Computers & Operations Research 57 (2015) 1–16

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Optimization-based scheduling for the single-satellite, multi-ground station communication problem

Sara Spangelo [a], James Cutler [a], Kyle Gilson [b], Amy Cohn [b,*]

[a] Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, United States
[b] Department of Industrial and Operations Research, University of Michigan, Ann Arbor, MI 48109, United States

ARTICLE INFO

Available online 9 December 2014

Keywords:
Scheduling
Resource allocation
Integer programming
Satellite operations

ABSTRACT

In this paper, we develop models and algorithms for solving the *single-satellite, multi-ground station communication scheduling problem*, with the objective of maximizing the total amount of data downloaded from space. With the growing number of small satellites gathering large quantities of data in space and seeking to download this data to a capacity-constrained ground station network, effective scheduling is critical to mission success. Our goal in this research is to develop tools that yield high-quality schedules in a timely fashion while accurately modeling on-board satellite energy and data dynamics as well as realistic constraints of the space environment and ground network. We formulate an under-constrained mixed integer program (MIP) to model the problem. We then introduce an iterative algorithm that progressively tightens the constraints of this model to obtain a feasible and thus optimal solution. Computational experiments are conducted on diverse real-world data sets to demonstrate tractability and solution quality. Additional experiments on a broad test bed of contrived problem instances are used to test the boundaries of tractability for applying this approach to other problem domains. Our computational results suggest that our approach is viable for real-world instances, as well as providing a strong foundation for more complex problems with multiple satellites and stochastic conditions.

1. Introduction

The potential for small satellites to perform novel science, observation, and technology missions is being realized worldwide across academic, government, and industry settings. Satellites gather and store data as they orbit the Earth, and require a supporting network of ground stations to receive and distribute this data [1–7]. The existing ground station infrastructure for these missions has largely been monolithic, designed and used by independent organizations for a handful of satellite missions to date. However, federated ground station networks, consisting of autonomous, globally distributed ground stations, are now beginning to form [8]. The need to efficiently download data from the growing number of satellites to the existing and future ground station network gives rise to scheduling problems that are particularly challenging due to the exchange of limited resources between space and ground entities.

The goal of the research presented in this paper is to develop models and algorithms for building communication schedules for a single satellite and a heterogeneous network of globally distributed ground stations, so as to maximize the amount of data downloaded to Earth; we call this the *Single-Satellite Multiple-Ground Station Scheduling Problem* (*SMSP*).

The contributions of our research are three-fold. *First*, we develop models and algorithms that enable us to solve real-world instances of *SMSP* in acceptable run times. *Second*, our research lays the foundation for solving more complex satellite scheduling problems, such as those in which there is stochasticity in the system (e.g. uncertainty as to the availability of the ground stations) as well as problems in which there are multiple satellites competing for the same ground station resources. *Third*, the theoretical insights gained in this research have relevance for many other applications in which tasks must be scheduled subject to resource acquisition, storage, and utilization constraints.

The remainder of the paper is outlined as follows. Section 2 places our research problem in the context of the related existing literature. In Section 3, we formally state the problem and present a continuous-time model to demonstrate the physical dynamics of the system. In general, the associated non-linear optimization problem is not tractable except for limited special cases. We therefore develop a discretized *mixed-integer programming (MIP)*

* Corresponding author.
E-mail addresses: saracs@umich.edu (S. Spangelo), jwcutler@umich.edu (J. Cutler), kygils@umich.edu (K. Gilson), amycohn@umich.edu (A. Cohn).

http://dx.doi.org/10.1016/j.cor.2014.11.004
0305-0548/© 2014 Elsevier Ltd. All rights reserved.

formulation in Section 4, which under-constrains the problem. A special case in which this model is guaranteed to yield optimal solutions to the original problem is discussed in Section 5, with corresponding computational results provided. For the more general case, we present an iterative algorithm in Section 6 that progressively tightens the constraints to converge to a feasible and thus optimal solution. Note that this approach differs from conventional cutting-plane algorithms in that it introduces new variables and constraints and solves a newly defined (and larger) problem at each iteration. Computational experiments are presented in this section as well, to demonstrate tractability and investigate problem structure. We conclude in Section 7 with a summary and suggestions for future research.

2. Literature review

There is a large body of related research on spacecraft operations and scheduling; however we have not encountered any other research explicitly studying *SMSP* as it is formulated in this paper. First, we review the well-studied problem of scheduling imaging spacecraft due to its extensive literature and similarities to the spacecraft communication scheduling problem. Next, we discuss approaches to solving spacecraft downlinking optimization problems. Finally, we describe limitations of the existing work and how our approach fits within the context of the existing literature.

A very common scheduling problem addressed in the literature is the *Earth Observing Satellites*scheduling problem (*EOS*). In *EOS*, the goal is to take the maximum number of high-priority observations with on-board spacecraft sensors during a given time period. This problem is similar to *SMSP* in that they both consist of scheduling a set of complex tasks involving the exchange of limited resources between an orbiting spacecraft and Earth-based targets. [Key differences between the problems are discussed in the following paragraph.] Additionally, data and energy are collected and consumed in both problems, providing restrictions on when and how the desired tasks can be performed. *EOS* is often generalized to a more common problem structure, such as a knapsack problem [9–12], a packing problem [13], a single-machine scheduling problem [14], or a network flow problem [15]. Constraint programming is used by others [11,12,16,17]. One of the most common approaches in solving *EOS* is a greedy algorithm based on spacecraft priorities [11,13,18–21]. Other common techniques include dynamic programming [11,15], heuristic approaches [19,22,23], and genetic [9,13,14,22,24]. Look-ahead methods are used by [13], look-behind pre-emption methods by [25], repair-based iterative schemes by [26] and [27], and particle swarm optimization by [16]. Other methods used to solve *EOS* include prune and search trees [28], branch and bound procedures [29], and tabu searches with intensification and diversification [10,30]. Finally, [24] provides a comparison of several strategies for solving *EOS*, including a genetic algorithm, hill climbing, simulated annealing, squeaky wheel optimization, and iterated sampling implemented as permutation-based methods.

Despite the similarities between *EOS* and *SMSP*, there are differences in the problem objectives, decisions, and constraints. For example, in addition to decisions on *when* to download, *SMSP* includes decisions on *how* to download (i.e. what data rate, energy utilization, and efficiency), which governs the relationship between the energy and data consumed during downloads. Thus *SMSP* must be modeled and solved in a new way. However, much of the *EOS* literature is informative in developing these models and algorithms.

Conventional approaches to satellite operations often use greedy scheduling, where the satellite performs mission operations and downloads data at every feasible opportunity. This approach may be sub-optimal since in some cases it may be preferable to store resources for a future time period with more beneficial download options.

Only a handful of researchers have studied the scheduling of spacecraft downlinks [22,23,31–33]. Ref. [31] considers multiple spacecraft downlinking. Polynomial time algorithms are used to solve several special cases, including a greedy algorithm and an approach based on exploiting a longest-path formulation in a directed acyclic graph. Ref. [32] focuses on minimizing the communication time required to meet the download constraints of a system with multiple spacecraft and ground stations. They formulate and solve a non-linear constrained optimization problem, showing results for small network examples with a simplified set of communication parameters and constraints. Ref. [23] studies a hierarchy of successively more complex spacecraft scheduling problems and propose a tight time-indexed formulation. A Lagrangian relaxation heuristic is implemented to solve the scheduling problem and results are shown for the GALILEO constellation; however spacecraft energy collection, storage, and consumption are not considered in the analysis.

In general, the formulations and approaches in the literature towards optimizing satellite downlink schedules ignore important logistical constraints (e.g. on-board energy and data buffers) necessary in *SMSP* and treat only small-scale problem instances. More sophisticated models relative to those presented in the literature are required to accurately represent the realistic constraints of *SMSP*, including representations of on-board data storage, communication systems, and energy management systems. Higher levels of modeling fidelity are also required, as is the ability to extend the model to incorporate the even greater challenges of multiple spacecraft and multiple ground stations concurrently, as well as to address system stochasticity. Developing such models, as well as the corresponding algorithms to find high-quality solutions in acceptable run times, is the goal of our research.

3. Problem description and system dynamics

The goal of *SMSP* is to maximize the amount of data downloaded to a network of ground stations from a single spacecraft orbiting the Earth. In this problem, we assume the following:

- A single satellite is orbiting Earth, collecting both data (via on-board instruments) and energy (via solar panels) along its orbit.
- The satellite's energy and data acquisition rates may vary over time. For example, the collection of data depends on whether the spacecraft is in view of a target of interest (e.g. a science or surveillance target) and the collection of energy depends on the line of sight of the solar panels relative to the sun.
- Energy is required to conduct basic operational functions of the spacecraft and download data.
- The satellite has finite limits on the amount of energy and data that can be stored at any given time.
- We assume that the spacecraft orbit is deterministic and known such that the access times to the globally distributed ground stations are known a priori.
- There are multiple ground stations to which the satellite can download, each of which periodically comes into view of the satellite. Note that the view periods have variable durations, which depends on the geometry of the orbit relative to the ground station.
- More than one ground station may be in view of the satellite simultaneously, but the satellite can only download to one ground station and at one data rate at a time.
- Ground stations may vary in their characteristics, both with respect to the rate with which they may receive data (bits-per-second) and the energy utilization from the satellite required to do so (Joules-per-bit). Ground stations also vary in the *efficiency* of the data download (i.e. the fraction of transmitted data that

is actually captured by the ground station, where the losses are due to communication system and transmission inefficiencies [34]). Furthermore, each ground station may have multiple download *options* from which to choose, where an option is defined by a combination of values for data rate, energy utilization, and efficiency.

A *schedule* defines the time periods during which downloads are to take place and, for each of these time periods, which option is to be used. The objective is to maximize the total amount of data collected, subject to the constraints of the system. The mission scheduling horizon is the time for which we optimize the download schedule, $[0, T]$. Within this horizon, we define a set of *intervals*, I. Whenever a ground station comes in or out of view of the spacecraft, a new interval, i, starts. The intervals may be of variable duration, however we define that within an interval the set of ground stations in view remains constant. Intervals in which there are one or more ground stations in view of the spacecraft are download *opportunities*. See Fig. 1 for an example.

$O(t)$ denotes the set of download options at time $t \in [0, T]$. Each download option, $o(t)$, is defined by the data download rate, $\phi_{o(t)}$ (bits per second), the energy utilization, $\alpha_{o(t)}$ (Joules per bit), and the data download efficiency, $\eta_{o(t)}$ (fraction of transmitted data that is successfully received by the ground station). The no-download option, with $\{\phi_{o(t)}, \alpha_{o(t)}, \eta_{o(t)}\} = \{0, 0, 0\}$, is also an option at every instance. When there is not a download opportunity, the no-download option is the only available one. Since the spacecraft can only download to one ground station and single rate at a time, we enforce that it can only download using a single option within any given interval.

3.1. Energy dynamics

Spacecraft collect solar energy by solar panels and store energy in a battery as they orbit the Earth. Energy is consumed to support the basic functionality of the spacecraft and to download data. There is also upper and lower bounds on the amount of energy that can be stored in the spacecraft's battery.

Let $e(0)$ be the initial energy level at time $t=0$, and $o(t)$ be the option at time t associated with a given feasible data download schedule. We can then compute $e(t)$, the energy stored in the battery at time t, recursively in the following way:

$$e(t) = \min \begin{bmatrix} e(t-\tau) + \left[\pi^{e+}(t-\tau) - \pi^{e-}(t-\tau) - \alpha_{o(t-\tau)} \cdot \phi_{o(t-\tau)}\right]\tau, \\ e_{\max} \end{bmatrix}, \tag{3.1}$$

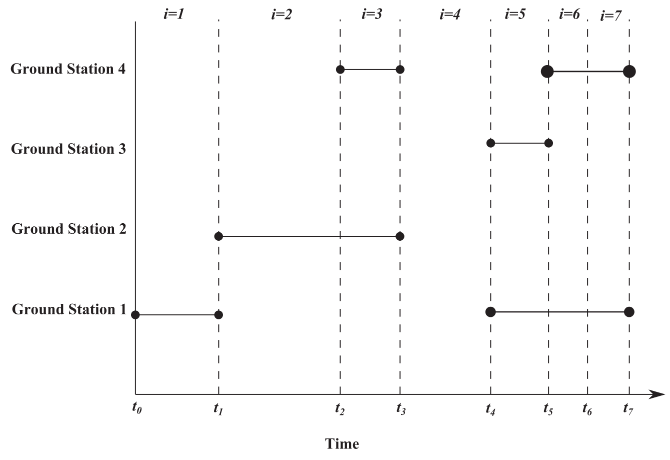

Fig. 1. Horizontal lines indicate when a ground station is in view of the spacecraft as a function of time. Vertical lines indicate the boundaries between intervals.

recognizing that if the energy levels exceeds the battery capacity, then the excess will be spilled.

In Eq. (3.1),

- $\tau \geq 0$ is a unit of time approaching 0 in the limit, measured in seconds.
- $\pi^{e+}(t)$ is the rate of energy collection at time t, measured in Joules per second.
- $\pi^{e-}(t)$ is the rate of energy consumption for nominal (non-downloading) functional operations at time t, measured in Joules per second.
- $\alpha_{o(t)}$ is the energy utilization to download using option $o(t)$, measured in Joules per bit.
- $\phi_{o(t)}$ is the data download rate using option $o(t)$, measured in bits per second.
- $e_{\max}$ is the upper limit on energy storage for the battery, measured in Joules.

Note that, by our assumption of the feasibility of the schedule, $e(t)$ will never drop below $e_{\min}$, which is defined to be the lower limit on energy storage for the battery, measured in Joules.

3.2. Data dynamics

Data dynamics are analogous to energy dynamics, with the exception that we do not need the $\alpha_{o(t)}$ term. There are upper and lower bounds on the amount of data that can be stored in the spacecraft's data buffer, denoted $d_{\max}$ and $d_{\min}$, respectively. Replacing π^{e+} and π^{e-}, with the analogous rates of data collection and consumption, π^{d+} and π^{d-}, we have

$$d(t) = \min \begin{bmatrix} d(t-\tau) + \left[\pi^{d+}(t-\tau) - \pi^{d-}(t-\tau) - \phi_{o(t-\tau)}\right]\tau, \\ d_{\max} \end{bmatrix}. \tag{3.2}$$

Note that, by our assumption of the feasibility of the schedule, $d(t)$ will never drop below $d_{\min}$.

3.3. System optimization

Our goal is to find a feasible schedule that maximizes the amount of data downloaded over the planning horizon. The discontinuities in the system dynamics caused by the upper and lower bounds on the allowable stored energy and data result in a difficult non-linear optimization problem. To overcome this challenge, we discretize the problem into a finite set of time periods to approximate the continuous-time dynamics. We initially require the energy and data constraints to be satisfied only at the start and end of each time period and ignore the dynamics within these time periods. We demonstrate that this under-constrained formulation may yield an infeasible solution. However, if the solution is feasible, then it is also optimal. In Section 4 we present the under-constrained formulation. In Section 5 we discuss special cases where the under-constrained formulation is guaranteed to yield a feasible and thus optimal solution. In Section 6 we present an iterative algorithm that progressively tightens the constraints of the under-constrained formulation until a feasible and thus optimal solution is achieved.

4. Problem formulation

4.1. Notation

4.1.1. Sets and subsets

- I is the set of intervals.
- O is the set of download options.

- $O_i \subseteq O$ is the subset of download options available during interval i, $\forall i \in I$.

4.1.2. Parameters

- η_{io} is the efficiency during interval i when downloading using option o, $\forall i \in I, o \in O_i$.
- t_i is the start time of interval i, measured in seconds, $\forall i \in I$.
- t_{i+1} is the end time of interval i, and coincides with the start of interval $i+1$, measured in seconds, $\forall i \in I$.
- Δt_i is the duration of time interval i ($\Delta t_i = t_{i+1} - t_i$), measured in seconds, $\forall i \in I$.
- ϕ_{io} is the data rate associated with downloading during interval i using option o, measured in bits/seconds, $\forall i \in I, o \in O_i$.
- α_{io} is the energy per data associated with downloading using option o during interval i, measured in Joules/bits, $\forall i \in I, o \in O_i$.
- e_{min} and e_{max} are the minimum and maximum allowable amount of energy to be stored in the battery, measured in Joules.
- e_{start} is the amount of energy stored in the battery at the beginning of the planning horizon, measured in Joules.
- d_{min} and d_{max} are the minimum and maximum allowable data stored in the data buffer, measured in bits.
- d_{start} is the amount of data stored in the data buffer at the beginning of the planning horizon, measured in bits.
- $\delta_i^{e+} \geq 0$ is the total amount of energy that can be acquired during interval i, measured in Joules, $\forall i \in I$.
- $\delta_i^{e-} \geq 0$ is the total amount of energy consumed during the interval i for non-download operations (e.g. data collection and nominal operations), measured in Joules, $\forall i \in I$.
- $\delta_i^{d+} \geq 0$ is the total amount of data that can be acquired during interval i, measured in bits, $\forall i \in I$.
- $\delta_i^{d-} \geq 0$ is the total amount of data lost during the interval i unrelated to data download (e.g. data degradation, expiration, etc.), measured in bits, $\forall i \in I$.

4.1.3. Variables

- $x_{io} \in \{0,1\}$ is the binary value representing the decision of whether to download using option o during some portion of interval i, $\forall i \in I, o \in O_i$.
- $q_{io} \in \mathbb{R}^+$ is the amount of data downloaded during interval i using option o, measured in bits, $\forall i \in I, o \in O_i$.
- $e_i \in \mathbb{R}^+$ is the amount of energy available at the beginning of interval i, measured in Joules, $\forall i \in I$.
- $d_i \in \mathbb{R}^+$ is the amount of data available at the beginning of interval i, measured in bits, $\forall i \in I$.
- $h_i^e \geq 0$ is the amount of excess energy spilled throughout interval i, measured in Joules, $\forall i \in I$.
- $h_i^d \geq 0$ is the amount of excess data spilled throughout interval i, measured in bits, $\forall i \in I$.

4.2. Under-constrained formulation

In the under-constrained formulation (*UCF*), we ignore the energy and data dynamics over the duration of the interval, only checking that constraints are satisfied at the start and end points. The formulation is as follows:

$$\max \sum_{i \in I} \sum_{o \in O_i} \eta_{io} q_{io} \tag{4.1}$$

s.t.

$$\sum_{o \in O_i} x_{io} \leq 1 \quad \forall i \in I \tag{4.2}$$

$$q_{io} \leq \Delta t_i \phi_{io} x_{io} \quad \forall i \in I,\ o \in O_i \tag{4.3}$$

$$e_0 = e_{start} \tag{4.4}$$

$$e_{min} \leq e_i \leq e_{max} \quad \forall i \in I \tag{4.5}$$

$$e_{i+1} = e_i + \delta_i^{e+} - \delta_i^{e-} - \sum_{o \in O_i} \alpha_{io} q_{io} - h_i^e \quad \forall i \in I \tag{4.6}$$

$$d_0 = d_{start} \tag{4.7}$$

$$d_{min} \leq d_i \leq d_{max} \quad \forall i \in I \tag{4.8}$$

$$d_{i+1} = d_i + \delta_i^{d+} - \delta_i^{d-} - \sum_{o \in O_i} q_{io} - h_i^d \quad \forall i \in I \tag{4.9}$$

$$x_{io} \in \{0,1\} \quad \forall i \in I,\ o \in O_i \tag{4.10}$$

$$q_{io} \in \mathbb{R}^+ \quad \forall i \in I,\ o \in O_i \tag{4.11}$$

The objective, Eq. (4.1), maximizes the total amount of data received over the planning horizon. Constraint (4.2) enforces that the satellite may download using at most one option o during each time interval i. Constraint (4.3) enforces that data can only be downloaded using the chosen option for any given interval, and that the amount of data downloaded is limited by the time of the interval and the chosen data rate. Note that we are defining how much data to download during an interval and therefore, implicitly, how much time of that interval. However, we are not defining when the download occurs, i.e. when the download starts and ends. Constraint (4.4) initializes the amount of energy stored at the start of the planning horizon. Constraint (4.5) ensures that the amount of energy stored at the beginning of each interval is within the battery limits. Constraint (4.6) defines the amount of energy stored at the beginning of an interval to be the amount stored at the start of the preceding interval plus the amount acquired minus the amount consumed for nominal operations minus the amount used to support download minus the amount spilled. Constraint (4.7) initializes the amount of stored data. Constraint (4.8) ensures that the amount of data stored at the beginning of each interval is within the data buffer limits. Constraint (4.9) defines the amount of data stored at the beginning of an interval to be the amount stored at the start of the preceding interval plus the amount acquired minus the amount lost to degradation or expiration minus the amount downloaded minus the amount spilled.

A feasible solution to *UCF* may be infeasible for the real-world problem in which continuous dynamics exist because the energy and data buffer constraints are only imposed at the start and end of each intervals, and are neglected throughout each interval. Consider the single-interval example in Fig. 2 where an interval begins with an empty battery. Assume that there is sufficient data to download at any time during the interval. Furthermore, suppose that for the first half of the interval no energy is acquired, and for the second half of the interval energy is acquired at the rate of 2 J/s. In the *UCF* model, a feasible solution would be to use 1 Joule/s throughout the entire interval, as the total amount of energy acquired is equal to the total amount consumed over the duration of the interval. In the true, continuous-time example, however, this would not be a feasible solution, see stored energy in Fig. 2.

Note that because all feasible solutions to the true problem (i.e. with continuous dynamics) are also feasible for this under-constrained formulation, then if *UCF* is feasible for the true problem, then it is also optimal for the true problem, as proved in the following section.

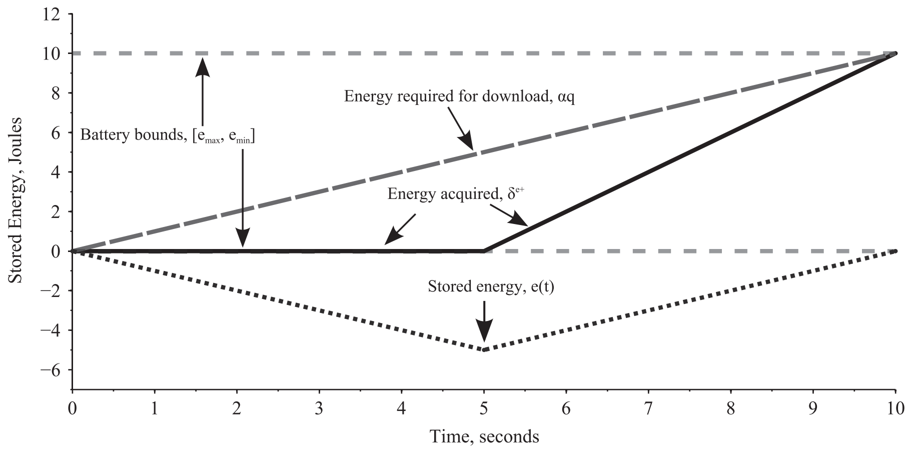

Fig. 2. Single-interval instance of *SMSP* where a feasible solution to *UCF* results in infeasibilities when applied to the continuous-time dynamics.

5. A special case: linear dynamics

The previous section introduced a formulation to optimize instances of *SMSP* with general forms of energy and data dynamics. This section focuses on the special case in which the energy and data dynamics are linear, which has important implications for the optimality of *UCF* solutions. We begin here by presenting a theorem and proof that, for problems with linear dynamics, *UCF* guarantees optimal solutions. In Section 5.1 we use this fact to study the tractability of several real-world instances of *SMSP* with a range of ground station networks and download options. We provide computational results in Section 5.2 for a wide range of instances of *SMSP* to show its general applicability beyond our specific application. We conclude in Section 5.3 with examples of special cases that may lead to computational intractability.

Theorem 1. *Consider an instance of SMSP in which the rates of energy and data acquisition (π^{e+} and π^{d+}), nominal energy consumption (π^{e-}) (i.e. required for nominal operations but not to support downloading data), and data loss (π^{d-}) are each constant with respect to time over the duration of any given interval $i \in I$. For any such instance, an optimal solution to the corresponding instance of UCF will satisfy the continuous-time dynamics (see Sections 3.1 and 3.2) throughout the entire interval duration. This is a sufficient condition to yield a feasible, and thus optimal, solution to SMSP.*

Proof. For simplicity of exposition, we focus here on the energy dynamics; the same logic applies to the data dynamics. Because energy acquisition and consumption are constant with respect to time over the duration of interval i, we replace $\pi^{e+}(t)$ and $\pi^{e-}(t)$ with π_i^{e+} and π_i^{e-}. By Lemma A.1 (see Appendix A) we can convert any *UCF* solution in which data is downloaded for only a portion of a given interval i at a chosen constant rate to an equivalent and feasible solution in which the same amount of data is downloaded at a (lower) constant rate, which we denote by θ_i^e, for the entire duration of interval i. For example, if the solution is to download B bits per second for half the interval, we instead consider downloading $B/2$ bits per second for the full interval. Similarly, for the case where energy is spilled for some fractional portion of interval i, we can assume without loss of generality, by Lemma A.2, that this spillage occurs at a constant rate, which we denote by s_i^e, for the entire duration of interval i. Since the rates at which energy is acquired, nominally consumed, consumed to support download, and spilled are all then constant over the duration of interval i, the net change in energy and data is linear with respect to time

$$e(t) = e_i + [\pi_i^{e+} - \pi_i^{e-} - \theta_i^e - s_i^e](t - t_i) \quad \forall\ i \in I,\ t \in [t_i, t_{i+1}]. \tag{5.1}$$

Thus, the energy at any time $t \in (t_i, t_{i+1})$ will lie on the line segment connecting the two points (t_i, e_i) and (t_{i+1}, e_{i+1}). Because the energy levels at times t_i and t_{i+1} (which are the end points of this line segment) are within the feasible energy range by Constraint (4.5), the energy level at any intermediate point along this line segment will be within this range and thus feasible as well. □

5.1. Real-world computational experiments

Theorem A.1 proves that using *UCF* for a problem with linear dynamics will yield an optimal solution to the underlying continuous-dynamics problem. In this section, we analyze the performance of this formulation, solved using standard branch-and-bound techniques, on real-world problem instances.

The computational experiments in this section are based on a real-world satellite mission studying space weather. Specifically, we model the Radio Aurora Explorer (RAX), the first NSF-funded CubeSat (a type of miniaturized satellite) [35]. RAX orbits the Earth every 97 min, with ground stations periodically coming in and out of view. RAX collects energy whenever it is in view of the sun. It collects scientific data whenever it is in view of its target of interest. This data collection occurs for short periods of time (≤ 5 min) at high data rates. These dynamics can be modeled by piece-wise linear functions. Thus, in accordance with *Theorem* 1, optimality is guaranteed when using *UCF*.

We analyze RAX in the context of three diverse and realistic networks, summarized in Table 1 [36]. N1 consists uniquely of RAX's primary ground station, located in Ann Arbor, MI. This ground station has a single download option. N2 is the ground station network primarily used by RAX. It consists of seven independently owned and operated stations around the world. Each station has a single download option; downloading characteristics vary from station to station. N3 is composed of 20 amateur radio ground stations with which RAX could potentially communicate. Each of these stations has three download options with unique characteristics.

For each of the three ground station networks, we evaluated several different planning horizons ranging from a few hours to 2 months. [Note that, in practice, the general planning horizon for RAX is on the order of days.] The computations were performed on an Intel Core i7 2.8 GHz processor with 8 GB of memory using the IBM ILOG Optimization Studio (CPLEX) 12.1 C++ API software package [37].

The problem size is given in Fig. 3, and results are shown in Fig. 4. All problem instances solved quickly (i.e. in a matter of seconds), even those with planning horizons significantly longer than that required in practice. N3 did demonstrate significantly slower results relative to N1 and N2. This is not surprising, however, given that N3 not only has more ground stations, but each of these stations has multiple download options, thereby increasing both the size and the complexity of the associated MIP. Even with the choice of multiple options per station, however, we observed no branching for any of the instances tested. This fact, in combination with the relatively small size of the MIPs, explains the fast run times.

To gain a more general understanding of our proposed formulation, we consider a collection of larger and more complex problem instances in Section 5.2, no longer limiting ourselves strictly to the real-world RAX applications. This enables us to test the boundaries of tractability for a more general class of problem instances. We then focus in Section 5.3 on exploring the integrality of the proposed formulation. In particular, we demonstrate problem instances for which significant branching (and thus slower run times) will occur.

5.2. *General case computational experiments*

In the previous subsection, we observed that several real-world satellite communication scheduling problems could be solved via *UCF*. Provably optimal solutions were found in under 1 min for all tested instances, including several instances with time horizons much longer than would be found in real-world practice. In this subsection, we consider whether these computational results are specific to the underlying RAX problem structure, or would be experienced in a broader class of problem instances as well.

For our broader computational experiments, we consider problem instances with parameters drawn from the distributions in Tables 2 and 3. Table 2 includes the distributions that are constant for all tests and Table 3 shows the distributions for 16 different test cases. Because all the parameters must be non-zero, if a sampled parameter drawn from a normal distribution is negative, it is rounded to zero. Similarly, if the efficiency parameter, η, exceeds its upper bound of one, it is rounded to one. Without loss of generality, we assume that no energy or data is nominally lost during each interval, i.e. $\delta_i^{e-}=0$ and $\delta_i^{d-}=0$. For each test case, we generated and solved 100 instances, sampling from the distributions in Tables 2 and 3.

Table 1
Parameters for sample ground station (GS) networks.

Network	Description	Number of GSs	Number of options at each GS
N1	Single Station: Ann Arbor, MI	1	1
N2	RAX Network	7	1
N3	Global Amateur Radio Network	20	3

Computations were performed using an Intel Core i7 2.8 GHz processor with 8 GB of memory using CPLEX 12.2 C++ API with an optimality gap of 0.01%. The statistics of the computational results appear in Fig. 5.

Fig. 5 shows that, similar to the RAX application, all the instances are solved very quickly – typically on the order of several seconds. Unlike the RAX application, where we observed no branching, there is branching observed in some of these tests. Most cases have a mean number of branch-and-bound nodes that is non-zero (often about 500 nodes). However, the minimum number of nodes is zero for all test cases, as shown in Fig. 5, which indicates that at least one of the 100 instances for each test case have no branching. Although there is not a perfect correlation between the number of nodes and the run time, there is, as expected, a fairly strong correlation between them.

Next, we conducted tests specifically targeted at studying the impact of the number of problem intervals and the number of download options, and therefore the size of the *MIP*, on computational performance. These tests were performed with the parameter distributions in Table 2 and parameter distributions for test case 0 in Table 3. The results for 100 randomly generated instances with variations in the number of intervals and options are shown in Fig. 6(a) and (b), respectively.

In Fig. 6(a), we observe that the run time increases, as expected, with the number of intervals. In our example, an instance with 1000 intervals has ≈ 4000 binary decision variables and ≈ 9000 constraints and an instance with 10,000 intervals has $\approx 40{,}000$ binary decision variables and $\approx 90{,}000$ constraints. The lack of apparent exponential growth in solve time with significantly larger problems indicates that there is a limited amount of branching. We hypothesize that this is because the impact of decisions in one time interval has limited impact on decisions several intervals into the future and thus the coupling between intervals is fairly loose except for those intervals very close together in time.

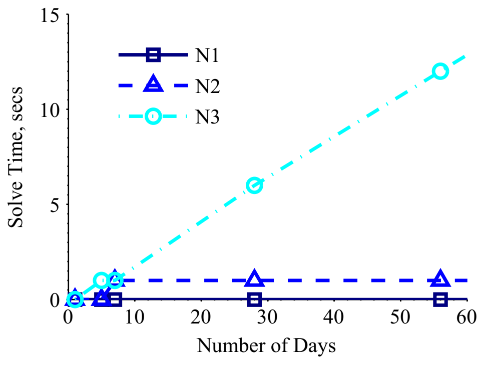

Fig. 4. Solve time as a function of planning horizon.

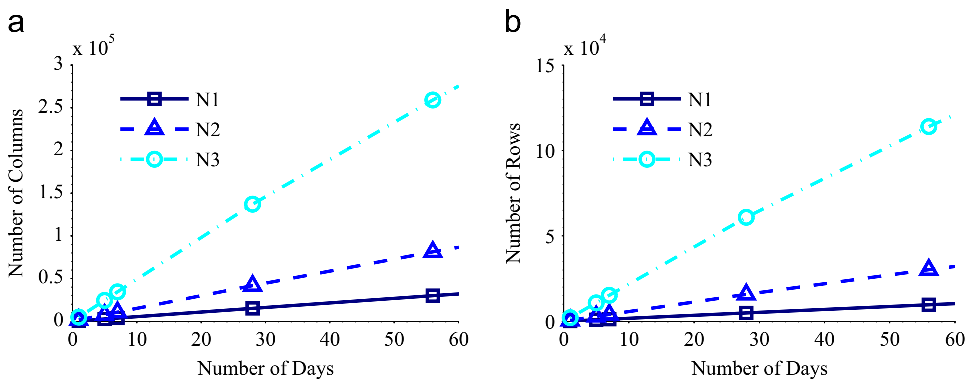

Fig. 3. Problem size for results in Fig. 4. (a) Number of columns (decision variables). (b) Number of rows (constraints).

Table 2
Parameters that are constant for all test cases 0–15, where $N(\mu, \sigma)$ denotes a normal distribution with mean μ and variance σ, and $P(X)$ denotes the probability of achieving outcome X.

Parameter	Description	Distribution
I	Number of intervals	1000
t_i	Duration of interval i	$\approx N(10, 4)$
$\|O_i\|$	Number of options for interval i	$P(3)=P(5)=0.5$
e_{min}, e_{start}	Energy minimum and starting levels	0, 0
d_{min}, d_{start}	Data minimum and starting levels	0, 0
η_{io}	Efficiency of options	$\approx N(1, 0.5)$

Table 3
Parameter distributions for test cases 0–15.

Parameter description	Maximum energy buffer	Energy collection	Maximum data buffer	Data collection	Energy utilization	Data rate
Test	e_{max}	δ^{e+}	d_{max}	δ^{d+}	α_{io}	ϕ_{io}
0	100	$\approx N$ (80,40)	100	$\approx N$ (50,25)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
1	300	$\approx N$ (80,40)	300	$\approx N$ (50,25)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
2	50	$\approx N$ (80,40)	50	$\approx N$ (50,25)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
3	100	$\approx N$ (80,40)	100	$\approx N$ (50,25)	$\approx N$ (5,2.5)	$\approx N$ (4,2)
4	100	$\approx N$ (80,40)	100	$\approx N$ (50,25)	$\approx N(10,5)$	$\approx N$ (2,1)
5	100	$\approx N$ (80,40)	100	$\approx N$ (50,25)	$\approx N(10,5)$	$\approx N$ (4,2)
6	100	$\approx N$ (80,40)	100	$\approx N$ (50,25)	$\approx N$ (2.5,1.25)	$\approx N$ (2,1)
7	100	$\approx N$ (80,40)	100	$\approx N$ (50,25)	$\approx N$ (5,2.5)	$\approx N$ (1,.5)
8	100	$\approx N$ (80,40)	100	$\approx N$ (50,25)	$\approx N$ (2.5,1.25)	$\approx N$ (1,.5)
9	100	$\approx N$ (40,20)	100	$\approx N$ (50,25)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
10	100	$\approx N$ (80,40)	100	$\approx N$ (25,12.5)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
11	100	$\approx N$ (40,20)	100	$\approx N$ (25,12.5)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
12	100	$\approx N$ (160,80)	100	$\approx N$ (50,25)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
13	100	$\approx N$ (80,40)	100	$\approx N$ (100,50)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
14	100	$\approx N$ (160,80)	100	$\approx N$ (100,50)	$\approx N$ (5,2.5)	$\approx N$ (2,1)
15	100	$\approx N$ (40,20)	100	$\approx N$ (25,12.5)	$\approx N(10,5)$	$\approx N$ (2,1)

In Fig. 6(b), we allow the number of options per interval to increase from one to 100 (recall that in the first set of experiments, each interval had either three or five download options from which to choose) for instances with 1000 intervals. In our example, an instance with 10 options per interval has $\approx$10,000 binary decision variables and $\approx$15,000 constraints and an instance with 100 options has $\approx$40,000 binary decision variables and $\approx$105,000 constraints. Despite a linear growth in the number of binary decision variables with increases in the number of options, the problem may become more complex as there are multiple competing options during every download opportunity. However, we continue to see run times remaining well within the range of tens of seconds and the solve time does not grow significantly as the number of options increases as there is limited branching. We suggest that although, in theory, there are many competing options at each interval that could lead to fractionality and associated branching, in practice as the number of options grows the likelihood of certain options being dominated by other options grows as well. Thus, the incentive for fractional solutions does not increase as quickly as the number of options. In the next session, we focus on identifying cases where significant branching *does occur*.

The experiments in this section indicate that a broad class of problems can be solved quickly using *UCF*. An important observation from these tests is that branching does occur in some instances. This confirms that integral optimal solutions, as observed in the RAX-specific examples, are not characteristic of the *UCF* problem structure, but are rather a consequence of the input data. All tests in this section have had fast run times, which is largely driven by a lack of significant branching, motivating the following section in which we seek instances where significant branching *does* occur.

5.3. *Non-integral solutions resulting in branching*

In this section we investigate conditions under which branching occurs, and the implications of this for tractability. In Section 5.3.1 we provide a demonstrative single-interval numeric example where branching occurs. We present theoretical conditions for non-integral optimal solutions to the *LP* relaxation of *UCF* for single-interval instances in Section 5.3.2 and extend this logic to instances with multiple intervals in Section 5.3.3. Finally, in Section 5.3.4 we perform computational tests on larger problem instances with these conditions.

5.3.1. *Single interval UCF non-integral solution example*

For simplicity, we consider a single interval of *UCF*, so there is a single decision: what download option to use. If one option is more desirable in all aspects (i.e. most efficient, largest data-to-

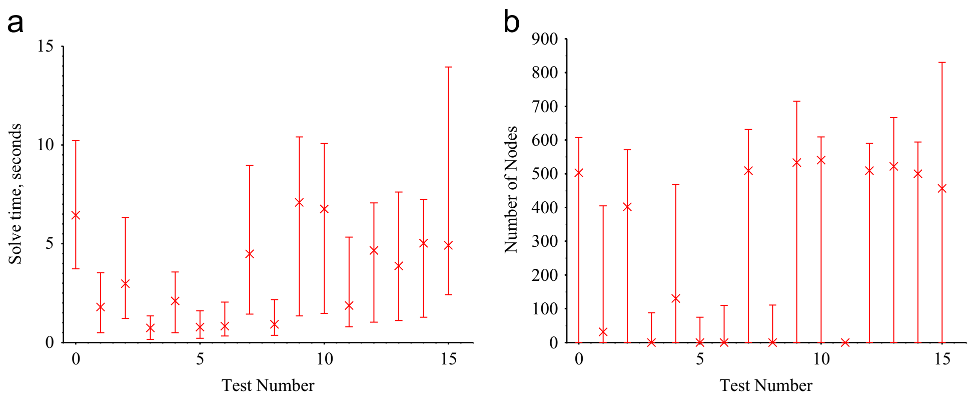

Fig. 5. Computational statistics for test cases in Table 3 with an integrality gap of 10^{-8}. The x represents the mean, and the upper and lower horizontal lines show the maximum and minimum values. (a) Solve time. (b) Number of Branch-and-Bound Tree Nodes.

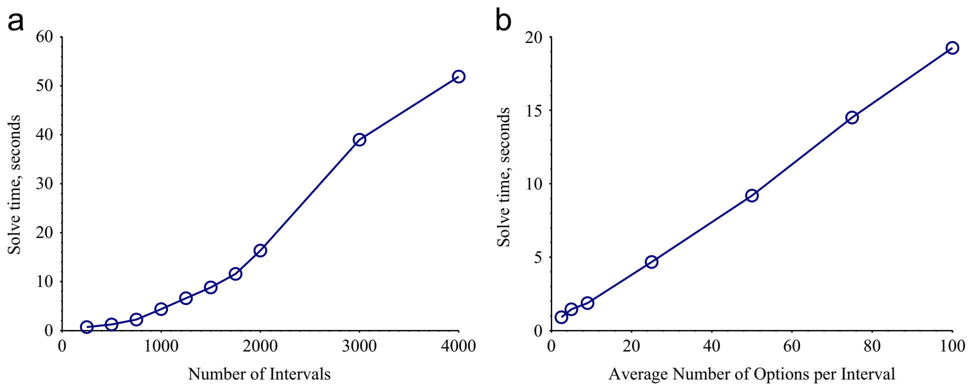

Fig. 6. The effects of the number of intervals and average number of options per interval on solve time (where circles represent a single test). (a) Effect of intervals. (b) Effect of options.

time ratio, and smallest energy-to-data ratio) compared to other available options, then the dominant option will be used exclusively, and the linear programming *(LP)* relaxation will yield an optimal integer solution. In this section, we investigate the case when there is no dominant solution and thus fractionality could potentially occur in an optimal solution to the *LP* relaxation.

Consider the two-option single-interval instance of *SMSP* in Table 4. The two feasible integer solutions and the optimal solution to the *LP* relaxation, which is non-integer, are given in Table 5.

Note that in this single-interval example, there is no advantage in reserving resources (energy, or data) for future intervals. Solution 1 exhausts the available energy by downloading at a high data rate, leaving behind unused time and data. Solution 2 exhausts the available time by downloading at a more energy-efficient rate, leaving behind unused energy and data. By taking a convex combination of these two options, Solution 3 is able to more effectively trade off time and energy to download a larger amount of data in a fractional solution.

5.3.2. *Single interval UCF conditions for non-integral solutions*

Next we generalize this example to develop a set of sufficient conditions such that the optimal solution to the *LP* relaxation of *UCF* is non-integer. The constraints for the *LP* relaxation of *UCF* for a two-option single-interval scenario are explicitly enumerated in Eqs. (5.2)–(5.11), where Δe and Δd represent the total amount of available energy and data at the start of the interval, respectively

$$\max \quad \{\eta_i q_1 + \eta_2 q_2\} \tag{5.2}$$

s.t.

$$q_1 \le t\phi_1 x_1 \tag{5.3}$$

$$q_2 \le t\phi_2 x_2 \tag{5.4}$$

$$\alpha_1 q_1 + \alpha_2 q_2 \le \Delta e \tag{5.5}$$

$$q_1 + q_2 \le \Delta d \tag{5.6}$$

$$x_1 + x_2 = 1 \tag{5.7}$$

$$q_1 \ge 0 \tag{5.8}$$

$$q_2 \ge 0 \tag{5.9}$$

$$x_1 \ge 0 \tag{5.10}$$

$$x_2 \ge 0 \tag{5.11}$$

To simplify the analysis of non-integer solutions, let the available data, Δd, be sufficiently large such that Constraint (5.6) is never binding. [Note that an analogous analysis can be conducted assuming that the energy level, Δe, is sufficiently large such that Constraint (5.5) is never binding.] In addition, for simplicity (and without loss of generality) assume that the efficiency of each option is equal to one. Finally, motivated by the observations of the relative option characteristics of our non-integer example, assume that Option 1 alone depletes time while leaving excess energy unused (i.e. if $x_1 = 1$, Constraint (5.3) will be binding and Constraint (5.5) will not be binding), and Option 2 alone depletes energy, leaving excess time (i.e. if $x_2 = 1$, Constraint (5.4) will not be binding and Constraint (5.5) will be binding). This set of conditions can be summarized as

$$\alpha_1\phi_1 < \frac{e}{t} < \alpha_2\phi_2. \tag{5.12}$$

Eqs. (5.2)–(5.11) form a linear program with four decision variables, and thus a basic solution will have four binding constraints. Given the conditions in Eq. (5.12), the optimal solution to the problem may yield three possible outcomes, which are summarized in Table 6 and developed in more detail in Appendix B. Cases 1 and 2 each consider when a single option is used, and their optimal solutions are computed based on Constraints (5.5) or (5.3) being active. In Case 3, both Options 1 and 2 are used, thus Constraints (5.6) and (5.8)–(5.11) are not active. As a result, the remaining four constraints (5.3), (5.4), (5.5) and (5.7) are active. The conditions for this case are $\phi_2 > \phi_1$, $\alpha_2 > \alpha_1$, see Appendix B for the procedure for computing these conditions.

The conditions for Case 3 are sufficient such that the optimal solution to the *LP* relaxation is non-integer. Note that branching may be caused by other conditions. For example, when energy is never a constraining resource, a similar argument can be made to

Table 4
Two-option example where LP relaxation yields a fractional optimal solution.

Interval parameters	**Option 1**	**Option 2**
$t=6$ s	$\phi_1=2$ bits/s	$\phi_2=3$ bits/s
$e=36$ J	$\alpha_1=2$ J/bit	$\alpha_2=4$ J/bit
$d=14$ bits	$\eta_1=1$	$\eta_2=1$

Table 5
Feasible solutions to the *MIP* and the optimal *LP* relaxation.

Solution	x_1	x_2	q_1 (bits)	q_2 (bits)	*Optimal solution* (q_1+q_2) (bits)	*Excess energy* (J)	*Excess data* (bits)	*Excess time* (s)
1	1	0	12	0	12	0 J	5	3
2	0	1	0	9	9	12 J	2	0
3	0.75	0.25	9.5	4	**13.5**	0 J	0.5	0

Table 6
Fractionality conditions for the single interval *UCF* with the conditions in Eq. (5.12).

Case	Conditions	x_1	x_2	q_1	q_2	**Optimal solution** (q_1+q_2)
1	$\alpha_2 \ge \alpha_1$ $\alpha_2\phi_1 \ge \frac{e}{t}$	1	0	$t\phi_1$	0	$t\phi_1$
2	$\phi_2 \ge \phi_1$ $\alpha_2\phi_1 \le \frac{e}{t}$	0	$\frac{e}{\alpha_2\phi_2 t}$	0	$\frac{e}{\alpha_2}$	$\frac{e}{\alpha_2}$
3	$\alpha_1 < \alpha_2$ $\phi_1 < \phi_2$	$\frac{\alpha_2\phi_2 - \frac{\Delta e}{t}}{\alpha_2\phi_2 - \alpha_1\phi_1}$	$\frac{\frac{\Delta e}{t} - \alpha_1\phi_1}{\alpha_2\phi_2 - \alpha_1\phi_1}$	$\frac{t\phi_1\left(\alpha_2\phi_2 - \frac{\Delta e}{t}\right)}{\alpha_2\phi_2 - \alpha_1\phi_1}$	$\frac{t\phi_2\left(\frac{\Delta e}{t} - \alpha_1\phi_1\right)}{\alpha_2\phi_2 - \alpha_1\phi_1}$	$\frac{t\phi_1\phi_2(\alpha_2-\alpha_1)+\Delta e(\phi_2-\phi_1)}{\alpha_2\phi_2-\alpha_1\phi_1}$

show that the *LP* relaxation will yield fractional solution under the conditions $\phi_1 t < \Delta d < \phi_2 t$ and $\eta_2 < \eta_1$.

5.3.3. *Fractionality with more than two options*

We extend the logic for the single-interval two-option case to consider cases with multiple options. The *LP* relaxation of a single-interval instance of *UCF* with three options will have six variables, thus an optimal basic feasible solution will have six active constraints. A basic feasible solution that utilizes all three download options (and thus is fractional) will have non-binding non-negativity constraints on all variables, i.e. $x_1 > 0, x_2 > 0, x_3 > 0, q_1 > 0, q_2 > 0, q > 3$. Thus, the six remaining equations must be binding, as in the following system:

$$\begin{cases} q_1 = t\phi_1 x_1 \\ q_2 = t\phi_2 x_2 \\ q_3 = t\phi_3 x_3 \\ \alpha_1 q_1 + \alpha_2 q_2 + \alpha_3 q_3 = \Delta e \\ q_1 + q_2 + q_3 = \Delta d \\ x_1 + x_2 + x_3 = 1 \end{cases} \tag{5.13}$$

Since the system contains six variables and six equations, there exists a unique solution corresponding to this basis, $[q_1^*, q_2^*, q_3^*, x_1^*, x_2^*, x_3^*]$. [Note that there may exist other basic feasible solutions with the same objective function value that use only one or two download options.]

We observe that it is *not* possible to have an optimal basic feasible solution for the single interval case that uses more than three options, as stated and proven in the following theorem.

Theorem. *For any single-interval instance of UCF, an optimal solution to the LP relaxation will at maximum have three download options.*

Proof. For a single-interval instance of *UCF* with $|O|$ options, the number of variables is $n = 2|O|$ and the number of constraints is $m = 3 + 3|O|$. A solution that utilizes all $|O|$ options requires that all $2|O|$ non-negativity constraints are non-active. A basic solution requires that $2|O|$ of the remaining constraints (i.e. $3 + |O|$) must be active, therefore $(|O|+3) \ge 2|O|$. The inequality holds only for $|O| = 0, 1, 2, 3$. If $|O| > 3$, then there are not enough remaining constraints to form a basis once the non-negativity constraints have been removed from consideration. □

5.3.4. *Computational experiments*

While all the instances of *SMSP* that we have considered thus far solve very quickly, contriving data instances that branch extensively and thus solve slowly is possible (which may be relevant for other applications). Thus, we investigate the number of branch-and-bound nodes and solve times for different sizes of contrived cases that capture the dynamics discussed in Section 5.3.3.

Results are shown in Fig. 7 for instances with up to 500 intervals to emphasize the trends for smaller instances. Large data sets, with up to 2500 intervals can be solved in less than 3 min with an optimality gap of 1%, see Table 7. However, instances with more than 150 intervals with an optimality gap of 0.01% and with more than 300 intervals with an optimality gap of 0.1% are intractable, and thus do not appear in Fig. 7 and Table 7. In these cases, the branch-and-bound tree becomes too large to store in allocated computer memory.

Fig. 7 and Table 7 demonstrate that a tighter optimality gap increases the amount of branching and hence the run time. This is not surprising, as the problem instances were designed to have fractional solutions. On the other hand, the *LP* relaxation is quite weak in these problem instances, and the added run time serves only to tighten the upper bound, with little impact on the objective value.

6. Applications to non-linear dynamics

In Theorem 1, we proved that a solution to *UCF* for an instance of *SMSP* in which the energy and data dynamics remain constant over the duration of any given interval will always yield solutions that are feasible, and thus optimal, for the original problem. When the dynamics are non-linear, i.e. the rates of energy and data acquisition and consumption are not constant during an interval, a solution to *UCF* may be infeasible when applied to the true, continuous-time problem. This occurs because *UCF* only considers the cumulative change in dynamics at the start and end of each interval and neglects the continuous-time dynamics that occur within the interval, as seen in the example in Fig. 2.

The infeasibilities that result from applying *UCF* solutions to instances with non-linear dynamics can be overcome by discretizing the continuous-time problem into sufficiently small intervals. As the interval sizes approach zero, *UCF* is guaranteed to yield a feasible, and thus optimal, solution by *Theorem* 1. The resulting *MIP*, however, may be computationally intractable.

In practice, relatively long time intervals may be sufficient to ensure a valid solution. Specifically, so long as the solution defined by the boundary conditions of each interval also satisfies the dynamics within each interval, the solution will be valid. A generic approach for accomplishing this is described in Section 6.1. We provide specific details for applying this approach to problem instances with piece-wise linear dynamics (or that can be approximated with piece-wise linear dynamics) and discuss computational implications in Section 6.2.

6.1. *Algorithm for solving non-linear SMSPs*

In this section we introduce the *Non-Linear SMSP Algorithm (NLSA)*, an approach for solving non-linear instances of *SMSP*. This algorithm can theoretically be applied to solve any instance of *SMSP*. The tractability of the approach depends on the effort required to identify intervals with infeasibilities and the degree of discretization required to achieve feasibility.

NLSA iteratively identifies intervals in a *UCF* solution that are infeasible and further discretizes these intervals until feasibility, and thus optimality is achieved, as seen in the pseudo-code below. Specifically, let I^k be the set of intervals at the start of iteration k. At the start of *NLSA*, I^0 is the set of original intervals, where an *original interval* starts and ends when and only when there is a change in the set of download opportunities, i.e. a ground station comes in or out of view, see Fig. 1. At every iteration k, *NLSA* solves an instance of *UCF* with I^k, and then checks the feasibility of each interval over which the solution to the discretized problem is applied to the true, continuous-time dynamics.

If the solution is feasible for all intervals, then it is optimal and the algorithm terminates. If any infeasibilities are found, then one or more infeasible intervals are split into *sub-intervals*. Constraints are then imposed to ensure that all resulting sub-intervals within an original interval use the same download option, i.e. x_i^k (the download decision at iteration k for interval i) must be equivalent for all i within a common original interval. The process repeats until a valid solution is obtained.

Algorithm 1. Non-Linear SMSP Algorithm (NLSA).

```
1. Initialize $k=0$ and $I^0$ to the set of original intervals
2. Solve UCF with $I^k$
3. Check feasibility of solution with respective to each sub-
   interval in $I_k$ when applied to the true continuous-time
   dynamics
• if all sub-intervals in $I_k$ are feasible
   – Terminate the algorithm because the solution is feasible
     and thus optimal
• else
   – Update $k \rightarrow k+1$
   – Split infeasible intervals, update $I^k \rightarrow I^{k+1}$
   – Impose constraints that all $x_i^k$ must be equivalent for all $i$
     within a common original interval
   – Return to Step 2
```

The key design issues in the implementation of *NLSA* are the approaches for checking the feasibility of a given solution with respect to an individual sub-interval and splitting infeasible sub-intervals, which are addressed next in the context of the specific case of instances with piece-wise linear dynamics.

6.2. *Special case: piece-wise linear dynamics*

In this section, we demonstrate the applicability of *NLSA* to special cases of *SMSP* where the dynamics are piece-wise linear (PWL). These are instances where the time horizon can be broken into a discrete number of *linear segments*, which are time periods over which the energy and data rates are constant. See Fig. 8 for an example instance with PWL energy and data dynamics.

A PWL instance is composed of a finite number of intervals with linear dynamics. Thus, in the worst case, *NLSA* will terminate after splitting the original intervals into the set of sub-intervals with linear dynamics. As a result, *NLSA* will converge in a finite number of iterations by *Theorem* 1.

6.2.1. *Assessing the feasibility of an interval*

A key design issue for the implementation of *NLSA* is the approach for testing the feasibility of a sub-interval k for a given solution to *UCF*. That is, for a given iteration i, we must test that

Table 7
Performance of worst-case instance of *UCF* under various optimality gaps.

Optimality gaps	1.000%			0.100%			0.010%		
Number of intervals	Time	Number of nodes	Obj. val. (bits)	Time	Number of nodes	Obj. val. (bits)	Time	Number of nodes	Obj. val. (bits)
25	0.4 s	135	336	0.4 s	208	336	0.6 s	350	336
50	0.2 s	57	669	2.1 s	3406	672	2.3 s	5321	672
75	0.3 s	84	1005	2.7 s	3244	1011	2.9 s	3166	1011
100	0.4 s	133	1338	7.8 s	20,471	1350	6.4 s	11,115	1350
125	0.8 s	520	1671	5.3 s	2393	1686	16.8 s	30,341	1686
150	1.1 s	492	2010	2.0 min	407,241	2022	1.7 h	2,122,780	2022
175	1.1 s	533	2340	8.0 min	866,052	2361	–	–	–
200	1.2 s	484	2679	9.2 min	977,586	2700	–	–	–
300	1.9 s	483	4011	9.9 min	1,973,530	4047	–	–	–
400	7.7 s	523	5352	–	–	–	–	–	–
500	8.0 s	490	6684	–	–	–	–	–	–
2500	3.0 min	29,682	33,441	–	–	–	–	–	–

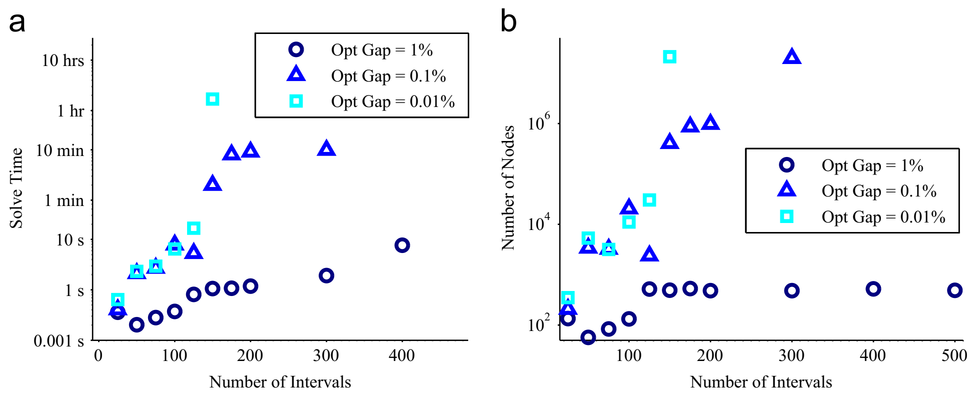

Fig. 7. Performance of worst-case instances of *UCF* under various optimality gaps. (a) Effect on solve time. (b) Effect on number of nodes.

the download amount q_i^* specified by the *UCF* solution is valid relative to the continuous dynamics of each sub-interval. A *UCF* solution is guaranteed to satisfy the system dynamics at the beginning and end of the interval, but not necessarily throughout the interval.

In the case where the time required to download q_i^* is less than the full duration of the interval, *UCF* does not specify when within the interval the download should take place (note that a feasible download may include multiple starts and stops). In this section, we present a constructive algorithm to determine when to download within an interval and, in the process, assess the feasibility of a given interval with respect to a given *UCF* solution.

Consider the question, for a given sub-interval and a given corresponding "optimal" amount of data to download, of when to download within the interval. In general, a greedy approach to downloading seems advantageous: we should download whenever possible, starting from the beginning of the interval. Conversely, delaying download until later in the interval can have the negative impacts of (a) causing energy and/or data to be spilled that will be required for downloading or nominal operations at a future time; and (b) running out of time within the interval before completely downloading q_i^*. Such a purely greedy approach fails, however, when energy and/or data must be reserved to meet the nominal operating conditions of the satellite later during the interval.

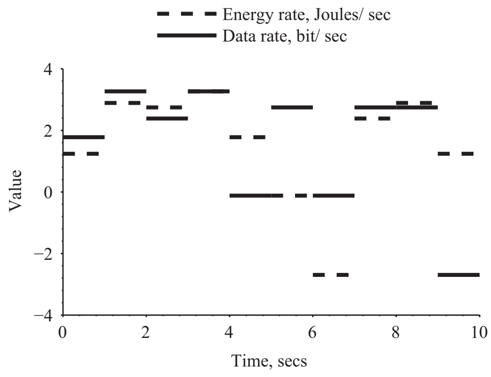

Fig. 8. Example instance with PWL dynamics.

Fig. 9 demonstrates a simple case where a greedy approach yields infeasibility, while other feasible download schedules exist. Fig. 9(a) depicts the energy available for downloading and nominal operations. Specifically, the interval starts with 1 J of energy in storage and there is no net change in energy over the first linear segment (0–1 min). In the next linear segment (1–2 min), 1 J per minute is consumed for nominal operations. In the third linear segment (2–3 min), 1 J per minute is acquired by solar energy.

Suppose that the *UCF* solution for the example in Fig. 9 is to download for 1 min over this interval, at a corresponding energy utilization rate of 1 J per minute. Assume that there is sufficient data available at the start of the interval for the download and no nominal data consumption throughout the interval such that the download can occur feasibly at any time during the interval. In Fig. 9(b), we show the energy level, assuming that the download occurs during the first linear segment. By using all the available energy during the first linear segment, there is no energy available in the second linear segment for nominal operations, leading to infeasibility. On the other hand, a feasible download schedule can be achieved if we wait until the third linear segment to download, as seen in Fig. 9(c). In this case, the initial energy can be used for nominal operations during the second linear segment, and the energy acquired during the third linear segment supports the download during this segment.

We propose an *Anticipative Greedy Assign and Check Algorithm (AGACA)* as an approach to check the feasibility of a *UCF* solution, $\{x_{io}^*, q_{io}^*, e_i^*, e_{i+1}^*, d_i^*, d_{i+1}^*\}$, throughout the duration of each interval i. Note that ϕ_i^* and α_i^* represent the energy and data utilization rates of the *UCF* solution. We begin by dividing the interval into its linear segments, starting a new segment whenever the energy or data rate changes. For example, Fig. 8 depicts 10 linear segments and Fig. 9 depicts three linear segments. Let J_i denote the set of linear segments for interval i, where $t_{i,j}$ denotes the start of linear

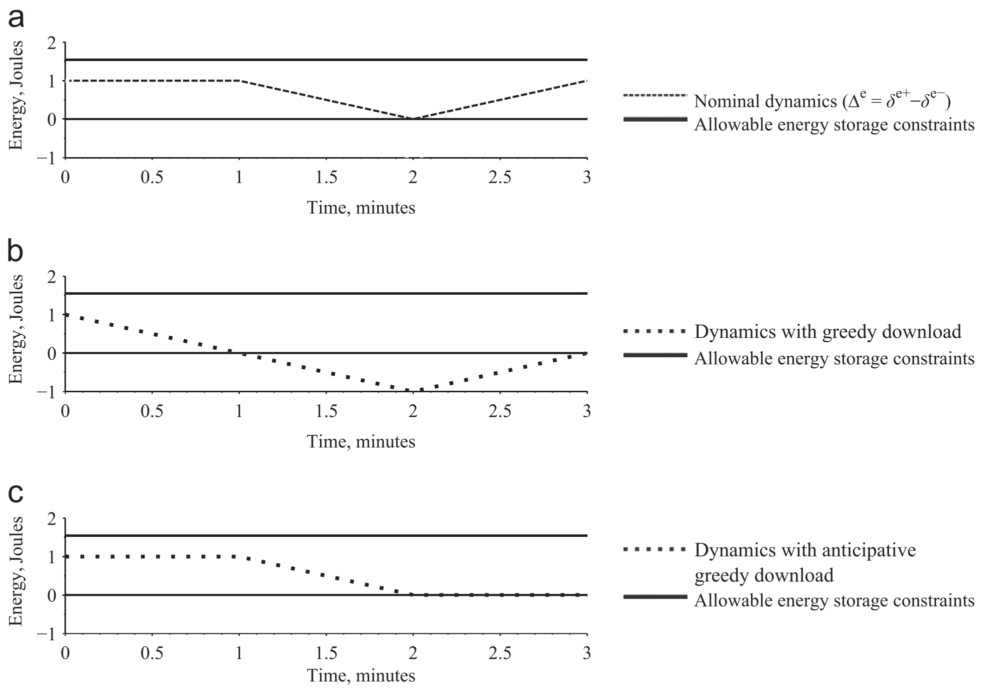

Fig. 9. Single interval instance of *SMSP* with piece-wise linear dynamics where a purely greedy download approach (downloading during the first segment) results in infeasibilities but an anticipative greedy approach (downloading during the third segment) is feasible. (a) Nominal energy dynamics. (b) Dynamics with greedy download (infeasible). (c) Dynamics with anticipative greedy download (feasible).

segment j and $t_{i,j+1}$ denotes the end of segment j, which corresponds to the start of linear segment $j+1$. Analogously, the subscripts for the stored energy and data, $d_{i,j}$ and $e_{i,j}$, denote the state levels during interval i at the start of segment j. Let m_i be the number of linear segments during interval i, i.e. $m_i = |J_i|$.

By *Theorem* 1, we know that for an interval with constant energy and data rates (in our case, this is defined as a linear segment), the system dynamics will be feasible throughout the duration of the interval so long as the energy/data levels are feasible at the beginning and end of the interval. Thus, our approach in *AGACA* is to move sequentially through each linear segment, determining how much data that can be feasibly downloaded during the segment while reserving adequate energy and data for future linear segments within the current interval.

We begin *AGACA* for a given interval i with e_i^* energy and d_i^* data. For each successive linear segment, we first determine the amount of data that can feasibly be downloaded during the jth linear segment of interval i, defined as Ω_{ij}. Ω_{ij} is the minimum of four terms, $\Omega_{ij} = \min(\beta_{ij1}, \beta_{ij2}, \beta_{ij3}, \beta_{ij4})$, which are defined by

$$1.\quad \beta_{ij1} = q_i^* - \sum_{k=1}^{j-1} \Omega_{ik}, \tag{6.1}$$

the remaining data to completely download q_i^*

$$2.\quad \beta_{ij2} = \phi_i^* \cdot (t_{i,j+1} - t_{i,j}), \tag{6.2}$$

the maximum data that can feasibly be downloaded given the duration of linear segment j

$$3.\quad \beta_{ij3} = \frac{1}{\alpha_i^*}\left(e_{ij} - e_{\min} + \min\left[0, \min_{p \in \{j,j+1,\ldots,m_i\}} \left\{\sum_{k=j}^{p} \Delta_{ik}^e\right\}\right]\right), \tag{6.3}$$

the maximum data that can feasibly be downloaded with the energy available at the start of the segment, while reserving enough energy to meet the energy demands of future segments with a net energy loss. Specifically, e_{ij} is the amount of energy available at the beginning of the jth segment of interval i and Δ_{ij}^e is defined to be $\delta_{ij}^{e+} - \delta_{ij}^{e-}$. If $\Delta_{ij}^e < 0$, there is a net loss of energy over segment j; sufficient energy must be reserved prior to this segment to ensure feasibility. Likewise, if for any consecutive sequence of future segments the net energy change is negative, that quantity must be reserved in the current segment

$$4.\quad \beta_{ij4} = d_{ij} - d_{\min} + \min\left[0, \min_{p \in \{j,j+1,\ldots,m_i\}} \left\{\sum_{k=j}^{p} \Delta_{ij}^d\right\}\right], \tag{6.4}$$

the maximum data that can feasibly be downloaded to ensure all future data constraints are satisfied, analogous to the energy constraint described above.

If the first term dominates, i.e. $\Omega_{ij} = \beta_{ij1}$, then we have downloaded the total amount of data in the solution, q_i^* and we terminate *AGACA* with a certificate of feasibility for the interval and a corresponding feasible download schedule.

Otherwise, if $\Omega_{ij} < \beta_{ij1}$, we continue to linear segment $j+1$. Before doing so, we update the amount of stored energy and data at the end of linear segment j

$$e_{i,j+1} = \min\{e_{max}, e_{i,j+1} + \Delta_{ij}^e - \alpha_i^* \Omega_{ij}\}, \tag{6.5}$$

$$d_{i,j+1} = \min\{d_{max}, d_{i,j+1} + \Delta_{ij}^d - \Omega_{ij}\}. \tag{6.6}$$

By construction, the energy and data levels will never be below zero, but to remain below e_{max} and d_{max}, it may be necessary to spill, as enforced in Eqs. (6.5) and (6.6).

At this point, we also check that we have enough remaining time, energy, and data to ensure feasibility during the remainder of the interval. In particular, Eqs. (6.7)–(6.9) must be satisfied, which enforce that there must be sufficient time, energy, and data to completely download the remainder of q_i^* during the remaining time in the interval, $t_{i+1} - t_{i,j+1}$

$$t_{i+1} - t_{i,j+1} \geq \frac{\beta_{ij1} - \Omega_{ij}}{\phi_{io}*} \tag{6.7}$$

$$e_{i,j+1} - e_{i+1} + \sum_{k=j+1}^{m_i} \Delta_k^e \geq \alpha_i^* (\beta_{ij1} - \Omega_{ij}) \tag{6.8}$$

$$d_{i,j+1} - d_{i+1} + \sum_{k=j+1}^{m_i} \Delta_k^d \geq \beta_{ij1} - \Omega_{ij} \tag{6.9}$$

If these three conditions are satisfied, we repeat the above process for the next linear segment. Otherwise, we exit *AGACA* with a certificate of infeasibility in Step 3 of *NLSA* and proceed with splitting the interval and re-solving the problem, as discussed next in the following section.

6.2.2. *Computational results*

In this section, we present computational results to demonstrate the effectiveness of using *NSLA* for solving instances of *SMSP* with PWL dynamics. In addition to assessing tractability, we specifically consider three alternative approaches (below) for searching for infeasible intervals. In all approaches, the intervals are checked for infeasibility according to *AGACA*. When *AGACA* detects infeasibility while evaluating some segment j in some interval i, we split interval i at both the start and the end of the segment j, typically forming three new segments.

In the *Split Once, Evaluate from Start* approach for identifying an infeasible interval, we begin each iteration's search with the first interval, and proceed forward chronologically until an infeasible interval is found. Once an infeasible interval is found, we split this interval and return to Step 2 in *NLSA*.

The *Split Once, Cycle* approach is similar to the first approach, except that rather than starting the search at the first interval each iteration, we start at the interval corresponding to the linear segment that was found to be infeasible in the previous iteration. If the last interval in the planning horizon is reached without finding a feasible interval, then we cycle back to the first interval. The intent here is to focus on searching areas of known infeasibility rather than initially searching over intervals that have been feasible in previous iterations and thus are more likely to still be feasible.

These first two approaches focus on reducing the amount of time per iteration spent on searching for infeasible intervals. The trade-off is a potential increase in the number of iterations. In the *Split All* approach, we instead check all intervals at each iteration, thereby potentially splitting multiple intervals per iteration and thus reducing the total number of iterations.

In the upcoming computational results we consider nine problem instances that all have five original intervals, 100 linear sections per original interval, and the data parameters as given in Table 8. The instances have three possible download options, where the download efficiency, energy utilization, and data rate, i.e. (η, α, ϕ), for Options 1, 2, and 3 are (1, 1, 1), (1, 1.5, 0.75), and (1, 0.75, 1.5), respectively. The set of available options for interval i is denoted O_i; which are identically $O_1 = \{1,2,3\}$, $O_2 = \{1,2\}$, $O_3 = \{2,3\}$, $O_4 = \{1\}$, $O_5 = \{1,2,3\}$ for all problem instances studied.

The nine problem instances are designed to represent a broad range of possible problem types and are summarized in Table 9. The instances have different levels and combinations of energy/data *irregularity* and *synchronization. Irregularity* is a measure of the change in the dynamics during each interval relative to the buffer capacity (ζ), which can be used to characterize either energy or data dynamics. *Synchronization* is a measure of the similarity between energy and data dynamics (γ_e and γ_d). For a more detailed description of these characteristics, see Appendix C.

We expect problem instances with low *irregularity* to be easier to solve because their dynamics may be well approximated by their original intervals, and problems with high *synchronization* to be easier to solve because energy and data may become available at the same time to support downloading. Our experiments are designed to assess these hypotheses and generally evaluate computational tractability.

Overall, all problem instances solved relatively quickly, as seen in the total solve times, T_{total}, plotted on a logarithmic scale in Fig. 10(a). In particular, most instances solved in well under 1 min, while all instances solved in under 6 min.

The total solve time is composed of the *UCF* solve time, T_{ucf}, and the total evaluation time to find infeasible intervals, T_{eval}, where $T_{total} = T_{ucf} + T_{eval}$. The total number of iterations, N, and the average T_{ucf} and T_{eval} per iteration, T_{ucf}/N and T_{eval}/N, are plotted in Fig. 10(b)–(d).

The average solve and evaluate times per iteration are relatively small, i.e. T_{ucf}/N and T_{eval}/N are always less than 1 s, thus T_{total} is largely a function of N. The *Split All* approach requires up to five times fewer N relative to the *Split Once* approaches, and thus has notably lower average T_{total} values for most data sets.

In Fig. 10, the data sets are presented in order of increasing average irregularity from A to I (from left to right). Fig. 11 quantifies the relationship between normalized average energy irregularity and T_{total}, where we only show the *Split Once, Cycle* results, because they are representative of the other *NSLA* split approaches (i.e. similar trends are observed for all approaches). These results suggest that there is a positive correlation between average irregularity and computational effort. Problem instances with higher irregularity may not be well approximated by the dynamics at the start and the end of a single interval that contains multiple linear segments. Thus, they may require more branching to make decisions and more iterations to find feasible solutions.

There was no correlation observed between synchronization and computational effort, which may be because of the dominating effect of data irregularity in our experiments. More extensive and targeted tests should be performed to assess this correlation in more detail.

Table 8
Constant data parameters for test cases A–I in Table 9.

Parameter	Description	Value
t_i	Duration of each original interval	500
$e_{min}, e_{min}, e_{start}$	Energy minimum, maximum, and starting levels	0, 100, 50
$d_{min}, d_{min}, d_{start}$	Data minimum, maximum, and starting levels	0, 100, 50

Table 9
Data sets with diverse irregularity and synchronization characteristics. Normalized values are the raw values divided by the maximum raw value of the sets.

Data set	Irregularity				Synchronization (ζ)	
	Energy (γ_e) (raw)	Data (γ_d) (raw)	Maximum (normalized)	Average (normalized)	Raw	Normalized
A	0.020	0.020	0.040	0.040	0.81	0.82
B	0.020	0.32	0.64	0.34	0.42	0.42
C	0.32	0.020	0.64	0.34	0.42	0.42
D	0.020	0.50	1.0	0.52	0.35	0.35
E	0.50	0.020	1.0	0.52	0.35	0.35
F	0.32	0.32	0.64	0.64	0.97	0.98
G	0.32	0.50	1.0	0.82	0.53	0.54
H	0.50	0.32	1.0	0.82	0.53	0.54
I	0.50	0.50	1.0	1.0	0.99	1.00

7. Conclusions

The small satellite community is growing as scientists and engineers from government, industry, and educational institutions propose and launch an increasing number of single and constellation small satellite missions every year. Small satellites are highly constrained, with limited on-board power generation, attitude and orbit control, energy and data storage, and computational resources. Scientists and engineers aim to download large amount of highly sought-after data from these small satellites to independently owned and operated ground networks. This goal, coupled with the aforementioned constraints, results in a challenging spacecraft operational problem. Thus, optimizing satellite schedules is critical to ensure that constrained resources, such as time, energy, and data, are efficiently allocated towards maximizing the data downloaded. This paper presents an optimization formulation, algorithms, and theoretical and computational results that address the satellite download scheduling problem.

We have defined the *Single-Satellite Scheduling Problem (SMSP)* as the problem of maximizing the total data downloaded from single small satellite to a ground station network considering realistic constraints on available and stored energy and data. We presented the *Under-Constrained Formulation (UCF)* to solve instances of *SMSP*. In the paper, we demonstrated how *UCF* enables us to solve linear real-world small satellite and generic problems rapidly and in a computationally tractable way. We identified theoretical conditions for instances of *SMSP* that yield non-integer *MIP* solutions and require branching to solve, and discuss their computational properties and results.

To address instances of *SMSP* with non-linear dynamics, we developed a *Non-Linear SMSP Algorithm (NLSA)*, which is an iterative approach to repeatedly solving and discretizing the problem until a solution is found. We proved that *NLSA* yields feasible, and thus optimal, solutions for data instances with piece-wise linear dynamics and can prevent computational intractability of large problem instances. We discussed computational tractability of *NLSA* in the context of piece-wise linear instances of *SMSP* with diverse data characteristics.

In summary, we developed a formulation and algorithms to address the satellite download scheduling problem, laying the ground work for more complex problems both within and outside the operational space domain. The formulations and algorithms address scheduling tasks subject to resource acquisition, storage, and utilization constraints, so the theoretical and computational insights are also relevant in other applications.

7.1. *Future work*

There are several areas for possible extensions to address scheduling needs in the small satellite community and using the foundational formulations presented in this paper.

7.1.1. *Multiple-satellite scheduling problem*

Beyond the single-satellite problem considered in this paper, there is the more challenging problem of scheduling multiple spacecraft interacting with the same ground system and/or other spacecraft simultaneously. For these missions, individual spacecraft or network goals and constraints may conflict with the goals and constraints of other spacecraft, missions, or networks. Future research should develop scheduling algorithms to accommodate the additional complexities of multi-satellite problems, such as network availability, conflict, priority, and financial cost constraints. Routing and prioritization to resolve conflicts, arising because of the high demand and competition for access to relay orbiters, ground-based networks, or deep space networks (e.g. DSN), must be considered. The foundational structure of the

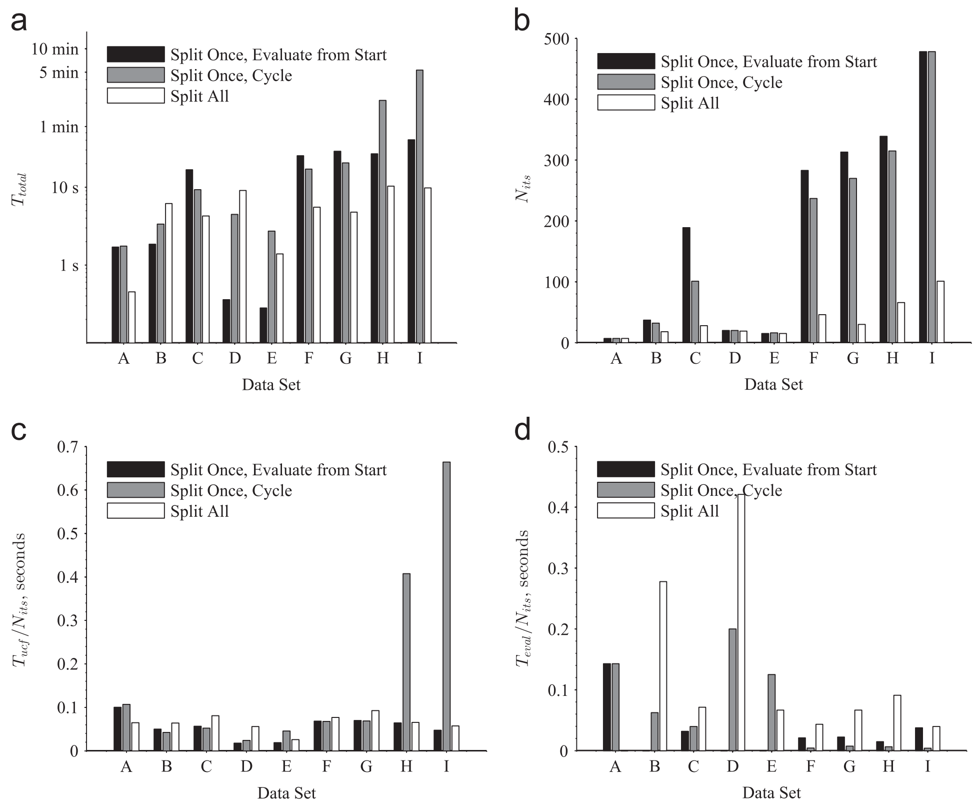

Fig. 10. Comparison of the three *NLSA* approaches when applied to solve PWL instances of *SMSP* for the data sets in Table 9. The data sets are presented in order of increasing average irregularity from A to I (from left to right). (a) Computational solve time (T_{total}). (b) Number of algorithm iterations *N*. (c) UCF solve time per iteration T_{ucf}/N. (d) Evaluation time per iteration T_{eval}/N.

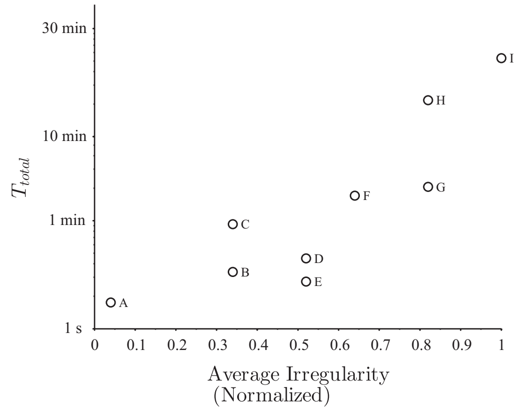

Fig. 11. Effects of average irregularity on computational performance for the *Split Once, Cycle NLSA* approach when applied to solve PWL instances of *SMSP* for the data sets in Table 9.

UCF can be extended to accommodate the multiple-satellite, multiple-ground station scheduling problem.

7.1.2. Stochastic scheduling problem

While the formulation and algorithms in this paper capture an approximation of realistic spacecraft operations, they neglect the non-deterministic nature of realistic satellite operational scenarios. There are several properties that are stochastic in *SMSP*, including energy and data acquisition, and the temporal opportunities and efficiency of both payload and download operations. Uncertainty in these parameters may be a complex function of multiple interacting factors. For example, download efficiency is influenced by the availability of the station, elevation and azimuth angle of the satellite relative to the ground station, and local time of day. Future research should aim to capture the deterministic factors impacting download efficiency (which are not modeled in the existing formulation). Next, formulations and algorithms should be developed for managing the impact of stochasticity on performance in both schedules that are developed a priori and dynamically updated in real time.

Acknowledgments

The first author, Sara Spangelo, is now employed at the Jet Propulsion Laboratory (JPL), and her contribution to the work in this paper was completed prior to her JPL employment. This research was sponsored by the National Science Foundation through National Science Foundation (NSF) Award no. CNS-1035236, the Canadian National Science and Engineering Research Council (NSERC), the Air Force Office of Scientific Research (AFOSR), and the Zonta Amelia Earhart Fellowship. We thank the CubeSat and amateur radio community for their support in completing the satellite and ground station surveys. We would also like to acknowledge the University of Michigan RAX Team and students from the College of Engineering SURE program.

Appendix A. Proofs

Lemma A.1. *Assume we are given a solution to an instance of UCF in which rate* ϕ^*_{io} *is used to download a total of* q^*_{io} *data during an interval i (where * denotes the solution). Then, download occurs for* $p = q^*_{io}/(\phi^*_{io} \cdot (t_{i+1} - t_1))$ *fraction of interval i, and we can construct an equivalent solution in which data is downloaded at the constant rate* $\phi^*_{io} \cdot p$ *over the entire duration of the interval.*

Proof. A solution to *UCF* specifies the amount of data to download during an interval (q_{io}^*) and the rate at which this download occurs (ϕ_{io}^*). For those cases where the specified amount of data and download rate will not fill the entire duration of the interval (i.e. $0 < p < 1$), the solution does not specify when in the interval the download must occur. In addition, the physics of the system do not require that the download occur over a single continuous interval. We may thus assume without loss of generality that the full interval is split into smaller sub-intervals of equal duration, and that during each interval we download for the first p fraction of time and then do not download for the remainder of this sub-interval. As the size of these sub-intervals approaches zero, the download rate approaches the constant download rate $\phi_{io}^* \cdot p$ over the entire duration of the interval i. □

Lemma A.2. *Given a solution to an instance of UCF in which a total of s_i^{e*} energy is spilled during interval i (where * denotes the solution), then spillage occurs for $f = s_i^{e*}/(\phi_{io}^* \cdot (t_{i+1} - t_1))$ fraction of interval i. We can construct an equivalent solution in which energy is spilled at the constant rate $s_i^{e*} \cdot f$ over the entire interval duration.*

Proof. The identical approach taken in Lemma A.1 can be used to prove Lemma A.2. □

Appendix B. Non-integral branching cases

A linear program with four decision variables will have four binding constraints in the optimal solution. This yields three possible cases:

Case 1: *Option* 2 *is not used*

When Option 2 is not used, $x_2 = 0$ and $q_2 = 0$. By (5.12), the solution is

$$q_1 = \min\left(t\phi_1, \frac{e}{\alpha_1}\right) = t\phi_1$$

$$q_2 = 0$$

with objective value

$$q_1 + q_2 = t\phi_1 \tag{B.1}$$

and binding constraints (5.4), (5.5), (5.7), (5.7) and (5.11)

Case 2: *Option* 1 *is not used*

When Option 1 is not used $x_1 = 0$ and $q_1 = 0$. By (5.12), the solution is

$$q_1 = 0$$

$$q_2 = \min\left(t\phi_2, \frac{e}{\alpha_2}\right) = \frac{e}{\alpha_2}$$

with the objective value

$$q_1 + q_2 = \frac{e}{\alpha_2} \tag{B.2}$$

and binding constraints (5.3), (5.5), (5.7), (5.8) and (5.10).

Case 3: *Options* 1 *and* 2 *are used*

When both options are used to download data, the variable constraints (5.8)–(5.11) are not active. Also, by the assumption that data is unlimited, constraint (5.6) is not active. Therefore, the remaining four constraints (5.3), (5.4), (5.5), and (5.7) must be. Since x_1 and x_2 are a convex combination define

$$\lambda = x_1$$

$$1 - \lambda = x_2$$

By solving the system of equations formed by the three binding constraints

$$\begin{cases} q_1 = t\phi_1\lambda \\ q_2 = t\phi_2(1-\lambda) \\ \alpha_1 q_1 + \alpha_2 q_2 = e \end{cases} \tag{B.3}$$

we get the solution

$$\lambda = \frac{\alpha_2\phi_2 - \frac{e}{t}}{\alpha_2\phi_2 - \alpha_1\phi_1} \tag{B.4}$$

and objective value

$$q_1 + q_2 = \frac{t\phi_1\phi_2(\alpha_2 - \alpha_1) + e(\phi_2 - \phi_1)}{\alpha_2\phi_2 - \alpha_1\phi_1} \tag{B.5}$$

The difference between the objective value for Case 3 and Case 1 is

$$\Delta_1 = \text{B.5} - \text{B.1} = \frac{\frac{e}{t} - \alpha_1\phi_1}{\alpha_2\phi_2 - \alpha_1\phi_1} t(\phi_2 - \phi_1)$$

If $\phi_2 > \phi_1$, then $\Delta_1 > 0$ and Case 3 provides a better solution than Case 1.

Similarly, the difference between the objective value for Cases 3 and 2 is

$$\Delta_2 = \text{B.5} - \text{B.2} = \frac{t\phi_1}{\alpha_2} \frac{\alpha_2\phi_2 - \frac{e}{t}}{\alpha_2\phi_2 - \alpha_1\phi_1}(\alpha_2 - \alpha_1)$$

If $\alpha_2 > \alpha_1$, then $\Delta_2 > 0$ and Case 3 provides a better solution than Case 2.

Therefore, if 5.12 holds in addition to $\phi_2 > \phi_1$ and $\alpha_2 > \alpha_1$, then a fractional solution is uniquely optimal.

Appendix C. Non-linear *SMSP* data characterizations

The data characterizations are defined below:

- *Irregularity* is a measure of the change in the dynamics during each sub-interval relative to the buffer capacity. This is measured independently for energy and data dynamics. Dynamics with a high level of irregularity will increase or decrease by as much as the buffer size over an interval, while data with low irregularity will be relatively constant (i.e. $\delta^{e+} + \delta^{e-} \approx 0$). Irregularity for energy and data dynamics, γ_e and γ_d, respectively, are defined analytically

$$\gamma_e = \frac{1}{n}\sum_{i=1}^{n} \frac{(\delta_i^{e+} - \delta_i^{e-})\Delta t_{av}}{(e_{max} - e_{min})(t_{i+1} - t_i)}, \tag{C.1}$$

$$\gamma_d = \frac{1}{n}\sum_{i=1}^{n} \frac{(\delta_i^{d+} - \delta_i^{d-})\Delta t_{av}}{(d_{max} - d_{min})(t_{i+1} - t_i)}, \tag{C.2}$$

  where changes in dynamics during interval i are normalized by the average interval duration of the planning horizon, Δt_{av}, divided by the duration of that interval, $t_{i+1} - t_i$. The number of intervals in the planning horizon is n.
- *Synchronization* is a measure of the similarity between energy and data dynamics. Note this only characterizes the nominal dynamics of a problem, and not the effects due to data download. Data with a high level of synchronization is characterized by energy and data dynamics that have slopes with the same sign, i.e. both increasing or decreasing, while for dynamics with low level of synchronization, the dynamics will have opposite signs, i.e. one resource will be increasing while the other is decreasing. Synchronization, ζ, is defined analytically as the sum of the intervals where the dynamics are synchronized normalized by the number of intervals in the planning horizon, n. $\{S\}$ represents the sum of the set S, where $\delta_i^{e+} \cdot \delta_i^{e-})$ checks

that the dynamics have the same sign

$$\zeta = \frac{1}{n}\left|\left\{ i \in I \middle| (\delta_i^{e+} - \delta_i^{e-}) > 0, (\delta_i^{e+} \cdot \delta_i^{e-}) > 0 \right\}\right| \tag{C.3}$$

References

[1] Baker Daniel N, Pete Worden S. The large benefits of Small-Satellite missions. Trans Am Geophys Union 2008;89(August (33)):301.
[2] Moretto T. CubeSat mission to investigate ionospheric Irregularities. Space Weather 2008;6:S11002. http://dx.doi.org/10.1029/2008SW000441.
[3] Earle G, Davidson R. Challenges and promise: CubeSat-based instrumentation for thermal plasma and neutral measurements. In: Fall AGU meeting; December 2009.
[4] Jorgensen T. The NSF CubeSat program: the promise of scientific projects. In: Fall AGU meeting; December 2009.
[5] Spence H, Moore T. A retrospective look forward on constellation-class geospace missions. In: Fall AGU meeting; December 2009.
[6] Swenson C, Sojka JC, Fish C, Larsen M, Bingham B, Young Q, et al. The high-latitude dynamic e-field (HiDef) explorer, a proposed network of 90 cubesats. In: QB-50 workshop; November 2009.
[7] Ridley A, Forbes J, Cutler J, Nicholas A, Thayer J, Fuller-Rowell T, et al. Armada Mission Team. In: The Armada mission: determining the dynamic and spatial response of the thermosphere/ionosphere system to energy inputs on global and regional scales. American Geophysical Union (AGU) Fall Meeting; December 2010, p.A7.
[8] J Cutler. Ground station markup language. In: IEEE aerospace conference; March 2004.
[9] Baek Seung-Woo, Cho Kyeum-Rae, Lee Dae-Woo, Bainum Peter M, Kim Hae-Dong. Development of scheduling algorithm and GUI for the autonomous satellite mission operation. In: Sixtieth international astronautical congress (IAC), vol. 7, Republic of Daejeon, Korea; 2009. p. 5527–34.
[10] Vasquez Michel, Haom J. A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. Comput Optim Appl 2001;20:137–57.
[11] Lemaitre M, Verfaillie G, Jouhaud F, Lachiver J-M, Bataille N. Selecting and scheduling observations of agile satellites. Aerosp Sci Technol 2002; 6(September (5)):367–81.
[12] Bensana E, Lemaitre M, Verfaillie G. Earth observation satellite management. Constraints 1999;4:293–9.
[13] Wolfe William, Sorensen Stephen. Three scheduling algorithms applied to the earth observing systems domain. Manag Sci 2000;46(1):148–66.
[14] Sun Baolin, Mao Lifei, Wang Wenxiang, Xie Xing, Qin Qianqing. Satellite mission scheduling algorithm based on genetic algorithm. In: Second international conference on space information technology (SICE), vol. 6795; 2007. p. 67950U.
[15] Martin William. Satellite image collection optimization. Opt Eng 2002;41 (9):2083–7.
[16] Jian Li, Cheng Wang. Resource planning and scheduling of payload for satellite with particle swarm optimization, USA, vol. 6795; 2007. p. 67951–1.
[17] Verfaillie Gerard, Pralet Cedric, Lemaitre Michel. How to model planning and scheduling problems using constraint networks on timelines. Knowl Eng Rev 2010;25(3):319–36.
[18] Sarah Burrowbridge. Optimal allocation of satellite network resources. In: Master of science in mathematics. Virginia Polytechnic Institute and State University, Blacksburg, VA; 1999.
[19] Pemberton J, Galiber F. A constraint-based approach to satellite scheduling. In: DIMACS workshop on constraint programming and large scale discrete optimization, Providence, RI; 2001. p. 101–14.
[20] Muraoka Taijiro Ohno Hiroyasu, Cohen Ronald H, Doi Naoyuki. Aster observation scheduling algorithm. In: International symposium space mission operations and ground data systems; 1998.
[21] Han Soon mi, Beak Seung woo, Cho Kyuem rae, Lee Dae woo, Kim Hae dong. Satellite mission scheduling using genetic algorithm. In: SICE annual conference; 2008. p. 1226–30.
[22] Barbulescu Laura, Watson Jean-Paul, Whitley L, Howe A. Scheduling space-ground communications for the Air Force satellite control network. J Sched 2004;7:7–34.
[23] Marinelli Fabrizio, Nocella Salvatore, Rossi Fabrizio, Smriglio Stefano. A Lagrangian heuristic for satellite range scheduling with resource constraints. In: Dipartimento di informatica, universitita degli studi di L'Aquila. Technical report TRCS 004; 2005.
[24] Globus Al, Crawford James, Lohn Jason, Pryor Anna. A comparison of techniques for scheduling earth observing satellites. In: Innovative applications of artificial intelligence conference, IAAI'04, AAAI Press, San Jose, CA; 2004. p. 836–43.
[25] Potter William, Gasch John. A photo album of earth: Scheduling landsat 7 mission daily activities. In: Fifth international conference on space operations, Tokyo, Japan; 1998.
[26] Soma P, Rao J, Padmashree G. Multi-satellite scheduling system for LEO satellite operations. In: SpaceOps conference; 1998.
[27] Rabideau G, Knight R, Chien S, Fukunaga A, Govindjee A. Iterative repair planning for spacecraft operations using the aspen system. In: Fifth international symposium on artificial intelligence, robotics and automation in space (ESA SP-440), Noordwijk, Netherlands; 1999. p. 99–106.
[28] Defence Harrison Price, Harrison SA, Price ME, Philpott MS. Task scheduling for satellite based imagery. In: UK planning and scheduling workshop, special interest group, University of Salford, UK; 1999. p. 64–78.
[29] Dutta Amitava, Rama Dasaratha V. An optimization model of communications satellite planning. IEEE Trans Commun 1992;40(9):1463–73.
[30] Bianchessi Nicola, Cordeau Jean-Francois, Desrosiers Jacques, Laporte Gilbert, Raymond Vincent. A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites. Eur J Oper Res 2007;177 (2):750–62.
[31] Arkali G, Dawande M, Sriskandarajah C. Scheduling support times for satellites with overlapping visibilities. Prod Oper Manag 2009;17:224–34.
[32] Cheung Kar-Ming, Lee CH, Gearhart WB, Vo T, Sindi S. Link-capability driven network planning and operation. In: IEEE aerospace conference, vol. 7; 2002. p. 7-3281–5.
[33] Zheng W, Meng X, Huan H. Genetic algorithm for TDRS communication scheduling with resource constraints. In: International conference on computer science and software engineering; December 2008.
[34] Spangelo SC, Cutler JW, Klesh AT, Boone DR. Models and tools to evaluate space communication network capacity. IEEE Trans Aerosp Electron Syst 2012;48(July (3)):2387–404.
[35] Bahcivan H, Kelley M, Cutler J. Radar and rocket comparison of UHF radar scattering from auroral electrojet irregularities: implications for a nano-satellite radar. J Geophys Res; 2009;114(June).
[36] Cutler JW, Mann J. CubeSat Ground Station Survey. ⟨http://gs.engin.umich.edu/gs_survey⟩; 2009–2011.
[37] IBM. IBM ILOG CPLEX Optimization Studio. ⟨http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/⟩; March 2010.