Heuristic Scheduling of Space Mission Downlinks: A Case study from the Rosetta Mission

Gregg Rabideau[1], Federico Nespoli[2], Steve Chien[1]

[1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA
{gregg.rabideau, steve.chien}@jpl.nasa.gov

[2]European Space Agency, Noordwijk, Netherlands / Telespazio VEGA UK Ltd, Luton, UK.
fnespoli@esa.int

Abstract

Space mission planning/scheduling is determining the set of spacecraft activities to meet mission objectives while respecting mission constraints.

One important type of mission constraint is data management. As the spacecraft acquires data via its scientific instruments, it must store the data onboard until it is able to downlink it to ground communications stations. Because onboard storage is limited, this can be a challenging task.

This paper describes a formulation of the data downlink scheduling problem used for the Rosetta orbiter, a European Space Agency cornerstone mission currently investigating the comet 67P/Churyumov-Gerasimenko. We first describe the abstract problem and the Rosetta mission specific problem, along with desirable features of downlink schedules. We outline several algorithms (including the Rosetta operational algorithm) and we compare their performance on both actual mission data.

Introduction

Spacecraft enable us to measure and explore a wide range of targets spanning Earth, to the planets and bodies of our solar system, to bodies beyond our galaxy to the furthest reaches of the universe.

Mission planning and scheduling is an extremely challenging part of operating these space missions. While in the space community it is termed mission planning, from an Artificial Intelligence perspective the issue is more scheduling than planning as the challenge is to find appropriate times to schedule observations to achieve mission objectives that conform to the operations constraints of the spacecraft. Space mission planning represents a fertile applications area for Artificial Intelligence-based planning and scheduling techniques with a wide range of deployed systems (for a survey see [Chien et al. 2012]).

One particular challenge for space mission planning is downlink planning. In this problem the data acquired onboard from engineering telemetry and science observations is stored onboard. This onboard storage is limited and is often pre-partitioned in an inflexible allocation. Commonly, first a schedule is negotiated between the space mission and a ground communications station provider (or providers). Once this schedule has been determined, a prior version of a mission plan is adapted to ensure that all data is preserved - determining exactly which portions of onboard storage are downlinked when so as to enable the science and engineering data to be acquired and downlinked without loss of data.

Many variants of this downlink problem exist. For example, there may be some uncertainty as to the volume of acquired data. There may be certain types of data that have deadlines for downlink. We describe a particularly challenging downlink problem in which data generation may occur over extremely long periods of time overlapping downlink periods.

Problem Definition

We formalize the data downlink problem as follows:

Given:
a set of buffers $B = \{b_1, b_2, \ldots b_n\}$
where each b_i has
an initial fill state: $init\_fill_i$
(fill state of buffer at start of planning interval)
end fill limit: end_fill_i

(hard limit on fill state of buffer
at end of planning interval)
hard capacity: $capacity_i$
desired margin: $margin_i$
(soft limit on buffer fill volume
at any point in the interval)
and the operations plan dictates for each b_i in B, there is a

fill_function fill(b_i,t)→rate where rate is bits/s

and there is a set of downlinks D={d_1,...d_m} where each

d_i=<start_d_i, end_d_i, rate_d_i>

(we assume that no downlinks are overlapping)

for each downlink specify a downlink rate from each buffer such that the sum of all buffers downlink rates is <= downlink data rate

∀b_i, a function empty(b_i,t)→rate (bits/s) such that

∀downlink d_i
start_d_i ≤ t ≤ end_d_i Σ empty(b_i,t) ≤ rate_d_i
(i.e. at any point in time we can only downlink up to our downlink capacity)

∀t ∈ current_schedule current_fill_state(b_i, t) ≤ $capacity_i$

also it is desired that peak fill state and end fill state meet their targets. Specifically:

no peak margins are violated
∀t in current_schedule, for all b_i
max(current_fill_state(b_i, t)) ≤ $margin_i$

no end margins are violated
for t = end of current_schedule,
∀b_i current_fill_state(b_i, t)) ≤ end_fill_i

In reality, as we will see, flight software on actual missions is not designed to allow for arbitrary downlink policies, so that our ability to control the empty(b_i,t) function is not as flexible as desired.

The Rosetta Downlink Scheduling Problem

The Rosetta onboard data storage is partitioned into a set of buffers, called packet stores, for different types of science and engineering data. Each instrument is designated a packet store with a specified hard upper volume limit that cannot be changed during routine scheduling.

The behavior of each downlink can only be controlled by two commands: SET_SCI_DW_LEVEL and STOP_DUMP.

- The first, SET_SCI_DW_LEVEL, is issued at the start of the downlink and assigns a priority to each of the packet stores.
- The second, STOP_DUMP, can be issued any time during the downlink and stops the downlink of data from the specified packet store for the remainder of the current downlink. Note that once a packet store has been stopped, it cannot be restarted for that downlink.

These two commands, along with their timing and parameters, make up the decision variables available to the scheduler for controlling the "empty" functions described earlier. To fully understand how these variables affect the "empty" function, we must examine the onboard software that controls the data downlink. We summarize the behavior of the downlink software in the following set of rules.

- Two of the packet stores (used for high-priority engineering data) have fixed priorities and cannot be stopped with STOP_DUMP commands.
- A packet stores remains "active" until the STOP_DUMP command is issued, after which no data will be downlinked regardless of priority.
- When more than one active packet store has data waiting to be downlinked, the one with higher priority will be dumped first.
- If more than one active packet store all have the same priority, data will be downlinked in a round-robin fashion.
- Each packet store has a predefined packet size which defines the minimum amount of data that will be downlinked on each round-robin cycle.
- When a packet store contains less than one packet, downlink for that packet store will stop, possibly allowing downlink to start on the next highest priority packet store.
- If, at any time during the downlink, data is added to an empty but active packet store, downlink for that packet store will restart, preempting any downlink from lower-priority packet stores.
- Both the "active" state and the priority are reset at the end of the downlink.

Given the two available commands, and the set of downlink rules, the primary job of the downlink scheduler is to assign priorities and decide when to stop dumps in order to prevent overflow on all packet stores. The secondary goal of the scheduler is to make selections that prevent margin violations. Last, for some of the packet stores, there is a desire for the scheduler to keep margins as large as possible.

To achieve these goals, the scheduler must first model the behavior of the packet stores so that volume and overflows can be accurately predicted. Fill rates from observations, and dump rates from downlinks, are all provided as inputs to the scheduler. When constructing the schedule for the first time, the scheduler must decide on which observations to include as well as which downlink commands to issue to best satisfy science requests. In this paper, we focus on the scheduling of downlink commands only, assuming an observation schedule is fixed. Note that this type of re-scheduling of the downlink commands is necessary during short-term planning when certain last-minute changes must be made (e.g. due to the loss of a downlink). However, in the larger mission planning/scheduling process, observation scheduling and downlink scheduling are performed simultaneously.

With a model of how data is collected and downlinked, the scheduler can generate a profile for each packet store that predicts the data volume at any point during the planning period. This profile can be used not only to predict overflows, but also provides information to the scheduler about when, and by how much, data will overflow. This information can then be used to make decisions about which priority values to assign at the start of each downlink, and when to stop the dump during each downlink. For example, after a given downlink, if there is one particular packet store that will overflow sooner, or exceed its limit by more than any other packet store, then that packet store should be given higher priority or more time to downlink.

In our original implementation, we used a fixed set of pre-assigned priorities that were mostly unique, and selected only the length of time for each dump. Due to the serial nature of the resulting dump schedule, this method proved brittle to communication loss (packet stores scheduled near the end of the downlink would be unfairly impacted). To address this problem, we implemented the priority-based method, which assigned different priorities for each downlink but did not issue STOP_DUMP commands. Ideally, both SET_SCI_DW_LEVEL and STOP_DUMP would be used to select the best possible dump schedule, measuring both quality and robustness of the schedule. This is left for future work.

In this paper, we focus on the priority-based method, since this is the default method used in current operations. Therefore, in this formulation, the control variables are:

for each downlink: $d_1,...d_i$
for each packet store: $s_1,...s_j$
assign a priority to $P_{a,b}$ for a =1...i, and b = 1...j

Note that these priorities, together with the packet store initial states and incoming data, effectively define an empty(b_i,t) function.

Figure 1 contains the pseudo-code for scheduling downlink priority commands for the Rosetta spacecraft. The initial schedule contains only fill activities that generate data into packet stores with continuously increasing volumes (beyond their limits). Lines 3-6 heuristically assign a priority to each packet store of each downlink, generating a list of overflows that result. If an overflow occurs before the end of a downlink, the schedule will perform limited backtracking to reschedule at most two of the previous downlinks (line 8).

The function priorityHeuristic (lines 11-17) implements the operational heuristic for making priority-based buffer allocations for a given downlink. Here, packet stores are given a priority that is inversely proportional to the number of the downlinks that occur before the first overflow (lines 15-17). This ensures that high priority is given to packet stores with more urgent need for downlink. A similar heuristic is used to choose STOP_DUMP times in the time-based method. As an example, if two or more packet stores have future overflows at around the same time, then they will likely be assigned the same priority (or same amount of time). If an *immediate* overflow has been identified for the given packet store, then it will be assigned the highest priority (line 12-13). An immediate overflow is defined as one that occurs before the next downlink. Note that downlink parameters are chosen independent of other downlinks and packet stores. Choices made for one downlink have only an indirect impact on the choices that will be made for future downlinks.

```
1.  scheduleDownlinks(downlinks)
2.    sortByStartTime(downlinks)
3.    for each d in downlinks
4.      for each ps in packet stores
5.        p = priorityHeuristic(d, ps)
6.        setDumpPriority(d, ps, p)
7.      if overflows exist
8.        backtrack
9.    return overflows
10.
11. priorityHeuristic(d, ps)
12.    if ps has an immediate overflow
13.      return MAX_PRIORITY
14.    else
15.      o = findFirstOverflow(ps)
16.      n = numberOfDownlinksBetween(d, o)
17.      return MAX_PRIORITY - n
```

Figure 1: Scheduling algorithm

For evaluation purposes only, we consider three additional heuristics for selecting priorities. First, as a baseline, we randomly select priorities. Second, we assign the highest priority to the packet store with the largest

volume measured as percent of capacity. The remaining buffers are assigned the lowest priority. Last, we implement a heuristic that assigns priorities by normalizing the percent full values across the available priorities (e.g. with 10 priority values, a packet store with volume <10% is assigned the lowest priority). All four heuristics are compared in the empirical evaluation section of this paper.

Finally, certain packet stores may contain time-sensitive data (e.g. data which may impact future plans). For these "urgent" packet stores, the downlink latency (i.e. time between collection and downlink) can be reduced by increasing the required margin. To find the largest margin without creating overflows, we wrap the scheduleDownlinks function in a binary search loop. Each iteration of the loop either increases or decreases the margin, depending on the existence of overflows. The result is a schedule with large margins, keeping the data volume low, and reducing the time that data waits in the packet store. This technique is limited, however, to data collection schedules that have feasible downlink schedules (i.e. a solution must exist with no overflows).

Estimated Computational Complexity of the Scheduling Algorithm

Our analysis of the above scheduling algorithm indicates the following factors in its computational complexity

scheduleDownlinks = O(D * P * F)

where
D = # downlinks,
P = # packet stores,
F = # fill rate changes

Finding the best margin only adds a constant factor lg100 (binary search on a percentage between 1 and 100).

For the Rosetta mission, downlink planning is typically processed over a "Medium term plan" or MTP, which is generally 4 weeks in length. For Rosetta there are 16 packet stores, and for one MTP, there are typically 30+ downlinks and hundreds of fill rate changes.

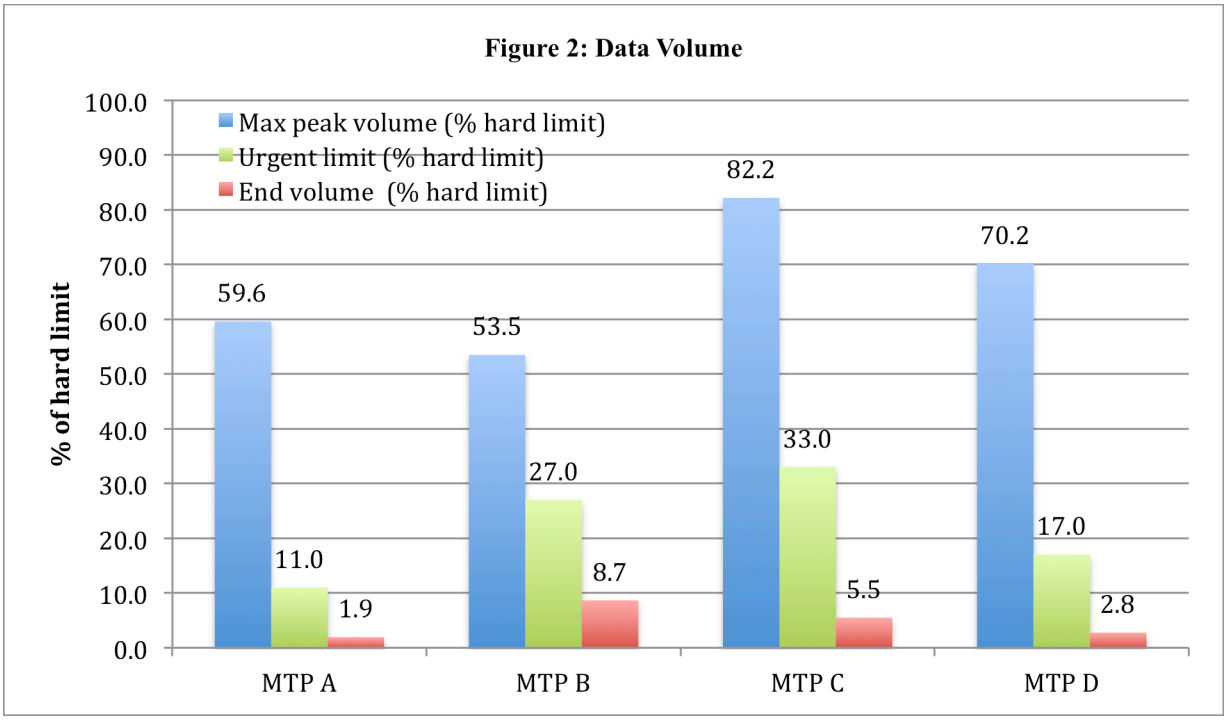
Empirical Evaluation of the Scheduling Algorithm

To date, we have conducted an empirical evaluation of the priority-based scheduling algorithm, the primary method used in operations. We use data from 4 medium-term planning (MTP) periods during the Rosetta mission, with each period spanning approximately 4 weeks. The CPU time required to generate each MTP downlink schedule was less than 1 minute running on a typical Windows laptop. The results are summarized in Figure 2 and Figure 3, with more details provided in Appendix A. The actual names of the packet stores and MTPs have been replaced for security reasons.

We evaluate its performance using the following metrics:

- Max peak volume percent: the maximum percent volume consumed for any packet store at any time during the MTP (no overflow if less than 100%)
- End volume percent: the percent volume consumed at the end of the MTP period
- Urgent limit: the smallest limit (i.e. largest margin) found by the schedule for the packet stores designated as containing urgent data
- Data collected: how much total data was collected from observations during the MTP

Figure 2: Data Volume



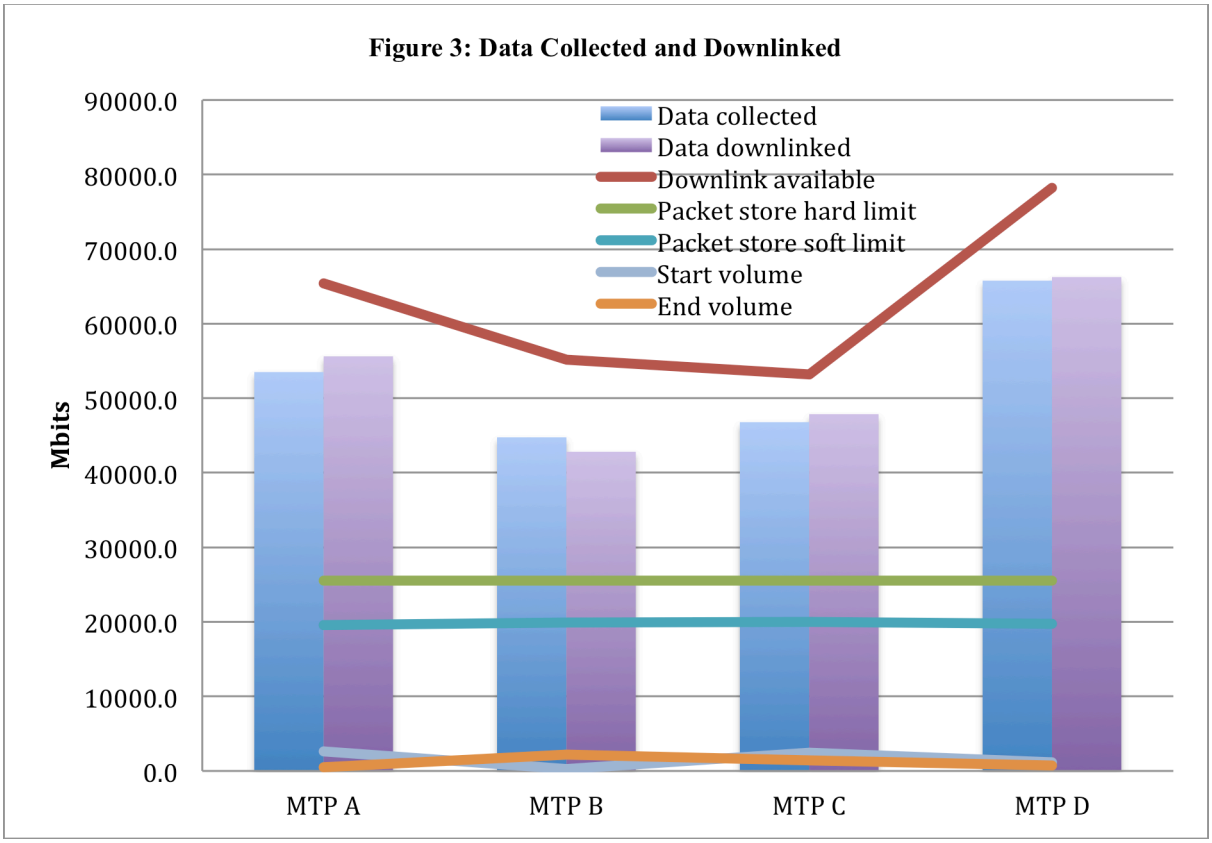
- Data downlinked: how much total data was downlinked out of the packet stores during the MTP
- Downlink available: how much downlink was theoretically available during the MTP (downlink duration multiplied by the bit rate).
- Packet store hard limit: what was the physical limit on the packet stores, and how do the data amounts compare
- Packet store soft limit: what was the operational limit imposed (including margin), and how do the data amounts compare
- Start and end volume: what was the volume of the packet store at the start and end of the MTP

The "Max peak volume" reported in Figure 2 give the maximum percent volume consumed for any packet store at any time during the MTP as a percent of the capacity for that packet store. This shows that, during the given 16-week period, at no point are any of the packet stores predicted to overflow. In addition, the data in Appendix A shows that at no point are any of the packet stores expected to exceed the desired "soft" limit. Some margin (typically 20%) on the packet store volume is maintained in order to account for uncertainties in the data collection and downlink model, which can occur from such things as variable data compression rates and communication outages. The "End volume" series in Figure 2 shows the percent volume consumed at the end of the MTP. Operationally, there is a preference to have minimal carry-over from one MTP to the next. For these four MTPs, the end volume stays below 10% of the capacity.

In each MTP, the data in two or three of the packet stores was considered urgent (designated with a '*' in the tables in Appendix A), and the downlink scheduler searched for the largest feasible margin on the packet stores. This meant keeping the volume low for the urgent packet stores without overflowing the other packet stores. In this way, the urgent data is not left to accumulate in the packet stores over long periods of time. The "Urgent limit" series in Figure 2 gives the smallest limit found by the scheduler for the packet stores containing urgent data.

In Figure 3, we see that the data collected amounts are very similar to the downlinked amounts for each MTP. It also shows that both values stay greater than 80% of the theoretical downlink available. The available downlink increases in the last MTP due to an increase in downlink rate, which occurs as the spacecraft exits solar conjunction. Finally, we can see that the data collected in each MTP is roughly between 2x and 3x the total packet store limit.

Figure 3: Data Collected and Downlinked



	Peak Volume			
	random	full	even	ops
MTPA	68.4	69.2	60.9	59.6
MTPB	64.7	52.9	54.4	55.3
MTPC	204.6	231.3	119.4	82.2
MTPD	77	69.5	66.1	70.2
Avg	103.675	105.725	75.2	66.825

Table 1: Peak Volume

	Urgent Limit			
	random	full	even	ops
MTPA	17.2	14.1	14.1	10.9
MTPB	31.2	32.8	29.7	28.1
MTPC	100	100	100	32.8
MTPD	32.8	18.7	20.3	17.2
Avg	45.3	41.4	41.025	22.25

Table 2: Urgent Limit

We should mention that there are certain packet stores that are designated as containing high-priority, engineering data (marked with '**' in the tables in Appendix A). The data in these high-priority packet stores is downlinked first, and all of the data is downlinked before time is given to the other packet stores. Note that peak volume (column 3) stays very low for these packet stores.

Additional runs were performed to compare different heuristics for selecting priorities. Tables 1 and 2 show the results. In the "random" heuristic, priorities were selected at random to create a baseline for comparison. The "full" heuristic assigns the highest priority to the packet store with the highest volume measured as a percent of buffer capacity. The "even" heuristic assigns priorities by normalizing the fill percentages across the available priority values. The "ops" heuristic is the operational heuristic described in this paper. Table 1 records the maximum peak volume that resulted from applying each of the heuristics to each MTP. Table 2 records the smallest limit achieved for the "urgent" packet stores. The most

Figure 4: GUI

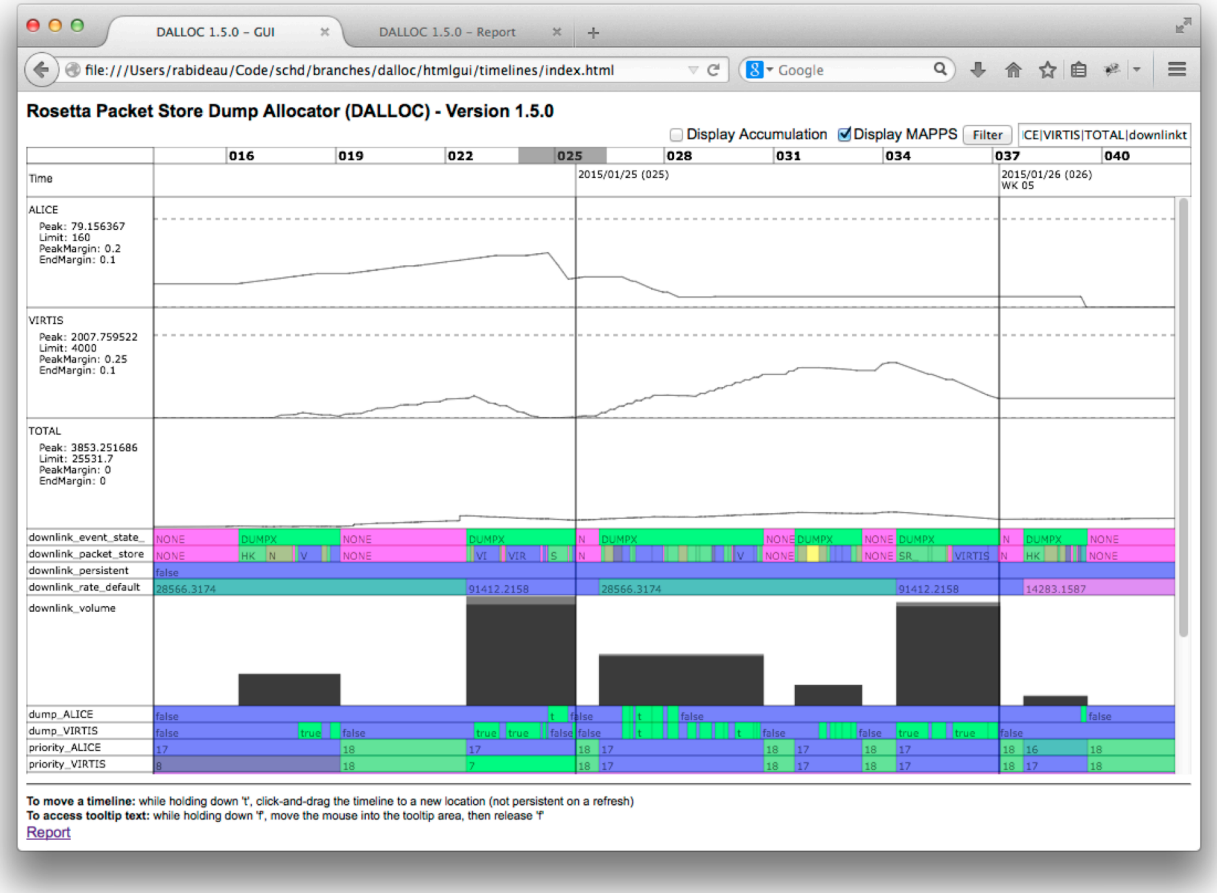
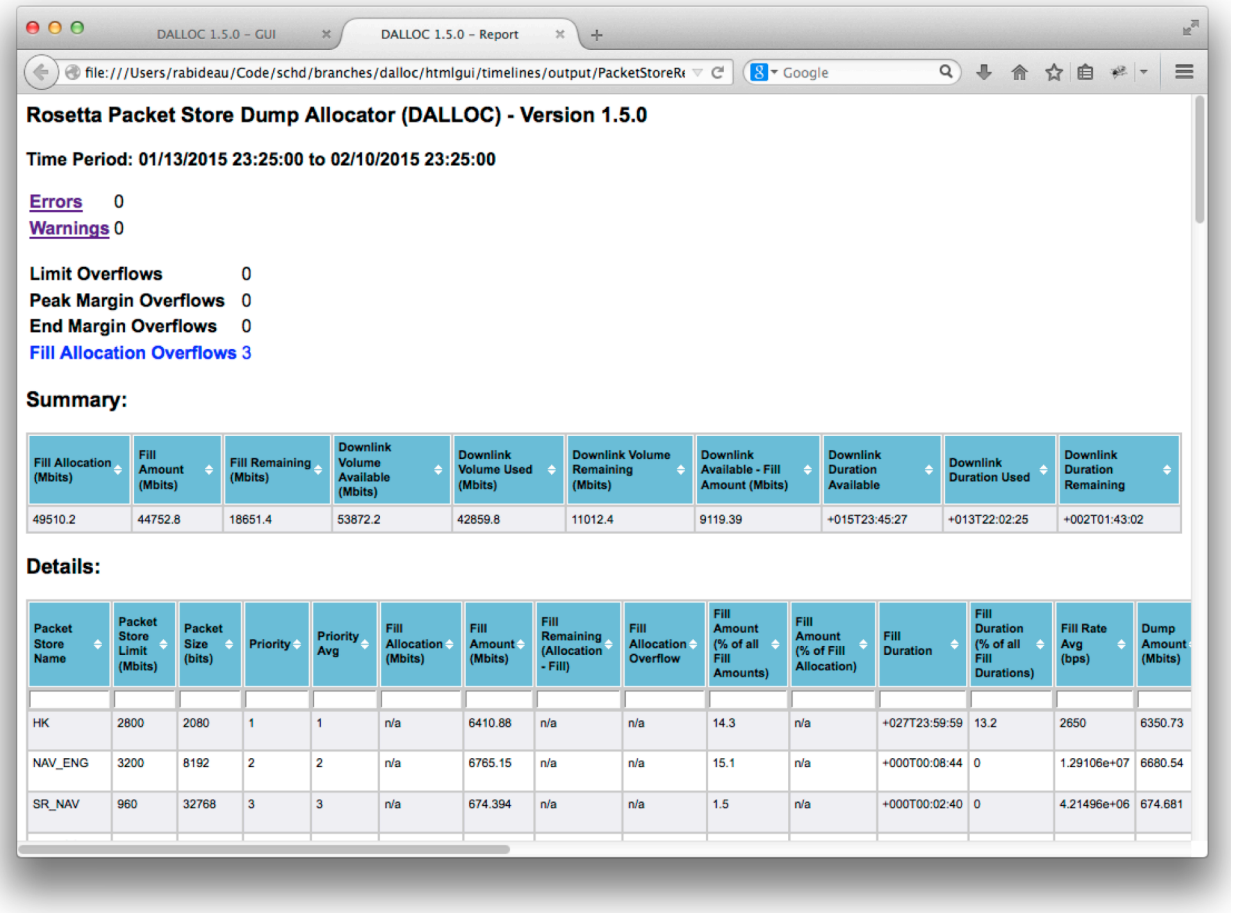


Figure 5: Report



significant difference can be seen in MTPC, which has the least amount of downlink available. In this case, only the operational heuristic generates a downlink schedule that does not result in packet store overflow. Note that the operational heuristic finds the smallest "urgent" limit in all four cases, but to achieve this, will sometimes create a higher maximum peak volume.

The GUI used in operations to evaluate the downlink schedule can be seen in Figures 4 and 5. The first is a zoom-able, scrollable, interactive graph that shows how the packet store volumes change over time within the given MTP. The bottom of the graph contains data on the various parameters that affect the volume, some of which the scheduler can control (e.g. priorities) and others that it cannot (e.g. downlink volume). The second page of the GUI provides a report on the resulting downlink schedule. The values in Appendix A were taken from this report.

In the future, we plan to evaluate the time-based scheduling method, as well as a combined method that selects both priority and stop time for each packet store, which in theory should produce the best results. Also, we plan to conduct scaling tests to validate our analysis of the computational complexity of the algorithm. We also plan on developing synthetic problem generators to further explore the performance of the downlink scheduling algorithms.

Discussion

Automated downlink planning is in operational use for the Mars Express [Cesta et al. 2007] mission. However in their data model observations produce data instantaneously, whereas in the Rosetta model data producing activities have rates that have significant temporal extent (such as engineering production continuously over the entire mission, and science observations doing the same). Interestingly, Cesta et al. characterize the problem as a planning problem (that of producing the sequence of downlink controlling commands). We take an opposing view that the end product is a scheduling/resource allocation problem, that of providing a downlink profile. MEX-MDP (Mars Express

Spacecraft Memory Dumping Problem) also takes into account the size of the plan (this is not an issue for Rosetta). Their robustness metric is similar to our margin requirement.

Onboard downlink management [Pralet et al. 2014] is proposed in order to address challenges of uncertainty in data generation (due to the uncertainty of effectiveness of content-dependent compression schemes). This formulation of the problem adds even several more complexities such as antenna pointing, multiple channels, data latency, and encoding table time. Again for a typical earth imager, the data production is effectively instantaneous, in contrast to the Rosetta problem.

Most other deployed automated planners must also solve some version of the downlink planning/scheduling problem however in most cases it is not the focus of the overall scheduling problem (e.g. Hubble Space Telescope [Johnston and Miller 1994], Earth Observing One [Chien et al. 2005, 2010] or Orbital Express [Knight et al. 2013]).

The Philae Lander for the Rosetta Mission has a science scheduling with downlink problem [Simonin et al. 2012]. They use ILOG-scheduler in a system called MOST to solve for most of the scheduling constraints except data management. They examine the problem of scheduling science experiments with fixed science experiment storage and downlink buffer storage but with a fixed priority downlink strategy. This problem is analogous to the full Rosetta scheduling problems [Chien et al. 2014]. However, one key difference is that MOST does not have the ability to re-program buffer priorities dynamically as we have on the Rosetta Orbiter (and described here in DALLOC).

Summary

We have described the downlink scheduling problem, a well defined subproblem within the overall space mission planning and scheduling problem. While this problem can be and often is solved in isolation, it is also addressable concurrently with the overall scheduling problem.

We then described the Rosetta downlink scheduling problem as a specific instantiation of the general downlink scheduling problem - with additional constraints. We describe two implemented heuristic solutions to this problem and we present complexity analysis and empirical evaluation on actual Rosetta mission data.

Acknowledgements

Portions of this work were performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and N. Policella. "An Innovative Product for Space Mission Planning: An A Posteriori Evaluation." Proc Intl Conf on Automated Planning and Scheduling, pp. 57-64. 2007.

S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandl, S. Frye, B. Trout, S. Shulman, D. Boyer, "Using Autonomy Flight Software to Improve Science Return on Earth Observing One, Journal of Aerospace Computing, Information, & Communication, April 2005, AIAA.

S. Chien, D. Tran, G. Rabideau, S. Schaffer, D. Mandl, S Frye, "Timeline-based Space Operations Scheduling with External Constraints," International Conference on Automated Planning and Scheduling, Toronto, Canada, May 2010.

S. Chien, M. Johnston, N. Policella, J. Frank, C. Lenzen, M. Giuliano, A. Kavelaars, A generalized timeline representation, services, and interface for automating space mission operations Space Operations (SpaceOps 2012). Stockholm, Sweden. June 2012.

S. Chien, G. Rabideau, D. Tran, J. Doubleday, D. Chao, F. Nespoli, M. P. Ayucar, M. C. Sitja, C. Vallat, B. Geiger, N. Altobelli, M. Fernandez, F. Vallejo, R. Andres, M. Kueppers, Activity-based Scheduling of Science Campaigns for the Rosetta Orbiter: An Early Report on Operations International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS 2014). Montreal, Canada. June 2014.

M. D. Johnston and G. Miller. "Spike: Intelligent scheduling of hubble space telescope observations." Intelligent Scheduling (1994): 391-422.

Russell Knight, Caroline Chouinard, Grailing Jones, Daniel Tran, Leveraging Multiple Artificial Intelligence Techniques to Improve the Responsiveness in Operations Planning: ASPEN for Orbital Express, AI Magazine, Vol 35, No 4, 2014.

C. Pralet, G. Verfaillie, A. Maillard, E. Hébrard, N. Jozefowiez, M.-J. Huguet, T. Desmousceaux, P. Blanc-Paques, J. Jaubert. Satellite Data Download Management with Uncertainty about the Generated Volumes. Proc. of the 24th International Conference on Automated Planning and Scheduling (ICAPS-14), Portsmouth, NH, USA, 2014.

C. Simonin, C. Artigues, E. Hebrard, and P. Lopez, "Scheduling Scientific Experiments on the Rosetta/Philae Mission," "Scheduling scientific experiments on the Rosetta/Philae mission." In *Principles and Practice of Constraint Programming*, pp. 23-37. Springer Berlin Heidelberg, 2012.

Appendix A

	MTP A						
	#1	#2	#3	#4	#5	#6	#7
PS1**	2800.0	127.8	4.6	80.0	101.2	3.6	30.0
PS2**	3200.0	154.9	4.8	80.0	103.3	3.2	30.0
PS3**	960.0	67.5	7.0	80.0	0.0	0.0	30.0
PS4	800.0	273.3	34.2	80.0	10.8	1.4	30.0
PS5	50.3	21.0	41.7	80.0	12.3	24.4	30.0
PS6	160.0	91.8	57.4	80.0	17.5	10.9	30.0
PS7	320.0	86.8	27.1	80.0	4.8	1.5	30.0
PS8*	640.0	67.5	10.5	11.0	18.7	2.9	30.0
PS9*	320.0	33.8	10.6	11.0	18.8	5.9	30.0
PS10	1600.0	494.6	30.9	80.0	41.9	2.6	30.0
PS11	7516.2	3645.7	48.5	80.0	0.0	0.0	30.0
PS12	0.8	0.0	0.0	80.0	0.0	0.0	30.0
PS13	800.0	288.1	36.0	80.0	81.2	10.2	30.0
PS14	4000.0	2385.0	59.6	80.0	0.0	0.0	30.0
PS15	0.8	0.0	0.0	80.0	0.0	0.0	30.0
PS16	880.0	325.4	37.0	80.0	62.2	7.1	30.0
PS17	595.6	25.4	4.3	80.0	14.9	2.5	30.0
PS18	888.0	57.4	6.5	80.0	10.0	1.1	30.0
TOTAL/AVG	25531.7	8146.1	31.9	72.3	497.6	1.9	30.0

	MTP B						
	#1	#2	#3	#4	#5	#6	#7
PS1**	2800.0	173.2	6.2	80.0	131.2	4.7	30.0
PS2**	3200.0	206.6	6.5	80.0	193.2	6.0	30.0
PS3**	960.0	67.4	7.0	80.0	0.0	0.0	30.0
PS4	800.0	353.8	44.2	80.0	155.6	19.5	30.0
PS5	50.3	18.8	37.3	80.0	14.0	27.8	30.0
PS6	160.0	85.6	53.5	80.0	33.0	20.7	30.0
PS7	320.0	0.0	0.0	80.0	0.0	0.0	30.0
PS8*	640.0	168.5	26.3	27.0	6.7	1.0	30.0
PS9*	320.0	28.4	8.9	27.0	24.6	7.7	30.0
PS10	1600.0	193.8	12.1	80.0	57.2	3.6	30.0
PS11	7516.2	1839.7	24.5	87.0	729.1	9.7	30.0
PS12	0.8	0.0	0.0	87.0	0.0	0.0	30.0
PS13	800.0	174.4	21.8	80.0	75.1	9.4	30.0
PS14	4000.0	1799.1	45.0	75.0	753.2	18.8	30.0
PS15	0.8	0.0	0.0	75.0	0.0	0.0	30.0
PS16	880.0	304.2	34.6	80.0	28.0	3.2	30.0
PS17*	595.6	29.3	4.9	27.0	8.6	1.4	30.0
PS18	888.0	0.0	0.0	80.0	0.0	0.0	30.0
TOTAL/AVG	25531.7	5442.8	21.3	71.4	2209.5	8.7	30.0

#1 Packet store hard limit
#2 Peak volume
#3 Peak volume (% hard limit)
#4 Peak volume soft limit (% hard limit)
#5 End volume
#6 End volume (% hard limit)
#7 End volume soft limit (% hard limit)

* Contains urgent data
** Contains high-priority engineering data

	MTP C						
	#1	#2	#3	#4	#5	#6	#7
PS1**	2800.0	164.3	5.9	80.0	2.5	0.1	30.0
PS2**	3200.0	275.4	8.6	80.0	68.7	2.1	30.0
PS3**	960.0	337.3	35.1	80.0	0.0	0.0	30.0
PS4	800.0	516.4	64.6	80.0	89.3	11.2	30.0
PS5	50.3	27.2	54.0	80.0	18.7	37.2	30.0
PS6	160.0	87.3	54.6	80.0	58.3	36.4	30.0
PS7	320.0	0.0	0.0	80.0	0.0	0.0	30.0
PS8*	640.0	209.5	32.7	33.0	21.8	3.4	30.0
PS9*	320.0	80.7	25.2	33.0	28.2	8.8	30.0
PS10	1600.0	797.1	49.8	80.0	146.0	9.1	30.0
PS11	7516.2	6177.7	82.2	87.0	804.1	10.7	30.0
PS12	0.8	0.0	0.0	87.0	0.0	0.0	30.0
PS13	800.0	410.2	51.3	80.0	102.8	12.8	30.0
PS14	4000.0	2732.3	68.3	75.0	0.0	0.0	30.0
PS15	0.8	0.0	0.0	75.0	0.0	0.0	30.0
PS16	880.0	447.1	50.8	80.0	28.8	3.3	30.0
PS17*	595.6	43.4	7.3	33.0	16.4	2.8	30.0
PS18	888.0	25.0	2.8	80.0	17.6	2.0	30.0
TOTAL/AVG	25531.7	12330.8	48.3	72.4	1403.3	5.5	30.0

	MTP D						
	#1	#2	#3	#4	#5	#6	#7
PS1**	2800.0	172.0	6.1	80.0	46.7	1.7	30.0
PS2**	3200.0	232.9	7.3	80.0	77.5	2.4	30.0
PS3**	960.0	67.5	7.0	80.0	0.0	0.0	30.0
PS4	800.0	265.1	33.1	80.0	50.7	6.3	30.0
PS5	50.3	29.7	59.0	80.0	13.8	27.3	30.0
PS6	160.0	67.0	41.9	80.0	42.1	26.3	30.0
PS7	320.0	0.0	0.0	80.0	0.0	0.0	30.0
PS8*	640.0	83.4	13.0	17.0	20.2	3.2	30.0
PS9*	320.0	50.5	15.8	17.0	6.4	2.0	30.0
PS10	1600.0	424.1	26.5	80.0	32.1	2.0	30.0
PS11	7516.2	4465.6	59.4	87.0	0.0	0.0	30.0
PS12	0.8	0.0	0.0	87.0	0.0	0.0	30.0
PS13	800.0	217.5	27.2	80.0	29.1	3.6	30.0
PS14	4000.0	2806.2	70.2	75.0	330.0	8.2	30.0
PS15	0.8	0.0	0.0	75.0	0.0	0.0	30.0
PS16	880.0	337.0	38.3	80.0	27.4	3.1	30.0
PS17*	595.6	42.7	7.2	17.0	26.7	4.5	30.0
PS18	888.0	0.1	0.0	80.0	0.1	0.0	30.0
TOTAL/AVG	25531.7	9261.3	36.3	69.7	702.9	2.8	30.0