large-scale, Earth-observing small satellite constellations promise great benefits due to the additional spatial and temporal diversity they can provide. Traditional human-in-the-loop operations approaches will not scale to the constellations of potentially hundreds of networked satellites envisioned for the future. Operating these constellations is particularly challenging due to their intermittent communications and limited onboard resources. For this reason, automated planning and scheduling is needed.

Previous work has demonstrated the effectiveness of scheduling of observation tasks, routing data in networks with intermittent communications links, the management of limited onboard resources (energy, data storage), and onboard replanning for changing circumstances. However, existing approaches have not 1) demonstrated the ability to execute successfully for large problem sizes (long planning horizons, large constellations), 2) demonstrated an onboard replanning approach that leverages a centralized, ground-based algorithm for data routing scheduling with the full set of relevant constraints included, or 3) examined many of the metrics of interest for constellation operations.

This thesis formulates and implements an integrated planning and scheduling system that addresses the main operational constraints for an Earth-observing, crosslinked small satellite constellation, and simulates the execution of that system in repre-

sentative constellation geometries and scenarios to assess performance. The major contributions of this work are summarized below.

7.1.1 Contribution 1: Software Infrastructure Creation

The CIRCINUS simulation pipeline and the Constellation Simulator were developed and tested. The simulation pipeline facilitates the propagation of orbit geometries, calculation of communications link rates, running of the constellation simulation, and visualization of results. While this software currently depends on one commercial tool, the MATLAB language and environment, the backbone is written in Python and may be adapted to run entirely without for-purchase tools. The Constellation Simulator simulates the execution of the plans created by the planning and scheduling (P&S) algorithms developed in this thesis. It serves as a mechanism to separate operations execution from planning, and helps verify that P&S algorithms are able to continually produce feasible schedule solutions in a time-evolving manner, rather than only as one-off schedule artifacts from a possibly non-representative initial constellation state. The Constellation Simulator was designed to be as agnostic as possible about the P&S algorithms themselves, and handles all state information associated with agents (satellites, ground stations, the ground station network) in the simulation environment. Accurate execution of the constellation simulator was verfied in section 5.4. The software is currently versioned within multiple Git repositories, with a single unifying, wrapper repository that can run all components. We intend to make the code available to the general public. [1]

[1] top-level CIRCINUS repository: `https://github.mit.edu/star-lab/CIRCINUS` (only accessible to MIT persons). See STAR Lab website for details on open-source availability `http://starlab.mit.edu/`. Note we intend to preserve the Git commit history in the open source code, so that commit values referenced in this thesis remain valid.

7.1.2 Contribution 2: Planning and Scheduling Algorithm Development and Scalability Demonstration

The centralized, ground-based Global Planner algorithm and the onboard Local Planner algorithm were formulated, implemented, and evaluated in this thesis. The fast version of the Global Planner, GP-Fast, was validated against an optimal variant, GP-Optimal, and found to produce slightly worse, but acceptable quality schedules while running significantly faster. For multiple planning window sizes with a 6 satellite constellation, constellation throughput was above 92% of optimal and latency was within 92% and 104% of optimal (section 6.1.1, table 6.2) [2]. For the 1000 minute window, GP-Fast runtime was 46.5 seconds and GP-Optimal runtime was 7386 seconds (table 6.1). GP-Fast demonstrated execution on larger problem sizes than the state-of-the-art, successfully running with much larger planning windows (up to 1000 minutes versus 120 minutes for Zhou *et al.* [124]) and on much larger constellation scenarios (100 satellites versus 6 for Zhou *et al.* [124] and 18 for Kondrateva *et al.* [71]). With moderate parallelization it can produce a schedule for 100 satellites over a 120 minute planning window in under 30 minutes of execution time (about 1200s).

7.1.3 Contribution 3: Algorithm performance analysis in representative EO constellation cases

The GP-Fast algorithm was simulated with two representative constellation cases, SSO Ring and Walker (with parameters summarized in A.2). One simulated a case with relatively limited earth surface coverage and downlink availability, while the other had relatively more. Table 7.1 summarizes key findings from the performance analyses. Each entry in the table should be read as the change in the metric value when switching from the Dlnk Only communications context to the Dlnk + Xlnk communications context. In all cases, metrics were improved when using crosslinks. Dlnk + Xlnk assumed no inter-activity transition time constraints. A third communications context, Dlnk + Xlnk, constrained, was investigated to simulate constraints

[2] Latency for GP-Fast is generally worse than the optimal value, hence values larger than 100%

for a 6U satellite bus with more limited receive and transmit capabilities (*e.g.* only able to transmit in the velocity direction for intra-orbit crosslinks); performance this for this case was slightly worse than Dlnk + Xlnk, but still significantly better than Dlnk Only in all metrics.

The large increase in data throughput for Dlnk + Xlnk was largely due to the conservative data storage allowance (4 GB). While perhaps too constrained, the results illustrate the ability of the Global Planner to effectively route data by better balancing data buffering across the constellation.

Table 7.1: Summary of Key Results From Constellation Simulation Cases

Item	SSO Ring	Walker
Percent of potential data throughput	59.77% to 70.32%	43.62% to 70.49%
Percent of potential observations executed	85.64% to 94.55%	80.01% to 90.63%
Median satellite-average data margin	49.0% to 60.8%	31.8% to 56.2%
Median observation data latency	63.24 to 15.2 mins	14.24 to 8.07 mins
Median target-average AoI, with routing	5.42 to 5.19 hours	2.38 to 1.85 hours
Median satellite-average command TT&C AoI	1.29 to 0.90 hours	0.59 to 0.31 hours
Median satellite-average telemetry TT&C AoI	1.29 to 0.82 hours	0.59 to 0.27 hours

7.1.4 Contribution 4: Demonstration of urgent, "injected" observation data routing

The utility of the Local Planner was demonstrated in the 6-Sat simulation scenario with injected observation events. The Local Planner was able to successfully route this spontaneously arising data, utilizing slices of throughput from the data routes that had been previously planned by the Global Planner. Median latency for injected

observations was 42.87 minutes, versus 37.99 minutes for regular observation data. Results from this run were compared to a simulation without injected observations. While it was found that the injected observations increased latency for the existing observation plans (median latency increased from 19.95 minutes to 37.99 minutes, and interquartile range increased from 46.67 to 62.46 mins.), all of the existing observations were successfully executed, demonstrating that the LP does not significantly reduce the quality of the original schedule produced by the GP.

In addition, the LP exhibited much shorter execution times, under 10 seconds in general (see section 4.3). It is able to make decisions for injected data routing almost instantaneously, much faster than running the GP again for the full constellation. This difference in execution time between the GP and LP speaks to the importance of a multi-layered P&S system: the centralized layer makes decisions from a full constellation perspective and balances resource usage in the long term, whereas the onboard layer can leverage the previous decision made (data routes created) by the centralized layer to quickly reprioritize data routing to handle changed circumstances. Though the results presented here assumed an external communications network to distribute planning information, they show that the LP still plays an important role by reducing the need to frequently replan with the GP.

7.2 Future work

Upgrades to Local Planner

Currently the LP can only take data volume from data routes and activity windows that have already been scheduled by the GP. It would be a useful capability for the LP to introduce new scheduled activities itself and create entirely new data routes through them. For the injected observations simulation case that was presented, it may even be possible to fully route all injected observation data volume through new activity windows scheduled by the LP, thus avoiding negative effects on already-planned observation data routing. Such a mechanism has an important side-effect: if one satellite introduces a new data route, then it is important to know that the rest

of the satellites along the route will actually execute it. Some form of consensus, or at least hierarchical delegation of authority to satellites to make changes to plans, will be needed to ensure that these new data routes are successfully executed.

Many interesting architectures could be envisioned for a consensus mechanism, including a completely distributed consensus like that used by Choi *et al.* [18], or the inclusion of certain "team planner" (TP) satellites that are allowed to make decisions for their neighbors within the constellation network topology. In this work, we focused on the LP's use for routing injected data, with little to no subsequent interaction between the LP and GP. In an architecture with multiple team planners, there would likely need to be much more interaction between the LP and TP instances, to ensure that the increased decentralization of decision making does not overly reduce global schedule quality. In addition, the LP or other onboard health assessment algorithms could be useful for informing the GP of the state of health of satellites within the constellation. The work by Zheng *et al.* illustrates a potential approach for including health assessment [123].

Improvements to Route Selection

The route downselection algorithm (RS Step 2) in this thesis does not scale particularly well with problem size. As noted in 6.1.2, when GP-Fast was run with 80 observation targets for the 30 satellite Walker case, RS step 2 took around 3000 seconds to execute. A more scalable approach might involve a closed-loop coupling of downselection and activity scheduling. This could be done in a "gradual commitment" method, where a small set of down-selected routes are provided to Activity Scheduling, the results of which are used to seed the selection of a larger set, which is provided to Activity Scheduling, and so on until schedule quality stops improving significantly. A genetic algorithm might be a good implementation, because it could mix the population of scheduled routes at each iteration with new route possibilities. Note that this genetic algorithm would effectively be an additional optimization layer wrapped around the Activity Scheduling optimization.

Another benefit of this approach would be improved inter-activity transition time

handling. The current route selection algorithm performs well for constructing data routes when inter-activity transition times are not required or are small, but it performs poorly when the times are large. This is because it doesn't currently fully compute the temporal overlap between two routes when transition times are required, only checking whether given activity windows are present in both routes. Route downselection is not particularly robust to situations where certain subsets of data routes are more easily joint-schedulable than others (*e.g.* when the selection of one route tends to bias the satellite to point in a certain direction at a certain time); it simply considers routes individually. By iteratively changing the set of downselected routes considered by Activity Scheduling, an optimization layer wrapper might more effecively be able to choose a final set of downselected routes.

Higher Fidelity Modeling of Observation Targets

In this work, observation targets are modeled as basic latitude, longitude points and the satellites are assumed to collect arbitrary, equally valuable observation data at any point during an overpass of a target. For more realistic applications, it would be useful to include a higher fidelity model of satellite pointing relative to targets, and the ability to place more weight on collecting data during certain periods of an overpass or with certain observation geometries. Other EO instrument operations concepts should be considered, including continuous scanning instruments, configurable Field of View restrictions, and specific ground lighting constraints.

These observation concepts were not investigated in depth in this work because we anticipate they will require a much finer temporal granularity of observation windows than the current version of the GP can handle, particularly at RS2. A more robust route selection algorithm will help extend the capability of GP-Fast to handle more observation windows. Also, observation windows could be decoupled from data routes: routes could be constructed at regular time intervals on a satellite rather than only at the end of an observation window, and then data from any observation window within a given interval would be available for routing at the end time of that interval. We call this the "observation window aggregation" approach. This approach would

allow for many more observations to be included because they would not all generate their own data routes, and might cut down a large amount of redundancy between routes.

More Responsive Observation Execution

It would be good to allow "agile" observation targeting (term from [74]), where a satellite can roll or pitch from a nominal attitude to access more observation targets from a given orbit vantage point. Though even with a more streamlined RS algorithm, the inclusion of explicit decision making about satellite pointing in the GP planner would add a great deal of computational complexity. One potential approach for this would be to include significant pre-processing of the pointing angles for activity windows, generating a large lookup table for transition time requirements between each pair of activities. This would require a large amount of memory for large problem sizes (every activity window must be compared with every other one) but would be highly parallelizable.

Coordinated observation timing across satellites would be fairly straight-forward to add to the GP model; constraints could be included that restrict observation activity execution on two satellites to happen at the same time or not at all (*e.g.* using the activity indicator, I_a, variables). In general, such constraints would actually lead to quicker solutions for the Activity Scheduling MILP by reducing the solution space. However, any increase in the number of observation windows would likely cancel this benefit.

A mechanism for observation target prioritization should be included. Zhou *et al.* use simple weighting factors for each target [124], however these do not handle hard requirements on observation window exection (*e.g.*a given obs must be scheduled before allowing any others to be scheduled). We examined these requirements somewhat in previous work [65], but only in the context of a simple greedy scheduling algorithm. For a more operationally-responsive model of observation target priority, something similar to the priority heat map discussed by Monmousseau for Planet Inc. [83], could be incorporated. Such a heat map could be used to model temporally and

spatially changing observations regions, to simulate both changing target knowledge and mission completion.

Explicit Optimization of Data Routes for Planning Information Updates, Investigation of Alternative Communications Architectures

The current version of the GP does not explicitly reward selection of data routes for maximizing planning information freshness (TT&C data), but this would be a useful mechanism to include. It would also be useful to have additional data routes available for scheduling that are not primarily for bulk data routing, but rather focused on planning information (PI) routing. This could avoid periods of reduced TT&C freshness that were seen in the constellation sim runs, caused by the lack of activities being executed for bulk data routing (*e.g.* because no observations or downlinks were present).

The results in this thesis assumed distribution of updated PI from the GP via an external communications network, but it would be useful to examine the case where no external network is available and all PI is shared over intra-constellation links. We were not able to model this effectively here because of the lack of optimization for PI freshness.

Integrated Bulk and Small/Bursty Data Routing

The primary focus of this work was bulk observation data routing scheduled in advance, as compared routing of smaller packets that might appear in a more unpredictable, bursty (large numbers for short periods) manner. While the Local Planner does incorporate handling of injected packets, it still is focused on bulk data, whereas other approaches focus on algorithms more tailored to smaller-packet cases [76, 35]. In future work, it might be useful to examine a two-layer data routing system that incorporates both route scheduling for bulk data and smaller, bursty data. The bulk data would likely be scheduled in advance and plans would be mostly fixed, whereas the "bursty" layer could operate on a slice of througput dedicated to rapidly-reprioritizable routing of small packets. Links would be scheduled between satellites

to ensure good network connectivity for the bursty layer, whereas the bulk routing layer would operate more in a store-and-forward manner, in parallel.

Additional Analysis of Observation AoI

Additional analysis of AoI for observation data is merited, in two aspects. The first aspect is the incorporation of a more explicit objective term in GP-Fast for optimizing observation AoI. Currently the AoI improvement for the Dlnk + Xlnk context (discussed in section 6.2.3) is a result of optimizing for low initial latency for every observation. We observed that because the algorithm seeks to execute as many observations as possible (*i.e.* to achieve delivery of $MinDV$, 100 Mb, for each obs), and the obseration snapshot data volume ($MinDV$) is low relative to downlink and crosslink throughputs, usually almost all the observation windows get scheduled successfully in a given GP-Fast execution for the sim cases examind in this thesis. In the general case, it is likely that GP-Fast will need to also consider past history of executed observation windows, to keep track of which observation targets have been seen, and which of them need an update to reduce AoI. There currently is no explicit mechanism incorporated in the Constellation Simulator code to keep track of this history, but our previous work [64] details an approach. Using this execution history, GP-Fast could potentially include an observation AoI objective term that favors an even temporal spread of executed observations for each target, perhaps by binning the observation activity indicator variables (I_a) into different time periods.

The second aspect of additional AoI analysis includes a more thorough examination of different constellation geometries, as well as larger observation $MinDV$ requirements. Constellation geometry has a large effect on observation AoI, and we expect that in order to achieve the < 1 hour average AoI goal for observation targets, a larger, more distributed constellation than the Walker case in this thesis would be needed.

More Fidelity in Satellite Model

The current satellite model does not include modeling of satellite pointing; all attitude constraints are abstracted through transition time requirements between activities. Inclusion of a dedicated pointing model, or additional pre-processed pointing information, in the GP-Fast algorithm and Constellation Simulator could improve the utility derived from observation activities by allowing a larger range of observation geometries to be considered. One extreme example is for BRDF (Bidirectional Reflection Distribution Function) measurement, where imaging from multiple angles is necessary to fully assess the properties of an observation target [87].

For the satellite energy model, we assumed a constant value for energy production when the satellite is out of eclipse, as opposed to varying solar power input based on satellite attitude. In future work, it would be useful to calculate input power values as a function of time given a nominal satellite attitude. It would be simple to incorporate these pre-processed values into the Global Planner algorithm as currently implemented (see inequalities 3.23 and 3.24, for example), though it would be much more computationally complex to include an input power value calculated from a model of satellite attitude. Also, we have modeled dedicated recharging periods for the satellite in past work [64], but did not include this activity type in the current model; these may be useful to re-integrate.

It would also be helpful to refine the transition times requirements between activities on a given satellite, or between downlinks executed by a ground station, with a more thorough analysis of setup and takedown times for the activities. There may be processing time required after an observation before data can be routed, or a minimum amount of time for setting up a link activity. In addition, it would be good to allow for execution of multiple activities at once on a given satellite.

When executing real communications links, satellites will experience occaisional connection dropouts. It would be useful to include a probabalistic model of link dropouts and examine the effect these have on schedule quality. The LP may be used to replan data routing to handle the effects of such dropouts.

Finally, the data storage constraints in the GP-Fast algorithm (inequalities 3.38 and 3.39) are overly conservative. They require the data volume for a given data route to be considered as present on a satellite for the entire original duration of any activity along the data route (not the scheduled duration). Moreover, they force both satellites executing a crosslink to count the data as stored onboard. A more refined model that allows a gradual reduction or increase of data storage over the course of an activity (perhaps modeled using the average data rate for the activity) would be more representative.

Optimization of Constellation Design with Multi-Displinary Systems Design Optimization (MSDO) Methodology

The algorithms and simulation infrastructure developed in this thesis may serve as tool for investigation of a wide range of different constellation geometries and operations parameters. To do this, an optimization layer could be wrapped around the CIRCINUS software pipeline. It would rerun the pipeline (or at least the constellation simulation part) over different constellation parameters and use the performance metrics returned by the simulation as guidance for new parameter choices. A number of different optimization approaches could be used, including heuristic algorithms such as a Genetic Algorithm or Simulated Annealing.

Promising investigations for this approach include: 1) varying the number of satellites in a given orbit plane or the number of planes (particularly to investigate the stepped buildup of a larger constellation from many launches) 2) varying the distribution of ground stations 3) varying communications parameters, particularly the power and pointing requirements of the crosslink payload 4) varying the amount of access to an external communications constellation and 5) varying design parameters to reflect different mission costs .

Modeling of Heterogeneous Constellations

In this work homogeneous constellations of similar payload instruments and communications links were simulated. A large potential benefit of small satellite constellations

is that they can unify many different assets of different types to operate in an overall system ("federation" or "fractionation"). For example, there could be a dedicated ground communications or crosslink relay nodes, or multiple different observation instrument types spread over different satellites. The algorithms and software tools developed in this thesis would serve as a good starting point for modeling a heterogeneous constellation. Many different heterogeneous designs could be investigated in an automated manner by applying the aforementioned MSDO approach.

Incorporation of Orbit Phasing Adjustment and Propulsion

Here we only considered orbits that are fixed over a timescale of about a day. Future constellations may wish to incorporate changes in orbit phasing (position of a satellite along its orbit) or propulsion. In general, the P&S system developed in this work can handle such capabilities by incorporating orbit changes in the pre-processing step when activity windows are calculated (the Orbit Propagation module in the CIRCINUS Simulation Pipeline, see section 2.5). Phasing techniques like differential drag take days or weeks to see significant effect [38], so the GP would likely just be run with an updated orbit specification each time. For large propulsive maneuvers, orbit changes can happen quickly; these effects would need to be included explicitly in the propagation of the satellites' orbits. The Orbit Propagation module currently does not include this capability, but only small changes would be needed to add it.

Application to Other Planning and Scheduling Problems

The algorithms developed in this work were specifically focused on bulk data routing in a smallsat constellation, however they could be applied to similar problems featuring delivery of data over a delay-tolerant network with constrained resources. One potential application is for Unmanned Underwater Vehicles (UUVs), where communication is often constrained due to vehicles operating underwater, and bulk data delivery can be focused in time periods where the vehicles come to the surface. The Global Planner formulation could be adapted to schedule the observation requirements for a set of resource-constrained UUVs while also planning for data delivery to

relay nodes such as satellites.

Another application might include satellites coordinating with Unmanned Aerial Vehicles (UAVs), both to provide increased situational awareness on the ground through a wider field of view from orbit, and communications availability for bandwidth limited or security-sensitive operations. The UAVs may introduce new scheduling constraints not present in the GP, such as real-time task reallocation amongst the UAVs, but they could potentially be modeled as a single agent that produces bulk observation data for offloading and requires frequent TT&C updates.

Within the area of satellite EO, the Global Planner algorithm could be augmented to include scheduling for multiple observation or communication payload spot beams on a single satellite. These could be modeled as additional, temporally overlapping observation windows, from which a satellite has to choose one or more to execute at a given time. This could lead to significant computational complexity, but a more scalable route selection approach (as previously detailed) could help to handle this case.

Appendix A

Simulation Case Parameters

Glossary:

- a - semi-major axis
- i - inclination
- Ω - Right Ascension of the Ascending Node
- ν - true anomaly
- e - eccentricity
- M - Mean anomaly

General notes

All target and ground station heights were fixed at 0 meters.

A.1 6-Sat Simulation Case Parameters

A.1.1 Parameters Overview

The full 24 hour scenario for 6-Sat is from 2016-02-14T04:00:00 to 2016-02-15T04:00:00.

Unless otherwise specified, the following parameters are used for GP-Fast in simulation runs. The number of routes chosen in downselection (RS2) are $\rho_1^{RS2} =$

$6, \rho_2^{RS2} = 6, \rho_3^{RS2} = 30$ ("highest data volume", "lowest latency", and "least overlap", respectively; see section 3.3.1 for context). Objective weightings are all set to 1.0, $\nu_1 = \nu_2 = \nu_3 = \nu_4 = 1.0$ (see section 3.3.2 for definition). The observation execution data volume requirement ($MinDV$) is set to 100 Mb; *i.e.* 100 Mb of an observation window must be delivered by a data route for that observation to be counted as executed, and for that data route to count towards the initial latency reward factor for that observation (see inequalities 3.53 in section 3.3.2).

The allowed optimality gap for the MILP solver in AS is 1%. Time horizons are 210 minutes for observations, 210 minutes for crosslinks, and 840 minutes for downlinks. A maximum of 3 downlink after the end of the crosslink planning window is allowed to be scheduled for a given GP run (see section 3.1.5 for context). Resource delta t (Δt for T^{rsrc}) is 10 seconds. The GP replan interval is 6300 seconds, and a 60 second delay is included before plans are released. Maximum crosslink activity window durations (the length of activity window slices) are 200 seconds, maximum downlink activity window durations are 20 minutes, and no maximum limitation is placed on the window durations for observations.

A.1.2 Satellite Bus Model Parameters

More background on the satellite bus model can be found in sections 2.2.1 to 2.2.3. The values in the below tables are specifically for the 6-Sat simulation case. These values are the same as, or negligibly different from, those used by Zhou *et al.*[124].

Table A.1: Payload and Link Data Rate Parameters

Parameter	Value
Observation payload	50 Mbps above 60° target elevation mask
Downlink	20 Mbps above 10° ground station elevation mask
Crosslink	10 Mbps, line-of-sight required

Table A.2: Satellite Bus Model Parameters

Parameter	Value
Power Usage	Base: 10 W Obs: 25 W Dlnk: 20 W Xlnk, Tx: 20 W Xlnk, Rx: 5 W
Solar Charging	Constant 30 W in sunlight
Energy Storage	13.89 Wh max, 80% discharge allowed (min 2.78 Wh)
Sim Initial Energy Storage State	12 Wh
Charge/ Discharge Efficiency	charge: 1.0; discharge: 1.0
Data Storage	max 12 Gb, min 0 Gb

Table A.3: Activity Minimum Durations

Activity	Minimum Duration
Obs	15
Dlnk	60
Xlnk	60

A.1.3 Geometry Parameters

Note that in the 6-Sat case, satellites 1 and 5 are co-located at the orbit ascending node. This constellation design was borrowed from Zhou *et al.*, where this is true as well [124]. It was assumed in this work that in reality the satellites are able to maintain a small but safe distance from each other at all times. Also in the 6-Sat case, crosslink lines-of-sight between some satellites cross just at the surface of the earth or slightly below. While not realistic for an actual constellation, these cross-links are allowed for two reasons: 1) to maintain consistency with the scenario as defined by Zhou *et al.*, and 2) because a solution that avoids this inaccuracy, moving the satellite slightly closer in mean anomaly, has negligible effect on the quantitative results of

interest.

Table A.4: 6-Sat Orbit Parameters

Satellite Index	a **(km)**	e	i (°)	Ω (°)	ω (°)	M (°)
0	7378	0	97.86	0	0	90
1	7378	0	97.86	0	0	180
2	7378	0	97.86	0	0	270
3	7378	0	83.86	0	0	60
4	7378	0	83.86	0	0	120
5	7378	0	83.86	0	0	180

Table A.5: 6-Sat Observation Target Parameters

Target Index	**Name**	**Latitude (°)**	**Longitude (°)**
0	Himalaya	28.0	87.0
1	Mamiraus	-2.0	-66.0
2	Cape York	-11.0	142.5
3	Alaska Coast	60.0	-148.0
4	Greenland	69.0	-49.0

Table A.6: 6-Sat Ground Station Parameters

GS Index	**Name**	**Latitude (°)**	**Longitude (°)**
0	Beijing	40.0	116.0
1	Kashi	39.5	76.0
2	Sanya	18.0	109.5
3	Xi'an	34.0	108.0

A.2 SSO Ring and Walker Simulation Case Parameters

A.2.1 Parameters Overview

The full 24 hour scenario for SSO Ring and Walker is from 2018-01-18T00:00:00 to 2018-01-19T00:00:00.

Unless otherwise specified, the following parameters are used for GP-Fast in simulation runs. The number of routes chosen in downselection (RS2) are $\rho_1^{RS2} = 6, \rho_2^{RS2} = 6, \rho_3^{RS2} = 30$ ("highest data volume", "lowest latency", and "least overlap", respectively; see section 3.3.1 for context). Objective weightings are all set to 1.0, $\nu_1 = \nu_2 = \nu_3 = \nu_4 = 1.0$ (see section 3.3.2 for definition). The observation execution data volume requirement ($MinDV$) is set to 100 Mb; *i.e.* 100 Mb of an observation window must be delivered by a data route for that observation to be counted as executed, and for that data route to count towards the initial latency reward factor for that observation (see inequalities 3.53 in section 3.3.2).

The allowed optimality gap for the MILP solver in AS is 1%. Time horizons are 95 minutes for observations, 95 minutes for crosslinks, and 760 minutes for downlinks. A maximum of 1 downlink after the end of the crosslink planning window is allowed to be scheduled for a given GP run (see section 3.1.5 for context). Resource delta t (Δt for T^{rsrc}) is 10 seconds. The GP replan interval is 3000 seconds, and a 60 second delay is included before plans are released. Maximum crosslink activity window durations (the length of activity window slices) are 2 minutes, maximum downlink activity window durations are 10 minutes, and no maximum limitation is placed on the window durations for observations.

A.2.2 Satellite Bus Model Parameters

The tables below are repeated from sections 2.2.1 to 2.2.3 for convenience, see those sections for more detail.

Table A.7: Payload and Link Data Rate Parameters

Parameter	Value
Observation payload	127.5 Mbps above 60° target elevation mask
Downlink	80 Mbps above 20° ground station elevation mask
Crosslink	73 to 3.2 Mbps, up to 5000 km range, line-of-sight required. See appendix B, table B.1

Table A.8: Satellite Bus Model Parameters

Parameter	Value
Power Usage	Base: 12.8 W Obs: 10 W Dlnk: 12.3 W Xlnk, Tx: 12 W Xlnk, Rx: 5 W
Solar Charging	24 W orbit-average power (constant 38 W in sunlight)
Energy Storage	40 Wh max, 60% discharge allowed (min 24 Wh)
Sim Initial Energy Storage State	40 Wh
Charge/ Discharge Efficiency	charge: 0.7; discharge: 0.9
Data Storage	max 4 GB (32 Gb), min 0 GB

Table A.9: Activity Minimum Durations

Activity	Minimum Duration
Obs	15
Dlnk	60
Xlnk	60

A.2.3 Geometry Parameters

Table A.10: Walker Orbit Parameters

Satellite Index	a **(km)**	e	i (°)	Ω (°)	ω (°)	M (°)
0	6978	0	30	0	0	0
1	6978	0	30	0	0	36
2	6978	0	30	0	0	72
3	6978	0	30	0	0	108
4	6978	0	30	0	0	144
5	6978	0	30	0	0	180
6	6978	0	30	0	0	216
7	6978	0	30	0	0	252
8	6978	0	30	0	0	288
9	6978	0	30	0	0	324
10	6978	0	30	120	0	12
11	6978	0	30	120	0	48
12	6978	0	30	120	0	84
13	6978	0	30	120	0	120
14	6978	0	30	120	0	156
15	6978	0	30	120	0	192
16	6978	0	30	120	0	228
17	6978	0	30	120	0	264
18	6978	0	30	120	0	300
19	6978	0	30	120	0	336
20	6978	0	30	240	0	24
21	6978	0	30	240	0	60
22	6978	0	30	240	0	96
23	6978	0	30	240	0	132
24	6978	0	30	240	0	168
25	6978	0	30	240	0	204
26	6978	0	30	240	0	240
27	6978	0	30	240	0	276
28	6978	0	30	240	0	312
29	6978	0	30	240	0	348

Table A.11: SSO Ring Orbit Parameters

Satellite Index	a **(km)**	e	i (°)	Ω (°)	ω (°)	M (°)
0	6978	0	98	0	0	0
1	6978	0	98	0	0	36
2	6978	0	98	0	0	72
3	6978	0	88	0	0	108
4	6978	0	88	0	0	144
5	6978	0	88	0	0	180
6	6978	0	88	0	0	216
7	6978	0	88	0	0	252
8	6978	0	88	0	0	288
9	6978	0	88	0	0	324

Table A.12: SSO Ring Ground Station Parameters

GS Index	**Name**	**Latitude (°)**	**Longitude (°)**
0	Prudhoe Bay	70.37	-148.75
1	Argentina	53.16	-70.92
2	Norway	67.31	14.78

Table A.13: Walker Ground Station Parameters

GS Index	**Name**	**Latitude (°)**	**Longitude (°)**
0	New Mexico	32.78	-106.32
1	Hawaii	19.89	-155.58
2	Florida	26.75	-80.93
3	Brazil	-18.41	-45.63
4	South Africa	-25.89	27.68
5	Dubai	25.20	55.27
6	Singapore	1.35	103.81
7	Guam	13.44	144.79
8	Japan	37.51	139.66

Table A.14: SSO Ring and Walker Observation Target Parameters

Target Index	Name	Latitude (°)	Longitude (°)
0	Lower US Upper Mexico 1	27.25	-101.75
1	Mexico and Caribbean Islands 0	19.75	-102.75
2	Mexico and Caribbean Islands 2	19.75	-86.35
3	Panama and Upper South America 1	11.25	-79.25
4	Panama and Upper South America 3	11.25	-62.25
5	Upper Mid South America 1	-14.87	-66.37
6	Upper Mid South America 3	-14.87	-43.37
7	Upper Mid South America 4	-8.62	-77.87
8	Upper Mid South America 6	-8.62	-54.87
9	Upper Mid South America 9	-2.37	-66.37
10	Upper Mid South America 11	-2.37	-43.37
11	Upper Mid South America 14	3.87	-54.87
12	Lower Mid South America 0	-27.0	-69.0
13	Lower Mid South America 2	-27.0	-47.67
14	Sub-Saharan Africa 2	5.85	8.48
15	Sub-Saharan Africa 4	5.85	31.82
16	Sub-Saharan Africa 7	12.15	-3.18
17	Sub-Saharan Africa 9	12.15	20.15
18	Sub-Saharan Africa 11	12.15	43.48
19	Lower Africa 3	-20.19	13.27
20	Lower Africa 5	-20.19	34.60
21	Lower Africa 9	-7.11	13.27
22	Lower Africa 11	-7.11	34.60
23	Madagascar 0	-23.0	46.3
24	Upper Arabian Sea and Coast 2	27.5	49.5
25	India 3	14.37	79.62
26	India 4	20.62	71.12
27	India 7	26.87	79.62
28	Upper Southeast Asia 5	27.25	97.25
29	Upper Southeast Asia 7	27.25	116.25
30	Lower Southeast Asia 1	-6.37	104.12
31	Lower Southeast Asia 3	-6.37	123.12
32	Lower Southeast Asia 4	0.87	94.62
33	Lower Southeast Asia 6	0.87	113.62
34	Lower Southeast Asia 11	8.12	123.12
35	Lower Southeast Asia 12	15.37	94.62
36	Lower Southeast Asia 14	15.37	113.62
37	Papua New Guinea and Islands 1	-9.0	143.0
38	Papua New Guinea and Islands 3	-3.0	132.0
39	Upper Australia 1	-16.0	134.0

A.3 Simulation Cases for Scalability With Number of Satellites Analysis

The following parameters are used for GP-Fast in these simulation runs (note: only GP-Fast is run for the number of satellites scalability analysis, not the full Constellation Simulator). The number of routes chosen in downselection (RS2) are $\rho_1^{RS2} = 6, \rho_2^{RS2} = 6, \rho_3^{RS2} = 10$ (see section 3.3.1 for context). Objective weightings are all set to 1.0, $\nu_1 = \nu_2 = \nu_3 = \nu_4 = 1.0$ (see section 3.3.2 for definition). The allowed optimality gap for the MILP solver in AS is 1%, though certain runs did not attain this small of a gap. Time horizons are 120 minutes for observations, 120 minutes for crosslinks, and 120 minutes for downlinks.

Table A.15: Sim Case Parameters for Scalability With Number of Satellites Analysis

Number of Satellites	**Orbit Geometry**	**Observation Targets**	**Ground Stations**
10	Same as "SSO Ring", see table A.11	table A.14	table A.16
18	30°:18/3/1 Walker Delta	table A.14	table A.16
30	Same as "Walker", see table A.10	table A.14	table A.16
60	60°:60/3/1 Walker Delta	table A.14	table A.16
100	60°:100/5/1 Walker Delta	table A.14	table A.16
140	60°:140/5/1 Walker Delta	table A.14	table A.16

Table A.16: Ground Station Parameters for Scalability With Number of Satellites Analysis

GS Index	Name	Latitude (°)	Longitude (°)
0	New Mexico	32.78	-106.32
1	Hawaii	19.89	-155.58
3	Brazil	-18.41	-45.63
4	South Africa	-25.89	27.68
5	Dubai	25.20	55.27
6	Singapore	1.35	103.81
7	Guam	13.44	144.79

A.4 Injected Observations and Parameters for "Injects" Simulation Case

The LP is run with a planning horizon of 210 minutes, with the following weighting factors (see section 4.2 for context): $\nu_1 = 1.0, \nu_2 = 1.0, \nu_3 = 1.0, \nu_4 = 0, \nu_5 = 0, \nu_6 = 5.0$. The weighting factors ν_4, ν_5 are set to zero to reduce emphasis on data volume for injected observations, and focus reward on latency for them.

Table A.17: Injected observations used in "Injects" simulation case. All observation capacities are 300 Mb

Obs Index	**Satellite Index**	**Start/End, on 2016-02-14**
0	4	09:54:37.16 / 09:55:37.16
1	5	11:59:35.48 / 12:00:35.48
3	5	21:22:23.29 / 21:23:23.29
5	1	18:09:00.70 / 18:10:00.70
6	5	04:15:54.45 / 04:16:54.45
8	0	15:11:02.82 / 15:12:02.82
9	3	07:05:41.52 / 07:06:41.52
10	4	16:36:02.41 / 16:37:02.41
11	1	10:05:12.82 / 10:06:12.82
13	2	09:54:20.71 / 09:55:20.71
15	1	20:38:46.38 / 20:39:46.38
17	4	10:54:33.46 / 10:55:33.46
19	1	21:54:09.27 / 21:55:09.27
20	0	18:45:56.19 / 18:46:56.19
21	4	19:42:40.34 / 19:43:40.34
22	0	15:12:31.77 / 15:13:31.77
24	0	14:11:12.05 / 14:12:12.05
25	4	16:21:00.81 / 16:22:00.81
26	2	10:00:28.15 / 10:01:28.15
27	4	08:39:39.03 / 08:40:39.03
28	0	05:01:51.50 / 05:02:51.50
29	5	21:52:18.70 / 21:53:18.70
31	0	19:25:09.14 / 19:26:09.14
34	0	13:59:13.61 / 14:00:13.61
35	3	18:09:56.97 / 18:10:56.97
36	3	17:35:05.82 / 17:36:05.82
37	4	21:30:32.70 / 21:31:32.70
38	5	13:33:49.86 / 13:34:49.86

Appendix B

Optical Crosslink Data Rates

Table B.1: Crosslink rate for optical transceiver as function of inter-satellite range

Range (km)	Data Rate (Mbps)	Range (km)	Data Rate (Mbps)
0	73.113	2600	9.9184
100	73.113	2700	9.9184
200	73.113	2800	9.9184
300	73.113	2900	5.6897
400	73.113	3000	5.6897
500	73.113	3100	5.6897
600	60.927	3200	5.6897
700	60.927	3300	5.6897
800	60.927	3400	5.6897
900	43.007	3500	5.6897
1000	43.007	3600	5.6897
1100	43.007	3700	5.6897
1200	27.694	3800	5.6897
1300	27.694	3900	5.6897
1400	27.694	4000	3.2067
1500	27.694	4100	3.2067
1600	16.872	4200	3.2067
1700	16.872	4300	3.2067
1800	16.872	4400	3.2067
1900	16.872	4500	3.2067
2000	16.872	4600	3.2067
2100	9.9184	4700	3.2067
2200	9.9184	4800	3.2067
2300	9.9184	4900	3.2067
2400	9.9184	5000	3.2067
2500	9.9184		

Bibliography

[1] Christopher Amato, Girish Chowdhary, Alborz Geramifard, N Kemal Ure, and Mykel J Kochenderfer. Decentralized Control of Partially Observable Markov Decision Processes. In *2013 IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 2398–2405, 2013.

[2] Christopher Amato, George D. Konidaris, and Leslie P. Kaelbling. Planning with Macro-Actions in Decentralized POMDPs. In *Proceedings of the Workshop on Planning and Robotics (PlanRob) at the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*, pages 1273–1280, Portsmouth, NH, 2014.

[3] Baran Tan Bacinoglu, Elif Tugce Ceran, and Elif Uysal-biyikoglu. Age of Information under Energy Replenishment Constraints. In *Information Theory and Applications Workshop*, 2015.

[4] A.H. Ballard. Rosette Constellations of Earth Satellites. *IEEE Transactions on Aerospace and Electronic Systems*, AES-16(5):656–673, 1980.

[5] Dimitris Bertsimas and Robert Weismantel. *Optimization Over Integers*, volume 13. Dynamic Ideas Belmont, 2005.

[6] W. J. Blackwell, S. Braun, R. Bennartz, C. Velden, M. DeMaria, R. Atlas, J. Dunion, F. Marks, R. Rogers, B. Annane, and R. V. Leslie. An Overview of the TROPICS NASA Earth Venture Mission. *Q J R Meteorol Soc. Accepted Author Manuscript.*, 2018.

[7] William J Blackwell, G Allen, C Galbraith, R Leslie, I Osaretin, M Scarito, Mike Shields, E Thompson, D Toher, D Townzen, A Vogel, R Wezalis, Kerri Cahoy, David W Miller, Anne Marinan, Ryan Kingsbury, Evan Wise, Sung Wook Paek, Eric Peters, Meghan Prinkey, Pratik Davé, and Brian Coffee. MicroMAS : A First Step Towards a Nanosatellite Constellation for Global Storm Observation. In *27th Annual AIAA/USU Conference on Small Satellites, SSC13-XI-1*, Logan, UT, 2013. AIAA/USU.

[8] Christopher R. Boshuizen, James Mason, Pete Klupar, and Shannon Spanhake. Results from the Planet Labs Flock Constellation. In *28th Annual AIAA/USU Conference on Small Satellites, SSC14-I-1*, Logan, UT, 2014. AIAA/USU.

[9] Timo Bretschneider, Tanya Vladimirova, and Siti Yuhaniz. Image Processing Capabilities On-Board Micro- Satellites for Disaster Monitoring. In *Proceedings of the 2nd Asian Space Conference*, 2005.

[10] California Polytechnic State University. CubeSat Design Specification Rev. 13. Technical report, Cal Poly SLO, San Luis Obispo, CA, 2014.

[11] Valdemir Carrara. An Open Source Satellite Attitude and Orbit Simulator Toolbox for Matlab. In *Proceedings of the XVII International Symposium on Dynamic Problems of Mechanics*, Natal, Brazil, 2015.

[12] Jeremy Castaing. Scheduling Downloads for Multi-Satellite, Multi-Ground Station Missions. In *28th Annual AIAA/USU Conference on Small Satellites, SSC14-VIII-4*, Logan, UT, 2014.

[13] Steve Chien, Joshua Doubleday, Kevin Ortega, Daniel Tran, John Bellardo, Austin Williams, Jordi Piug-Suari, Gary Crum, and Thomas Flatley. Onboard Autonomy and Ground Operations Automation for the Intelligent Payload Experiment (IPEX) CubeSat Mission. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Noordwijk, Netherlands, 2012. European Space Agency.

[14] Steve Chien, B Engelhardt, R Knight, G Rabideau, R Sherwood, Eric Hansen, A Ortiviz, C Wilklow, and S Wichman. Onboard Autonomy on the Three Corner Sat Mission. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Noordwijk, Netherlands, 2001. European Space Agency.

[15] Steve Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau. Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling. In *The Fifth International Conference on Artificial Intelligence Planning Systems*, pages 300–307, Menlo Park, CA, 2000. The AAAI Press.

[16] Steve Chien, Gregg Rabideau, R Knight, Rob Sherwood, B Engelhardt, D Mutz, Tara Estlin, B Smith, F Fisher, T Barrett, G Stebbins, and D Tran. ASPEN - Automated Planning and Scheduling for Space Mission Operations. In *International Conference on Space Operations (SpaceOps 2000)*, Toulouse, France, 2000. AIAA.

[17] Steve Chien, Rob Sherwood, Michael Burl, Russell Knight, Gregg Rabideau, Barbara Engelhardt, Ashley Davies, Paul Zetocha, Ross Wainwright, Pete Klupar, Pat Cappelaere, Derek Surka, Brian C. Williams, Ronald Greeley, Victor Baker, and James Doan. The Techsat-21 Autonomous Sciencecraft Constellation. In *6th International Symposium on Artificial Intelligence and Robotics & Automation in Space*, Noordwijk, Netherlands, 2001. European Space Agency.

[18] Han Lim Choi, Luc Brunet, and Jonathan P. How. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics*, 25:912–926, 2009.

[19] John Robert Claybrook. Feasibility Analysis On The Utilization Of The Iridium Satllite Communications Network For Resident Space Objects In Low Earth Orbit. Master's thesis, Air Force Institute of Technology, 2013.

[20] Emily Clements, Raichelle Aniceto, Derek Barnes, David Caplan, James Clark, Inigo del Portillo, Christian Haughwout, Maxim Khatsenko, Ryan Kingsbury, Myron Lee, Rachel Morgan, Jonathan Twichell, Kathleen Riesing, Hyosang Yoon, Caleb Ziegler, and Kerri Cahoy. Nanosatellite Optical Downlink Experiment: Design, Simulation, and Prototyping. *Optical Engineering*, 55(11):111610, 2016.

[21] Brian G Coffee, Kerri Cahoy, and Rebecca Bishop. Propagation of CubeSats in LEO using NORAD Two Line Element Sets : Accuracy and Update Frequency. In *AIAA Guidance, Navigation, and Control Conference*, Boston, MA, 2013.

[22] Cesium Consortium. Cesium: An open-source JavaScript library for world-class 3D globes and maps. `https://cesiumjs.org/`, accessed 2018-07-02.

[23] Brian Cooper. Corvus-BC Manufacturing Lessons Learned. In *CubeSat Developer's Workshop*, 2016.

[24] Alex Da Silva Curiel, Lee Boland, John Cooksley, Mohammed Bekhti, Paul Stephens, Wei Sun, and Martin Sweeting. First results from the disaster monitoring constellation (DMC). *Acta Astronautica*, 56(1-2):261–271, 2005.

[25] Sylvain Damiani, Gérard Verfaillie, and Marie-Claire Charmeau. An Earth Watching Satellite Constellation: How to Manage a Team of Watching Agents with Limited Communications. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems - AAMAS '05*, pages 455–462, New York, NY, 2005. ACM.

[26] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *The Basic George B. Dantzig, RW Cottle, Ed*, pages 24–32, 2003.

[27] Subrata Das, Curt Wu, and Walt Truszkowski. Enhanced Satellite Constellation Operations via Distributed Planning and Scheduling. In *Proceeding of the International Symposium on Artificial Intelligence and Robotics & Automation in Space*, Noordwijk, Netherlands, 2001. European Space Agency.

[28] Kiruthika Devaraj, Ryan Kingsbury, Matt Ligon, Joseph Breu, Vivek Vittaldev, Bryan Klofas, Patrick Yeon, and Kyle Colton. Dove High Speed Downlink System. In *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites, SSC17-VII-02*, Logan, UT, 2017.

[29] Delkin Devices. Industrial SD: Secure Digital Memory Cards. `https://www.delkin.com/products/industrial-sd/`, accessed 2018-07-02.

[30] Walt Everetts. personal communication.

[31] M A Fernandez, G Guillois, Y Richard, R Walker, and E S A Estec. A Game-changing radio communication architecture for cube/nano satellites. In *29th Annual AIAA/USU Conference on Small Satellites, SSC15-P-48*, Logan, UT, 2015.

[32] P. Festa. A Brief Introduction To Exact, Approximation, And Heuristic Algorithms For Solving Hard Combinatorial Optimization Problems. In *International Conference on Transparent Optical Networks*, 2014.

[33] William A Fisher. The Optwise Corporation Deconfliction Scheduler Algorithms (As used in STK/Scheduler). Technical Report 510, Optwise Corporation, 2004.

[34] William A Fisher and Ella Herz. A Flexible Architecture for Creating Scheduling Algorithms as used in STK Scheduler. Technical report, Optwise Corporation and Orbit Logic Incorporated, 2013.

[35] J. A. Fraire, P. Madoery, S. Burleigh, M. Feldmann, J. Finochietto, A. Charif, N. Zergainoh, and R. Velazco. Assessing Contact Graph Routing Performance and Reliability in Distributed Satellite Constellations. *Journal of Computer Networks and Communications*, 2017.

[36] Juan A. Fraire, Pablo G. Madoery, and Jorge M. Finochietto. Traffic-aware contact plan design for disruption-tolerant space sensor networks. *Ad Hoc Networks*, 47:41–52, 2016.

[37] Jeremy Frank, Minh Do, and Tony T Tran. Scheduling Ocean Color Observations for a GEO-Stationary Satellite. In *The 26th International Conference on Automated Planning and Scheduling*, 2016.

[38] Joseph W Gangestad, Brian S Hardy, and David a Hinkley. Operations , Orbit Determination , and Formation Control of the AeroCube-4 CubeSats. In *27th Annual AIAA/USU Conference on Small Satellites, SSC13-X-4*, Logan, UT, 2013.

[39] E Gill, P Sundaramoorthy, J Bouwmeester, B Zandbergen, and R Reinhard. Formation flying within a constellation of nano-satellites : The QB50 mission. *Acta Astronautica*, 82(1):110–117, 2013.

[40] Matthew Gombolay, Ronald Wilcox, and Julie Shah. Fast Scheduling of Multi-Robot Teams with Temporospatial Constraints. In *Robotics: Science and systems (RSS)*, pages 24–28, Berlin, Germany, 2013. RSS.

[41] Michael R Greene and Robert E Zee. Increasing the Accuracy of Orbital Position Information from NORAD SGP4 Using Intermittent GPS Readings. In *Proceedings of the 23rd Annual AIAA/USU Conference on Small Satellites, SSC09-X-7*, Logan, UT, 2009.

[42] Paul T. Grogan and Olivier L. de Weck. Interactive Simulation Games to Assess Federated Satellite System Concepts. In *2015 IEEE Aerospace Conference*, Big Sky, MT, 2015.

[43] Om P Gupta. Iridium: A Global Communication Network. `http://www.stanford.edu/class/aa247/Iridium_Innovations.pdf`, accessed 2018-07-05.

[44] John Hanson, James Chartres, Hugo Sanchez, and Ken Oyadomari. The EDSN Intersatellite Communications Architecture. In *11th Annual Summer CubeSat Developers' Workshop, SSC14-WK-2*, 2014.

[45] John Hanson, Ken Oyadomari, Jasper Wolfe, Watson Attai, and Cedric Prical. Nodes : A Flight Demonstration of Networked Spacecraft Command and Control. In *30th Annual AIAA/USU Conference on Small Satellites, SSC16-WK-09*, Logan, UT, 2016.

[46] John M Hanson and Alexander N Lindenj. Improved Low-Altitude Constellation Design Methods. *Journal of Guidance, Control, and Dynamics*, 12(2):228–236, 1989.

[47] William E. Hart, Carl D. Laird, Jean-Paul Watson, David L. Woodruff, Gabriel A. Hackebeil, Bethany L. Nicholson, and John D. Siirola. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, second edition, 2017.

[48] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[49] Ayesha Hein. A Systems Analysis of CubeSat Constellations with Distributed Sensors. Master's thesis, Massachusetts Institute of Technology, 2017.

[50] Thomas M Herold, Mark R. Abramson, Alexander C. Kahn, Stephan E. Kolitz, and Hamsa Balakrishnan. Asynchronous, Distributed Optimization for the Coordinated Planning of Air and Space Assets. In *AIAA Infotech at Aerospace 2010*, Atlanta, Georgia, 2010.

[51] IBM. IBM ILOG CPLEX Optimization Studio. `https://www.ibm.com/products/ilog-cplex-optimization-studio`, accessed 2018-06-30.

[52] H. Iglseder, W. Arens-Fischer, and W. Wolfsberger. Small Satellite Constellations For Disaster Detection And Monitoring. *Advances in Space Research*, 15(11):79–85, 1995.

[53] Analytical Graphics Inc. Engineering Tools. https://www.agi.com/products/engineering-tools, accessed 2018-07-16.

[54] BitBeam Inc. BBSDR High Speed Software Defined Radio. http://www.spaceflight.com/wp-content/uploads/2015/05/BitBeam-BBSDR_Leaflet.pdf, accessed 2018-07-02.

[55] Bridgesat Inc. Ground Network and Services. http://www.bridgesatinc.com/products/#satellite, accessed 2018-07-03.

[56] Gurobi Optimization Inc. Gurobi Optimizer Reference Manual. http://www.gurobi.com, accessed 2018-06-30.

[57] Gurobi Optimization Inc. Mixed-Integer Programming (MIP) A Primer on the Basics. http://www.gurobi.com/resources/getting-started/mip-basics, accessed 2018-06-30.

[58] Spaceflight Industries Inc. Network PDF. http://spaceflight.com/wp-content/uploads/2017/06/SF-Networks-Data-Sheet.pdf, accessed 2018-07-02.

[59] Tethers Unlimited Inc. SWIFT - XTS: Flexible S/X Communications Solutions for Space Missions. http://www.tethers.com/SpecSheets/Brochure_SWIFT_XTS.pdf, accessed 2018-07-03.

[60] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a Delay Tolerant Network. *ACM SIGCOMM Computer Communication Review*, 34(4):145, 2004.

[61] Siegfried Janson, Richard Welle, Todd Rose, Darren Rowen, Brian Hardy, Richard Dolphus, Patrick Doyle, Addison Faler, David Chien, Andrew Chin, Geoffrey Maul, Chris Coffman, Stephen D La Lumondiere, I Nicolette, and David Hinkley. The NASA Optical Communications and Sensor Demonstration Program : Initial Flight Results. In *29th Annual AIAA/USU Conference on Small Satellities, SSC16-III-03*, 2015.

[62] Li Jun, Li Jun, Chen Hao, and Jing Ning. A data transmission scheduling algorithm for rapid-response earth-observing operations. *Chinese Journal of Aeronautics*, 27(2):349–364, 2014.

[63] Sanjit Kaul, Roy Yates, and Marco Gruteser. Real-Time Status: How Often Should One Update? In *The 31st Annual IEEE International Conference on Computer Communications: Mini-Conference*, pages 2731–2735, 2012.

[64] Andrew K. Kennedy and Kerri L. Cahoy. Performance Analysis of Algorithms for Coordination of Earth Observation by CubeSat Constellations. *Journal of Aerospace Information Systems*, 3097(February), 2016.

[65] Andrew K Kennedy and Kerri L Cahoy. Initial Results from ACCESS : An Autonomous CubeSat Constellation Scheduling System for Earth Observation. In *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites*, pages SSC17–V–04, Logan, UT, 2017.

[66] Jan King, Kyle Leveque, Michael Bertino, Jin Kim, and Houahang Aghahassan. Ka-Band for CubeSats. *29th Annual AIAA/USU Conference on Small Satellites, SSC15-VIâĂŘ8*, 2015.

[67] Ryan Kingsbury. *Optical Communications for Small Satellites.* PhD thesis, Massachusetts Institute of Technology, 2015.

[68] Brian Klofas. CubeSat Communication Systems Table, Version 16. `https://www.klofas.com/comm-table/`, accessed 2018-07-03.

[69] Bryan Klofas. Planet Labs Ground Station Network. In *13th Annual CubeSat Developers Workshop*, San Luis Obispo, CA, 2016.

[70] Ed Klotz and Alexandra M Newman. Practical Guidelines for Solving Difficult Mixed Integer Linear Programs. *Surveys in Operations Research and Management Science*, 18(1-2):18–32, 2013.

[71] Olga Kondrateva, D Holger, Hagen Sparka, and Andreas Freimann. Throughput-Optimal Joint Routing and Scheduling for Low-Earth-Orbit Satellite Networks. In *14th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 59–66, 2018.

[72] MIT Lincoln Laboratory. TROPICS Mission Overview. `https://tropics.ll.mit.edu/CMS/tropics/Mission-Overview`, accessed 2018-07-03.

[73] Robert Scott Legge. *Optimization and Valuation of Reconfigurable Satellite Constellations Under Uncertainty.* PhD thesis, Massachusetts Institute of Technology, 2014.

[74] Michel Lemaître, Gérard Verfaillie, Frank Jouhaud, Jean Michel Lachiver, and Nicolas Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6(5):367–381, 2002.

[75] G. X. Liu, X. L. Ding, Z. L. Li, Z. W. Li, Y. Q. Chen, and S. B. Yu. Pre- and coseismic ground deformations of the 1999 Chi-Chi, Taiwan earthquake, measured with SAR interferometry. *Computers and Geosciences*, 30(4):333–343, 2004.

[76] Christopher J. Lowe and Malcolm Macdonald. Resource-Considerate Data Routing Through Satellite Networks. *Journal of Aerospace Information Systems*, 14(8):1–11, 2017.

[77] Clyde Space Ltd. Platforms. `https://www.clyde.space/what-we-do/platforms`, accessed 2018-07-02.

[78] Kris Maine, Carrie Devieux, and P Swan. Overview of IRIDIUM Satellite Network. In *WESCON/'95.*, San Francisco, CA, 1995.

[79] Daniel Mandl, Gary Crum, Vuong Ly, Matthew Handy, Karl F Huemmrich, Lawrence Ong, Ben Holt, and Rishabh Maharaja. Hyperspectral Cubesat Constellation for Natural Hazard Response (Follow-on). In *Proceedings of the 30th Annual AIAA/USU Conference on Small Satellites, SSC16-XII-02*, Logan, UT, 2016.

[80] Jacob Mattingley, Yang Wang, and S P Boyd. Receding Horizon Control. *IEEE Control Systems*, 31(3):52–65, 2011.

[81] Snezana Mitrovic Minic. Collection Planning Management : Multi-Satellite Collection Scheduling. Technical report, MDA, 2016.

[82] H. Mittelmann. Benchmark of commercial LP solvers. `http://plato.asu.edu/ftp/lpcom.html`, accessed 2018-07-05.

[83] Philippe Monmousseau. Scheduling of a Constellation of Satellites : Improving a Simulated Annealing Model by Creating a Mixed-Integer Linear Model. Master's thesis, Royal Institute of Technology (KTH), 2015.

[84] Daniel Morgan, Soon-jo Chung, and Fred Y. Hadaegh. Spacecraft Swarm Guidance Using a Sequence of Decentralized Convex Optimizations. In *AIAA/AAS Astrodynamics Specialist Conference*, Minneapolis, Minnesota, 2012.

[85] Rachel Morgan and Kerri Cahoy. Nanosatellite Lasercom System. *AIAA/USU Conference on Small Satellites, SSC17-VIII-1*, 2017.

[86] Katta G Murty. *Linear Programming.* John Wiley & Sons, 1983.

[87] Sreeja Nag, Jacqueline Lemoigne, David W. Miller, and Olivier De Weck. A Framework for Orbital Performance Evaluation in Distributed Space Missions for Earth Observation. *IEEE Aerospace Conference Proceedings*, 2015-June, 2015.

[88] National Academies of Sciences. *Achieving Science with CubeSats: Thinking Inside the Box.* 2016.

[89] Massachusetts Institute of Technology. Introduction to lp_solve 5.5.2.5. `http://web.mit.edu/lpsolve/doc/`, accessed 2018-06-30.

[90] World Meteorological Organization. OSCAR Observing Systems Capability Analysis and Review Tool. `https://www.wmo-sat.info/oscar/requirements`, accessed 2018-07-02.

[91] J Brent Parham, Aleks Zosuls, Brian Walsh, and Joshua Semeter. Multipoint Measurements of the Aurora with a CubeSat Swarm. In *Proceedings of the 30th Annual AIAA/USU Conference on Small Satellites, SSC16-P4-14*, 2016.

[92] Nicola Pergola, Francesco Marchese, and Valerio Tramutoli. Automated detection of thermal features of active volcanoes by means of infrared AVHRR records. *Remote Sensing of Environment*, 93(3):311–327, 2004.

[93] Gregg Rabideau, Russell Knight, Steve Chien, Alex Fukunaga, and Anita Govindjee. Iterative Repair Planning for Spacecraft Operations Using the Aspen System. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, volume 440, pages 99–106, Noordwijk, Netherlands, 1999. European Space Agency.

[94] Kathleen Riesing. Orbit Determination from Two Line Element Sets of ISS-Deployed CubeSats. In *29th Annual AIAA/USU Conference on Small Satellites, SSC15-VIII-5*, Logan, UT, 2015.

[95] Michael Rubel and James Mason. Patent US9651946B1: Automated Schedule Calculation For Controlling A Constellation Of Satellites, 2017.

[96] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education Inc., 2003.

[97] Rainer Sandau. Status and Trends of Small Satellite Missions for Earth Observation. *Acta Astronautica*, 66:1–12, 2010.

[98] Andrew D Santangelo and Paul Skentzos. Utilizing the Globalstar Network for CubeSat and Small Satellite Communications. In *33rd AIAA International Communications Satellite Systems Conference and Exhibition*, 2015.

[99] Thomas Schetter, Mark Campbell, and Derek Surka. Multiple agent-based autonomy for satellite constellations. *Artificial Intelligence*, 145:147–180, 2003.

[100] Daniel Selva and David Krejci. A survey and assessment of the capabilities of Cubesats for Earth observation. *Acta Astronautica*, 74:50–68, 2012.

[101] Sam Siewert and Linden H. McClure. A System Architecture to Advance Small Satellite Mission Operations Autonomy. In *9th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, 1995.

[102] Doug Sinclair and Kathleen Riesing. The Rainbow Connection - Why Now is the Time for Smallsat Optical Downlinks. In *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites, SSC17-II-06*, Logan, UT, 2017.

[103] Benjamin Smith, Robert Sherwood, Anita Govindjee, David Yan, Gregg R Rabideau, Steve Chien, and Alex Fukunaga. Representing Spacecraft Mission Planning Knowledge in ASPEN. Technical report, AAAI TR WS-98-03, 1998.

[104] Capella Space. Technology. `https://www.capellaspace.com/index#section-technology`, accessed 2018-07-03.

[105] Sara Spangelo and James Cutler. Optimization of Single-Satellite Operational Schedules Towards Enhanced Communication Capacity. *AIAA Guidance, Navigation, and Control Conference*, 2012.

[106] Sara Spangelo and James Cutler. Analytical Modeling Framework and Applications for Space Communication Networks. *Journal of Aerospace Information Systems*, 10(10):452–466, 2013.

[107] Derek M. Surka, Margarita C. Brito, and Christopher G. Harvey. The Real-Rime ObjectAgent Software Architecture for Distributed Satellite Systems. In *2001 IEEE Aerospace Conference Proceedings*, volume 6, pages 2731 – 2741, Big Sky, MT, 2001.

[108] S. R. Tsitas and J. Kingston. 6U CubeSat design for Earth observation with 6-5m GSD, five spectral bands and 14Mbps downlink. *Aeronautical Journal*, 114(1161):689–697, 2010.

[109] George Tyc, John Tulip, Daniel Schulten, Manfred Krischke, and Michael Oxfort. The RapidEye mission design. *Acta Astronautica*, 56(1-2):213–219, 2005.

[110] Johannes Van der Horst and Jason Noble. Task allocation in networks of satellites with Keplerian dynamics. *Acta Future*, 5:143–151, 2012.

[111] Michel Vasquez and Jin Kao Hao. A "Logic-constrained" Knapsack Formulation and a Tabu Algorithm for the Daily Photograph Scheduling of an Earth Observation Satellite. *Computational Optimization and Applications*, 20(2):137–157, 2001.

[112] Hank Voss, Art White, Stefan Brandle, and Jeff Dailey. Globalstar Communication Link for CubeSats: TSAT, GEARRS1, GEARRS2. In *29th AIAA/USU Conference on Small Satellites*, Logan, UT, 2015.

[113] J.G. Walker. Continuous Whole-Earth Coverage by Circular-Orbit Satellite Patterns. Technical report, Royal Aircraft Establishment, 1977.

[114] David Wang and Brian C Williams. tBurton : A Divide and Conquer Temporal Planner. In *MIT Computer Science and Artificial Intelligence Laboratory, TR-2014-027*, Cambridge, MA, 2014. MIT.

[115] Yu Wang, Min Sheng, Weihua Zhuang, Shan Zhang, Ning Zhang, Runzi Liu, and Jiandong Li. Multi-Resource Coordinate Scheduling for Earth Observation in Space Information Networks. *IEEE Journal on Selected Areas in Communications*, 36(2):268–279, 2018.

[116] Richard Welle, Alexander Utter, Todd Rose, Jerry Fuller, Kristin Gates, Benjamin Oakes, and Siegfried Janson. A CubeSat-Based Optical Communication Network for Low Earth Orbit. In *Proceedings of the 31st Annual AIAA/USU Conference on Small Satellites, SSC17-XI-01*, Logan, UT, 2017.

[117] Richard P Welle, Siegfried Janson, Darren Rowen, and Todd Rose. Cubesat-scale Laser Communications. In *31st Space Symposium*, 2015.

[118] William J Wolfe and Stephen E Sorensen. Three Scheduling Algorithms Applied to the Earth Observing Systems Domain. *Management Science*, 46(1):148–166, 2000.

[119] Guohua Wu, Huilin Wang, Haifeng Li, Witold Pedrycz, Dishan Qiu, Manhao Ma, and Jin Liu. An adaptive Simulated Annealing-based satellite observation scheduling method combined with a dynamic task clustering strategy. page 23, 2014.

[120] Xiaofeng Wu, Tanya Vladimirova, Kawsu Sidibeh, and U K Gu. Signal Routing in a Satellite Sensor Network Using Optimisation Algorithms. In *IEEE Aerospace Conference Proceedings*, 2008.

[121] Jiqun Zhang, Chenghu Zhou, Kaiqin Xu, and Masataka Watanabe. Flood disaster monitoring and evaluation in China. *Environmental Hazards*, 4(2-3):33–43, 2002.

[122] Zixuan Zheng, Jian Guo, and Eberhard Gill. Swarm satellite mission scheduling & planning using Hybrid Dynamic Mutation Genetic Algorithm. *Acta Astronautica*, 137(October 2016):243–253, 2017.

[123] Zixuan Zheng, Jian Guo, and Eberhard Gill. Onboard autonomous mission re-planning for multi-satellite system. *Acta Astronautica*, 145(February):28–43, 2018.

[124] Di Zhou, Min Sheng, Xijun Wang, Chao Xu, Runzi Liu, and Jiandong Li. Mission Aware Contact Plan Design In Resource-limited Small Satellite Networks. *IEEE Transactions on Communications*, 65(6):2451–2466, 2017.

[125] Guy G. Zohar. Ad-Hoc Regional Coverage Constellations Of Cubesats Using Secondary Launches. Master's thesis, California Polytechnic State University, San Luis Obispo, 2013.