Resource Optimization Algorithms for an Automated Coordinated CubeSat Constellation

by

Andrew Kitrell Kennedy

B.S. Aerospace Engineering, University of Texas at Austin, 2011

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

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2015

Author ..... Signature redacted
Department of Aeronautics and Astronautics
August 20, 2015

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

Accepted by ..... Signature redacted
Paulo Lozano
Graduate Committee Chair

Resource Optimization Algorithms for an Automated Coordinated CubeSat Constellation

by

Andrew Kitrell Kennedy

Submitted to the Department of Aeronautics and Astronautics
on August 20, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

We present and analyze the performance of two algorithms that plan and coordinate activities for a resource-constrained Earth-observing CubeSat constellation. The first algorithm is the Resource-Aware SmallSat Planner (RASP), which performs low-level planning of observation and communication activities for a single satellite while simultaneously keeping the satellite's onboard resources within specified bounds. RASP utilizes a Mixed Integer Linear Program based formulation and Depth First Search for construction of consistent onboard activity timelines. The second algorithm is the Limited Communication Constellation Coordinator (LCCC), which performs high level coordination of observations across the constellation through a distributed, "weak" consensus mechanism.

The performance of the algorithms is tested with a 24 hour simulation of an eighteen satellite constellation over multiple orbital geometries and inter-satellite communication contexts. The orbital geometries include a modified Walker Star constellation and an "ad hoc" constellation defined by historical launches of CubeSats. The multiple communication contexts simulate different methods for sharing observation planning information between the satellites, and include sharing through inter-satellite crosslinks, downlink and uplink to ground stations, connection to a commercial communications constellation, and no sharing at all.

Five analyses of the algorithms' performance were conducted, including average revisit times achieved, the numbers of communications links executed, how effectively planning information was shared, the resource margins maintained by the satellites, and the average execution time for the planner. Information sharing significantly aided in balancing revisit times across multiple Earth regions and three sensor choices, reducing the disparity in average revisit times between sensors from 514 minutes to 10 minutes for the Walker case and 617 to 11 minutes for he Ad Hoc case. Significantly more crosslink opportunities were available on average for the Walker satellites than for Ad Hoc (89.2 versus 47.7) and more crosslinks were executed for the Walker case (30.3 versus 20.8). Crosslink was found to be less effective than downlink at sharing planning information across the constellation, with a lower average latency

(186 minutes versus 434, Walker) and better average initial timeliness (-35 minutes versus -287, Walker). Information sharing through both a commercial constellation and downlink outperformed sharing through just downlink or just crosslink, with an average latency and initial timeliness of 77 and 74 minutes (Walker). Average data storage and energy storage margins were kept high, as desired, for both constellations, at around 85 and 70 %. RASP planning time was found to scale roughly with the square of planning window length, but stays under a minute in all cases tested (achieving a maximum of 37.71 seconds).

Thesis Supervisor: Kerri Cahoy
Title: Assistant Professor of Aeronautics and Astronautics

Acknowledgments

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.

This work is also supported by NASA Earth Science Technology Office grant number NNX14AC75G and NASA Space Technology Research Grant NNX12AM30H.

The author would like to thank his labmates for their feedback and advice over the course of performing the work in this thesis as well as his advisor for her continual encouragements and crucial feedback. He would like to thank Adam Meghuid for his work in adapting the RASP algorithm.

Finally, the author would like to thank his family for their support over an already-long, and continually growing, academic career.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

The purpose of this thesis is to demonstrate the utility of the onboard automated coordination of Earth observation across a constellation of CubeSats.

To enable this demonstration, two algorithms were developed. The first algorithm is the Resource-Aware SmallSat Planner (RASP), which performs low-level planning of observation and communication activities for a single satellite while simultaneously keeping the satellite's onboard resources within specified bounds. The second algorithm is the Limited Communication Constellation Coordinator (LCCC), which performs high level coordination of observations across the constellation through a distributed, "weak" consensus mechanism. When working together, the algorithms share observation planning information across the constellation and use the information to coordinate choices of observation timing for individual satellites.

The algorithms are tested for a use case based on two existing CubeSat missions, MicroMAS and MiRaTA. CubeSats are used as the test case because of the increasing interest in this platform and the unique resource limitations that these small satellites often have. These resource limitations necessitate careful planning of activities in order to achieve good coordinated performance across a constellation.

This chapter introduces the context for these algorithms, first discussing CubeSats and their capabilities, then the characteristics of a coordinated CubeSat constellation, and finally previous work in the area of coordinated constellations.

1.1 CubeSats

1.1.1 Definition of a CubeSat

CubeSats are a class of small satellite that has become a popular platform for academic, research, and commercial institutions. The CubeSat is a standardized satellite bus defined by California Polytechnic State University's CubeSat Design Specification (CDS) [13]. These satellites fall in the Picosatellite, Nanosatellite, and the low end of the Microsatellite mass ranges of small spacecraft, as indicated in Table 1.1. CubeSats are composed of one or multiple 10 cm x 10 cm x 10 cm units, or U's, and are usually designed in a 1U, 1.5U, 2U, 3U, or 6U configuration. Figure 1-1 below illustrates some of these configurations. U's are usually limited to 1.33kg in mass. The largest impact of the CDS has been the standardization of the interface between launch vehicles and small satellite deployers. The specification lays out requirements for mechanical and electrical interfaces with the satellite deployment mechanism as well as operational and testing requirements for the satellite.

Table 1.1: Small Spacecraft Mass Categories [53]

Mass Category	Mass Range
Minisatellite	$\geq$ 100 kg
Microsatellite	10 - 100 kg
Nanosatellite	1 - 10 kg
Picoosatellite	0.01 - 1 kg
Femtosatellite	0.001 - 0.01 kg

1.1.2 Cubesat Launch and Deployment

Historically CubeSats have been launched either as auxiliary payloads on launch vehicles or deployed from the International Space Station (ISS). Several deployment mechanisms exist for secondary payloads, including the PolyPicosat Orbital Deployer (P-POD) [13] and ISIPOD [11]. Figure 1-2 shows an image of the P-POD deployer. The Nanoracks CubeSat Deployer (NRCSD) offers deployment directly from the ISS

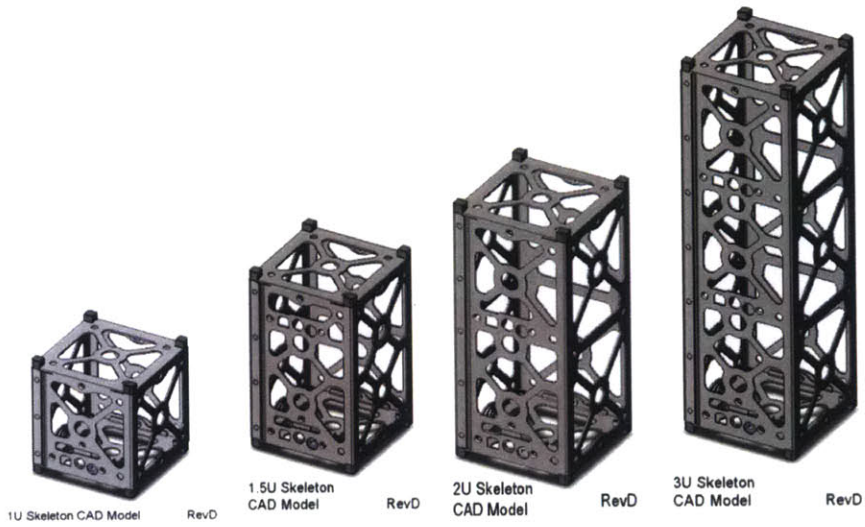


Figure 1-1: CubeSats are sized by number of units (U) and are typically built in 1U, 1.5U, 2U, or 3U sizes [35, 47]

[45]. CubeSats have benefited from significant government support and growing commercial services for obtaining launch opportunities over the past decade and a half. In the government sector, The National Aeronautics and Space Administration's (NASA) CubeSat Launch Initiative program offers rides to orbit for missions that address aspects of science, exploration, technology development, education, or operations [2]. In the commercial sector, a number of companies offer or plan to offer access to space for CubeSats. NanoRacks LLC, which provides the NRCSD, has played a particularly important role in providing launches in recent years: the company has deployed 61 CubeSats from the ISS since January 2014, and plans over 170 more in the near future [44]. Figure 1-3 displays the number of CubeSats launched per year since 2000. It is clear that launches have dramatically increased in recent years. This trend is expected to only increase as more commercial services come onto the market, particularly dedicated CubeSat and SmallSat launch vehicles such as those of Firefly Space Systems and Generation Orbit Launch Services [64] [55].

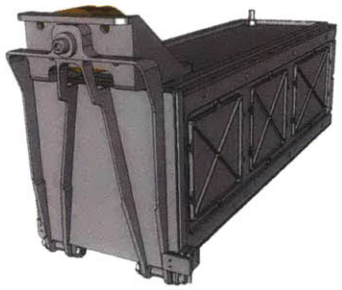


Figure 1-2: The PolyPicosat Orbital Deployer (P-POD) [13]

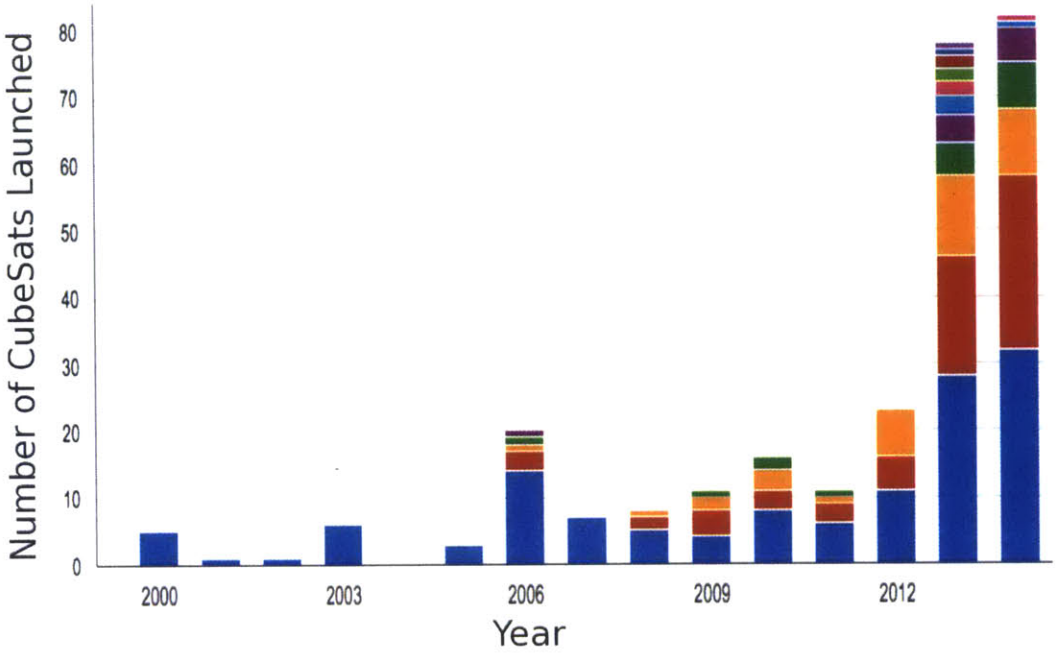


Figure 1-3: Graph of CubeSat launches per year from 2000 to 2014 [63]. The horizontal axis is the year, the vertical axis is the number of CubeSats launched per year. Different color bars correspond to individual launches and the height of said bars corresponds to the number of CubeSats on each launch. Note that similar color bars do not necessarily correspond to the same type of launch vehicle

1.1.3 Current Cubesat Bus Capabilities

The "spacecraft bus" is the main body of a spacecraft, which supports all the subsystems necessary for its successful operation. CubeSat bus technology is rapidly maturing, and more and more complex missions are being performed with this platform [60]. There are several key subsystems that tend to dictate the overall capability of a spacecraft bus, including communications (Comm), power, and the Attitude Determination and Control Subsystem (ADCS).

The Comm subsystem constrains how much data the spacecraft can send to the ground over a single operational day and the overall mission lifetime, as well as the ease of commanding the satellite. Up to this point, CubeSats have relied almost exclusively on radio frequency transceivers for nominal operational downlink and uplink to and from the ground [42]. The driving factors for Comm are the transmission (Tx) data rate to ground and the power the radio demands from the bus.

The Power subsystem collects solar power during periods of solar illumination, stores energy in onboard batteries, and manages the distribution of power over the satellite bus and payload. The driving factors for Power are the power production rate from the satellite's solar panels (which is maximized when the sun illuminates the panels from directly overhead) and the amount of energy the rechargeable batteries can store.

The ADCS is tasked with controlling the attitude, or pointing angle, of the spacecraft. This is usually achieved through a mix of sensors and actuators, which are used to determine the attitude of the vehicle and control its attitude, respectively. A common set of sensors for a CubeSat includes sun sensors, which aid in determining the direction of the sun, earth horizon sensors, which sense the direction of the Earth's horizon, and a magnetometer, which senses magnetic field direction and magnitude. Higher precision star sensors can be used as well, but are not yet common on CubeSats. A common set of actuators includes reaction wheels, which exert torques to absorb or provide angular momentum to the bus (thus controlling the angular rates of the bus), and magnetorquers, which act against the Earth's magnetic field to exert

torques. Magnetorquers are often used to desaturate stored angular momentum in the reaction wheels. The driving factors for the ADCS are the precision and accuracy of the sensors, which affects how well attitude can be known, and the control authority and accuracy of the actuators, which affects how well attitude can be controlled. Attitude cannot be controlled to better accuracy than it can be determined.

Table 1.2 summarizes the state-of-the-art in several key CubeSat subsystem technologies, mostly as reported by NASA Ames Research Center in 2014 [48]. Table 1.3 lists some example state-of-the-art Commercial-off-the-shelf (COTS) components available for these subsystems. Note that these technologies are intended for a 3U-sized CubeSat bus.

Table 1.2: Summary of State-of-the-Art in Key CubeSat Subsystems [48]

Subsystem	**Capability**	**State-of-the-Art**
Comm	Downlink to Ground	VHF/UHF/S-Band Transceivers: 1200 bps to ~3Mbps Tx data rate [1]
Power	Energy Production	Triple Junction Cells: solar conversion efficiency ~29%
Power	Energy Storage	Li-Ion, Li-Polymer batteries: specific energy 100 Wh/kg
ADCS	Attitude Control	~2 degree pointing accuracy (achieved on CXBN CubeSat [2])

1.1.4 Future Evolution of Cubesat Capabilities

Historically many of the CubeSats launched were technology demonstrations: they were an educational endeavor for the developer institution to build its own knowledge and improve its processes. Much of this effort was led by academic institutions and took place in the decade of the 2000's after the development of the CubeSat Design Specification.

[1] VHF: Very High Frequency, UHF: Ultra High Frequency
[2] CXBN: Cosmic X-ray Background Nanosatellite

Table 1.3: Example State-of-the-Art CubeSat Components

Subsys.	**Company**	**Component**	**Capability**
Comm	L-3 Communication Systems-West	Cadet UHF Radio	2.6 Mbps Tx rate, power consumption of 11.33W Tx (peak), 0.17 W Rx [43, 42]
Comm	Clyde Space Ltd.	CubeSat S-band Transmitter	$<$ 2 Mbps Tx rate, power consumption of $<$ 6 W [48]
Power	Pumpkin Inc.	PMDAS v5 7-cell panel	Power production of 7 W at beginning-of-life [39]
Power	Clyde Space Ltd.	Li-Polymer battery	8.2 V output, 1.24 A-Hr storage [48]
ADCS	Maryland Aerospace Inc.	3-axis integrated reaction wheel actuator set	Max angular momentum storage of 10.8 mNms, max torque of 0.63 mNm [48]
ADCS	Maryland Aerospace Inc.	Static Earth Sensors	One coarse field-of-view sensor (60 degrees), 3 fine field-of-view sensors (7 degrees) [34]

The community is now seeing a shift towards payload capability development and larger scale operational demonstrations. The plots by Buchen in Figures 1-4 and 1-5 provide some detail on this trend [10]. We see that from 2009-2013 the largest contingent (55%) of Nano/Microsatellites (including CubeSats) were in the spacecraft bus development-oriented "Technology" category, whereas future projections for 2014-2016 are largely dominated by "Earth Observation/Remote Sensing". The largest individual developer influence on this trend was that of Planet Labs Inc., which alone launched 39 Earth observation CubeSats in 2014 [9]. This company's successful funding, development, launch and operation of such a large group, or "constellation", of CubeSats has helped demonstrate the viability of the CubeSat as a mass-produced platform for commercial and scientific applications in orbit.

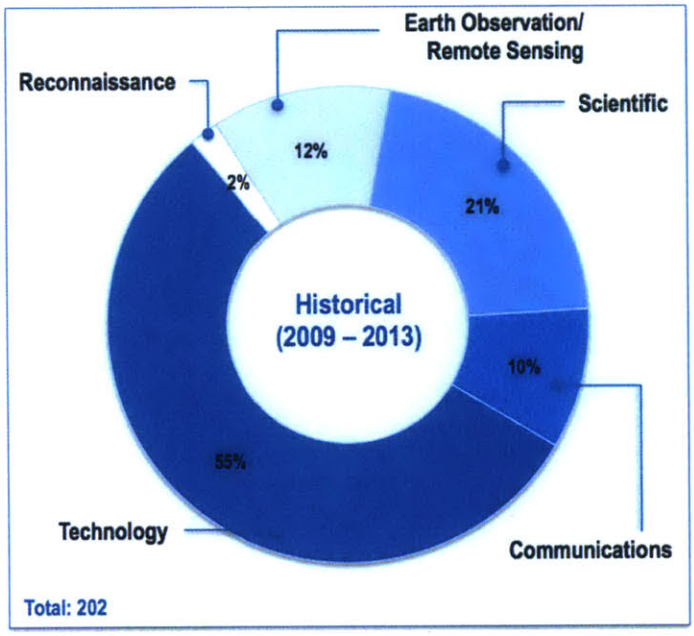


Figure 1-4: Historical Nano/Microsatellite Trends by Purpose (2009 - 2013) [10]

Buchen predicts continued growth in launches over the coming years, as shown in Figure 1-6. We see an expected 410 Nano/Microsatellite launches in 2020, with a full potential for 543 launches in the optimistic case. Whether or not these predictions are met, we can expect that the large number of satellites put into orbit will both drive significant advancement in commercial and scientific applications.

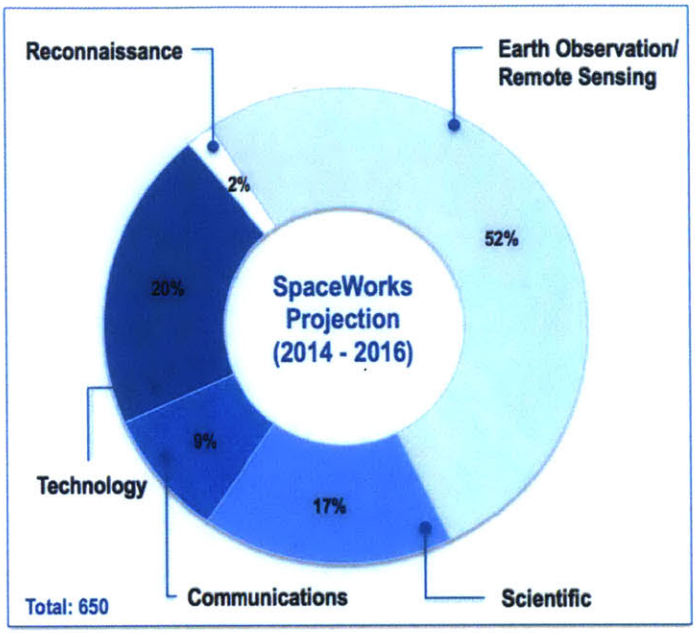


Figure 1-5: Future Nano/Microsatellite Trends by Purpose (2014 âĂŞ 2016) [10]

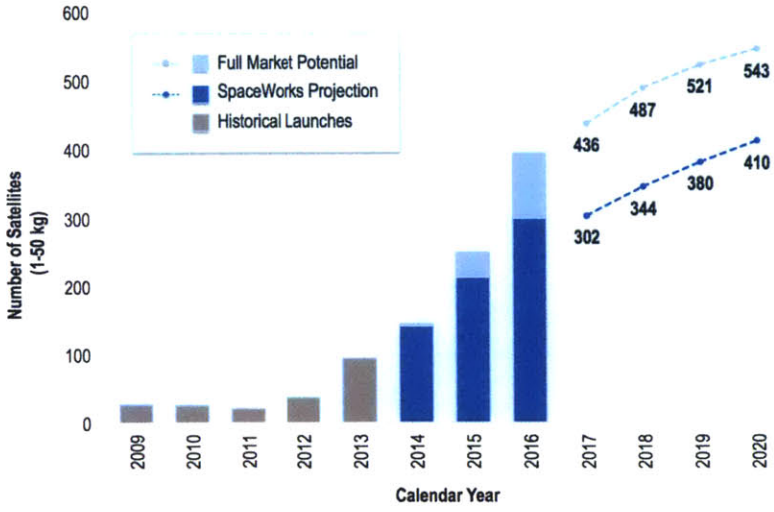


Figure 1-6: Nano/Microsatellite Launch History and Projection [10]

1.2 Coordinated CubeSat Constellations

Traditionally space missions have utilized only a single, monolithic spacecraft to take measurements and communicate with ground controllers, which fundamentally limits spacecraft activities to only a single point in space and time. Large-scale, Earth-orbiting constellations of many satellites promise great operational benefits due to the spatial and temporal diversity they can achieve. Such constellations are rapidly emerging as a real possibilty with the advance of small satellite technology, which makes it feasible to launch the many spacecraft. The following subsections detail the benefits, constraints, and applications of a "coordinated constellation", a particular way of operating a constellation studied for this thesis.

1.2.1 Definition of a Coordinated Constellation

A "constellation" is a specific type of Distributed Space System (DSS). Figure 1-7 shows several categories of DSS. The defining characteristic of all categories is that they utilize multiple spacecraft in some formation in orbit, performing a cooperative task. A constellation in particular features a "widespread" formation with large inter-satellite distances and relatively low controllability of those distances. Constellations have already been implemented to great effect for diverse applications in Low, Medium, and Geosynchronous Earth Orbit (LEO, MEO, GEO). Some well-known examples include:

- The Global Positioning System constellation: a set of 31 (currently) MEO satellites providing positioning services for ground and space-based receivers [52]
- The Iridium and Globalstar satellite telephony and low data rate communications constellations, with 66 (total) and 32 (second-generation) LEO satellites, respectively [33, 28]
- NASA's Tracking and Data Relay Satellite System (TDRSS), a set of 9 GEO satellites that provides continual communication between on-orbit satellites and ground stations [4]

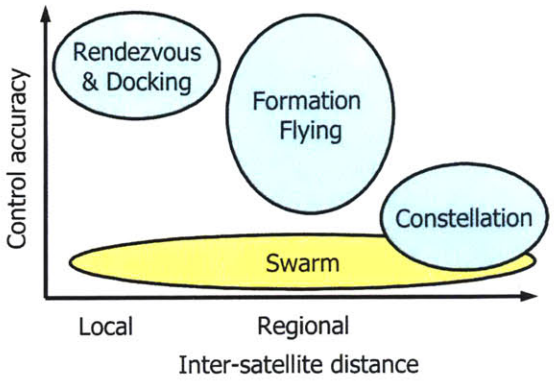


Figure 1-7: Categories of distributed space systems [26]

A "coordinated constellation" is considered for the purposes of this thesis to be a constellation that features a high degree of inter-satellite task planning and scheduling dependence to accomplish its operational goals. This coordination can be achieved through varying degrees of automation. In the traditional, low-automation case, ground operators might plan out the entire constellation's activities days in advance, and uplink these plans to the spacecraft during ground station overflights. In the highly automated case, the spacecraft might communicate with each other directly and plan out their own tasks.

1.2.2 Benefits of a Coordinated Constellation

Constellations offer benefits in many mission areas, including observation, remote-sensing (e.g. weather satellites), communications, and in-situ measurement (e.g. mapping of Earth's magnetosphere). A non-coordinated constellation - one in which satellites simply perform their own activities without considering overlap with other satellites' activities - offers the simple benefit of more data points, spread over all the satellites' positions. But non-coordinated constellations are unable to fully consider redundancy and temporal priority in data collection or take advantage of the data network provided by the rest of the constellation. Particular benefits of constellation-level coordination are discussed below.

Coordinated Observation and Remote Sensing

A significant advantage offered by a coordinated constellation is the ability to align measurements taken by different spacecraft, both spatially and temporally. Figure 1-8 illustrates four different methods for coordinating observations taken by spacecraft. Subfigure (a) represents the case where the same instrument is used to observe a single target on the Earth's surface simultaneously (or nearly simultaneously), which allows for both increased geometric diversity in the measurements being taken as well as cross-validation of one instrument with another of the same type. Subfigure (b) represents the case where two different instruments carried by two different spacecraft are used to simultaneously observe the same target region, which can be used for calibration of one instrument relative to the other. This illustration is meant to show microwave radiometer calibration through GPS radio occultation measurements. Subfigure (c) represents the simultaneous use of many different instruments and sensing modalities over a single target region. Subfigure (d) represents the simultaneous measurement of many different target regions, which could be useful for mapping a large-scale structure in the Earth's atmosphere, for example.

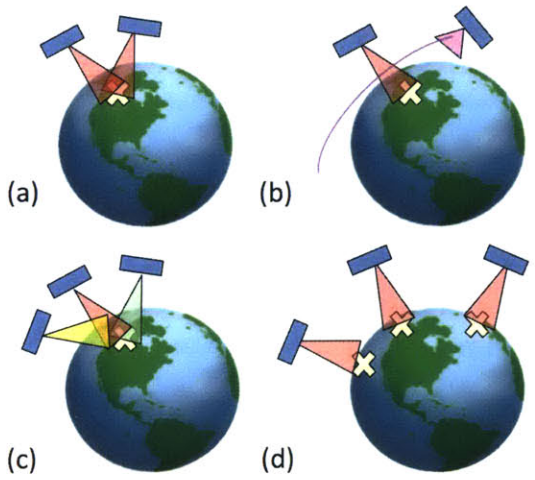


Figure 1-8: Methods for coordinating observations across multiple spacecraft. a) multiple simultaneous observation geometries for a single target region b) calibration of one observation instrument with another c) multiple simultaneous sensing modalities for a single target region and d) organized observation of multiple targets.

The observation scenarios in Figure 1-8 are in most cases infeasible without some form of explicit coordination between the spacecraft involved, due to each spacecraft's limited instrument field-of-view and the need to interleave observation activities with its own management activities (e.g. downlink to a ground station, attitude slews).

Faster Response to Spontaneous Observation Opportunities

Depending on the mission type, there may be opportunities that arise spontaneously for high-value observations. An example of this could be a forest-fire monitoring constellation, which is always on the lookout for fires over large, isolated regions. The satellites are tasked to scan different swathes of the region on every orbit, until one satellite spots a fire and automatically alerts the other spacecraft of its presence so they can perform rapid follow-up observations. Such a constellation could incorporate many small, cheap "scout" satellites that increase overall coverage, and large "mothership" satellites that perform precision follow-up measurements once alerted of high-priority targets.

In this case, constellation-level coordination provides the ability to perform rapid re-tasking and follow up without the need for a ground operator to first assess measurements and manually replan spacecraft activities.

More Effective Data Routing

Another high-impact benefit offered by a coordinated constellation is the ability to achieve more downlinks to ground stations by routing data through the other satellites in the constellation. This scenario is illustrated in Figure 1-9, in which a spacecraft performing an observation forwards its data to a spacecraft in view of a ground station for more rapid downlink. This capability could be used to reduce data latency-to-ground for high priority observations and to help achieve a larger volume of high quality data to ground overall by enabling the constellation to route data through dedicated communications nodes.

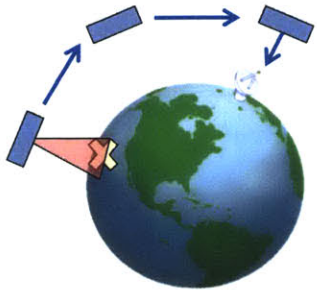


Figure 1-9: Routing data between spacecraft in the constellation

Robustness to Faults on Individual Spacecraft

One general benefit offered by small satellites is the fact that they are relatively cheap to design and build when compared to larger, monolithic spacecraft. In a large-scale CubeSat constellation, the importance of a loss of a single spacecraft is reduced even further by economies of scale. Yet the operational dropout of an individual satellite could still negatively impact the constellation if that satellite's future plans included high priority measurements. In a coordinated constellation, satellites could automatically identify faulted spacecraft via a simple liveness-testing communications protocol and change their plans accordingly. This mechanism could save valuable operational time which might normally be used in debugging the problem.

1.2.3 Constraints Imposed by a Coordinated CubeSat Constellation

While a coordinated constellation promises significant benefits, there are several significant constraints that must be considered for the efficient operation of the constellation. Some of these constraints arise from the tight cooperative nature of the constellation, whereas others arise from the limits of CubeSats in general.

Coupling of Coordinated Activity Planning

In order to achieve the best performance at the constellation level, activities should be scheduled and executed across multiple spacecraft. Any individual satellite has a

schedule of activities (e.g. remote-sensing observations) to perform in the near future, which is determined both based on that satellite's orbital geometry and its geometry relative to other spacecraft.

In a perfect world, there would be no uncertainty in observation target priorities, the satellites' orbital positioning, and onboard resource states. In this case, all observations could be scheduled far in advance. In reality, satellites have to replan their activities based on changing priorities and state updates. When one satellite's activity plan changes, this automatically affects the overall quality of planning across all the satellites in the constellation, and ideally the other satellites would replan their own activities in light of this new information.

The activity planning and scheduling process could be performed in a centralized manner on the ground, in a distributed manner on the satellites themselves, or in an implementation somewhere in between. For any coordinated planning architecture, the coupling in the planning process has two important constraints:

1. The architecture should consider the activity plans of all satellites when planning for any individual satellite

2. The architecture should obtain state updates from satellites and refresh satellite activity plans in a relatively short timeframe

Efficient Use of Limited Communications Availability

The coordinated planning constraints imply a need for relatively frequent communications with the satellites. Unfortunately, CubeSats tend to have limited communications capabilities due to their limited SWaP (Size, Weight, and Power), as evidenced by Klofas' 2013 survey of CubeSat communications systems [42]. Also, due to limited funding and staffing, CubeSat missions have traditionally operated only a single ground station, which can lead to communications blackouts of tens of hours or more due to orbital geometry restrictions.

For these reasons the constellation should be able to gracefully perform coordinated planning and scheduling even with limited ground contacts. One very useful

mitigation of this limitation is the use of inter-satellite links, or "crosslinks", which involve communication between satellites themselves without ground interaction. The use of such crosslinks will be discussed in more detail in Chapter 2.

Management of Onboard Resources

The choice of CubeSats as the constituents of the constellation imposes planning and scheduling constraints due to the limited resources available to the bus.

There are several types of onboard resource that must be carefully considered, including power, data storage, and attitude pointing capability. CubeSats generally do not have enough power available for continuous observation and communication, due to limited battery capacity and the lack of gimbaled solar arrays. Battery levels should be considered explicitly in the planning process to achieve good operational performance. Data is collected both from observation activities and from general engineering telemetry, and the satellites should try to reduce the latency-to-ground of this data as much as possible. Finally, CubeSats have very constrained attitude pointing capabilities, and must schedule any required attitude slews in such a way as to avoid saturation of actuators (e.g. remaining under angular momentum saturation point of reaction wheels).

1.2.4 Constellation Orbit Architecture

Many Earth remote-sensing constellation mission applications can benefit from the use a large number of satellites to reduce average revisit times and increase the daily Earth surface coverage percentage. Generally these satellites should be spread out in a pattern that maximizes their usefulness to the overall constellation. Such a dedicated constellation architecture would most likely be deployed using multiple launches of small sets of CubeSats, because this platform normally has limited propulsion abilities. Two successfully implemented traditional satellite constellation architectures are briefly discussed, followed by some detail on how CubeSats are launched today and where the community is headed.

Traditional Satellite Constellation Architectures

The Iridium communications constellation serves as an example of a successfully implemented constellation orbit architecture. Figure 1-10 shows the orbit architecture used by Iridium. The constellation features 66 satellites in polar low earth orbits at 780 km altitude, organized into 6 orbital planes, with multiple satellites sharing the same plane [33]. The figure depicts the crosslinks that are used between adjacent satellites to shift data around through the constellation. The widely spread-out pattern allows the satellites to achieve high percentage coverage over the Earth's surface and minimize average revisit times to any point on the Earth's surface. This design represents a good, but expensive, dedicated architecture for constellation applications that demand high coverage of the Earth's surface.

The NASA A-Train, or "Afternoon Train" of Earth observing satellites is another example of a successful, dedicated constellation design. The A-train is seen in Figure 1-11. The satellites all fly within a single orbit, with varying along-track distance. This design allows the constellation to deliver a large set of different, roughly simultaneous measurement types, covering the swath of Earth's surface directly below the orbit. While beneficial for achieving temporally coincident measurements, the choice of a single orbit greatly limits the revisit times achieved by the constellation.

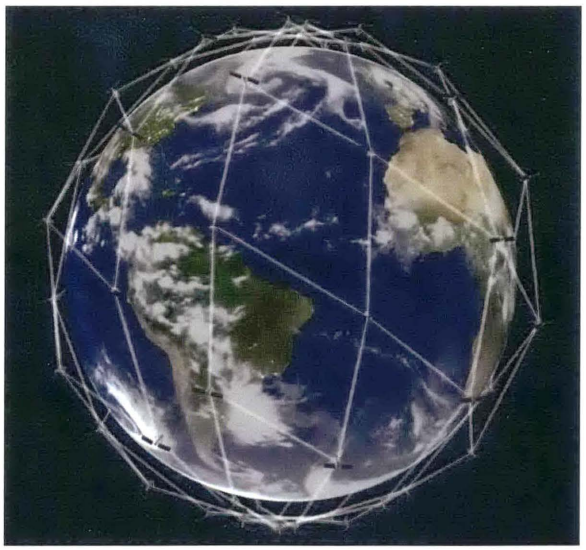


Figure 1-10: Routing data between spacecraft in the constellation [33]

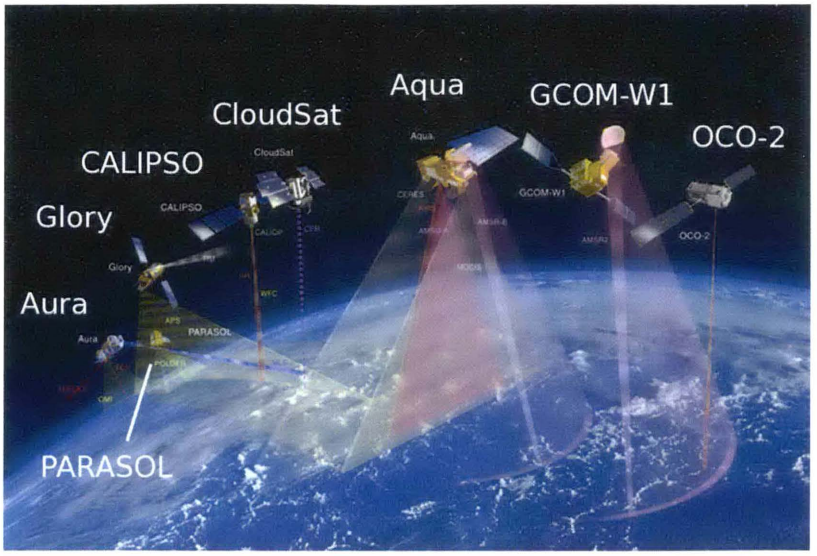


Figure 1-11: The NASA A-Train Constellation [3]

Current and Future CubeSat Constellation Architectures

Up to this point, functional CubeSat constellations have mostly been limited to ad hoc deployments, formed from multiple launches as auxiliary payloads. Planet Labs' Flock constellations are a good example of this [9]. The constellation was not built up in the best way to maximize Earth coverage, due to limited availability of launches.

Commercial CubeSat imaging companies are aiming for more distributed constellations in the future. Planet Labs plans to have a dedicated constellation of hundreds of satellites in a "variety of orbits" [9]. Blacksky, another company, aims to launch 60 Microsatellites by 2019 to achieve frequent revisit rates over 95 percent of the Earth's population [27]. SkyBox is also planning a similar LEO constellation [29]. Such dedicated constellations will be similar to the Iridium constellation in distribution, with the satellites widely spread out to minimize average revisit times and maximize coverage.

1.3 Previous work on Coordinated Constellations

An assessment was done of previous work in areas relevant to the automated operation of a coordinated satellite constellation. These areas included high level analyses of the benefits of Distributed Space Systems (DSS), specific architectures and algorithms for cooperation and coordination between satellites, and general algorithms for multi-agent coordination. The findings of various studies in these areas are discussed in more detail in the following subsections. Note that an "agent" is defined by Russel and Norvig to be "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators" [58]. In this thesis, an "agent" is taken to be synonymous with the flight software running onboard a single satellite. An agent is capable of making planning decisions and communicating with other agents as well as the ground.

1.3.1 Analyses of DSS Applications and Utility

General Utility of a DSS

Several works were examined for general background on the structure, benefits, and limitations of a DSS. These works illustrate the key role to be played by DSS and small satellites in future missions. Truszkowski et al. cover cooperative autonomy in great detail [66], discussing the need for this capability and various models and examples of how a mission using the capability would be structured. Figure 1-12 provides a schematic of how the cooperative planning process can be structured. It shows multiple agents planning their own activities, using these plans to perform activities in a timeline, and then perceiving their state. A mechanism exists for communication between the agents to share state and planning information. The algorithms developed in this thesis also follow this general structure. Shaw, Miller, and Hastings discuss the role of DSS as information transfer networks, and derive a generalized framework for classifying DSS [61]. The authors analyze some of the benefits of a DSS, most notably improved observation availability and improved reliability through redundancy. The authors also discuss the need for autonomous operations for DSS

to avoid domination of operations costs. Jilla and Miller examined the reliability of a multi-satellite interferometer composed of small satellites [38]. They found that the overall reliability of the DSS can be increased while simultaneously reducing required reliability and design effort for any individual spacecraft.

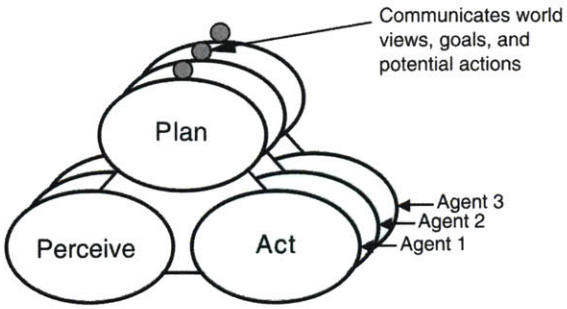


Figure 1-12: Mulitple agents participating in a cooperative planning process [66]

Minimizing Revisit Times

Multiple researchers have examined the capability of small satellite and CubeSat constellations to reduce revisit times and increase coverage abilities in Earth observation applications. Iglseder et al. found that a constellation of 32 small satellites, with 4 sun-synchronous orbit planes and 8 satellites in each plane, can achieve an average revisit time of about 0.5 hours [32]. Marinan, Nicholas, and Cahoy compared the revisit times achievable with an ad hoc CubeSat constellation with a dedicated constellation architecture (specifically, a Walker Delta constellation, discussed more in Chapter 2) [47]. Multiple ad hoc CubeSat constellations were examined, all composed of subsets of the CubeSats deployed as secondary payloads in the period from February 2013 to Jan 2014. The authors found that an ad hoc constellation can achieve a better average revisit time than the dedicated constellation, given the same number of satellites: 0.7 hours versus 0.8 hours, with 6 satellites per plane. But response time, the average time to the next visit to a desired region, suffers: increasing from 2 to 9 hours.

Gangestad et al. also investigated the revisit times achievable using ad hoc consellations assembled from scheduled CubeSat launches, in this case from 2015 to 2017

[24]. They found that an average revisit time of less than 1.0 hour was achievable with only 5 satellites, by strategically picking a combination of low and high inclination orbits. The authors also found that the revisit times of an ad hoc constellation can be more robust to the loss of a satellite than traditional dedicated architectures. Zohar analyzed the ability to achieve and maintain various CubeSat constellation geometries by choosing favorable launch opportunities and controlling the method of deployment from launch-vehicles [73]. He showed that a constellation will reach full deployment due to drift within 40 to 50 days, and that modest delta-V capabilities are required to maintain the constellation in full deployment.

These analyses of constellation geometry and revisit capabilities are enlightening for initial mission planning purposes, but lack analysis of several key consideration. The first consideration is the synergy or redundancy of data collected between the different satellites. A coordinated Earth-observing constellation must consider the utility of data collection at a given target and time to operate efficiently. Also, the analyses don't consider the effect limited onboard resources and operational timing constraints have on activity execution.

1.3.2 Architectures for Multi-Satellite Cooperation

A variety of architectures in the literature were examined. The subsections below delineate and discuss these in terms of their most important characteristics.

Expert-Agent Based Cooperation

Surka et al. and Schetter et al. discuss the implementation of a multi-satellite cooperative architecture and simulation environment based on "expert agents" [62, 59]. In this case, these agents are software modules that operate autonomously within an overarching software architecture. An agent can be in charge of a single satellite, a subset of satellites and agents, or even the entire system. The assignment of agents to satellites and the interfaces between the agents control how centralized the overall architecture is, as illustrated in Figure 1-13. The agents communicate with each

other through a natural-language-like message passing system. The agents are able to negotiate resources and tasks to according to their assigned goals and their existing capabilities (such as access to specific hardware components). The benefit of using expert agents is that they are flexible to wide range of architecture variations and are conceptually elegant in that a single agent can be assigned to a specific skill or set of hardware. However, there is a great deal of complexity and processing overhead associated with the agents, and they don't address the issue of infrequent communications over a widespread constellation.

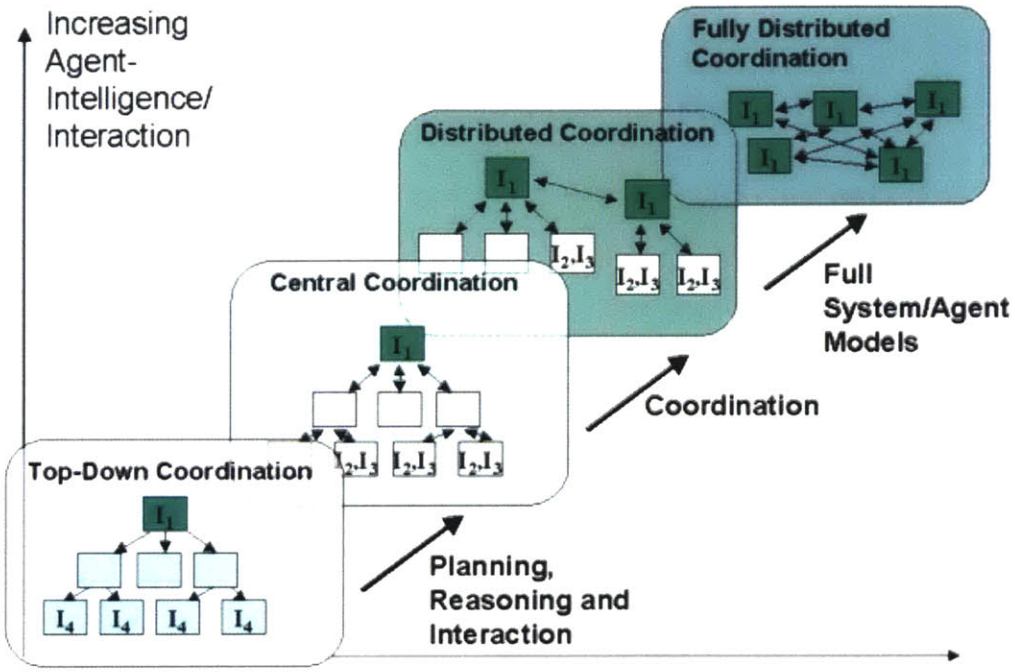


Figure 1-13: Mulitple agents participating in a cooperative planning process. Individual agents (the I's) can correspond to a single satellite or multiple satellites, depending on the overall architecture [59]

A constellation mission, TechSat-21, was designed in the early 2000's to implement this expert-agent based architecture, augmented with a lower-level single-spacecraft planning capability [17]. The mission would have proven several state-of-the-art autonomy capabilities in flight, but was unfortunately canceled. The Three Corner Sat Nanosatellite Mission was implemented to test a subset of these capabilities in advance, but failed to achieve orbit on launch [15]. To the author's knowledge, no

operational mission has demonstrated an expert agent based architecture.

Distributed Task Scheduling

Many researchers have looked into architectures that use a form of distributed scheduling, wherein goals or tasks are assigned to a constellation a whole, then the satellites either directly or indirectly negotiate and select tasks to perform. Damiani, Verfaillie, and Charmeau designed a system that features individual spacecraft planning their own activities and managing onboard resources, and a centralized ground station that distributes observation requests to satellites [20]. No direct crosslinks are used. Tripp and Palmer also discuss a system that doesn't feature crosslinks, but rather "stigmergy", effectively a feedback loop between satellites and a ground station that allows new tasks to be introduced and avoids duplicate task assignments [65]. Das, Wu, and Truszkowski design a system where the satellites accept high level goals and allocate tasks through crosslinks [23]. The system uses a high level hierarchical task syntax to reason about task decomposition.

Van der Horst takes a slightly different approach to task allocation, using a market based-scheme in which satellites make bids on tasks, and the highest bidding satellite performs the task [67, 68]. This scheme does not require a connection to a centralized ground station, and minimizes crosslink usage to only neighboring satellites. However, the architecture does not handle resource utilization onboard individual spacecraft.

Other Architectures: Swarms, Formation Flying, and Fractionation

A great deal of focus has been placed on formation flying in recent years, in which multiple spacecraft fly at precise separations to gain synergy between the instruments located on separate satellites. Maintenance of this separation is often achieved using differential GPS navigation and then close proximity sensors, as on the PRISMA mission [1]. A similar CubeSat-based formation flying demonstration is currently being designed for the QB50 constellation [25]. This maintenance inherently requires a degree of cooperative scheduling between the satellites.

Swarms are another example of a cooperative constellation or cluster, in that

they often use a limited amount of communication and interaction to achieve some global objective, such as a reconfiguration of an entire group of spacecraft. Multiple systems have been outlined that perform reconfiguration using a decentralized control algorithm [49, 37]. Rouff discusses an envisioned swarm mission that will explore the asteroid belts and autonomously decide how best to task individual craft with exploration objectives [57].

Fractionated spacecraft are another distributed architecture that separates out the functions of a traditional, monolithic spacecraft over many small "satlets". LoBosco et al. describe an architecture for national security purposes [46] Such systems are very cooperative, but require significant inter-satellite communications.

Hanson et al. discuss a distributed in-site measurement collection mission implemented on a set of CubeSats: the Edison Demonstration of Smallsat Networks (EDSN) [30]. The CubeSats fly in a cluster and take turns answering data requests from a single master satellite via crosslink. This is a pathfinder mission that represents one of the first actual applications of cooperative autonomy on a multi-small satellite mission.

1.3.3 Algorithms for Generalized Multi-Agent Coordination

There has also been a great deal of work on multi-agent cooperation outside of space applications. Chien et al. built and implemented a set of resource-aware automated planning algorithms which were to be included in the TechSat-21 mission [14, 16, 17]. The "tBurton" factored planner extends the classical planning approach to planning for interactions between multiple agents by reasoning about the temporal inter-dependencies of their tasks, and exploiting hierarchy to reduce the planning complexity [71, 72]. Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) can be used to find an optimal, decentralized plan across multiple agents by explicitly reasoning about uncertainty in the agents' performance of activities [5, 6]. Finally, Gombolay, Wilcox, and Shah, implemented a set of algorithms using Mixed-Integer Linear Programming (MILP) to quickly allocate and schedule sets of tasks to multiple robots on a factory floor.

These approaches generally do not consider how each agent can manage its own onboard resource constraints, and how that affects higher level planning. Market based task allocation schemes can incorporate this individual resource management to some extent; all agents make bids on tasks based on their own valuation of the value of the tasks [18]. Yet none of these approaches specifically deal with the question of how to gracefully manage each agent's limited resources in the context of the entire group.

1.4 Contributions and Organization of Thesis

This thesis details the derivation of two algorithms for the automated operation of a coordinated constellation, and assesses their performance in varying orbit geometries and levels of communication usage. The thesis makes four main contributions:

1. Development of an operational model for a resource-constrained, Earth-observing CubeSat platform which can be used for automated planning
2. Development of a receding-horizon, linear-programming-based algorithm for onboard activity planning which reasons about resource availability
3. Development of a consensus based algorithm for distributed prioritization of observations across a constellation, which is robust to limited access to inter-satellite crosslinks
4. Analysis of the effects of varying orbital geometry and crosslink usage on the performance of the constellation

The rest of this thesis is organized in the following way. Chapter 2 provides an introduction to the simulation cases run in this thesis, and background on the algorithms and metrics used for assessment. Chapter 3 details the Resource-Aware SmallSat Planner (RASP) algorithm, which is used for activity planning on a single satellite and the Limited Communication Constellation Coordination (LCCC) algorithm, which handles planning information sharing across the constellation to

maximize the utility of observations. Chapter 4 discusses the results obtained from the simulation cases run. Chapter 5 concludes with a discussion of the limitations of the algorithms and items of interest for future work.

Chapter 2

Approach

In this thesis, a simulation was constructed and executed for a coordinated CubeSat constellation that attempts to minimize average revisit times for a set of onboard sensors. "Revisit time" was defined in Chapter 1 as the time between one satellite (or sensor) ending an observation of a target region on the Earth's surface, and the start of the next observation. Two algorithms, the Resource-Aware SmallSat Planner (RASP) and the Limited Communication Constellation Coordinator (LCCC) are run onboard the satellites in a distributed fashion. The algorithms utilize planning information that is shared between the satellites to inform each individual satellite's activity planning. This shared information allows better decisions about whether or not to observe a given region and which type of sensor to use for the observation. This general framework is shown in Figure 2-1, for a three satellite constellation.

The rest of this chapter details the approach taken in modeling the coordinated constellation under investigation.

2.1 Constellation Orbital Geometry

The selection of orbits for the CubeSats in the constellation is critically important to achieving both good geometric coverage of the Earth's surface below, and for allowing sufficient communications connectivity between the satellites. As mentioned in Chapter 1, a constellation of 32 small satellites in sun-synchronous orbits can

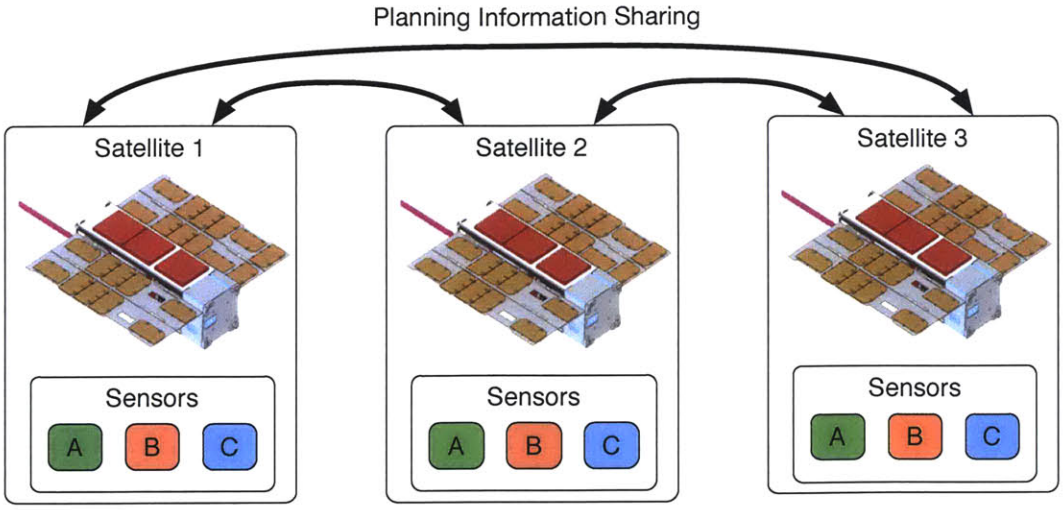


Figure 2-1: The framework for the coordinated constellation investigated in this thesis. Satellites make decisions about which sensor to use during observation opportunities, based on their most up-to-date knowledge of the constellations' plans as a whole. MiRaTA CubeSat image from Kennedy and Cahoy [40]

achieve an average revisit time of about 0.5 hours [32].

In this thesis, the cooperative constellation was investigated in two overall orbit architectures:

1. Stitched Walker Star constellation

2. Ad Hoc constellation

The Walker Star constellation, originally introduced by Walker [70], has been extensively studied in the literature [47, 24, 31, 51, 20, 54]. The Walker Star constellation spreads out the satellites over multiple, regularly spaced, polar orbits, which provides good coverage over most of the Earth's surface. Walker proposed the following notation for the general Walker-type constellation:

$$i : T/P/F \tag{2.1}$$

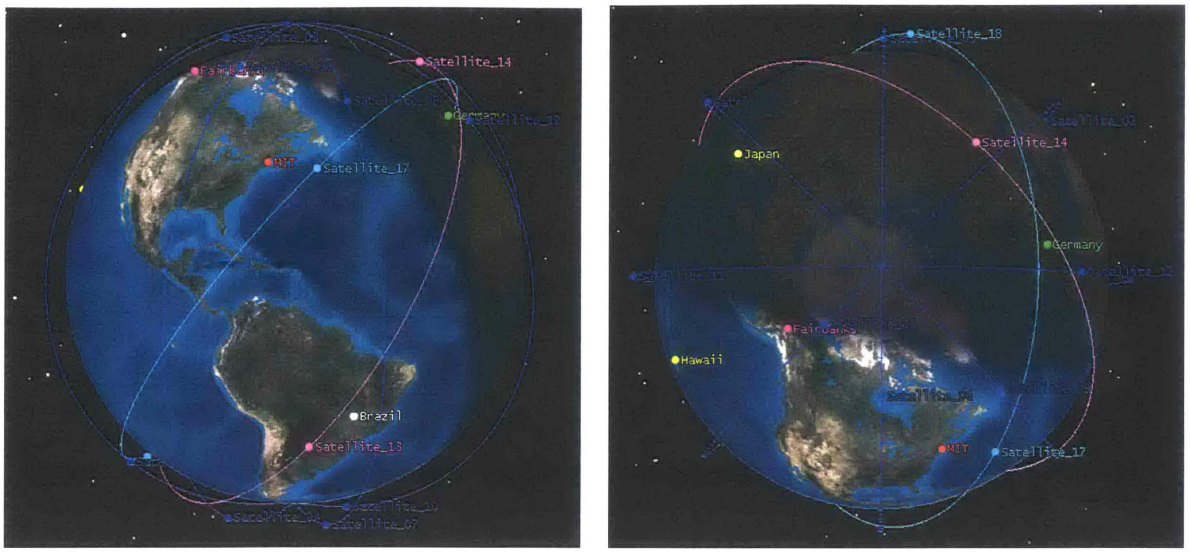
where a total number of T satellites are distributed in P evenly spaced orbit planes (by Right Ascension of the Ascending Node, RAAN) at an inclination of i. There is a

true anomaly phase difference of $F * 360/T$ degrees, where F is a phasing parameter constrained to 0 to $P - 1$ [70]. In a Walker Star configuration, i is set to 90, making the orbits polar. Normally the orbits are circular and at the same altitude. The phase difference is traditionally an integer, and should be large enough to prevent collisions between satellites in adjacent orbits.

A specially modified Walker Star was constructed, referred to as a "Stitched Walker Star". Figure 2-2 shows two views of this constellation. The main component is a $90 : 12/4/0.33$ Walker Star, with three satellites spaced at 120 true anomaly difference. The phasing parameter here is chosen to be small so that the star satellites pass closer to each other at the poles and thus have more crosslink opportunities. All the satellites that regularly pass each other at the poles are referred to as a "phase bracket". For this particular Walker Star architecture, the different phase brackets are too far apart to allow for crosslinks between them. For this reason, the constellation also incorporates two additional lower altitude orbits that "stitch together" the star orbits. That is, the satellites in these orbits are placed such that they are able to crosslink with multiple satellites in the star orbits, allowing more information exchange between the star satellites. This provides an information "conveyer belt" and mitigates the problem posed by the lack of crosslink opportunities between different phase brackets in the star orbits. Only 18 satellites were included in the constellation in order to limit the time required to run a full operational simulation.

The Stitched Walker Star constellation represents a best case scenario, in which there is total freedom to choose the satellites' orbits. However CubeSats are often launched as secondary payloads, and mission designers don't have a choice which orbit they end up.

The second constellation, the Ad Hoc, reflects a more feasible case. This constellation is formed opportunistically from CubeSat deployments over multiple launches. For this work, it is based on the second of the two ad hoc constellations analyzed by Marinan, Nicholas, and Cahoy from an assessment of launch opportunities for CubeSats in the 2013 calendar year [47]. This constellation is shown in Figure 2-3. The six different orbits are based on six different launches. Four are sun-synchronous,



(a) Side view

(b) Top view

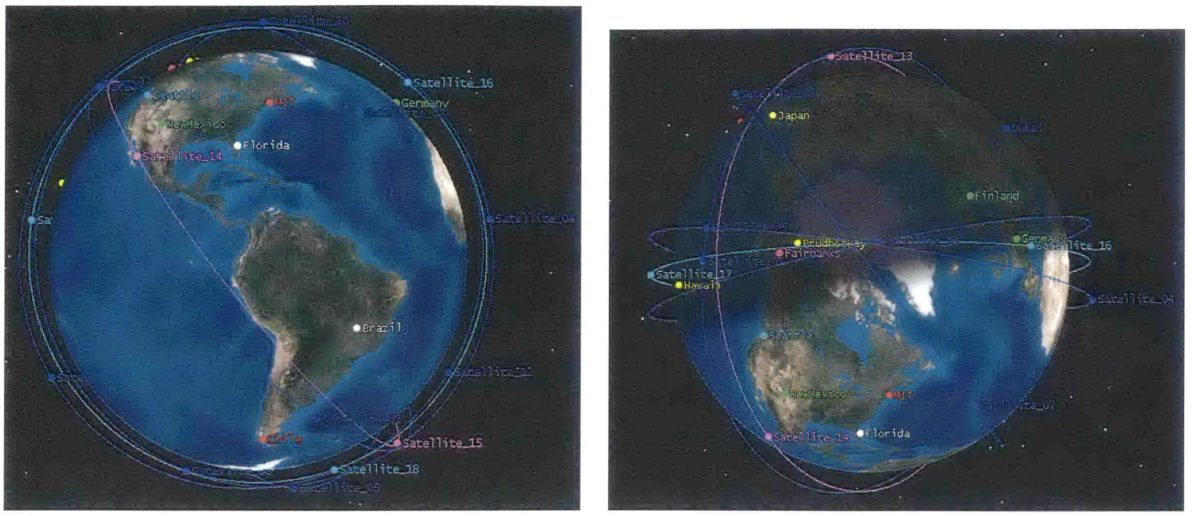
Figure 2-2: The "Stitched Walker Star" constellation. Blue orbits are components of a traditional 90 : 12/4/0.33 Walker Star. Cyan and magenta orbits cut through the star, providing more crosslink opportunities. Images produced with AGI's STK software.

with an inclination of 98 degrees. The other two orbits are at 51 and 52 degrees inclination. As in the stitched walker star case, three satellites were placed in each orbit and spaced at 120 degrees separation in true anomaly. This gives 18 satellites total over the six orbits.

Table 2.1 summarizes the high level parameters for each of the constellations. Note that It is assumed for this work that the satellites are able to achieve their initial true anomaly separation once released from the launch, and maintain this separation automatically. The specific orbital parameters for both constellations are summarized in Appendix A.

Table 2.1: Summary of Constellation Orbital Parameters

Constellation Geometry Type	**Altitude (km)**	**Number of Planes**	**Satellites per Plane**	**Inclination (degrees)**
Walker Star	500, 600	6	3	90
Ad Hoc	600 to 825	6	3	98, 51, 52



(a) Side view

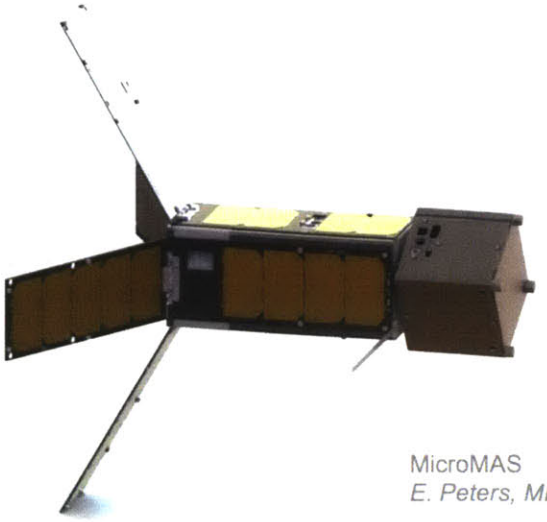
(b) Top view

Figure 2-3: The "Ad Hoc" constellation. Blue orbits are sun-synchronous, with an inclination of 98 degrees. Cyan and magenta orbits are at 51 and 52 degrees inclination, respectively. Images produced with AGI's STK software.

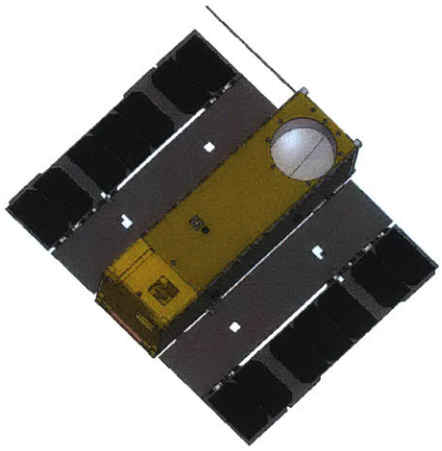
2.2 Satellite Attributes and Resource Constraints

For this work, a set of CubeSats are considered that are similar in performance to the MicroMAS (Micro-sized Microwave Atmospheric Satellite) and MiRaTA (Microwave Radiometer Technology Acceleration) CubeSat science missions [8, 7, 12, 41]. The MicroMAS and MiRaTA spacecraft are shown in Figure 2-4. These CubeSats were designed by the Massachusetts Institute of Technology (MIT) and MIT Lincoln Laboratory, in cooperation with The Aerospace Corporation, the Space Dynamics Laboratory, and the University of Massachusetts at Amherst. MicroMAS was launched in July 2014 and began operations upon deployment from the International Space Station in March 2015, and MiRaTA is currently under development for a launch in 2016. The CubeSats are representative of the increasingly complex missions that can be performed with small satellites, and their hardware is fairly standard for a 3U form factor. Also, the familiarity with these missions helped in developing a spacecraft model for use in the algorithms.

These CubeSats are approximately 10 cm x 10 cm x 34 cm in dimension and have a mass of a little above 4 kg. MiRaTA can store 20 Wh of energy and produce about



(a) The MicroMAS CubeSat [8]



(b) The MiRaTA CubeSat (Courtesy Timothy Cordeiro, MIT, RR number JPSS 15-038)

Figure 2-4: Example CubeSat missions

25 Watts of power when pointed directly at the sun. Mission lifetime depends on launch opportunity and orbit. The spacecraft are designed for 1 year lifetimes. Both spacecraft have a set of subsystems necessary for achieving their mission objectives, including systems for electrical power distribution and storage, communications, command and data handling, attitude determination and control (using a suite of sensors and actuators, but no propulsion), and a scientific payload. MiRaTA carries a GPS receiver that provides precise positioning. Both satellites collect science data on orbit and downlink it to the mission ground station at the NASA Wallops Flight Facility in Virginia.

For this work, the satellite models have similar resource constraints to MiRaTA: 20 Wh of energy storage and a goal to downlink science data to ground as soon as possible. Energy is consumed at different rates depending on the spacecraft mode, but it is assumed that energy can only be produced (at 24.8 W) during a dedicated recharge mode when the satellite points its solar panels directly at an optimal angle to the sun. The desired maximum depth of discharge (DOD) for the batteries is 30% (14 Wh). Science data is produced by the sensor payload at a rate of 63 kbps (kilobits per second) during observation activities, which are restricted to the times during which the satellites fly over specific target regions of the earth's surface. Engineering

telemetry is produced at a constant rate of 10 kbps during all onboard activities except for downlinks to the ground. Onboard data storage is fixed at an upper limit of 100 MB ($1MB = 1000^2 Bytes$). This amount, while less than what can be physically stored with standard commercial nonvolatile memory (on the order of GB), encourages the onboard planning process to avoid delay or loss of data prior to transfer to ground, given that current state of the art CubeSat radio downlink rates are at best on the order of 2 Mbps and more typically $<$ 100 kbps [42]. The satellite model also assumes an active 3-axis attitude control system, which performs slews to change the spacecraft's attitude between most activities.

It is assumed that there are three types of payload sensors available for the satellites to choose from when performing observations. This is indicated in Figure 2-1. Only a single sensor can be used for each observation activity. All sensors produce data at the same rate and consume energy at the same rate (6.7 W).

The satellites are modeled without any onboard propulsion system, and it is assumed they cannot change their orbit at all. It is also assumed that orbits can be determined well enough that satellite orbit position uncertainty has a negligible effect on onboard planning quality (achievable with the less than 1km uncertainty from two-line elements found by Coffee, Cahoy, and Bishop [19], or even occasional 10s of km uncertainty found by Riesing [56]) and that the effects of orbital perturbations on planning are negligible over the course of a 24 hour period.

The satellites' attributes are summarized in Table 2.2.

2.3 Communications Architecture

There are two types of communications link that each satellite uses. First, the satellites must communicate with ground stations in order to downlink collected science data and engineering telemetry. Second, the satellites must communicate with each other via cross-links in order to share planning information. For this work, it is assumed that crosslinks are reserved only for sharing planning information, not for transferring science and engineering data.

Table 2.2: Summary of Satellite Attributes and Resource Constraints

Attribute/Resource	Summary
Observation Sensors	3 total; sensors A, B, and C
Energy Storage	20 Wh (Watt-Hours), max 30% DOD
Energy Production	24.8 W, only in dedicated recharge mode
Energy Consumption	Varies by mode; sensors consume 6.7 W each
Data Storage	100 MB
Data Production	Engineering telemetry always produced at 10 kbps, Sensor payloads produce 63 kbps
Data Downlink	2.6 Mbps * 70%

2.3.1 Downlink

The constellation can downlink through a number of ground stations distributed through multiple countries. Nine ground stations were chosen for the simulation, with their placement based on the commercial, CubeSat-oriented ground station network being built by the Spaceflight Industries company [36]. These ground stations are shown in Figure blah. The satellites use a downlink data rate of 2.6 Mbps uncoded, based on the MiRaTA satellite's onboard radio capabilities [41]. For this work, that rate is reduced by a factor of 70%, to represent time lost to link maintenance and processing overhead. It is assumed the satellites can maintain this data rate above an elevation mask of 10 degrees as viewed by the ground station. For this work, it is assumed that any overlapping downlinks from multiple satellites to the same ground station can be handled.

2.3.2 Crosslink

The satellites use inter-satellite crosslinks to share planning information between each other. As an example, Figure 2-5 shows a multi-satellite crosslink event. The crosslinks are performed using a simple, low data rate, omnidirectional radio link,

readily achievable with commercial-off-the-shelf hardware for CubeSats [17]. Link budget calculations with representative hardware indicate that a 9600 bps crosslink can be achieved at inter-satellite distances up to about 2400 km, using 4W of transmit power, FSK modulation, assuming no atmospheric attenuation, and maintaining 3 dB of link margin. A simple geometric calculation with two conservatively low 300 km circular orbits shows that crosslinks at this distance pass well above the top of the atmosphere at 100 km. The satellites must only maintain a crosslink for a brief time, on the order of 10 seconds, to transfer planning information for a 30-satellite constellation.

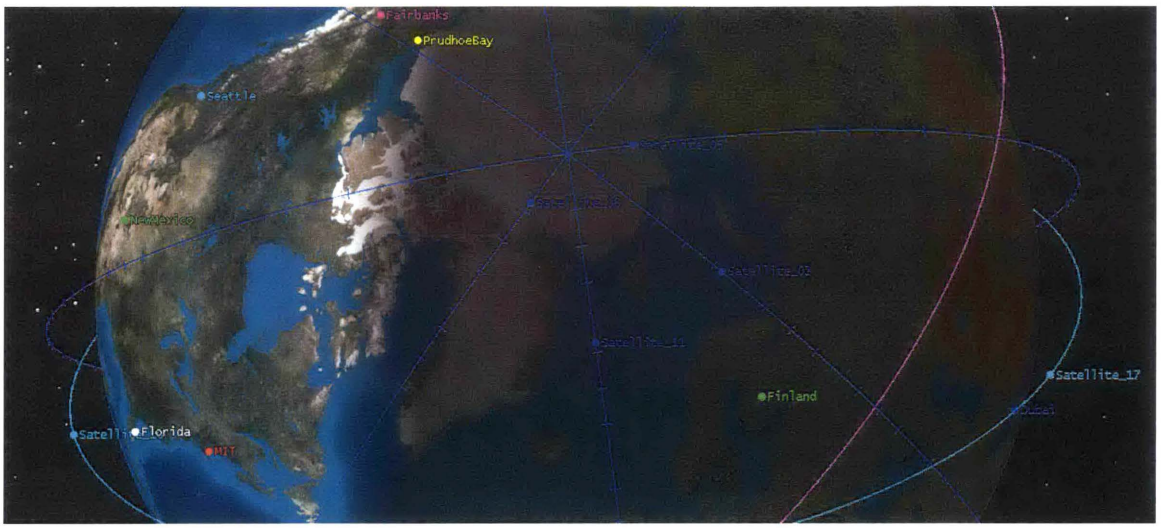


Figure 2-5: An example, multi-satellite crosslink event. All intersatellite distances are less than 2400 km. Image produced with AGI's STK software.

2.3.3 Commlink

A final type of communications link, the "commlink", gives the satellites access to a commercial LEO communications constellation, such as the Iridium or Globalstar constellations [33, 28]. This is helpful for simulating a situation where the satellites have regular access to a backbone communications network. The onboard commlink radio is assumed to also be omnidirectional and have the same parameters as the one used crosslink, except that access to the backbone constellation is available from any

point in LEO. This is not an unrealistic assumption, considering that such communications constellations are generally composed of large satellites with robust radio systems.

Table 2.3 summarizes the important communications architecture details. The detailed parameters for the grund stations are included in o B.

Table 2.3: Summary of Satellite Attributes and Resource Constraints

Item	Summary
Downlink Availability	Above 10 degree ground station elevation mask
Ground Stations	9 total: Brazil, Fairbanks (Alaska), Germany, Hawaii, Japan, MIT (Boston, Massachusetts), New Zealand, Singapore, South Africa
Crosslink Availability	Satellites within 2400 km, both in crosslink mode
Commlink Availability	Anywhere in LEO

2.3.4 Simulation Communications Contexts

The constellation is simulated in multiple "communications contexts", as summarized in Table 2.4. The communications context refers to satellites' use of communications links to share planning information between each other.

The "No Info Sharing" context models a non-cooperative constellation: the satellites only use their own knowledge to decide when to perform science observations and what sensors to use. In the "Downlink" context, the satellites share their latest planned observation activities with the ground station whenever they perform a downlink, and also retrieve the ground's latest knowledge about other satellites' observation plans. It is assumed that all ground stations are linked via a dedicated network and have a single database of the latest planning information obtained from the satellites.

In the "Crosslink" context, the satellites only share planning information via crosslinks. This happens whenever the satellites are both in crosslink mode and

within 2400 km of each other. Crosslink events can also include more than two satellites, either when all the satellites are within 2400 km or via a "multi-hop" topology through a common intermediary satellite. Each satellite keeps its own database of the latest known planning information for all satellites, and updates the information for the respective satellite whenever new information is obtained, either directly from the satellite itself or via a multi-hop information exchange. Note that each satellite individually decides when to perform a crosslink, with the result that some crosslink opportunities are not executed, because not all the satellites commit to them. However, satellites individually try to perform as many cross-links as possible.

The "Crosslink + Downlink" context utilizes both of these mechanisms for information sharing. The "Commlink + Downlink" context uses regularly-spaced commlinks instead of crosslinks. The backbone communications constellation is assumed to be connected to the same shared information database as the ground stations. The satellites must also use resources to perform a commlink, but they have frequent access to planning information from across the constellation. For a given constellation orbit architecture, it is expected that the Commlink + Downlink context will perform the best.

2.4 Coordination Algorithms

Two algorithms were developed for planning the satellites' onboard activities. These algorithms include the Resource-Aware SmallSat Planner (RASP) and the Limited Communication Constellation Coordinator (LCCC) algorithms. The algorithms are organized in a hierarchy as shown in Figure 2-6. The low-level RASP algorithm runs onboard each spacecraft, or "Agent", planning the activities of that agent alone. Conceptually, RASP can be thought of as an standalone module in the software running on the satellite's computer. The high-level LCCC algorithm runs in a distributed manner across all the satellites, and is not a standalone software module but rather an interface between the agents. It organizes the sharing of activity planning information through communications links, and feeds this information to RASP. RASP uses

Table 2.4: Summary of Constellation Communications Contexts

Context Number	Communications Context	Description
1	No Info Sharing	Satellites do not share planning information with each other
2	Downlink	Planning info shared via downlink, with single ground database
3	Crosslink	Planning info shared via crosslink, with separate databases on each satellite
4	Crosslink + Downlink	Planning info shared via both downlink and crosslink
5	Commlink + Downlink	Satellites routinely share info with an external comm constellation backbone, as well as the ground

the planning information to inform its decisions about what activities to perform.

RASP and LCCC will be discussed In detail in Chapter 3. The following subsections give general background for the algorithms and describe the software tools used in their development and simulation.

2.4.1 Algorithm Background

RASP

The RASP algorithm is based on an optimization technique known as Mixed Integer Linear Programming. Linear programs are well-known in both the research community and everybody operations and logistics. They have been extensively studied and documented in the literature since their introduction by George Danzig in the 1940's [21, 50, 58]. 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 such:

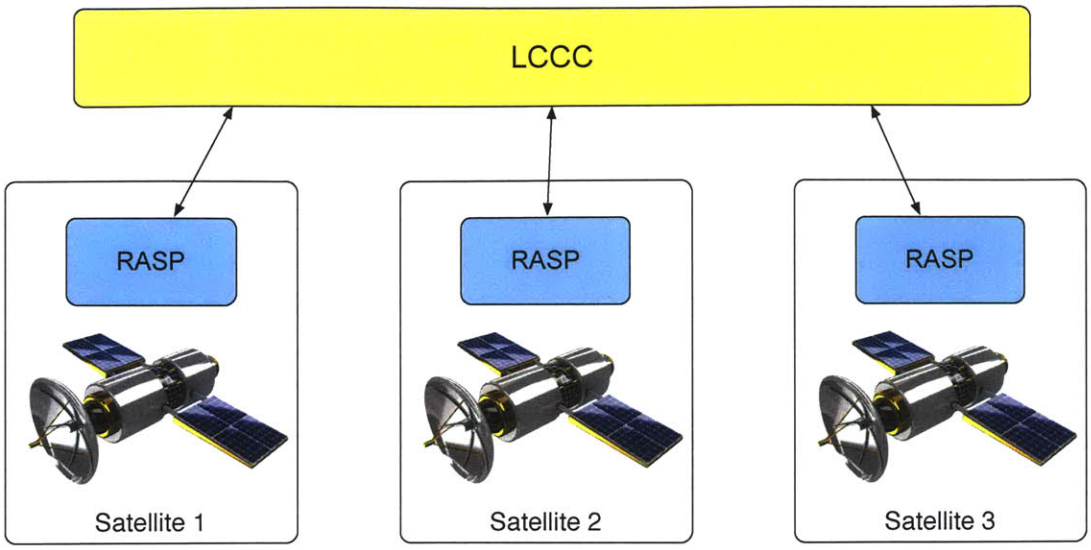


Figure 2-6: The hierarchical organization

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

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

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

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. $\mathbf{x}$ can represent anything: for example, the start and end times for a set of activities or varying amounts of materials to choose from several stockpiles. Flexibility of this formulation was key to its widespread adoption in the industry and academia.

A Mixed Integer Linear Program (MILP) Is a variant on a linear program that constrains some of the decision variables to be from a finite set, for example the integers. Such integer variables are often useful representing disjunctive conditions in the problem; for example, requiring certain activities to be performed before others, or specifying different resource stockpiles in different situations.

One common technique for solving linear programs is Dantzig's Simplex method

[22], which attempts to find an optimum point by moving along the edges of the convex region. A faster technique, known as the Dual-Simplex [69], is used for solving linear programs in this work. Solving MILP-based problems generally involves solving linear programs, with an extra added layer of searching through assignments to the finite decision variables. RASP accomplishes this through a depth first search (DFS) technique, as explained in Chapter 3.

LCCC

The LCCC algorithm takes inspiration from market-based task allocation techniques, such as those analyzed by Choi [18] and Van der Horst [67]. These techniques feature a bidding process, in which each agent submits a bid for each task from a set of tasks. This bid generally reflects the reward that the agent can derive from performing the task, or the cost for the agent to perform the task. A higher level algorithm manages the selection of winning bids from the agents, as well as ensuring that there are no conflicts in task assignments. These algorithms are "distributed", in that is not necessary for a central manager to assign tasks to the agents; the agents figure out their assignments themselves. However, finding an optimal task assignment and deconflicting task assignments is expensive from a communications perspective, often requiring many iterations between the agents.

2.4.2 Software Tools Utilized and Developed

A simulation environment was developed to analyze the performance of the RASP and LCCC algorithms. Several software packages were used or developed as part of this process.

Orbital geometry was analyzed using the Systems Tool Kit (STK) software package from AGI. Input data was extracted based on the satellites' orbits and fed into the constellation simulation, including the Earth-Centered, Earth-Fixed (ECEF) coordinates of the spacecraft, eclipse durations and timing, and the ECEF coordinates of the set of ground stations.

The RASP algorithm was implemented in the MATLAB programming language from MathWorks, using the linprog() function and dual-simplex optimizer for linear program solution. The dual-simplex method was found to be the fastest for RASP, at least in its current form. RASP was run in a receding horizon fashion: the algorithm was used to plan an activity timeline over a certain planning horizon, the satellite's state was propagated forward for a period of time using that timeline, and then activities were replanned.

A software package was developed in the Python programming language, from scratch, to simulate the on-orbit operation of a constellation with an arbitrary number of satellites planning and scheduling though RASP and LCCC. The simulation environment initially ingests orbit data from the STK analysis and uses this to determine activity windows for every satellite. Satellite information is stored in a custom Python class, with an instance for every individual satellite. The simulation keeps track of a global clock for the constellation, propagates satellite resource states forward, calls the RASP algorithm for replanning at regular intervals, maintains satellite and ground planning information databases, and shuttles planning information around as specified by the crosslink, commlink, and downlink activities. The global clock was configured to run with 1 second ticks.

2.5 Algorithm Performance Metrics

Several metrics were used for assessing the coordinated performance of the constellation.

The first is the average revisit time for a given region and sensor combination. The average revisit time for a given region and sensor is calculated by averaging all the time differences between the end of one observation of that region by any satellite and the start of the next by any satellite (as well as the start and end time of the overall scenario). These average times were subsequently averaged across all satellites for a given simulation case.

The second metric is the number of information-sharing communications links

available and performed by the constellation. This metric serves as an initial assessment of how well a given constellation should perform at information sharing.

A third set of metrics assesses the actual information-sharing performance of the constellation by examining how long it takes for information to propagate across the constellation, and how relevant and timely that information is when it arrives. These metrics are summarized in Table 2.5. Average initial creation latency measures how long it takes for information to propagate between all pairs of satellites in the constellation. Average initial creation timeliness measures how timely the shared planning information was for the receiving satellite, when that information was heard for the very first time. We restrict this information only to "matching" observations, or those that have a start time within a fixed cutoff before an observation of the same region by the receiving satellite. Using this cutoff forces the metric to only look at timely and relevant observations for planning purposes, as opposed to ones that occur days beforehand. Last change timeliness is roughly the same thing, but based on when the receiving satellite heard about the "last change" to the observation: the last time the originating satellite either created the observation, changed the sensor type, or swapped between planning and canceling the observation. This last change represents a change in the originating satellite's plans, and could cause the receiving satellite to change its plans as well. Finally, the ratio of matching observations measures how large of an effect one satellite can have on the plans of another satellite. The larger the ratio, the more important knowledge of the originating satellite's plans is for the receiving satellite.

A fourth set of metrics measures the resource management effectiveness of the RASP algorithm. These metrics are the average energy storage (ES) and data storage (DS) margin, measured at the end of every activity in an activity timeline created by RASP. ES margin is the difference between the ES state and the ES lower limit, and DS margin is the difference between the DS state and the DS upper limit.

The final metric is the average execution time taken by RASP to successfully create an activity timeline.

Table 2.5: Summary of Information Sharing Metrics Used In Simulation Assessment

Metric	**Description**
Average Initial Creation Latency	Time from first scheduling of observation activity by originating ("From") satellite to first reception by receiver ("To") satellite, averaged over all observations received by receiver satellite
Average Initial Creation Timeliness	Difference between first reception time of observation of originating ("From") satellite by receiver ("To") satellite and start of next matching observation of receiver satellite, within given cutoff, averaged over all observations received by receiver satellite
Average Last Change Timeliness	Difference between reception time of "last change" to observation of originating ("From") satellite by receiver ("To") satellite and start of next matching observation of receiver satellite, within given cutoff, averaged over all observations received by receiver satellite
Ratio of Matching Observations A/B	Out of B observations planned or planned and canceled by ("From") satellite, the number A of those observations received by and matched to observations of ("To") satellite, within given cutoff, averaged over all observations received by receiver satellite

Chapter 3

The RASP and LCCC Algorithms

The operations automation problem was broken down into two phases: 1. coordination of observations at the constellation level and 2. planning of onboard activities on individual spacecraft. In this work, an "observation" constitutes a fly-over of a target region on the Earth's surface. When the sub-satellite point crosses into or exits the region, the observation starts or ends, respectively. At the constellation level, satellites share information about planned observations via crosslinks, downlinks, and comm links to a communications backbone constellation. Each satellite uses its latest knowledge about other satellites' plans to determine preference weightings for its own possible observation activities. Using these preferences, the satellite performs a lower level planning process during which it selects an achievable set of observations and crosslinks while keeping its own onboard resource constraints in check.

The following sections detail the satellite operational model used for the planning process, the low-level RASP planner, and the constellation-level LCCC algorithm.

3.1 Small Satellite Operational Model

A simple but practical operational model is used for the CubeSats that compose the constellation. Figure 3-1 shows the Concept of Operations (ConOps) for the constellation. The satellites individually decide when to perform observations of regions on Earth's surface. Crosslinks are used to trade planning information between

the satellites. Commlinks are for communicating with a communication backbone constellation. The satellites collect science data during observations and engineering telemetry all the time, which they must downlink to ground stations to stay within onboard data storage limits. They periodically perform dedicated sun-tracking recharge activities to maintain energy storage limits. For this simple model we assume that all activities are mutually exclusive; that is, the satellite can only perform one activity at a time. Also note that either crosslinks or commlinks (but not both) will be used, depending on the communications context.

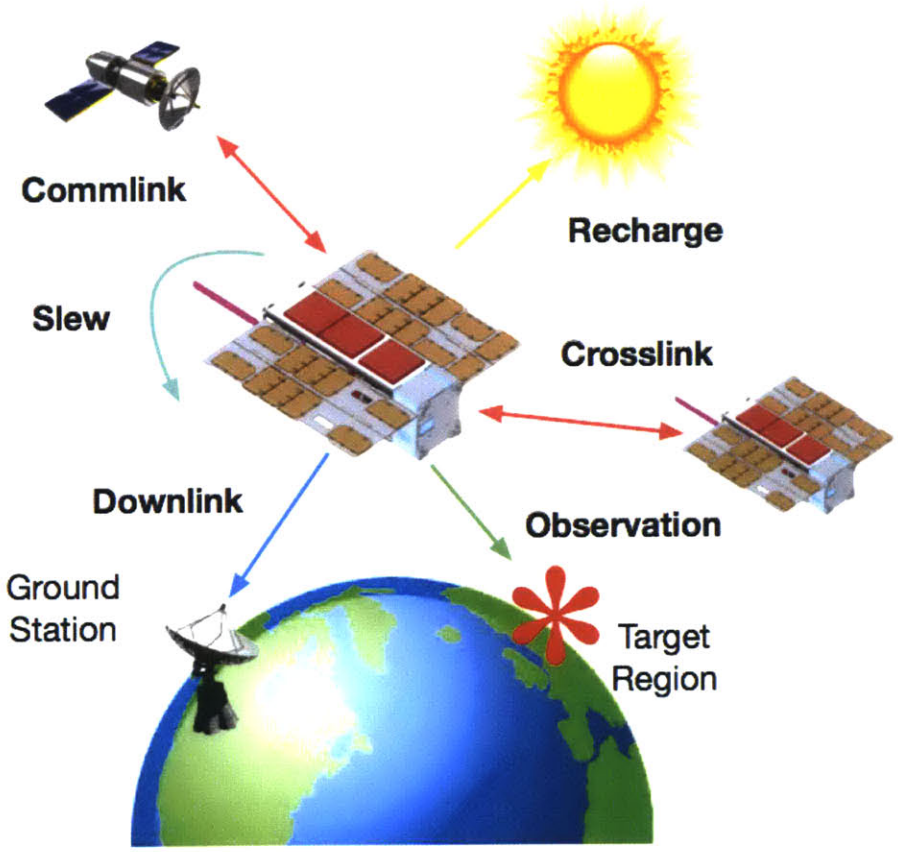


Figure 3-1: Illustration of the operational ConOps for a coordinated CubeSat constellation. MiRaTA CubeSat image from Kennedy and Cahoy [40]

This small satellite operational model is encoded as a state machine for the purposes of planning, as illustrated in Figure 3-2. This diagram shows all the possible transitions between activities or "operational modes" onboard the satellite. In general, all indicated transitions are possible; however, slews are treated specially, as

discussed below.

The coordination activities are those that the satellite plans based on its knowledge of the plans of other satellites. Resource management activities are those that the satellite uses to keep its onboard resources in an acceptable range. The transition activities are used to transition between the other modes. The slew activity is when the spacecraft uses the ADCS actuator suite to change its attitude. A conservative assumption is made that a slew must always immediately precede observations, recharges, and downlinks, in order to put the spacecraft at the right attitude for those activities. An omnidirectional crosslink radio is utilized, so a slew is not necessary before the crosslink activity. The idle activity is a general, lowest-energy consumption state to which the spacecraft transitions when it is not performing other activities.

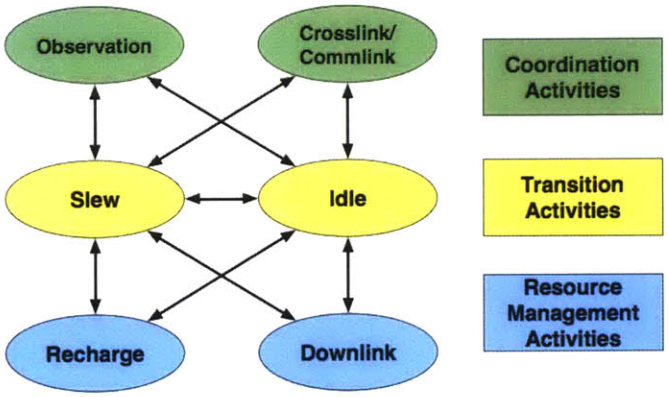


Figure 3-2: State machine representation of the operational activities for a single CubeSat

There are two types of onboard resource which the satellite must manage during operations planning: energy storage (ES) and data storage (DS). In this model, every activity produces or consumes each resource at a constant rate. The resource usage rates are broken down by activity in Table 3.1. Note that the recharge and downlink activities are used to manage the spacecraft's ES and DS, respectively. The final row specifies the minimum required duration of each activity.

The energy usage values in the table were based off of those for the MiRaTA spacecraft [41] and follow the details from Table 2.2. We assume here that energy is

only produced in recharge (Rech) mode (at 24.8 W, minus power consumption). The spacecraft will in actuality have solar power input during other modes, depending on orbital geometry. The restriction to recharge mode here simplifies the model while maintaining a conservative estimate of power production.

The data production values in the table include the production of spacecraft engineering telemetry as a base (10 kbps). For the observation (Obs) activity, the data production rate from the sensor payload (63 kbps) is added in, and assumed to occur over the entire duration of the activity. Downlink (Dlnk) uses the nominal radio downlink data rate, 2.6 Mbps, multiplied by the 70% reduction factor. The crosslink (Xlnk) mode energy usage is based on a radio with 10 W input power, and a transmit duty cycle of 33% during an actual crosslink.

The minimum duration values were chosen to be representative of what CubeSat operations would require. The slew activity is given a fairly long duration to be conservative. The Xlnk duration was based on data from the constellation simulation; it was set long enough to allow multi-satellite crosslink events to happen more often. Note that with the 2400 km crosslink limit, the physical windows for these activities were generally much longer than 1.5 minutes. The Commlink (Clnk) activity duration was chosen to allow enough time for establishing a link.

Table 3.1: Activity Resource Usage Rates and Minimum Durations

Parameter		**Operational State**						
Type	**Unit**	**Obs**	**Xlnk**	**Clnk**	**Rech**	**Dlnk**	**Slew**	**Idle**
Energy (ES)	Watts	-14.1	-9.3	-9.3	17.0	-16.0	-7.8	-6.4
Data (DS)	kbps	73	10	10	10	-1820	10	10
Minimum Duration	minutes	5	1.5	1	1	1	3	0

3.2 The Resource-Aware SmallSat Planner (RASP)

The RASP algorithm was developed to autonomously plan and schedule activities onboard a resource-constrained small satellite. The discussion of the algorithm here largely follows its previous introductions by Kennedy et al [41, ?]. The algorithm has some similarities with the ASPEN/CASPER algorithms developed at NASA JPL [?] in that it evaluates the feasibility of performing activities based on onboard resource usage, but it *a*) uses a simpler model focused specifically on a resource-constrained satellite and *b*) it constructs an entire activity sequence in a single algorithm, as opposed to creating an initial high level sequence for later onboard refinement.

Activity planning constitutes the selection of a set of activities (an "activity sequence") from the operational state machine (Figure 3-2) that allows the satellite to execute as many observation and crosslink activities as possible while maintaining onboard resources within constraint limits. Scheduling is the assignment of a set of start and end times to every activity in the plan (an "activity timeline") such that an overall score function is maximized as well as the determination of acceptable trajectories for onboard resource states. Figure 3-3 illustrates two example resource trajectories that are kept within resource bounds.

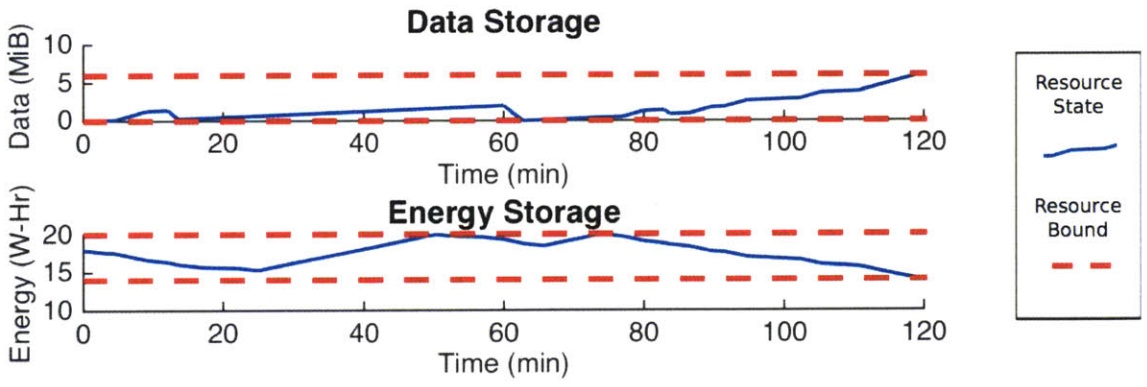


Figure 3-3: Example data storage and energy storage trajectories over an activity timeline

The RASP algorithm finds a suboptimal but acceptable activity timeline within a given planning horizon (t_h) given a set of initial observation and crosslink windows.

In order to limit the required computational time it does not attempt to determine the optimal timeline. The planner is used in a receding horizon fashion; that is, the satellite plans a set of activities for itself within t_h, executes those for a certain time, and replans from that new time using an updated state. This repeats for the duration of the scenario.

The following subsections detail the main components of RASP. The first subsection discusses the inputs RASP uses to create an initial activity sequence, the second discusses the scheduling of an optimal activity timeline from a specified activity sequence, the third, fourth, fifth, and sixth discuss in detail RASP's search mechanism for modifying this initial sequence to arrive at a feasible sequence, and the seventh discusses how RASP modifies the initial activity sequence if no feasible activity timeline was found during the search.

3.2.1 RASP Inputs

A simulation of the satellites' orbits is run to derive observation, crosslink, recharge, and downlink windows. Observation activity windows correspond to the times when the sub-satellite point (the intersection of the Earth's surface with the vector from the Earth's center to the satellite) is within a target region of latitude and longitude ranges. Multiple non-overlapping target regions can manifest themselves as sequential observation activities in the satellite's plan. Crosslink windows correspond to the times that the satellite is within 2400 km of another satellite. Downlink windows occur whenever the satellite is above a fixed elevation mask as viewed by the ground station. Recharge windows occur whenever the satellite is illuminated by the sun.

Given the time windows over a specified time horizon (t_h), RASP constructs an initial activity sequence by assuming a single observation or crosslink activity occurs during each of their respective windows. Observation activities are weighted based on their importance for coordinated performance across the constellation, as explained in Subsections 3.2.2 and 3.3.2. Figure 3-4 depicts a notional initial activity timeline over the course of an arbitrary orbit, i. Note that slew activities must occur between maneuvers in order to restore the desired attitude, and idle activities occupy the

non-used times.

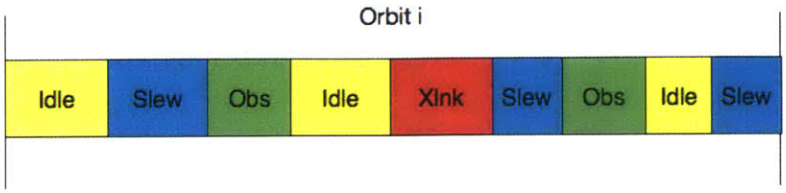


Figure 3-4: Notional initial activity sequence, with Obs, Xlnk, Slew, and Idle activities

It is important to note that in the current version of RASP, no special consideration is given to the importance of some crosslinks over others. The planner simply tries to schedule all crosslinks, at least initially. If an observation and crosslink overlap, preference is given to the observation. This lack of consideration for the relative importance of crosslinks in the overall constellation communications network is an important limitation of RASP in its current form, and is an item for future work. Also note that the satellites do not prioritize downlinks for information sharing; they only downlink for the purpose of offloading stored data. The inclusion of some mechanism for scoring downlinks for this purpose is another item for future work.

3.2.2 Activity Timeline Optimization

Given an activity sequence, the scheduler component of RASP attempts to find an optimal activity timeline. An activity timeline consists of an ordered list of timepoints $t^S_{a,i}$ and $t^E_{a,i}$ where $i \in [1, N]$, which represent the start and end times of each activity, respectively. N is the number of activities. The symbol a signifies a high-level activity, such that $a \in Act \triangleq Obs \cup Xlnk \cup Dlnk \cup Rech \cup Slew \cup Idle$, where each set in the overall union contains all the observation, crosslink, downlink, recharge, slew, and idle activities, respectively. This optimization is formulated as a Mixed Integer Linear Program (MILP):

$$\max\ [\sum_{Obs} w'_o(t_a^E - t_a^S) - w_d \sum_{Dlnk} \dot{d}_i(t_a^E - t_a^S) + w_e \sum_{i=1}^{N} \sum_{j=1}^{i} \dot{e}_j(t_a^E - t_a^S)] \tag{3.1}$$

subject to

$$t_{a,1}^S = 0\ , t_{a,i}^S \leq t_{a,i}^E\ , t_{a,N}^E = t_h; 1 \leq i \leq N \tag{3.2}$$

$$t_{a,i}^E = t_{a,j}^S; i \geq 1, j \leq N \mid j = i+1 \tag{3.3}$$

$$t_a^S \geq t_a^{S,W}\ , t_a^E \leq t_a^{E,W}; \forall a \in Obs \cup Dlnk \cup Rech \tag{3.4}$$

$$t_a^S = t_a^{S,W}\ , t_a^E = t_a^{E,W}; \forall a \in Xlnk \tag{3.5}$$

$$t_a^E - t_a^S \geq MinDur_a; \forall a \in Act \tag{3.6}$$

and

$$\begin{aligned}
RS_{init} + \dot{r}_1(t_{a,1}^E - t_{a,1}^S) &\leq UB_{RS} + M(1 - z_{RS,1}^{UB}) \\
RS_{init} + \sum_{i=1}^{2}[\dot{r}_i(t_{a,i}^E - t_{a,i}^S)] &\leq UB_{RS} + M(1 - z_{RS,2}^{UB}) \\
&\cdots \\
RS_{init} + \sum_{i=1}^{N}[\dot{r}_i(t_{a,i}^E - t_{a,i}^S)] &\leq UB_{RS} + M(1 - z_{RS,N}^{UB})
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
RS_{init} + \dot{r}_1(t_{a,1}^E - t_{a,1}^S) &\geq LB_{RS} - M(1 - z_{RS,1}^{LB}) \\
RS_{init} + \sum_{i=1}^{2}[\dot{r}_i(t_{a,i}^E - t_{a,i}^S)] &\geq LB_{RS} - M(1 - z_{RS,2}^{LB}) \\
&\cdots \\
RS_{init} + \sum_{i=1}^{N}[\dot{r}_i(t_{a,i}^E - t_{a,i}^S)] &\geq LB_{RS} - M(1 - z_{RS,N}^{LB})
\end{aligned} \tag{3.8}$$

The score function in Equation 3.1 attempts to maximize three items: the weighted sum of all observation durations in the activity timeline (summation 1), the total amount of data downlinked over the activity timeline (summation 2), and the average amount of ES margin over the course of the activity timeline (double summation). The value w'_o is a weighting given to each specific observation by LCCC, w_o, normalized by the total time summed up across all observation window lengths.

The $\dot{d}_i$ and $\dot{e}_i$ terms correspond to the DS usage rate and ES usage rate for activities i and j, respectively. ES margin here refers to the difference between the ES state at the end of an activity and the ES lower limit. The outer summation ($i = 1$ to N) accounts for the ES margin at the end of all activities in the activity sequence, and the inner summation propagates the ES state forward through the activity timeline by accounting for ES changes over all activities j up to activity i. The weighting terms w_d and w_e are calculated as:

$$w_d = u_{DS}/(UB_{DS} - LB_{DS}) \tag{3.9}$$

$$w_e = u_{ES}/(UB_{ES} - LB_{ES})/N \tag{3.10}$$

Equation 3.9 expresses that the total amount of data downlinked over an activity timeline is normalized by the range between DS bounds (where UB_{DS} and LB_{DS} represent the upper and lower bounds respectively) and multiplied by a unitless "urgency factor", u_{DS}, which effectively tunes the algorithm's preference for downlinking data. If this factor is set to 0, RASP will not care at all about downlinking data outside of its necessity to keep DS within bounds. Equation 3.10 is a similar expression, except that the additional normalization by the number of activities, N, means that the algorithm minimizes average ES margin.

Equations 3.2 enforce a planning window from 0 to t_h and ensures that the end of every activity follows its start. Equations 3.3 force activity j to follow activity i. Equations 3.4 force the *Obs*, *Dlnk*, and *Rech* activities to fall within their windows; $t_a^{S,W}$ and $t_a^{E,W}$ signify the start and end of the relevant time window. Equations

3.5 force $Xlnk$ activities to start and end exactly on their window bounds, which is helpful for making sure satellites commit to crosslink at the same time. Equation 3.6 enforces activity minimum durations. The N equations in 3.7 and 3.8 enforce resource constraint upper bounds (UB) and lower bounds (LB), respectively; the RS signifies that these equations hold for both resource types: ES and DS. We use the "Big M" method to select whether specific constraints will or will not be enforced [?]; hence, M is a large integer and $z_A^B \in \{0, 1\}$ is a variable that decides whether the constraint for the given activity number is enforced.

3.2.3 Activity Sequence Construction Through Greedy Search

RASP uses the selective enforcement of constraints in Equations 7 and 8 as a mechanism for determining where to add downlink and recharge activities to arrive at a final plan with all constraints enforced. At the highest level the algorithm performs a Depth-First Search (DFS) through a tree of modified activity sequences constructed from the initial activity sequence. Children activity sequences are created by adding a single resource management activity - an activity of type $Dlnk$ or $Rech$ - at a time to the parent activity sequence. $Slew$ and $Idle$ activities are added as necessary to maintain conformity to the semantics of the operational state machine. This process of search through incremental activity sequence modifications is shown in Figure 3-5.

Adding these activities allows the algorithm to progressively enforce more of the driving constraints ($DSUB$ and $ESLB$), pushing towards the goal state of having all constraints enforced. When a new activity is added, the algorithm attempts to solve the MILP with the appropriate resource constraint set enforced up to the location where the activity was added. The scores of the children activity sequences produced along the way inform the algorithm's choice of the next node to expand.

A heuristic function is used to push the algorithm to progressively enforce more constraints, while also trying to increase the score for the timeline. The function favors downlinks first because of their small, rare time windows, followed by recharges. When a timeline is found that satisfies all the constraints in the MILP, it is returned. For practical purposes, an unsuccessful search is limited to a timeout period (7 sec-

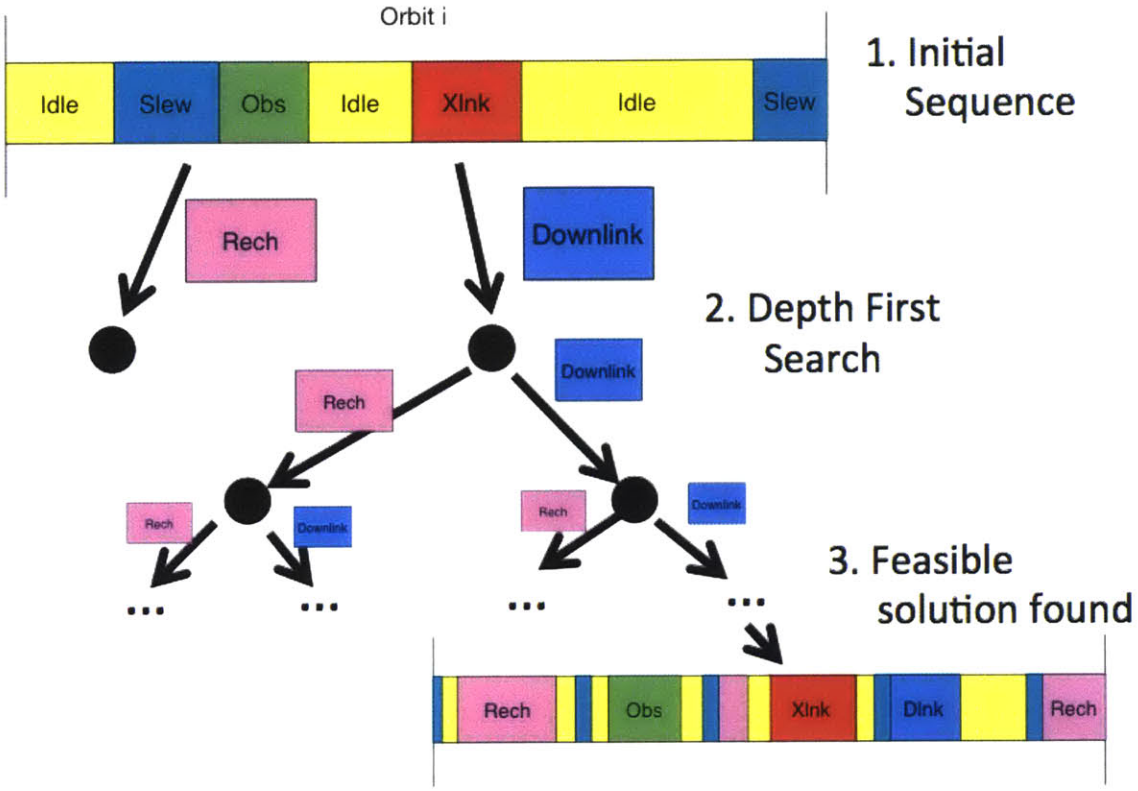


Figure 3-5: Illustration of Depth First Search process used to find a feasible activity timeline. The top activity sequence (1) cannot produce a consistent timeline, due to resource usage limit violations. The algorithm attempts to add recharge or downlink activities at various locations in sequence (2), progressively enforcing resource constraints it searches through the tree. Eventually a feasible timeline is found (3), consisting of the original sequence plus the recharges and downlinks added

onds), after which a reduction is made to the input activity sequence (the problem is simplified) and RASP is run again. The RASP algorithm as currently implemented is non-optimal and non-complete, but is strongly believed to be sound (if it returns what it believes to be a solution, that solution is in fact correct and reliable).

The following sections discuss in detail the algorithms used in RASP.

3.2.4 RASP High Level Algorithm: Depth-First Search

The high-level search algorithm is presented in Algorithm 1. Lines 2 and 3 set the non-driving resource constraints and driving constraints to be on and off, respectively. Initially fixing the non-driving constraints on reduces the size of the search space and still provides acceptable performance. Line 4 initializes the root node of the search tree with the initial activity sequence A_{init} as well as all the enforced constraints for this activity sequence from the previous lines, z^*. A node can store many values describing its activity sequence; more values will be introduced in the lower level algorithms. Lines 7 through 11 are a standard search formulation; the best child, $Next$, is popped from the search queue, Q, tested to see if it's a Goal state (all constraints enforced), and if not, then its children are added to the search queue. Detailed discussion of algorithms POPBEST() and GETCHILDREN() follows.

Algorithm 1 The high-level RASP search algorithm

```
1:  procedure RASP(A_init)
2:      z^UB_ES,i , z^LB_DS,i ← 1 ,   ∀ 1 ≤ i ≤ N      ▷ assume enforceable
3:      z^LB_ES,i , z^UB_DS,i ← 0 ,   ∀ 1 ≤ i ≤ N      ▷ set non-enforced
4:      Q ← MAKENODE(A_init, z*)                        ▷ initialize search queue
5:      SolutionFound ← False
6:      while ¬isempty(Q) ∧ ¬SolutionFound do
7:          Next , Q ← POPBEST(Q)                       ▷ See Algorithm 3
8:          SolutionFound ← GOALTEST(Next)
9:          if ¬SolutionFound then
10:             Q ← Q ∪ GETCHILDREN(Next)               ▷ See Algorithm 2
11:         end if
12:     end while
13: end procedure
```

3.2.5 GETCHILDREN: Find Children Through One-Step Modification

The heart of the RASP algorithm lies in the GETCHILDREN() procedure in Algorithm 2. GETCHILDREN produces all possible one-step modifications to the parent activity sequence and returns the modified activity sequences as nodes for addition to the search queue. A one-step modification consists of replacing an idle in the parent activity sequence with a resource management activity (of type *Dlnk* or *Rech*), as well as any transition activities necessary to make the resulting modified sequence valid.

The GETCHILDREN procedure first grabs and ranks by length all the *Idle* activities from the parent activity sequence, in Line 4. The longer an *Idle* is, the better candidate it is for replacement. The procedure then loops through all driving constraint types (Line 5) and all parent *Idle* activities (Line 6; in rank order) and replaces the *Idle* with the corresponding management activity (Lines 7 to 9). In lines 10 and 11, it attempts to find a valid activity timeline from the modified activity sequence by solving a relaxed version of the MILP in Equations 1 to 10. It does this while enforcing *a*) all its parents' constraints from constraint sets other than *RS*, and *b*) all constraints in set *RS* up to the activity that was just replaced. If a valid activity timeline cannot be found, then this modification is likely not useful, and no child is created for this particular management activity - *Idle* location combination. If a valid timeline is found, then the algorithm attempts to produce a valid timeline with all constraints enforced from set *RS* (Lines 13 and 14). Attempting to enforce all constraints in *RS* drives the algorithm towards the goal state more effectively. If no all-enforced timeline can be created, a new child node is added with enforcement up to the replacement location and with a record of the evaluated score function, *Score*, for this new timeline (Lines 26 and 27).

Lines 16 to 24 serve a special purpose in driving the algorithm towards a high quality solution, by determining the usefulness of adding another resource management activity corresponding to set *RS* even if all the constraints in *RS* have already

Algorithm 2 Find all consistent single-add activity sequences

```
1: procedure GETCHILDREN(Parent)
2:     Children ← ∅
3:     A ← GETACTSEQ(PARENT)
4:     indices ← RANKIDLES(A)
5:     for all RS ∈ {ES LB, DS UB} do
6:         for all k ∈ indices do
7:             switch RS do
8:                 case ES LB : A_temp ← REPLACE(A, k, rech)
9:                 case DS UB : A_temp ← REPLACE(A, k, dlnk)
10:            z_i^RS ← 1 ∀ 1 ≤ i ≤ k
11:            Solvable, Score ← SOLVERELAXEDLP(A_temp, Parent.z*, z_i^RS)
12:            if Solvable then
13:                z_i^RS ← 1 ∀ 1 ≤ i ≤ N
14:                Solvable, Score ← SOLVERELAXEDLP(A_temp, Parent.z*, z_i^RS)
15:                if Solvable then
16:                    ΔScore ← Score − Parent.Score
17:                    NumTimesUseful ← TIMESUSEFUL(Parent, RS)
18:                    if NumTimesUseful = 0 then
19:                        NumTimesUseful = 1
20:                    else if ΔScore > 0 then
21:                        NumTimesUseful = NumTimesUseful + 1
22:                    end if
23:                    NewChild ← MAKENODE(A_temp, Parent.z*, z_i^RS, Score, NumTimesUseful)
24:                    Children ← Children ∪ NewChild
25:                else
26:                    NewChild ← MAKENODE(A_temp, Parent.z*, z_i^RS, Score)
27:                    Children ← Children ∪ NewChild
28:                end if
29:            end if
30:        end for
31:    end for
32: end procedure
```

been enforced. The intuition here is that a certain placement of management activities may barely satisfy constraints, without any margin between resource usage and resource upper/lower bounds, so adding more activities may introduce more margin and give the flexibility to achieve a higher quality solution. Each node stores the value $NumTimesUseful$ for each driving constraint set RS, which tells the algorithm how many times set RS has been "usefully" fully enforced along this particular branch of the search tree. These three values are initially set to 0. The first time RS is fully enforced, its value is set to 1 (Lines 18 and 19). For every subsequent time RS is enforced, this value is incremented if the corresponding activity replacement caused an increase in score function from the parent activity sequence (Lines 20 to 21). This value is used to guide the selection of the next child to expand in Algorithm 3. The TIMESUSEFUL() procedure on Line 17 merely returns $NumTimesUseful$ for RS from the given node. After the update check, a new child node is added in Lines 23 and 24 with the updated value.

3.2.6 POPBEST: Select Best Child to Expand

The POPBEST() procedure in Algorithm 3 selects the best child to expand next in the search tree. It uses a series of filters, FILTERQUEUE(), to eliminate candidates from the search queue and hone in on this best child. Each successive filter only grabs those children from Q with the maximum value of the second argument. POPBEST was designed to cause the RASP algorithm to first enforce all constraints in set $DSUB$, then those in set $ESLB$, because of the increased ease of meeting time windows for *Rech* activities.

Algorithm 3 Remove best child from Q, return child and modified Q

1: **procedure** POPBEST(Q)
2: $Q_{temp} \leftarrow$ FILTERQUEUE(max, TIMESUSEFUL($DS\ UB$))
3: $Q_{temp} \leftarrow$ FILTERQUEUE(max, TIMESUSEFUL($ES\ LB$))
4: $Q_{temp} \leftarrow$ FILTERQUEUE($max, z^{UB}_{DS,i}$)
5: $Q_{temp} \leftarrow$ FILTERQUEUE($max, z^{LB}_{ES,i}$)
6: $Q_{temp} \leftarrow$ FILTERQUEUE($max, Score$)
7: **end procedure**

The first set of filters in Lines 2 to 3 causes the algorithm to prefer adding as many of each resource management activity as appears useful, before attempting to achieve the goal state. The filters in Lines 4 to 5 push the algorithms towards the actual goal state of enforcing all constraints. Line 6 selects from the filtered children the child with the best score.

3.2.7 Modification of Observation and Crosslink Activities Selection

If it is found that the input activity sequence, with its set of observation and crosslink windows, cannot be solved through the search process, either an observation or crosslink is subtracted from the input sequence to make the problem more feasible. Multiple windows can be progressively removed if the search process fails more than once.

RASP first attempts to remove observations with very low weightings (as determined by LCCC), because observation activities are particularly resource-demanding. If all low-weight observations below a specified cutoff are removed and the search process still fails, crosslinks are progressively removed. Crosslinks are ranked by the number of satellites involved. Crosslinks with fewer satellites are first removed, because of their lower potential for information sharing. After all low-weight observations and crosslinks are removed, RASP proceeds to remove the remaining observations, preferring the lowest weight observation first. Note that the first activity in the input sequence will never be removed because it is the satellite's current activity.

If all possible observations and crosslinks are removed and RASP still cannot find a consistent activity timeline, resource bounds are progressively relaxed until a solution is found. In the current algorithm, there is no method to reconcile relaxing a bound past a physical limit, so resource bounds should in general be set tighter than physical bounds.

3.3 Limited Communication Constellation Coordinator (LCCC)

The RASP algorithm is responsible for constructing a high scoring activity sequence, given a set of observations (with respective weightings) and crosslinks. The LCCC algorithm calculates observation weightings from planning information obtained via a "weak" form of distributed consensus. The weightings are selected in order to direct RASP to minimize the average revisit times for all regions being observed, over all three sensor types, by allocating all satellites' observation activities in the best way possible.

3.3.1 Weak Consensus over Shared Planning Information

The weak distributed consensus mechanism works in the way that each satellite maintains its own database of the most up-to-date observation planning information obtained from all satellites, and updates this database whenever new information becomes available through crosslinks, downlinks, or commlinks. The mechanism is described as "weak" because the satellites are not actually required to come to a consensus on the information stored in each of their separate databases. Such an approach would simply be infeasible from a communications perspective. However, in the hypothetical situation where all satellites have continuous, instantaneous access to all other satellites' updated plans, these databases would all contain exactly the same information, and a true consensus would be achieved. The satellites need not talk to the actual satellite that originated the observation; they can hear it second-hand, multiple times, and still unambiguously identify which information is the most up-to-date. An illustration of the weak consensus process is shown in Figure 3-6.

Every satellite, or "agent", maintains a list, $\mathbf{o}_i$, of the observations both planned and already executed for every satellite in the constellation, including itself. All observation activities across the constellation are uniquely identified by the originating satellite, the region of Earth being targeted, the selected sensor type, and the time

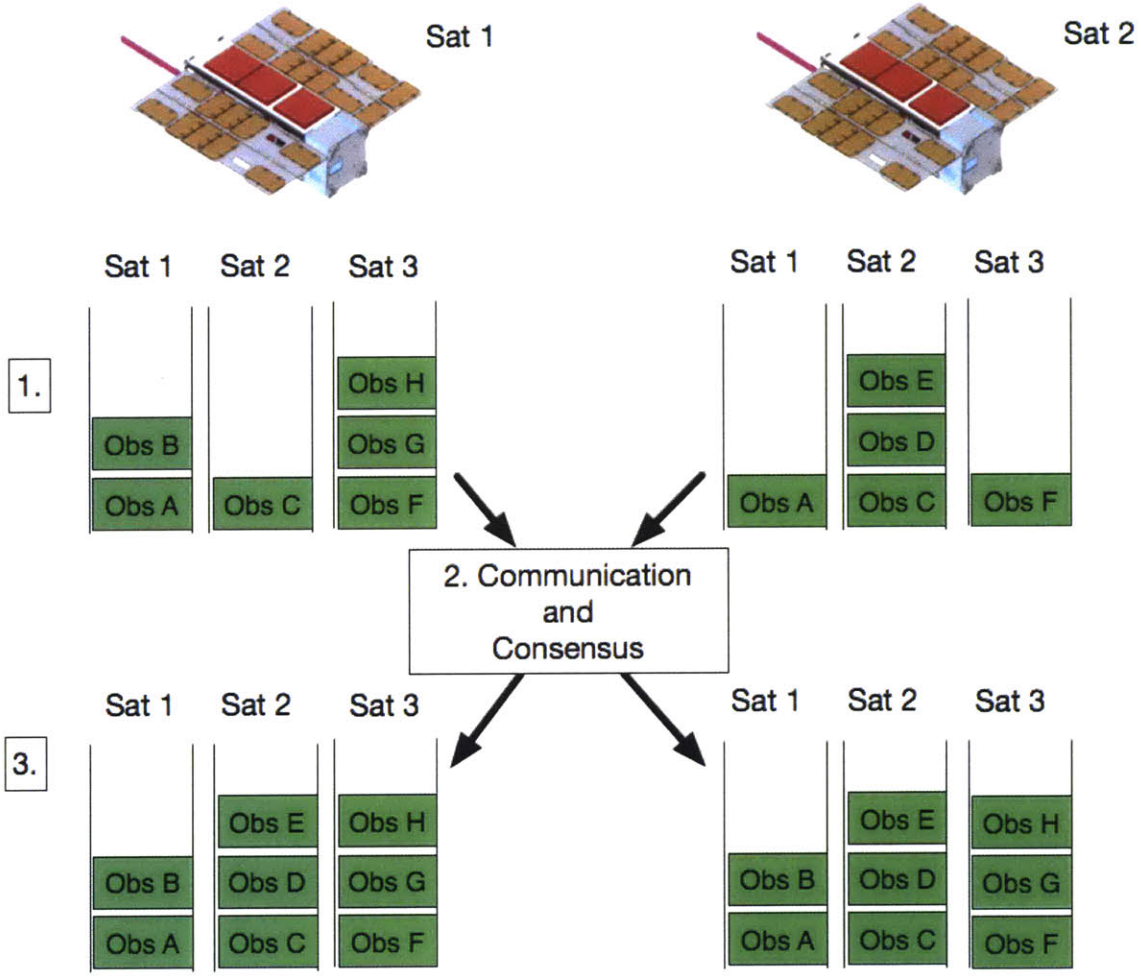


Figure 3-6: Illustration of the weak consensus mechanism employed in LCCC. In (1), satellites 1 and 2 start out with different knowledge of planned observations for themselves and an additional satellite, 3. In (2) satellites 1 and 2 trade planning information through a crosslink, downlink, or commlink. In (3), satellites 1 and 2 have come to a consensus on the planned observations for satellites 1, 2 and 3. MiRaTA CubeSat image from Kennedy and Cahoy [40].

window derived from the satellite's orbit. Whenever a satellite settles on an activity sequence (up to its current time plus time horizon t_h), it uses its list of planned observations to update $\mathbf{o}_i$. If it has already scheduled an observation during a previous planning period, that observation is updated with the new start and end times. If the observation is canceled (RASP decides not to perform a previously planned observation), the start time is set equal to the end time. If the observation is totally new, it is simply added to $\mathbf{o}_i$. All updated or added observations are tagged with an appropriate update time. During crosslink activities, satellites trade their lists of planned observations between each other and update any observations in their own list that are out-of-date, i.e. the update tag on their version of the observation is earlier than the update tag provided by another satellite.

This process is reflected in Algorithm 4. Agent i receives list $\mathbf{o}_j$ from all other agents j, and uses these to update its own list. The FINDMATCHING() procedure on Line 4 returns a matching observation if it exists, or returns the empty set if satellite i has not heard about this observation yet. Line 8 tests if the other agent's matching observation is more up-to-date, and updates the agent's observation (Line 9) if it is. Line 12 stores the other agent's observation if it is completely unknown.

Algorithm 4 Update satellite i's knowledge of planned observations for all satellites in crosslink

```
1: RECEIVE o_j from j,   ∀j ∈ X | j ≠ i
2: procedure UPDATEOBSERVATIONS(j, o_j)
3:     for theirObs ∈ o_j do
4:         myObs ← FINDMATCHING(theirObs, o_i)
5:         theirLastUpdateTime ← LASTUPDATED(theirObs)
6:         if myObs ≠ ∅ then
7:             myLastUpdateTime ← LASTUPDATED(myObs)
8:             if myLastUpdateTime < theirLastUpdateTime then
9:                 myObs ← theirObs
10:            end if
11:        else
12:            o_i ← o_i ∪ theirObs
13:        end if
14:    end for
15: end procedure
```

One important assumption in this model is that all satellites have access to the same global clock; this is achievable with GPS time for LEO constellations. An important requirement for this method to work well is a high degree of dynamic network connectivity. That is, as the satellites orbit and they perform opportunistic crosslinks and downlinks, we desire a given agent to hear from all other agents that could potentially affect its weighting for a particular observation.

3.3.2 Selecting Observation Weightings from Shared Planning Information

A given observation's weighting is based on what the agent knows about textitall agents' planned observations. Put simply, an observation is weighted more heavily the farther it is away from the closest preceding observation of the same region, with the same sensor. Algorithm 5 determines, for every possible observation from the satellite's current time to time horizon t_h, the closest preceding observation for every sensor type. It assigns a weight for performing the observation with each sensor type, and selects the highest weighting over all sensors. The highest weighting is RASP's w_o input for that observation. This weighting algorithm is called by RASP every time it tries to plan for a new set of observation and crosslink windows (see Subsection 3.2.7). After RASP plans and selects a set of observations to actually perform, those observations timings as well as the selected sensor type are updated in the planning observations list $\mathbf{o}_i$ for the satellite.

Algorithm 5 works in the following way. A list of observation windows from the current planning window (current time to t_h), $\mathbf{o}_i^W$, is provided along with $\mathbf{o}_i$ and the start and end times for the weighting window $\mathbf{o}_i$ and $t^S_{weighting}$. The observation windows are provided instead of actual observation activity instances, because these are the required input for RASP in its current form. Note that $\mathbf{o}_i^W$ is assumed to be sorted in ascending order, based on the start of the observation windows. No observation windows can overlap, due the use of non-overlapping regions of Earth's surface for observation opportunities.

A list of weights for all observations is instantiated on Line 2. Any observation activities previously scheduled from the current planning window are removed from $\mathbf{o}_i$ and the results is stored in $\mathbf{o}'_i$ (Line 3). This is required because the algorithm considers incrementally adds observation window back into $\mathbf{o}'_i$ as it assigns them weightings, in order to consider the case where the planning horizon is far enough away for there to be multiple observation windows for a given region in $\mathbf{o}_i^W$.

Algorithm 5 Calculate the weighting of all observation windows within current planning horizon

1: **procedure** CALCULATEWEIGHTINGS($\mathbf{o}_i^W, \mathbf{o}_i, t^S_{weighting}, t^E_{weighting}$)
2: $\quad\mathbf{w}_o \leftarrow \emptyset$
3: $\quad\mathbf{o}'_i \leftarrow \mathbf{o}_i \setminus \mathbf{o}_i^W$ ▷ remove from set
4: $\quad$**for** $o^W \in \mathbf{o}_i^W$ **do**
5: $\quad\quad weights_W \leftarrow \emptyset$
6: $\quad\quad$**for** $sensor \in \{A, B, C\}$ **do**
7: $\quad\quad\quad t_{prec} \leftarrow$ FINDNEARESTPRECEDING($o^W, \mathbf{o}'_i, t^S_{weighting}, sensor$)
8: $\quad\quad\quad \Delta t \leftarrow$ START(o^W) $- t_{prec}$
9: $\quad\quad\quad weight_W \leftarrow \Delta t / (t^E_{weighting} - t^S_{weighting})$
10: $\quad\quad\quad weights_W \leftarrow weights_W \cup weight_W$
11: $\quad\quad$**end for**
12: $\quad\quad\mathbf{w}_o \leftarrow \mathbf{w}_o \cup$ MAX($weights_W$)
13: $\quad\quad\mathbf{o}'_i \leftarrow \mathbf{o}'_i \cup o^W$
14: $\quad$**end for**
15: $\quad$**return** $\mathbf{w}_o$
16: **end procedure**

Each observation window, o^W, is looped through in temporal order (Line 4). A list of sensor-specific weights, $weights_W$, for the window is instantiated on line 5. All sensors are looped through and added to this list in Lines 6 to 11. The FINDNEARESTPRECEDING() procedure finds the nearest preceding observation activity or window matching o^W's region from $\mathbf{o}'_i$, for the specified *sensor*. The procedure returns the start time of the preceding activity or window. If no preceding observation is found after time $t^S_{weighting}$, then that time is returned. A time difference is calculated from the preceding time and the start of o^W (Line 8) and normalized by the length of the weighting window (Line 9) to give a weight for the current sensor. This weight is added to $weights_W$ (Line 10).

After all sensor-specific weights are determined, the sensor corresponding to the

maximum weight for the observation is selected on Line 12. The observation window, along with its selected weighting, is added to $\mathbf{o}'_i$ for consideration in weighting later windows. Finally, a list of final weights for every observation window, $\mathbf{w}_o$, is returned.

An important feature of this weighting algorithm is that it has an explicit preference for *earlier* observations. This means that if two satellites have two observations with start times that are separated by a small temporal distance (say on the order of seconds), then given that both satellites know about the two observations, the earlier observation will be weighted much higher than the later one. This degree of "objectivity" encourages satellites to commit early to observations and stick with their decisions across multiple RASP planning horizons, which is important for ensuring that shared planning information is timely and relevant.

Chapter 4

Constellation Simulation Results

The Python- and MATLAB-implemented simulation environment was used to analyze the performance of the RASP and LCCC algorithms for both the Stitched Walker and Ad Hoc constellations from Table 2.1, in the five communications contexts from Table 2.4: No Sharing, Downlink, Crosslink, Crosslink + Downlink, and Commlink + Downlink. Each combination of constellation and communications context is referred to as a "case". There were 10 cases in total.

Five major analyses were performed on the simulation cases: *1*) the effect of constellation choice and communications context on revisit times achieve, *2*) the amount of communications links executed over all the cases, *3*) how effectively planning information was shared across the constellations, *4*) the average resource usage margins maintained by the satellites, and *5*) the average time taken for planning by RASP.

The first section of this chapter discusses the context for the simulation cases, then the subsequent sections go into detail on each analysis.

4.1 Simulation Context

A 24 hour simulation was run in each simulation case. The same bounds were set on onboard resource usage for every satellite, per Table 2.2: DS ranging from 0 to 100 MB (1000^2 Bytes) and ES from 14 to 20 Wh. RASP planning was performed over a 90 minute planning window ($t_h = 90$), and satellites replanned every 20 minutes, or

after every crosslink, downlink, or commlink in which they obtained updated planning information. RASP utilized the resource usage rates in Table 3.1 for planning purposes. Satellite states were propagated forward using the same rates, with a small amount of noise added on top of the rates to simulate model imperfections. The noise was normally distributed about the nominal usage rate, with a standard deviation of 1% of the nominal rate; the noisy rates were allowed to saturate at 95% and 105% of the nominal rate.

Observations regions were chosen arbitrarily to be well-spread-out over the Earth's surface. They were identical across all simulation cases. Seventeen were chosen in total, and are shown in Figure 4-1.

Communications link usage adhered to the parameters in Table 2.3. As mentioned in 3.2.1, the satellites attempted to perform all crosslinks available to them from orbital geometry; no intelligent selection was made of the best crosslinks to use. In the Clnk + Dlnk context, the commlink windows were spaced every 20 minutes, starting from 10 minutes into the 24 hour simulation. The commlink windows occurred at the same time across all satellites, for ease of modeling. The locations of the ground stations are also shown in Figure 4-1.

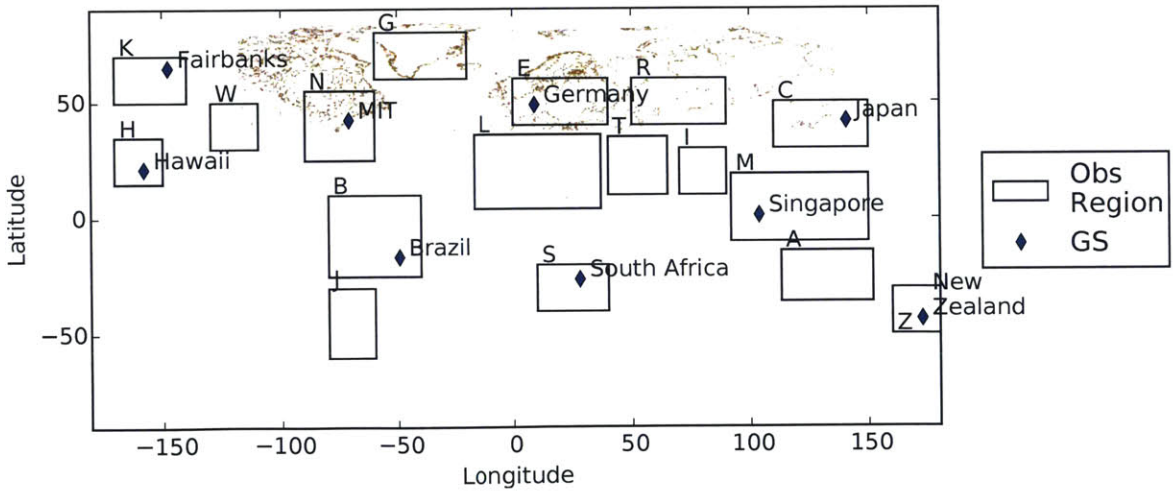


Figure 4-1: Regions (17 total) and Ground Stations (9 total) used in constellation simulations. Observation regions are used to define observation windows, based on when the subsatellite point falls within them.

Figure 4-2 shows an example activity timeline output by a RASP planning session.

The timeline illustrates the performance of observation and crosslink activities as well as all the use of downlinks and recharges to stay within resource bounds. The dashed color lines represent activity windows (when an activity can be performed) and the solid black lines represent the activity timeline actually chosen. Note that there are frequent, zero-length dips into *Idle* activities between other activities. These idles are used in the algorithm as padding around the other activities, and would not be executed in reality.

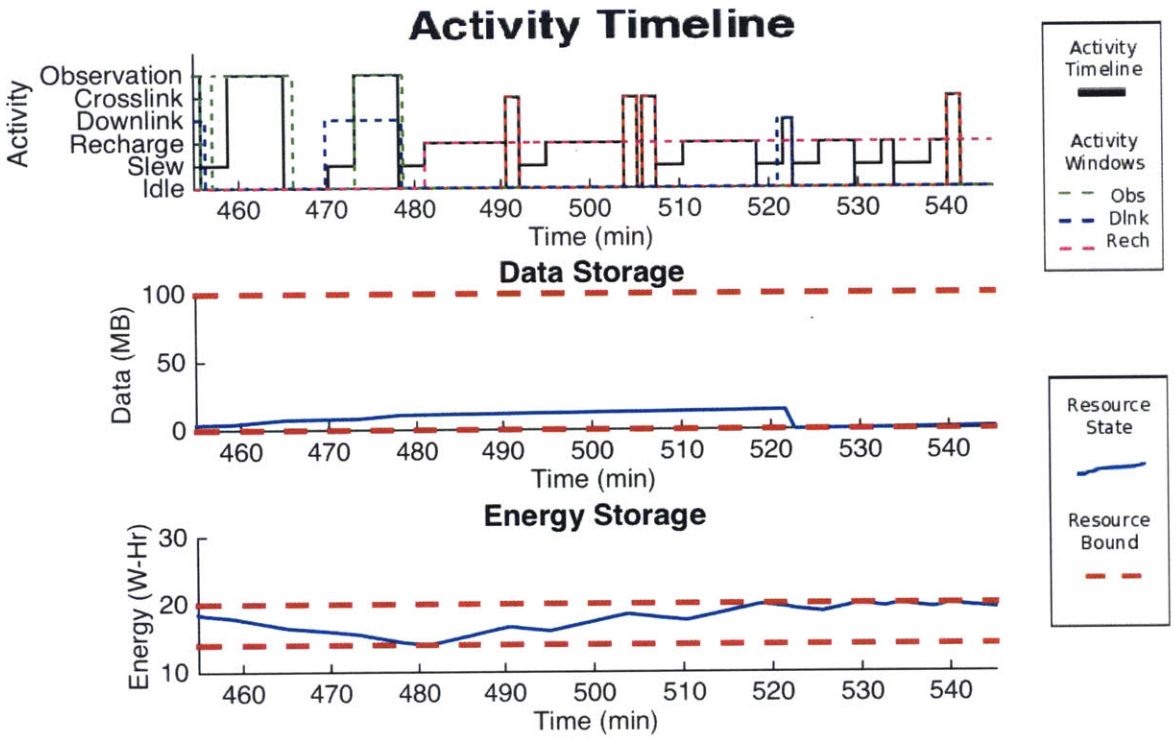


Figure 4-2: Example execution of the RASP algorithm, for satellite 8. The upper plot shows the RASP-derived activity timeline with the heights of the black activity line matching to the activity labels on the left or right y-axis. The lower three plots show corresponding resource states during this timeline. The resource bounds are [0,100] MB for DS and [14,20] Wh for ES

4.2 Revisit Times Performance

The first analysis was performed on the revisit times achieved in each simulation case. As context, Figures 4-3 and 4-5 show the executed observations over all sensor types, organized by satellite number, for the Stitched Walker Star constellation in two different communications contexts. Note that the observation windows are exactly the same in both figures, because they stem from constellation geometry. We see that with no information sharing (Figure 4-3), the sensor types are poorly spread out over the 24 hour period. Many observations are executed with sensor A up to about 10 hours in, then sensor B and C begin to mix in. Much better mixing is seen from the beginning when planning information is shared widely (Figure 4-5), which tends to lower average revisit times across all sensors. We also see that across the different contexts, roughly the same number of observations are executed for any given satellite across all sensor types.

When no planning information is shared, every single satellite tries to minimize revisit times for each sensor over all regions by itself. The satellite starts with sensor A in the first 10 hours simply because it is the first sensor chosen by the observation weightings algorithm in LCCC if no observations have yet been performed on a given region. At the same time, there are other agents with overlapping observations of some of the same regions, also tying to minimize revisit times over sensor A. As a whole, the constellation over-observes with sensor A during this time. When planning information is shared, the satellites are aware of overlapping observations, and the later-in-time observations are swapped to a different sensor by the owner satellite.

Table 4.1 supports the trends seen in Figures 4-3 and 4-5. With no information sharing the sensor A average revisit time is 168 minutes, whereas the sensor C average revisit time is a very large 682 minutes (almost half the 24 hour period) - a disparity of 514 minutes. Again, this is due to sensor C observations starting to mix in only around 10 hours into the simulation. Things look much better for the "Commlink + Downlink" context: average revisit time are roughly the same across all sensors, and the disparity has reduced to 10 minutes.

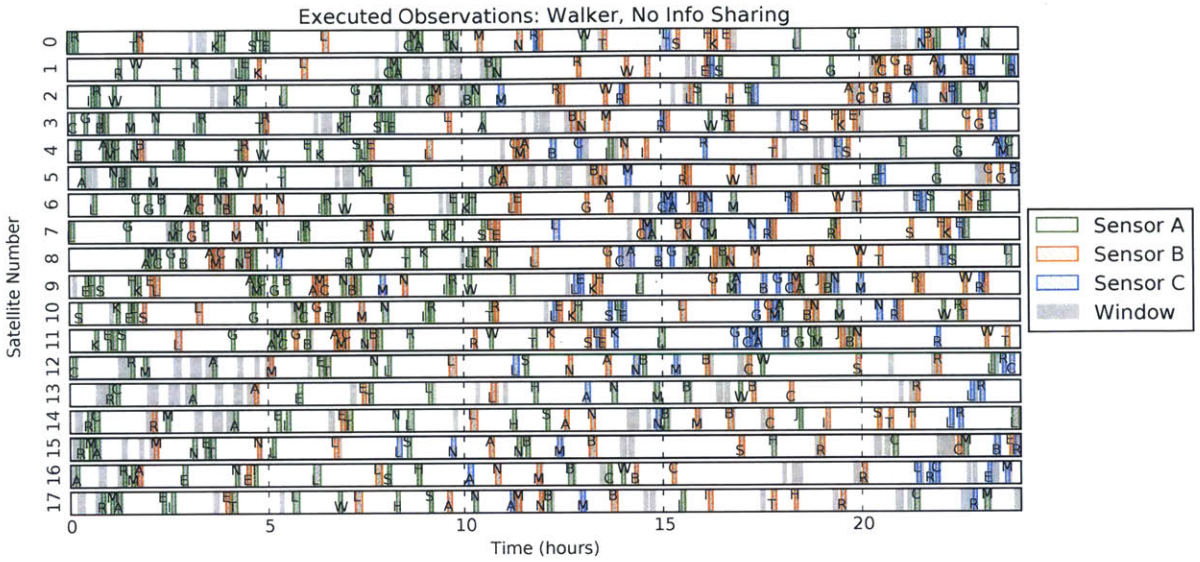


Figure 4-3: Executed observations and observation windows for Walker constellation in the "No Info Sharing" communications context

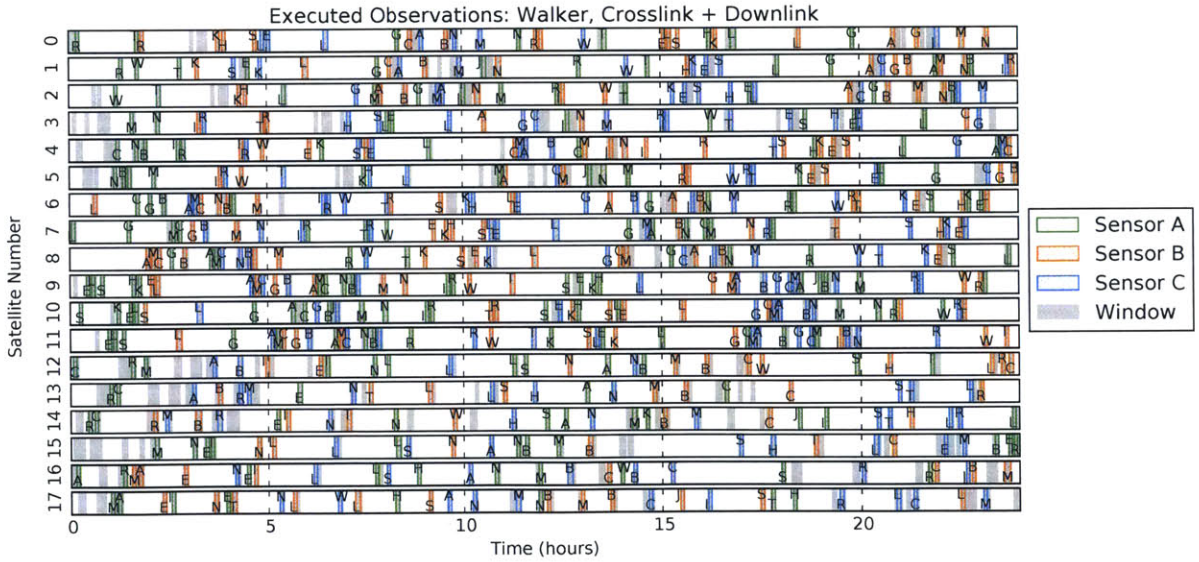


Figure 4-4: Executed observations and observation windows for Walker constellation in the "Crosslink + Downlink" communications context

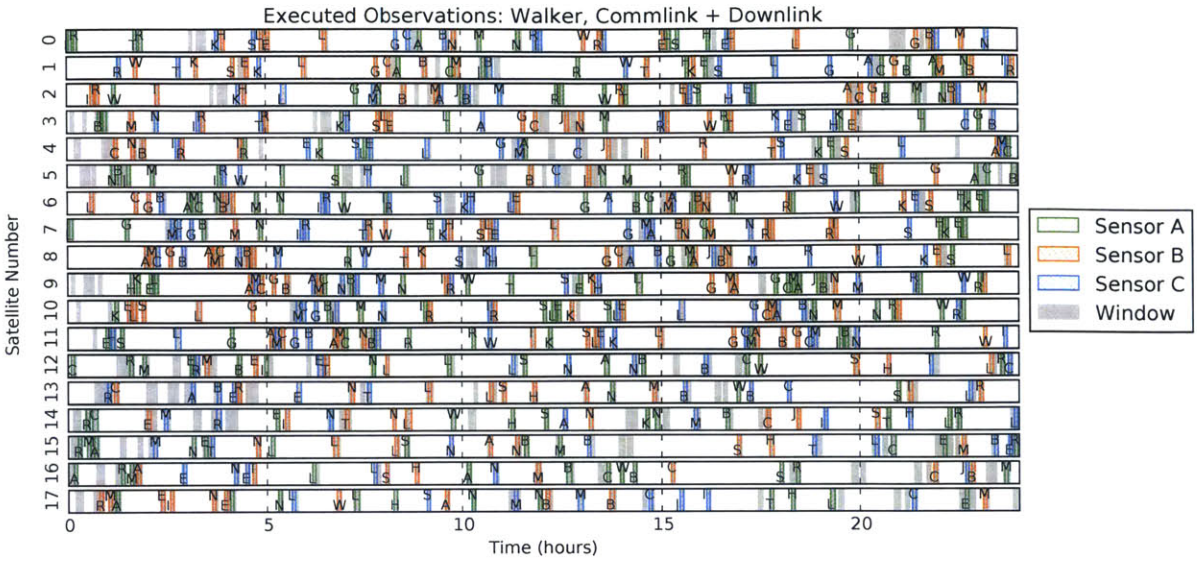


Figure 4-5: Executed observations and observation windows for Walker constellation in the "Commlink + Downlink" communications context

The sensor A average revisit time does however increase with more information sharing; the better performance for sensor C had to come from somewhere. The "Best Possible" row indicates the revisit time that would be achieved if all windows were dedicated to one sensor type. That is, if all windows were executed, and they were executed with only sensor A, an average revisit time of 126 minutes would be achieved. This is the physical limit of performance for the constellation. It is because of this limit that sensor A performs worse when sensor C performs better. As long as the percentage of executed observation windows stays roughly constant, the actual average revisit times are not going to approach any closer to the limit. Table 4.2 shows that this percentage does in fact stay roughly constant across the communications contexts.

The crosslink and downlink context performs on par with the commlink and downlink context in terms of performance, with a lower disparity of 8 minutes, but slightly higher times across the sensors. The other communications contexts perform somewhere in the middle. We see that the context with only crosslink performs slightly better (disparity of 92 minutes) than with only downlink.

Table 4.1: Average Revisit Times, Averaged Over All Regions, for Walker (minutes)

Sensor	Communications Context				
	No Sharing	**Dlnk**	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
A	168	165	176	201	198
B	206	239	224	207	203
C	682	354	268	209	208
Best Possible	126	126	126	126	126

Table 4.2: Number of Executed Observations Per Satellite, Averaged Over All Satellites, for Walker (average number of possible observation windows = 42.2)

Communications Context				
No Sharing	**Dlnk**	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
33.0	33.3	32.7	33.3	33.6

Tables 4.3 and 4.4 summarize the revisit results for the Ad Hoc constellation. The general trends are the same as for the Walker constellation, with an even higher disparity (617 minutes) in the no sharing context and about the same in the commlink and downlink context (11 minutes). The best possible average revisit time is larger, at 138 minutes versus 126 minutes for Walker. This time, the downlink context peforms slightly better than the crosslink context, which is probably due to the lower average number of crosslinks executed for this context in the Ad Hoc constellation versus the Walker constellation (20.8 versus 30.3, from Tables 4.5 and 4.6).

We see that roughly the same number of observation windows were executed on average across all contexts and both constellations (Tables 4.2 and 4.4). These are two quite different constellation geometries, with different average numbers of observation windows per satellite (42.2 versus 38.1). These results suggest that 33 observations over 24 hours is roughly the saturation point for this version of the RASP planner and the given satellite parameters.

Table 4.3: Average Revisit Times, Averaged Over All Regions, for Ad Hoc (minutes)

Sensor	Communications Context				
	No Sharing	Dlnk	Xlnk	Xlnk + Dlnk	Clnk + Dlnk
A	175	209	199	207	220
B	245	238	230	235	225
C	792	256	273	246	231
Best Possible	138	138	138	138	138

Table 4.4: Number of Executed Observations Per Satellite, Averaged Over All Satellites, for Ad Hoc (average number of possible observation windows = 38.1)

Communications Context				
No Sharing	Dlnk	Xlnk	Xlnk + Dlnk	Clnk + Dlnk
33.1	34.0	33.0	33.4	33.1

4.3 Communication Links Performance

The second analysis conducted was on the numbers of information-sharing communications links that were performed during the simulations. This analysis provides context for understanding the effectiveness of information sharing across the constellation.

Figures 4-6 and 4-7 show the communications links executed in the Walker constellation in the cases with the most information sharing. The text labels over the executed crosslinks in Figure 4-6 indicates, from the perspective of the satellite called out on the left side of the plot, which satellites were expected to be involved in the crosslink.

One important takeaway from Figures 4-6 is that there are often times when a satellite tries to perform a crosslink, but the other satellite (or satellites) sharing the crosslink do not perform it. This speaks to an important limitation of the RASP and LCCC algorithms currently: there is no mechanism for explicitly agreeing on crosslinks across satellites. All satellites simply try to perform all crosslinks available to them, rather than weight certain crosslinks more heavily when they know the

other satellite is likely to perform them. Such a mechanism could be very helpful in improving information sharing performance, and is an important item of future work.

We also see that many of the crosslink, commlink, and downlink windows are not actually executed, and those that do happen to be executed follow some intriguing patterns. Figure 4-6 features clear bands of executed crosslinks and downlinks progressing from bottom left to top right. Figure 4-7 hints at the same behavior. This is likely simply related to orbital geometry and resource availability: the bands of executed links fall in the times that the satellites are out of eclipse, or when they're not performing many observations and are more free to perform communications links. This performance relates again to the need to add a more sophisticated mechanism for ranking crosslinks and commlinks, so that they can be performed when actually needed as opposed to when it's convenient from a resource perspective

Appendix C shows the communication link plots for the Walker downlink only and crosslink only contexts.

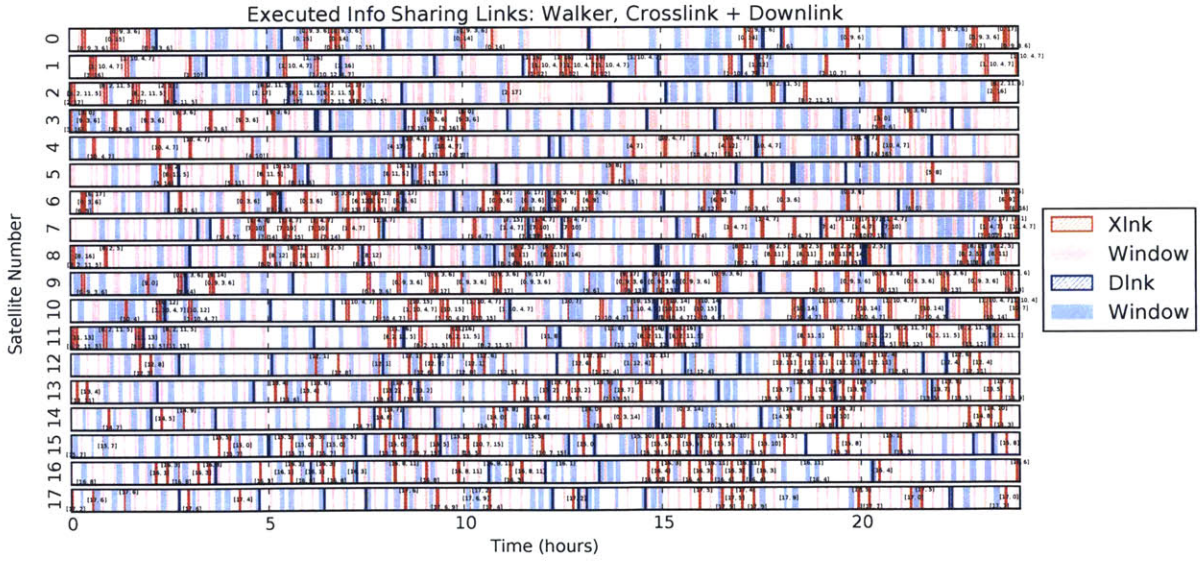


Figure 4-6: Executed information sharing communications links in crosslink + downlink context for Walker constellation. Text labels over executed crosslinks indicates which satellites were expected to participate in a given crosslink

Figure 4-8 shows the executed crosslink and downlinks for the Ad Hoc constel-

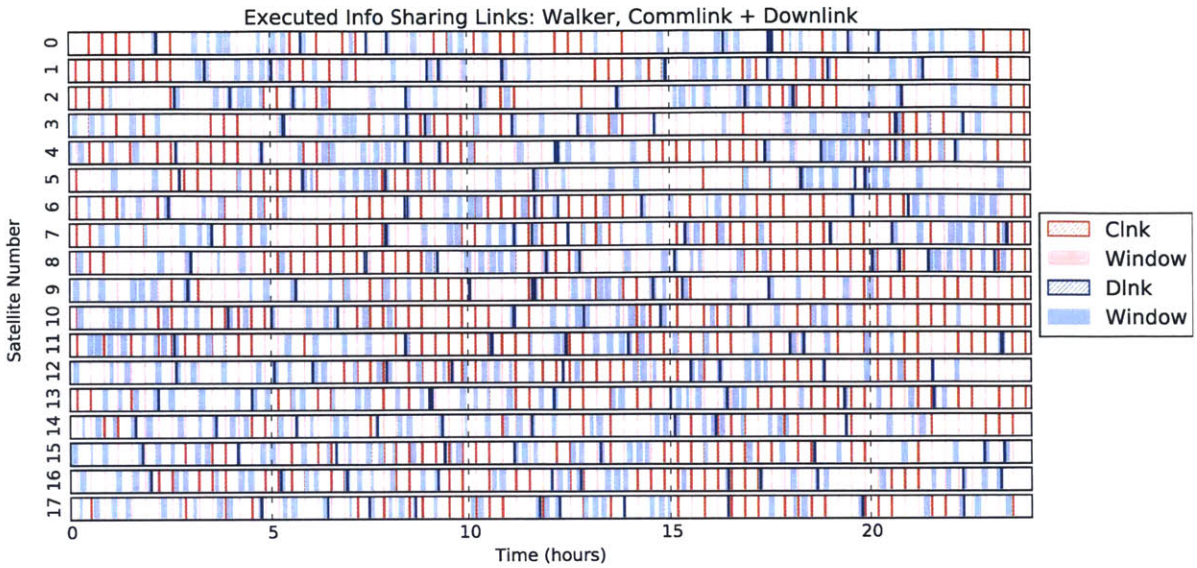


Figure 4-7: Executed information sharing communications links in commlink + downlink context for Walker constellation. Commlinks overlap across all satellites

lation in the crosslink + downlink context. We see that there is a large reduction in crosslink availability: many fewer crosslink windows are available. This shows a fundamental limitation in the ability of the Ad Hoc constellation to share planning information; its "dynamic network connectivity" is much more limited. This makes sense, considering the "stitched" part of the Stitched Walker Star constellation: two orbits were specifically designed to add connectivity across the constellation.

Tables 4.5 and 4.6 summarize the trends in Figures 4-6, 4-7, and 4-8 quantitatively. We see that there are many more crosslink windows for the Walker constellation, at 89.2 versus 47.7. For this reason, fewer crosslinks ended up being executed in the Ad Hoc constellation, roughly 20 as compared to 30. Ad Hoc does have more downlink opportunities, at 39.4 versus 34.4 for Walker.

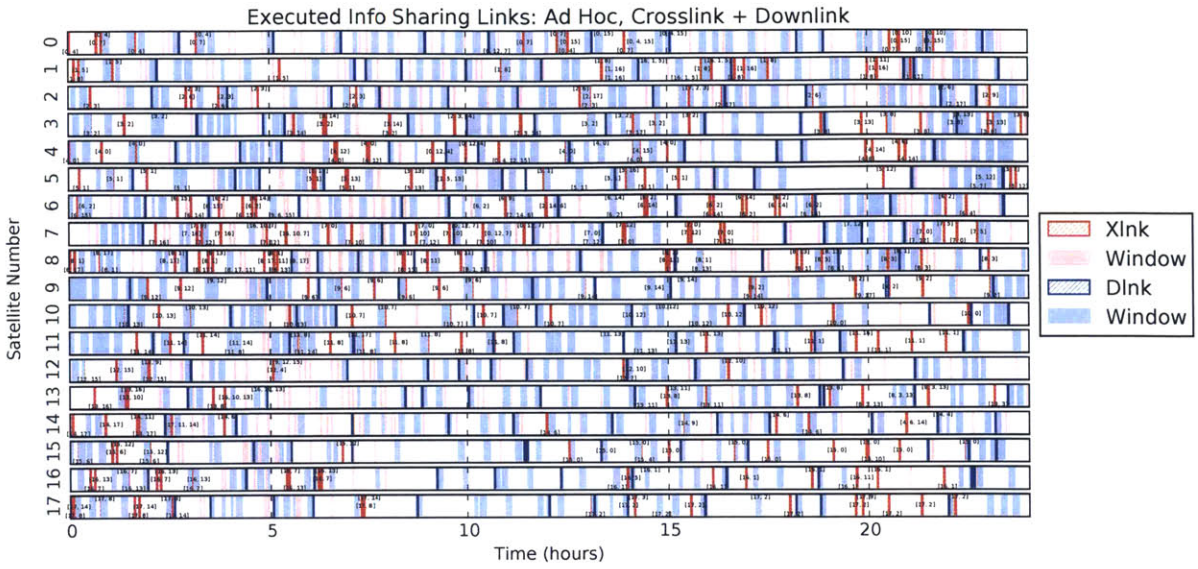


Figure 4-8: Executed information sharing communications links in crosslink + downlink context for Ad Hoc constellation. Text labels over executed crosslinks indicates which satellites were expected to participate in a given crosslink

Table 4.5: Numbers of Windows and Executed Crosslinks, Commlinks, and Downlinks, Averaged Over All Satellites, for Walker (* Dlnks not used for info sharing)

Item	**Communications Context**				
	No Sharing	**Dlnk**	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
Dlnk Windows	* 34.4	34.4	* 34.4	34.4	34.4
Dlnks Executed	* 8.9	8.6	* 8.9	8.6	8.8
X/Clnk Windows	0	0	89.2	89.2	72
X/Clnks Executed	0	0	30.3	29.8	33.3

Table 4.6: Numbers of Windows and Executed Crosslinks, Commlinks, and Downlinks, Averaged Over All Satellites, for Ad Hoc (* Dlnks not used for info sharing)

Item	**Communications Context**				
	No Sharing	**Dlnk**	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
Dlnk Windows	* 39.4	39.4	* 39.4	39.4	39.4
Dlnks Executed	* 11.2	11.6	* 11.1	11.1	10.6
X/Clnk Windows	0	0	47.7	47.7	72
X/Clnks Executed	0	0	20.8	20.7	39.9

4.4 Information Sharing Performance

The third analysis was on the planning information sharing effectiveness across the constellation. Information sharing performance should be dependent on both communications link performance and orbital geometry: the more information sharing communications links performed and the more connected the constellation geometry is, the faster and farther information should spread across the constellation. This expectation is assessed in a series of metrics taken on the Walker constellation, and then the Walker and Ad Hoc constellations are quantitatively summarized and contrasted with final tables.

Figure 4-9 shows the average initial creation latencies for the Walker constellation, which basically measures how long it takes for information to propagate between all pairs of satellites in the constellation. In this commlink + downlink context, we see that information propagates across the constellation relatively quickly: most satellites hear about the observations of other satellites within a single planning window's length. This confirms that, on average, the satellites are pretty well connected when there are many communications links.

Figure 4-10 shows the average initial creation timeliness for the Walker constellation, in the same communications context. This metric effectively measures how timely the shared information about observations being performed was for the receiving ("To") satellite, when those observations were heard about for the very first time. We restrict it only to observations received that occur at most 180 minutes before an observation of the same region by the receiving satellite; that is; relevant observations. The greener the boxes are in Figure 4-10, the better decisions the receiving satellite can make about what regions to observe and what sensors to use for its observations. We see that overall the constellation receives a lot of relevant planning information, and the satellites generally hear about other satellites' *initial* plans in a timely manner. This behavior matches well with the good average revisit time performance of the commlink + downlink context in general. We do see that there are certain groups of satellites, in the top left of Figure 4-10, that don't share any

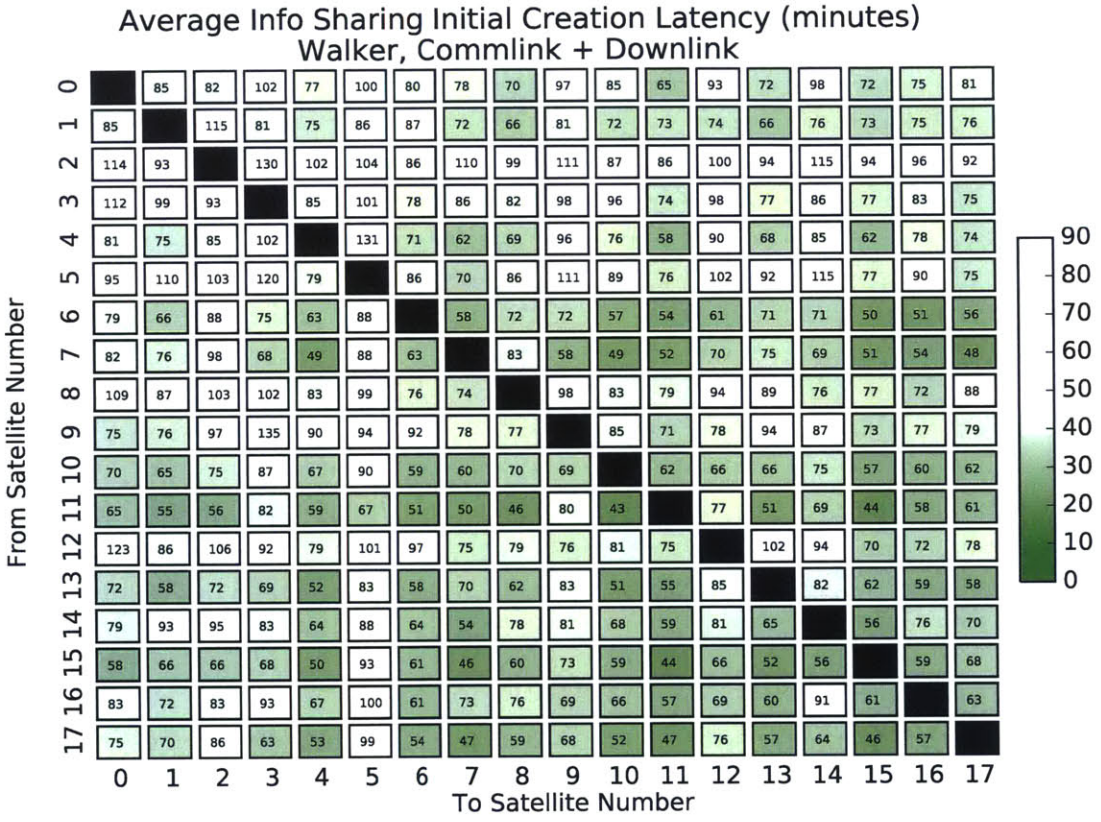


Figure 4-9: Average initial creation latency for Walker constellation, commlink + downlink context. Colors normalized to the 90 minute planning window length

relevant planning information. This observation is supported by Figure 4-12: these satellites simply don't overlap much in their observations.

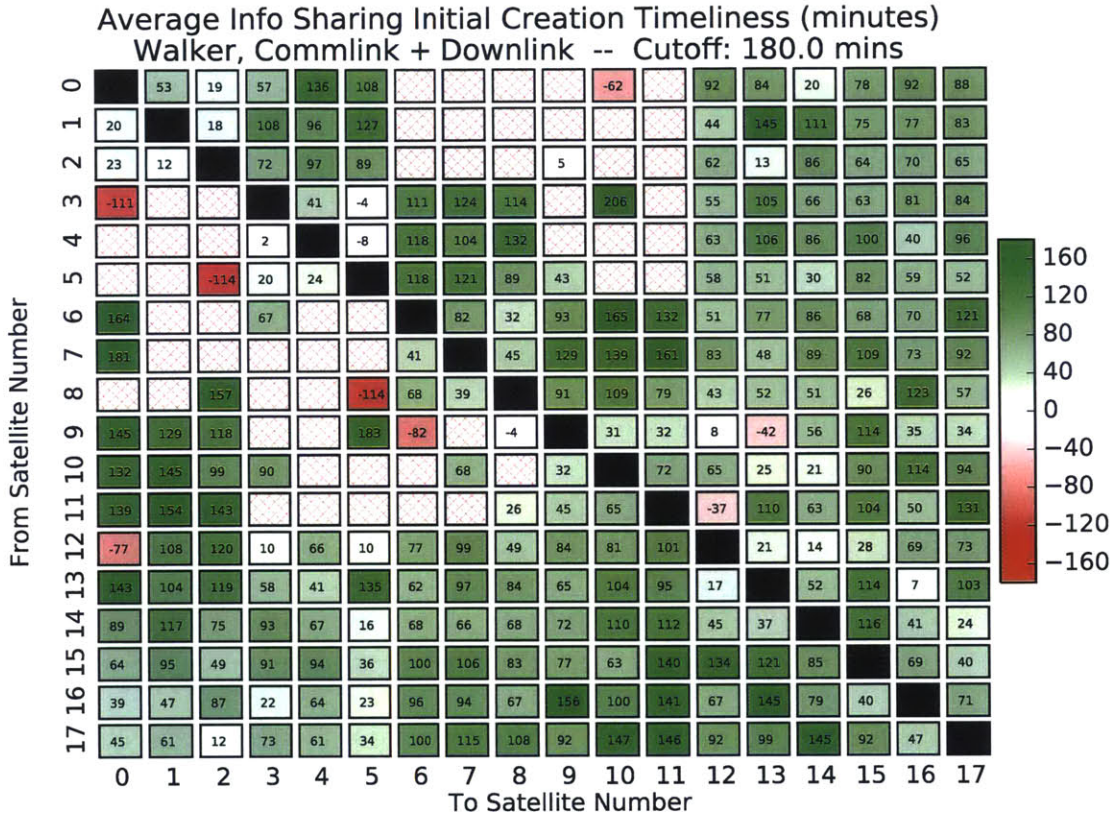


Figure 4-10: Average initial creation timeliness for Walker constellation, commlink + downlink context. Colors normalized to 180 minute cutoff time period

Figure 4-11 shows roughly the same thing as Figure 4-10, except that in this case the timeliness is based on the "last change" made to the observation by the originating satellite: the last time the satellite either created the observation, changed the sensor type, or swapped between planning and cancelling the observation. This last change represents a change in the originating satellites' plans, and could cause the receiving satellite to change its plans as well. We see that the last change timeliness is not quite as good as the initial creation timeliness, but it appears that things are still pretty timely on average. This is a good sign, meaning that, at least in this well-connected communications context, the satellites tend on average not to change their plans at the last minute.

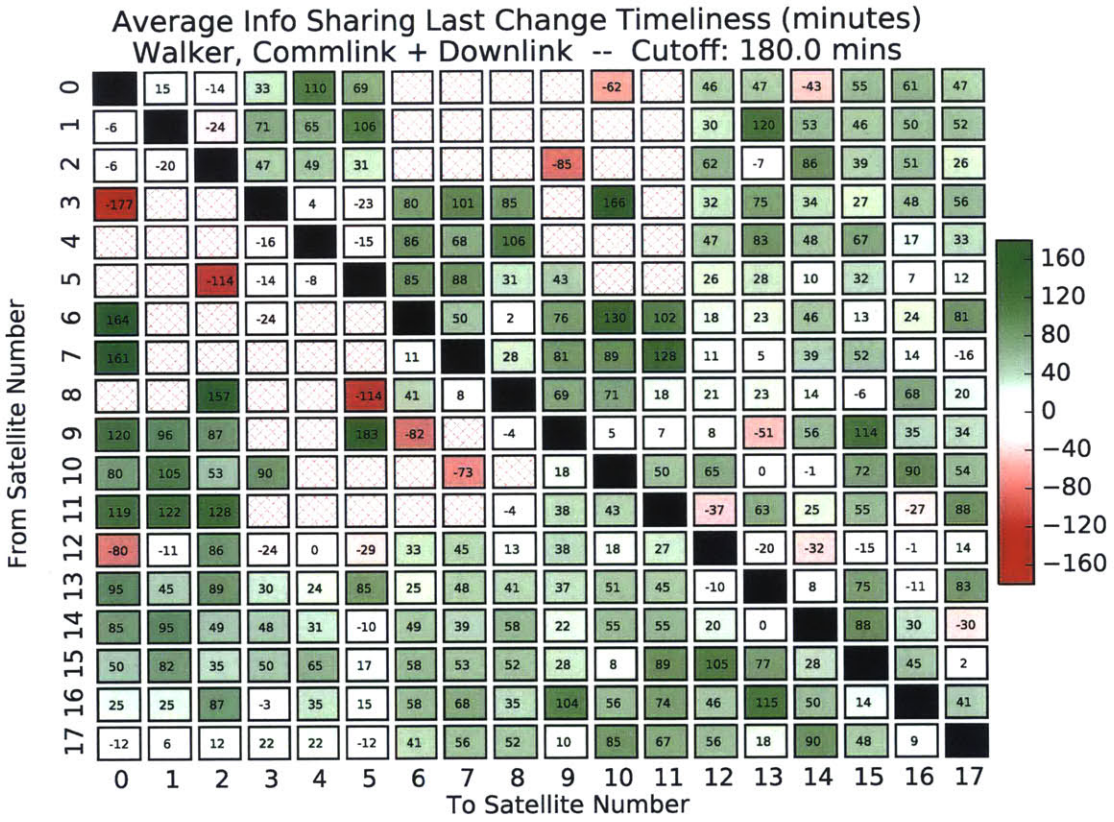


Figure 4-11: Average last change timeliness for Walker constellation, commlink + downlink context. Colors normalized to 180 minute cutoff time period

Finally, the ratio of matching observations plot in Figure 4-12 shows how many of the "From" satellite's observations were both received by the "To" satellite and relevant to the satellite (both observations started within 180 minutes of each other). The larger this ratio, the more effect the originating satellite has on the plans of the receiving satellite. We see that there are "blocks" of interconnected satellites planning-wise. These blocks align mostly with satellites in the same orbit; which makes sense, because those satellites tend to see a lot of the same regions over a relatively brief period. Comparing with Figures 4-10 and 4-11, we see that most of the highly interconnected satellites in the lower right, lower left, and top right of Figure 4-12 receive timely planning information. Those in the top left receive comparatively less timely information. Nonetheless, the combination of timely planning information over many planning-connected satellites illustrates why the commlink + downlink context performs best at minimizing average revisit times over all sensors.

Figures 4-13, 4-14, and 4-15 show a progression of average initial creation timeliness plots over communications contexts downlink, crosslink, and crosslink + downlink. We see relatively good timeliness across the board for the crosslink + downlink context, and okay performance for the downlink context. We see very bad performance for the crosslink context, most likely due to the inability of the satellites to explicitly synchronize their performance of crosslinks between each other.

Interestingly, , even though we see bad information sharing performance in the crosslink context, crosslink outperformed downlink (in terms of disparity between average revisit times for sensor A and C) in Table 4.1. This appears to be an idiosyncrasy of the number of sensors in the simulations (three total) and the revisit time metric used to assess performance; even with very little information sharing, the crosslink context was able to perform well. With more sensors or another type of coordinated task that fundamentally requires more information sharing, this might not happen.

Tables 4.7 and Tables 4.8 summarize the final results for info sharing performance for the Walker and Ad Hoc constellations. "Directions Heard" is the percentage of satellites that heard from other satellites in the constellation, counted in both direc-

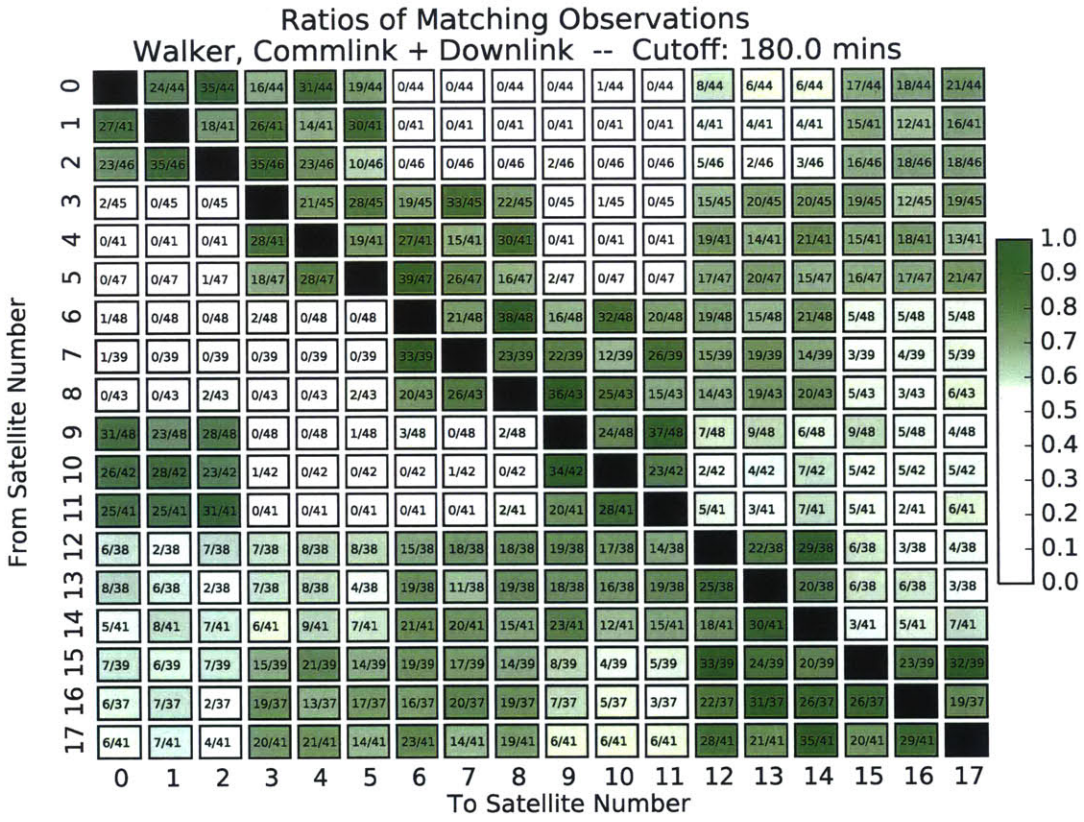


Figure 4-12: Ratios of matching observations, A/B, for Walker constellation, comm-link + downlink context

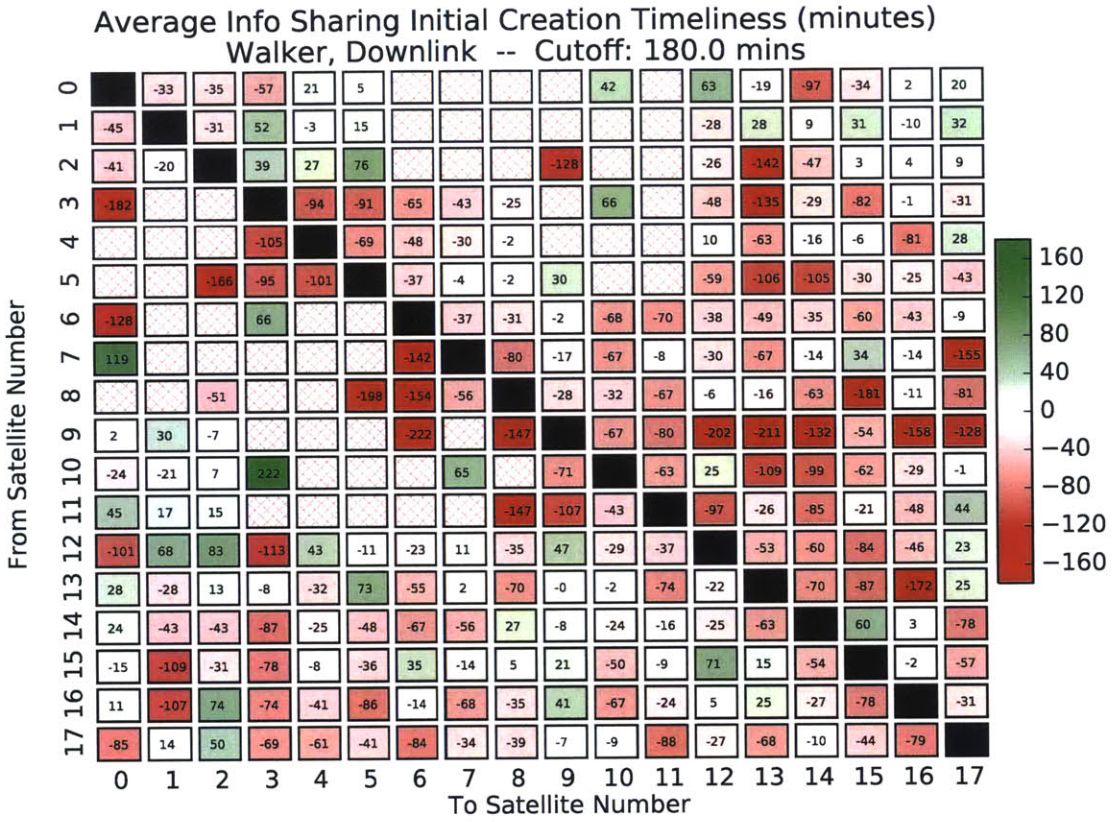


Figure 4-13: Average initial creation timeliness for Walker constellation, downlink context. Colors normalized to 180 minute cutoff time period

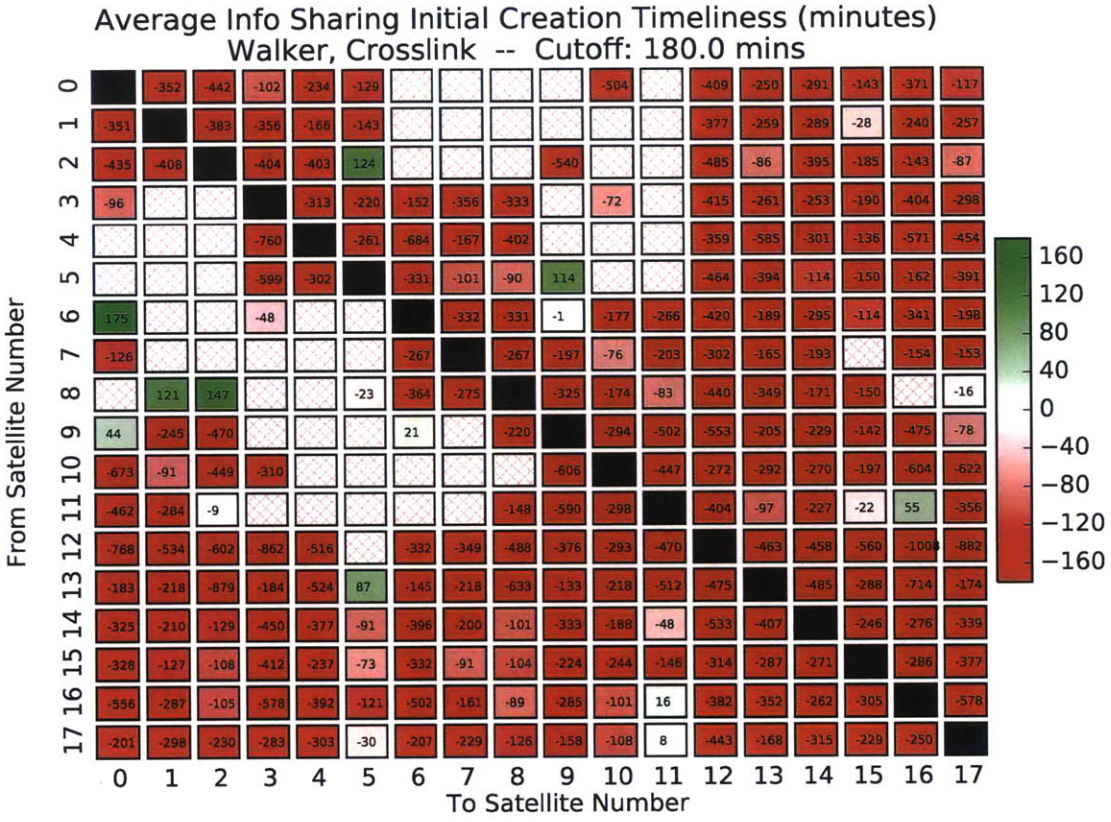


Figure 4-14: Average initial creation timeliness for Walker constellation, crosslink context. Colors normalized to 180 minute cutoff time period

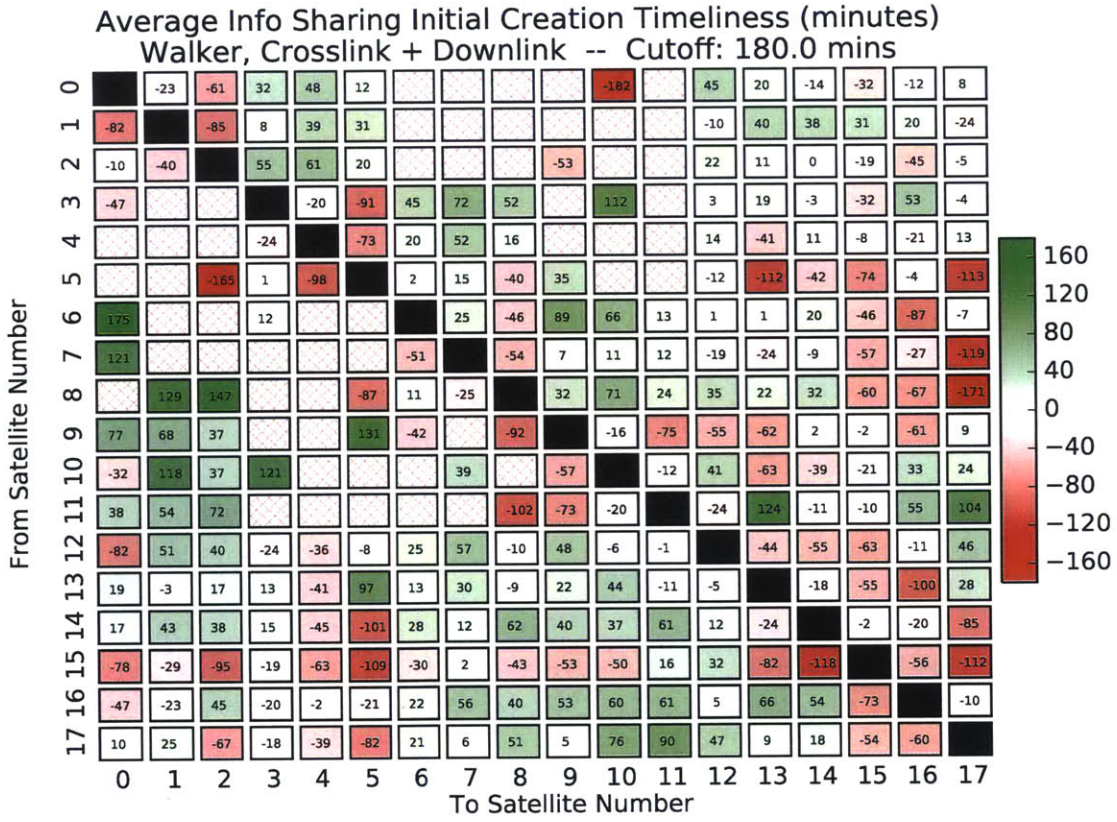


Figure 4-15: Average initial creation timeliness for Walker constellation, crosslink + downlink context. Colors normalized to 180 minute cutoff time period

tions. We see that the commlink + downlink performs best in both constellations, and that the crosslink context performs worse. The Ad Hoc constellation appears to perform better overall, with lower latencies and better timeliness. This is likely due to the increased availability of downlinks for Ad Hoc. It appears that the constellations do not make effective use of crosslinks, and most information ends up passed through downlinks. Again, this is likely because of the lack of an explicit method for prioritizing those crosslinks that are more likely to be executed by both satellites and provide useful planning information.

Table 4.7: Info Sharing Performance for Walker

Item	**Unit**	**Communications Context**			
		Dlnk	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
Average Latency	mins	186	434	155	77
Average Initial Timeliness	mins	-35	-287	-3	74
Average Last Change Timeliness	mins	-79	-318	-43	39
Directions Heard	%	100	99.7	100	100
Average Ratio of Matching Obs	%	26.9	23.8	28.7	29.3

Table 4.8: Info Sharing Performance for Ad Hoc (Averages: over all satellites)

Item	Unit	Communications Context			
		Dlnk	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
Average Latency	mins	146	589	139	57
Average Initial Timeliness	mins	6	-455	10	88
Average Last Change Timeliness	mins	-33	-457	-29	53
Directions Heard	%	100	75.8	100	100
Average Ratio of Matching Obs	%	33.5	13.9	33.7	34.2

4.5 Resource Usage Performance

Tables 4.9 and Tables 4.9 summarize the average resource margins maintained by the satellites over the 24 hour simulation period. Overall the resource margins are large, which is desired. We see that in general the Ah Hoc constellation has higher DS and ES margin, likely due to better access to ground stations and better illumination from the sun. Also, we see a slight drop in average ES margin when crosslinks and commlinks are used, due to the higher resource usage for all of the communications links performed.

Table 4.9: Individual Satellite Average Resource Margin, Averaged Over All Satellites, for Walker (%)

Item	Communications Context				
	No Sharing	**Dlnk**	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
Average DS Margin	86	86.7	83.7	83.1	84
Average ES Margin	68.3	68.6	62.8	63.1	65.4

Table 4.10: Individual Satellite Average Resource Margin, Averaged Over All Satellites, for Ad Hoc (%)

Item	Communications Context				
	No Sharing	**Dlnk**	**Xlnk**	**Xlnk + Dlnk**	**Clnk + Dlnk**
Average DS Margin	88.8	89.4	88.2	88.6	88
Average ES Margin	72.4	73.1	70.5	70.4	68.7

4.6 RASP Timing Performance

An analysis was done on the execution time performance of RASP, as already reported by Kennedy et al [41]. The average time for RASP to create a successful plan was measured over a 24 hour simulation, as a function of planning window length. The simulation was run on a slightly different version of RASP with different input parameters, but was similar enough for the timing analysis to apply to the current version of RASP as well, at least at a high level. The analysis was run on a 2013 Macbook Pro running a 2 GHz Intel Core i7 (quad core) processor, with 8 GB of RAM. The results for this timing analysis are shown in Table 4.11. We see that aver-

age successful execution time roughly scales with the square of the planning window length.

Table 4.11: Average Time for RASP To Create a Successful Activity Timeline [41]

Item	**Planning Window Length (t_h)**				
	30	**60**	**90**	**120**	**240**
Average Time (sec)	0.66	1.74	4.35	8.97	37.71

The timing results show that on a capable computer, RASP can successfully execute in under a minute on average. This suggests that RASP could feasibly be implemented in a lower level language, such as C, and run on an embedded computer typical for CubeSat missions. Example microprocessor boards include the BeagleBone Black, with an ARM Cortex A-8 based computer that can run at 1 GHz with 512 MB of RAM [?], and the Raspberry Pi (model B) running at 900 MHz with 1 GB of RAM [?]. Such computers do take a relatively large amount of power for a CubeSat (2.5 W), but with how the short the planning process is and how long the planning window is, they could be run infrequently.

Chapter 5

Conclusion

5.1 Summary of Results

We presented and analyzed the performance in simulation of two novel algorithms for managing a coordinated CubeSat constellation: RASP and LCCC. The analysis examined two constellation orbital geometry configurations, the Stitched Walker Star and Ad Hoc. Five communications contexts were investigated, with various degrees of planning information sharing: no information sharing, information sharing only through downlink, only through crosslink, through both crosslink and downlink, and through communication with a background constellation ("commlink") and downlink.

The results show promise for the benefits of information sharing. Sharing via a backbone communications constellation and through downlink appears to perform best. Overall average revisit times were much better balanced across the three sensor with information sharing incorporated: the disparity between sensors A and C decreased from 514 to 10 minutes for the Walker constellation and from 617 to 11 minutes for the Ad Hoc constellation. The Walker constellation achieved slightly better average revisit times than the Ad Hoc in the full commlink + downlink case, with times around 200 minutes versus 225 minutes. Roughly the same number of observations, 33, was performed across both constellations and all communications contexts.

It was found that while plenty of crosslink opportunities were available for "Walker"

and "Ad Hoc" (at an average of 89.2 crosslink windows and 47.7, respectively) and many of these were executed (29.8 and 20.7, respectively), the crosslinks were not very effective at reducing information sharing latency and increasing timeliness. This is likely due to the naive approach the algorithms take to crosslink performance, simply trying to execute as many as possible. But it was found that downlinks to a shared ground database do significantly enhance the performance of information sharing.

Several information sharing metrics were looked at for Walker, and it was found that in the commlink + downlink communications context, the information sharing mechanism works quite well. Shared observation information was timely (at 74 minutes and 39 minutes in initial and last change timeliness, respectively), latency was lower than the planning window length (77 minutes), all satellites heard from each in both directions, and on average about on third (29.3%) of satellites' observations were relevant (matching) when shared with others. Info sharing performed similarly, but not quite as well, for Ad Hoc. With crosslink + downlink, the metrics were significantly worse. The context with only crosslink proved to be the worst mechanism for information sharing. It is important to note though that the crosslink contexts still performed well in terms of revisit times, likely due to a peculiarity of the coordinated task of the constellation; a tiny bit of information sharing went a long way in balancing revisit times.

Resource margins were managed well across all simulation cases, being kept quite high. Average DS and ES margin were around 85% and 70% for Walker and 89% and 70% for Ad Hoc. RASP planning time was found to scale roughly with the square of planning window length, but stays under a minute in all cases tested (using the parameters in [41]). This suggests that RASP could feasibly adopted for running onboard an embedded processor on a CubeSat

5.2 Limitations of Algorithms

One important limitation of the current implementation of RASP is that it is only capable of scheduling a single onboard activity at a time. In this work is was assumed

that only a single observation can occur at a time. Significant development would be needed to generalize the algorithm to a simultaneous, multi-activity model while maintaining computational tractability.

The algorithms are naive in their handling of crosslinks and downlinks for information sharing; they simply perform as many as they can, or as many as are needed, respectively. Significant performance gains could be achieved by explicitly reasoning about and weighting the importance of these activites for information sharing.

The algorithm also does not reason about the latency of the science data it has collected and is storing onboard; data is treated as a simple bulk item that is of equal value no matter how or when it is produced. This approach is of limited value operationally because in reality operators do want to be able to treat time-sensitive science data and engineering telemetry differently.

Finally, the algorithm does not model energy consumption and production separately, constraining it to a simple model of constant energy usage rate by spacecraft activity mode. A conservative assumption was used here, namely that energy can only be produced in recharge mode.

5.3 Future Work

Particularly pressing items of future work include addressing the limitations mentioned above: scheduling multiple activities at once, reasoning about the value of crosslinks and downlinks for information sharing in the overall network, handling data latency, and modeling energy consumption and production separately.

Another important item includes adapting the RASP code for implementation on an embedded spacecraft processor. This steps in this process would include translating the code to a lower-level language, e.g. C, and timing its performance on the processor.

A particularly exciting avenue of investigation involves investigating different science and operational applications for such a coordinated constellation. Application possibilities include disaster monitoring, coordination with ground and air assets, and multi-point measurements of large weather systems. More work needs to be done on

how the distributed consensus mechanism could be applied to such problems.

Appendix A

Constellation Orbit Parameters

Satellites for Stitched Walker Star Constellation						
Satellite Number	Alt (km)	Inc (deg)	RAAN (deg)	True Anom (deg)	Eccentricity	Arg of Perigee (deg)
0	600	90	0	0	0	0
1	600	90	0	120	0	0
2	600	90	0	240	0	0
3	600	90	45	10	0	0
4	600	90	45	130	0	0
5	600	90	45	250	0	0
6	600	90	90	20	0	0
7	600	90	90	140	0	0
8	600	90	90	260	0	0
9	600	90	135	30	0	0
10	600	90	135	150	0	0
11	600	90	135	270	0	0
12	500	56	270	96	0	0
13	500	56	270	216	0	0
14	500	56	270	336	0	0
15	500	56	225	50	0	0
16	500	56	225	170	0	0
17	500	56	225	290	0	0

Figure A-1: Parameters for Walker Constellation

Satellites for Ad Hoc Constellation						
Satellite Number	Semimajor axis (km)	Inc (deg)	RAAN (deg)	True Anom (deg)	Eccentricity	Arg of Perigee (deg)
0	7128.14	51	0	110	0	0
1	7128.14	51	0	230	0	0
2	7128.14	51	0	350	0	0
3	7153.14	98	280	10	0	0
4	7153.14	98	280	130	0	0
5	7153.14	98	280	250	0	0
6	7078.14	98	235	35	0.01412801	0
7	7078.14	98	235	155	0.01412801	0
8	7078.14	98	235	275	0.01412801	0
9	7203.14	98	300	90	0	0
10	7203.14	98	300	210	0	0
11	7203.14	98	300	330	0	0
12	6978.14	52	10	30	0	0
13	6978.14	52	10	150	0	0
14	6978.14	52	10	270	0	0
15	7028.14	98	290	50	0	0
16	7028.14	98	290	170	0	0
17	7028.14	98	290	290	0	0

Figure A-2: Parameters for Ad Hoc Constellation

Appendix B

Ground Station Parameters

Ground Stations			
Station	**Number**	**Lat (deg)**	**Long (deg)**
Brazil (Goiania)	1	-16.71	-49.29
Fairbanks	2	64.84	-147.73
Germany (Stuttgart)	3	48.74	9.1
Hawaii	4	21.3	-157.86
Japan (Hakodate)	5	41.78	140.76
New England (MIT)	6	42.36	-71.09
New Zealand (Christchurch)	7	-43.54	172.73
Singapore	8	1.27	103.84
South Africa (Johannesburg)	9	-26.2	28.04

Figure B-1: Ground stations used in constellation simulations

Appendix C

Additional Executed Info Sharing Link Plots

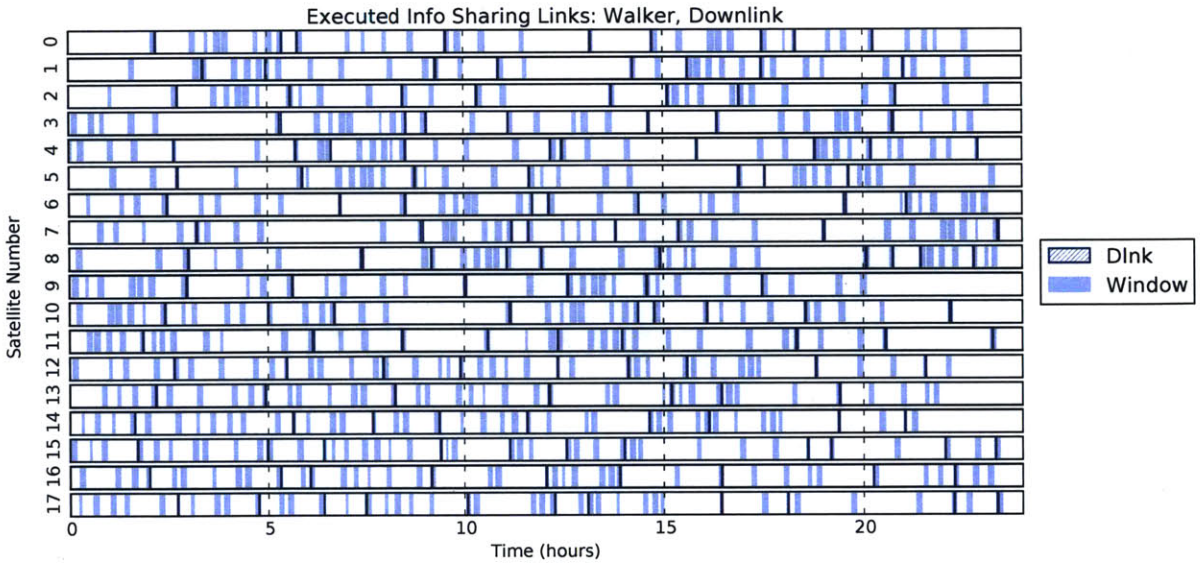


Figure C-1: Executed information sharing communications links in downlink only context for Walker constellation

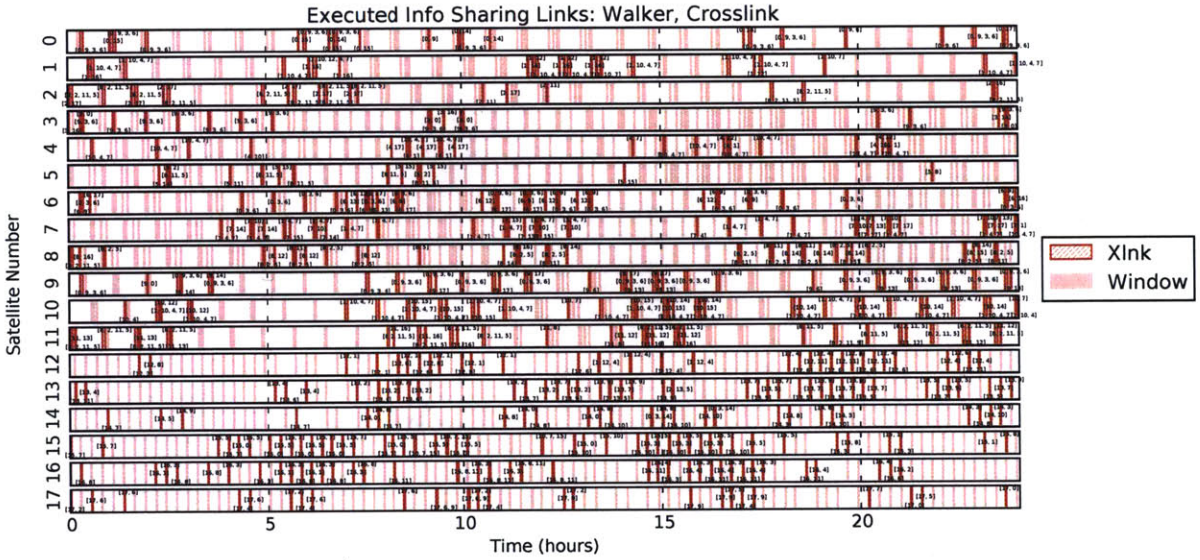


Figure C-2: Executed information sharing communications links in crosslink only context for Walker constellation. (note: downlinks were executed during this simulation, but were not used for information sharing)

Bibliography

[1] Flight Results From PRISMA Formation Flying and Rendezvous Demonstration Mission. In *International Astronautical Congress*, page 12, 2010.

[2] National Aeronautics and Space Administration. Cubesat launch initiative (csli), 2015.

[3] National Aeronautics and Space Administration. Introducing the a-train, 2015.

[4] National Aeronautics and Space Administration. Tracking and data relay satellite (tdrs) fleet, 2015.

[5] Christopher Amato, Girish Chowdhary, Alborz Geramifard, N Kemal Ure, and Mykel J Kochenderfer. Decentralized Control of Partially Observable Markov Decision Processes. In *52nd IEEE Conference on Decision and Control*, pages 2398–2405, Florence, Italy, 2013.

[6] 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.

[7] beagleboard.org. Beaglebone black system reference manual, 2015.

[8] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005.

[9] William J Blackwell, Greg Allan, Greg Allen, Dennis Burianek, Franz Busse, D Elliott, Christopher Galbraith, Robert Leslie, Idahosa Osaretin, Mike Shields, Erik Thompson, David Toher, Kerri Cahoy, Pratik Dave, Andrew Kennedy, Ryan Kingsbury, Anne Marinan, Eric Peters, Christopher Pong, Meghan Quadrino, James Mic Byrne, Rebecca Bishop, James Bardeen, Neal Erickson, Chad Fish, and Erik Stromberg. Microwave Radiometer Technology Acceleration Mission (MiRaTA): Advancing Weather Remote Sensing with Nanosatellites. In *28th Annual AIAA/USU Conference on Small Satellites*, pages P4–12, Logan, UT, 2014.

[10] 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*, pages XI–1, 2013.

[11] 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*, pages I–1, 2014.

[12] Elizabeth Buchen. SpaceWorks âĂŹ 2014 Nano / Microsatellite Market Assessment. In *28th Annual AIAA/USU Conference on Small Satellites*, pages SSC14–I–3, 2014.

[13] Innovative Solutions In Space B.V. Isipod cubesat deployer, 2015.

[14] Kerri L Cahoy, Anne Marinan, Weston Marlow, Timothy Cordeiro, William J. Blackwell, Rebecca Bishop, and Neal Erickson. Development Of The Microwave Radiometer Technology Acceleration (MiRaTA) Cubesat For All-Weather Atmospheric Sounding. In *IGARSS 2015*, pages 5304–5307, 2015.

[15] California Polytechnic State Univ. CubeSat Design Specification Rev. 13. Technical report, 2014.

[16] 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 for Space*, number 1, Turin, Italy, 2012.

[17] Steve Chien, B Engelhardt, R Knight, G Rabideau, R Sherwood, Eric a Hansen, a Ortiviz, C Wilklow, and S Wichman. Onboard Autonomy on the Three Corner Sat Mission. *International Symposium on Artificial Intelligence, Robotics and Automation for Space*, 2001.

[18] Steve Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau. Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling. *The Fifth International Conference on Artificial Intelligence Planning and Scheduling*, pages 300–307, 2000.

[19] 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 âĂŞ Automated Planning and Scheduling for Space Mission Operations. In *International Conference on Space Operations (SpaceOps 2000)*, pages 1–10, Toulouse, France, 2000.

[20] 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: i-SAIRAS 2001*, 2001.

[21] 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.

[22] 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*, 2013.

[23] 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*, page 455, 2005.

[24] 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.

[25] George B Dantzig, Alex Orden, Philip Wolfe, et al. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, 1955.

[26] Subrata Das, Curt Wu, and Walt Truszkowski. Enhanced Satellite Constellation Operations via Distributed Planning and Scheduling. In *Proceeding of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS 2001*, St. Hubert, Quebec, Canada, 2001.

[27] Raspberry Pi Foundation. Raspberry pi 2 model b, 2015.

[28] Joseph W Gangestad, James R Wilson, Kristin L Gates, and John V Langer. RIDESHARE-INITIATED CONSTELLATIONS : FUTURE CUBESAT ARCHITECTURES WITH THE CURRENT LAUNCH MANIFEST. In *31st Space Symposium*, number April, pages 1–18, Colorado Springs, 2015.

[29] 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.

[30] Eberhard Gill. Together in Space: Potentials and Challenges of Distributed Space Systems. Technical report, TU Delft, Faculty of Aerospace Engineering, Delft, Netherlands, 2008.

[31] Blacksky Global. Blacksky global (main webpage), 2015.

[32] LLC Globalstar. Constellation, 2015.

[33] Oliver Guinan. Skybox Imaging: Commercial Imaging Constellation Meets Cloud Computing. In *Ground Systems Architectures Workshop*, 2013.

[34] John Hanson, James Chartres, Hugo Sanchez, and Ken Oyadomari. The EDSN Intersatellite Communications Architecture. In *11th Annual Summer CubeSat Developers' Workshop*, number SSC14-WK-2, 2014.

[35] SeungBum Hong, Hyungho Na, and Jaemyung Ahn. Assessment of architectural options for a dual-mode disaster monitoring constellation supported by on-orbit propellant depots. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 228(11):2108–2122, 2013.

[36] 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.

[37] Iridium Communications Inc. Satellite constellation, 2015.

[38] Maryland Aerospace Inc. Static Earth Sensor Product Specification. Technical report, Maryland Aerospace Inc., 2015.

[39] Pumpkin Inc. 3d models of the cubesat kit, 2015.

[40] Spaceflight Industries Inc. Site locations, 2015.

[41] Dario Izzo. Autonomous and Distributed Motion Planning for Satellite Swarm. 30(2), 2007.

[42] C Jilla and D Miller. A Reliability Model for the Design and Optimization of Separated Spacecraft Interferometer Arrays. In *11th AIAA/USU Conference on Small Satellites*, Logan, UT, 1997.

[43] Andrew E Kalman. Enhanced Power Systems for CubeSats. In *CubeSat Developer's Workshop*, 2012.

[44] Andrew Kennedy and Kerri Cahoy. The MiRaTA CubeSat FSW Architecture & Scaling CubeSat FSW to Cooperative Constellations. In *2014 Workshop on Spacecraft Flight Software*, Pasadena, CA, 2014.

[45] Andrew Kennedy, Anne Marinan, Kerri Cahoy, James Byrne, Timothy Cordeiro, Zachary Decker, Weston Marlow, William J Blackwell, Michael Diliberto, R Vincent Leslie, Idahosa Osaretin, Erik Thompson, and Rebecca Bishop. Automated Resource-Constrained Science Planning for the MiRaTA Mission. In *29th Annual AIAA/USU Conference on Small Satellites*, pages SSC15–VI–1, Logan, UT, 2015.

[46] Andrew K Kennedy and Kerri L Cahoy. Onboard Operations Scheduling For A Cooperative Earth Remote Sensing Small Satellite Constellation. In *Proceedings of the 9th Annual International Workshop on Spacecraft Constellations and Formation Flying*, Delft, Netherlands, 2015.

[47] Bryan Klofas and Kyle Leveque. A Survey of CubeSat Communication Systems: 2009-2012. Technical report, 2013.

[48] Edward W Kneller, Kevin L Hyer, Todd Mcintyre, David K Jones, and Salt Lake City. Cadet: A High Data Rate Software Defined Radio for SmallSat Applications. In *26th Annual AIAA/USU Conference on Small Satellites*, pages SSC12–X–4, Logan, UT, 2012.

[49] NanoRacks LLC. Nanoracks completes historic third round of space station cubesat deployments, 2015.

[50] NanoRacks LLC. Nanoracks cubesat deployer (nrcsd) interface control document, 2015.

[51] David LoBosco, Richard Golding, Glen Cameron, and Theodore Wong. Pleiades Fractionated Space System Architecture and the Future of National Security Space. *AIAA SPACE 2008 Conference & Exposition*, pages 1–10, 2008.

[52] Anne Marinan, Austin Nicholas, and Kerri Cahoy. Ad hoc CubeSat constellations: Secondary launch coverage and distribution. In *IEEE Aerospace Conference Proceedings*, Big Sky, MT, 2013.

[53] Mission Design Division Staff. Small Spacecraft Technology State of the Art. Technical Report TPâĂŞ2014âĂŞ216648 Small, NASA Ames Research Center, Moffett Field, CA, 2014.

[54] Daniel Morgan, Soon-jo Chung, and Fred Y. Hadaegh. DECENTRALIZED MODEL PREDICTIVE CONTROL OF SWARMS OF SPACECRAFT USING SEQUENTIAL CONVEX PROGRAMMING. In *Proc. 23rd AAS/AIAA Space Flight Mechanics Meeting*, pages 1–25, Kauai, Hawaii, 2013.

[55] Katta G Murty. Linear programming. 1983.

[56] Sreeja Nag. DESIGN AND ANALYSIS OF DISTRIBUTED NANOSATELLITE SYSTEMS FOR MULTI-ANGULAR, MULTI-SPECTRAL EARTH OBSERVATION. In *64th International Astronautical Congress*, pages 1–6, Beijing, China, 2013.

[57] National Oceanic and Atmospheric Administration. Frequently asked questions, 2015.

[58] NASA Office of the Chief Technologist. Small spacecraft technology, 2012.

[59] Sung Wook Paek. Reconfigurable Satellite Constellations for Geo-spatially Adaptive Earth Observation Missions. (May):1–149, 2012.

[60] Anthony J Piplica, John Olds, and Brad St Germain. GOLauncher 2: Fast, Flexible, and Dedicated Space Transportation for Nanosatellites. In *28th Annual AIAA/USU Conference on Small Satellites*, number SSC14-IV-2, 2014.

[61] Kathleen Riesing. Orbit Determination from Two Line Element Sets of ISS-Deployed CubeSats. In *29th Annual AIAA/USU Conference on Small Satellites*, pages VIII–5, 2015.

[62] Christopher Rouff. Intelligence in Future NASA Swarm-based Missions. pages 112–115, 2002.

[63] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, Inc., 2003.

[64] Thomas Schetter, Mark Campbell, and Derek Surka. Multiple agent-based autonomy for satellite constellations. *Artificial Intelligence*, 145:147–180, 2003.

[65] Daniel Selva and David Krejci. A survey and assessment of the capabilities of Cubesats for Earth observation. *Acta Astronautica*, 74:50–68, 2012.

[66] Graeme B. Shaw, D. W. Miller, and D. E. Hastings. Generalized Characteristics of Communication, Sensing, and Navigation Satellite Systems. *Journal of Spacecraft and Rockets*, 37(6):801–811, 2000.

[67] 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.

[68] Michael Swartwout. The First x100x x200x 272 Cubesats. In *Electrical, Electronic, and Electromechanical (EEE) Parts for Small Missions Workshop 2014*, number September, 2014.

[69] Firefly Space Systems. Firefly alpha, 2015.

[70] Howard Tripp and Phil Palmer. Stigmergy based behavioural coordination for satellite clusters. *Acta Astronautica*, 66(7-8):1052–1071, 2010.

[71] Walt Truszkowski, Harold L Hallock, Christopher Rouff, Jay Karlin, James Rash, Mike Hinchey, and Roy Sterritt. *Autonomous and Autonomic Systems: With Applications to NASA Intelligent Spacecraft Operations and Exploration Systems.* Springer, 2009.

[72] Johannes van der Horst. *Market-based Task Allocation in Distributed Satellite Systems.* PhD thesis, University of Southampton, 2012.

[73] Johannes van der Horst and Jason Noble. Task allocation in networks of satellites with Keplerian dynamics. *Acta Future*, 5:143–151, 2012.

[74] Harvey M Wagner. The dual simplex algorithm for bounded variables. *Naval Research Logistics Quarterly*, 5(3):257–261, 1958.

[75] J.G. Walker. Satellite Constellations. *Journal of the British Interplanetary Society*, 37:559–571, 1984.

[76] David Wang and Brian C Williams. tBurton : A Divide and Conquer Temporal Planner. Technical report, 2014.

[77] David Wang and P I Brian Williams. tBurton : Model âĂŘ based Temporal Generative Planning. In *KISS Workshop: Engineering Resilient Space Systems: Leveraging Novel System Engineering Techniques and Software Architectures*, Pasadena, CA, 2012.

[78] Guy G. Zohar. *AD-HOC REGIONAL COVERAGE CONSTELLATIONS OF CUBESATS USING SECONDARY LAUNCHES.* PhD thesis, California Polytechnic State University, San Luis Obispo, 2013.