Managing Spacecraft Memory Buffers with Overlapping Store and Dump Operations

Gregg Rabideau[1], Steve Chien[1], Federico Nespoli[2], Marc Costa[3]
[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
[3]European Space Astronomy Center (ESAC-ESA), Villanueva de la Cañada, Madrid, 28692, Spain
marc.costa@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 and communication opportunities are often limited, this can be a challenging task.

This paper describes a formulation of the overlapping Memory Dumping Problem (oMDP), which is a generalization of the Mars Express Memory Dumping Problem (MEX-MDP). We first describe the abstract problem of onboard data management for spacecraft. Then we focus on a more specific version that allows data downlink to be controlled by using either the priority or the maximum dump duration of each buffer.

Previous solutions to the MDP, including Max Flow and Linear Programming (LP) formulations, assume that data generation and downlink events do not overlap. We present a solution, called DALLOC, that uses a fast heuristic-based method to solve the more general oMDP. We then compare it to Max Flow as well as other heuristic methods using actual mission data from the European Space Agency's Rosetta mission. The ESA science operations team has been successfully using DALLOC to solve the oMDP in both strategic and tactical science planning.

1 Introduction

Spacecraft enable us to explore Earth, our solar system, and bodies beyond our galaxy to the furthest reaches of the universe. However, determining operations of these spacecraft (e.g. Mission planning and scheduling) is an extremely challenging part of 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 in the volume of acquired data, or deadlines for downlinking certain types of data, or buffers with dynamic priorities. We describe a particularly challenging downlink problem, the oMDP, in which data generation may occur over extremely long periods of time, overlapping with long downlink periods. We then describe the heuristic solution used by DALLOC, and compare it to two alternative heuristics and a Max Flow solution.

2 The Overlapping Memory Dumping Problem (oMDP)

Downlink scheduling is a sub-problem of the larger task of scheduling spacecraft activities. From a science planning/scheduling perspective, when constructing the schedule for the first time, the scheduler must decide on which observations to include, where they should occur, as

well as which downlink commands to issue to best satisfy science requests (e.g. for Rosetta [Chien et al. 2015]). When constructing and evaluating these observation schedules, the impact of the observations on spacecraft resources, such as memory, must be managed.

In this paper, we assume that a set of observations has been selected, and we focus on finding the best way to downlink data, thus freeing up memory used to store those observations. We focus on the scheduling of downlink commands only, assuming that the observation schedule cannot be changed. Fill rates from observations, and dump rates from downlinks, are all provided as inputs to the scheduler. As mentioned, this is a sub-problem of the strategic mission planning/scheduling process [Costa et al. 2016] where observation scheduling and downlink scheduling are performed either simultaneously or interleaved [Ayucar et al 2016]. In addition, this type of downlink re-scheduling is often necessary during short-term, tactical planning when certain last-minute changes must be made (e.g. due to the loss of a downlink).

Our problem was first discussed in [Rabideau et al. 2015] and is similar to the Mars Express Memory Dumping Problem (MEX-MDP) described in [Oddi and Policella 2004]. While we discuss this problem in the context of the Rosetta mission [Rosetta 2015], most space missions handle downlink/data volume scheduling similarly.

The general problem we solve is to specify an "empty" function that utilizes a fixed set of downlink periods to keep a set of onboard memory buffers well within their pre-defined limits. We formalize the data downlink problem as follows:

Given:

a time range T

a set of buffers $B = \{b_1, b_2, ... b_n\}$
where each b_j has
  an initial volume state: $init\_vol_j$
  a final volume requirement: $end\_vol\_req_j$
  a hard volume capacity: $capacity_j$
  a required margin: $margin_j$

a set of buffer fillers $F = \{f_1, f_2, ... f_n\}$
where each $f_i = <start\_f_i, end\_f_i, rate\_f_i>$

a set of downlinks $D = \{d_1, d_2, ... d_n\}$
where each $d_i = <start\_d_i, end\_d_i, rate\_d_i>$

Solve:

$\forall d_i, \forall b_j$: assign a function $empty(b_j,t) \rightarrow rate$ s.t.:

$\forall t \in [start\_d_i, end\_d_i]$: $\Sigma\ empty(b_j,t) \leq rate\_d_i$
(cannot empty more than the downlink capacity)

$\forall t \in T$: $volume(b_j, t) \leq capacity_j - margin_j$
(cannot exceed the buffer capacity minus margin)

$t = max(T)$: $volume(b_j, t) \leq end\_vol\_req_j$
(cannot exceed the end volume requirement)

$\forall b_j$: $minimize\ max(peak\_percent(b_j))$
(maximize robustness)

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 $empty(b_j,t)$ function is not as flexible as desired.

Note that in our problem formulation, the downlinks and fill function are specified over all time. Therefore, the data generation and downlink events can occur concurrently. Indeed, in Rosetta operations, downlinks cover greater than half of all time and on average seven data generation events are occurring at any point in time. Thus Rosetta represents a case where prior problem formulation assumptions of non-overlap between data production and downlink [Cesta et al. 2007, Righini and Tresoldi 2010] most definitely do not hold.

3 Controlling Data Downlink with Priority and/or Duration

As with most spacecraft, Rosetta onboard data storage is partitioned into a set of buffers, called packet stores, for different types of science and engineering data that is accumulated from observations. Each instrument has a designated buffer with a specified hard upper volume limit that cannot be changed during routine scheduling.

The behavior of each downlink can be controlled in two ways: by setting priority or by limiting duration.

- First, a priority can be assigned to each of the memory buffers, indicating a relative downlink order.
- Second, downlink of a specific buffer can be halted at any time, effectively limiting the duration of data dump from that buffer.

Priority and duration are the only decision variables available to the scheduler for controlling the "empty" function described earlier. Therefore, in this formulation, the control variables are:

priorities $P = \{p_{1,1}, p_{1,2}, ... p_{i,j}\}$ for each $d_i \in D$ and $b_j \in B$
durations $U = \{u_{1,1}, u_{1,2}, ... u_{i,j}\}$ for each $d_i \in D$ and $b_j \in B$

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 Rosetta downlink software in the following set of rules.

- Some of the buffers (used for high-priority engineering data) have fixed priorities and cannot be halted (they must dump first, and until they are empty).
- A buffer remains "active" until a command is issued to stop it, after which no data will be downlinked regardless of priority.
- When more than one active buffer has data waiting to be downlinked, the one with higher priority will be dumped first.

- If more than one active buffer all have the same priority, data will be downlinked round-robin.
- When a buffer becomes empty, downlink for that buffer will stop, allowing downlink to start on the next highest priority buffer.
- Downlink from a buffer will be preempted when new data is added to an active, higher-priority buffer.

Using these downlink rules, and the two control variables, the primary goal of the downlink scheduler is to prevent overflow on all buffers. The secondary goal of the scheduler is to make selections that respect a minimum margin and maximum carryover. And finally, it is preferred to have margins as large as possible, making the schedules more robust to uncertainties in data collection (e.g. compression ratios) and downlink availability.

To achieve these goals, the scheduler must first model the behavior of the buffers so that volume and overflows can be accurately predicted. This is accomplished using the activities and timelines of the ASPEN scheduling system. With a model of how data is collected and downlinked, ASPEN generates a profile for each buffer 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 buffer that will overflow sooner, or exceed its limit by more than any other buffer, then that buffer should be given higher priority or more time to downlink.

Because of the serial nature of the resulting dump schedules, using stop dump commands to allocate fixed downlink volumes is brittle to changes in those downlinks. If downlink times change (to start later, end earlier, or with an interruption in the middle), stop dump commands that fall during deleted downlink periods will be ignored. Losing these commands will cause some buffers to dump much longer than needed, consuming time needed by other buffers.

Originally the Rosetta mission used a fixed set of pre-assigned buffer priorities and selected only the duration for each dump. To address the brittleness of this approach, the Rosetta mission switched to a priority-based method, which assigns different priorities to buffers and does not explicitly halt data dumps. This method offers less control over the exact amount to downlink from each buffer, but is more robust to changes in the downlink schedule. Potentially, both priority *and* duration could be used to control the dump schedule and increase control and robustness - this topic is left for future work.

In this paper, we discuss a set of value selection heuristics for the overlapping Memory Dumping Problem (oMDP), and evaluate their performance on selecting either dump priorities or dump durations.

4 Downlink Parameter Value Selection Heuristics

In Rosetta operations, the DALLOC software tool is used to assign buffer priorities or durations based on the number of downlinks that exist before the first overflow of that buffer. Roughly speaking, this "downlink count" heuristic used by DALLOC will assign higher priorities or longer durations to buffers with earlier overflows. This ensures that more downlink time is given to the buffers with more urgent need.

For comparison purposes, we have implemented three alternative heuristics/methods for selecting either downlink duration or priority within the DALLOC framework. In all, we have:

1. Downlink count
2. Random
3. Percent full
4. Max flow

In the "random" heuristic, values were independently selected at random to create a lower bound for comparison. When assigning priorities, one of the available priority levels is randomly selected. When assigning durations, a set of random numbers for the buffers is normalized across the total available downlink duration.

The "percent full" heuristic assigns priorities (or durations) by normalizing the peak volume percentages across the available priority values (or across the downlink duration). In other words, the relative priority or duration assigned is proportional to the relative percent full for the peak of that buffer.

Finally, we compare our local heuristics against "max flow" which uses flow values that result from running the Edmond-Karp max-flow algorithm on the network constructed for the overlapping Memory Dumping Problem (oMDP). The MEX-MDP very closely matches the oMDP, allowing us to use a similarly constructed network. Here, "flow" represents data flowing into the buffers, out via downlinks, and carrying over to the next downlink (or end of the planning period). When the max-flow algorithm completes successfully, dump durations for each buffer can be extracted from the resulting graph. If selecting priorities, the "flow" value is converted to a priority by looking at it as a percent of the downlink available.

One distinction with Rosetta and the oMDP, however, is that buffer store and dump activities can occur over long periods of time (e.g. hours) and often overlap. In the max-flow formulation, these activities must be modeled as instantaneous events. The resulting flow values, therefore, are not guaranteed to prevent overflow when the buffer profile is created.

In "max flow", the entire schedule is evaluated to compute control variables (dump durations or priorities) for all downlinks at once. This can help ensure that selected values for one downlink do not adversely impact what can be done in a later downlink (i.e. prevents "painting into a corner"). However, the run-time for such a global evaluation can be significantly longer than local methods. All other heuristics use more local methods, selecting parameters for a downlink

without much consideration for other downlinks. However, because we update buffer volumes after scheduling a downlink, this new information can be used when applying the heuristic to subsequent downlinks. While this lack of global information may lead to sub-optimal heuristics, it makes the computation very fast. Efficiency is important when downlink scheduling must be performed repeatedly during the construction of observation schedules, as done in the strategic planning phase.

All heuristics are compared in the empirical evaluation section of this paper.

5 Schedule Robustness and Iterative Leveling

As in [Oddi and Policella 2004], we are interested in producing schedules that are robust to unpredictable events that occur after committing to the schedule (e.g. after uplink). For comparison purposes, we use the same approximation to schedule robustness, which uses the maximum percent full that any buffer is predicted to be at any time.

In [Oddi and Policella 2004], max-flow is used to find a solution for *all* downlinks. This does not maximize the flow through any *individual* downlink, which can produce solutions that contain buffers that are near capacity at specific times. These solutions are considered brittle, and an "Iterative Leveling" technique is presented to improve robustness. Here, more robust solutions are generated by assigning an epsilon smaller capacity to the brittle buffer after each iteration.

We present a variant of iterative leveling that reduces all capacities to the same level instead of one-at-a-time, and iterates using binary search instead of epsilon reduction. First, we recognize that limiting one buffer to a lower percent does not help robustness if other buffers are allowed to increase above the previously identified maximum. For example, if one buffer is limited to 90%, all should be limited to 90%. Therefore, binary search can find a more robust solution by using an artificial capacity that is reduced when a solution is found, or increased when the solution results in overflows.

6 Estimated Computational Complexity

Figure 1 contains high-level pseudo-code for downlink scheduling using a global max-flow formulation, using local heuristics, and finally the outer loop that adds iterative leveling with binary search. We use the following variables to analyze the computational complexity of these downlink scheduling methods:

D = downlinks
B = buffers
C = capacities
F = fill rate changes

First, we compute max-flow values using the Edmond-Karp implementation, which is $O(EV^2)$ where E is the number of edges and V is the number of nodes. In the MDP flow network, there are only a few nodes and edges for each buffer dump [Oddi and Policella 2004]. Therefore, E and V are each approximately equal to D*B, making the overall complexity of solving the MDP with max-flow $O(D^3B^3)$.

```
scheduleWithMaxFlow(D, B, C, F)
  M = computeMaxFlow(D, B, C, F)
  for each d in D
    for each b in B
      assignValue(d, b, M[d][b])

scheduleWithHeuristic(D, B, C, F)
  sort(D)
  for each d in D
    for each b in B
      c = C[d][b]
      heuristicallyAssignValue(d, b, c)
    for each f in F
      recalculateVolumeAt(f)

iterativeLeveling(D, B, C, F)
  Cdelta = 100
  Cprev = 0
  while(Cdelta > 1)
    Cdelta = abs((C – Cprev) / 2)
    Cprev = C
    if(USE_MAX_FLOW)
      r = scheduleWithMaxFlow(D, B, C, F)
    else
      r = scheduleWithHeuristic(D, B, C, F)
    if(r)
      C -= Cdelta
    else
      C += Cdelta
```

Figure 1: Scheduling with max-flow, with local heuristics, and iterative leveling

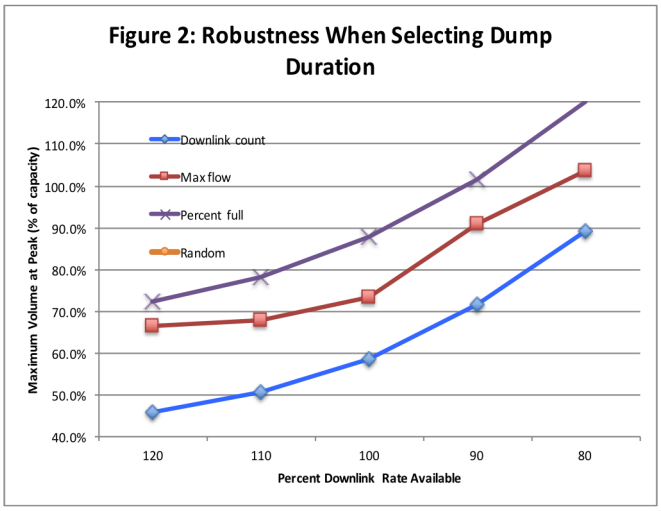
For all but "random", the local heuristics require an initial sort of the downlinks, and a propagation of volumes after each downlink assignment. Sorting the downlinks is O(DlgD). Because the downlinks are scheduled forward in time, recalculating volumes after the last scheduled downlink is simply proportional to the number of fill rate changes. The resulting complexity of scheduling using a local heuristic is O(DlgD + D*F).

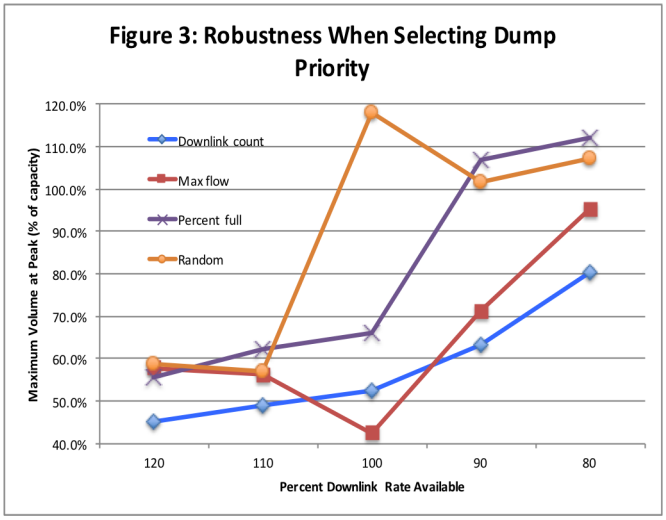
For any of the heuristics, performing iterative leveling with binary search will add a constant multiplier. This is because our implementation works on an integer percentage between 1 and 100, which will loop at most lg100 times (about 7 times).

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 buffers, and for one MTP, there are typically 30+ downlinks and hundreds of fill rate changes.

7 Empirical Evaluation

We have conducted an empirical evaluation of the scheduling algorithm using the four previously mentioned heuristics for assigning dump durations or priorities. Performance of the





heuristics based on run-time and schedule robustness. We use data from four medium-term planning (MTP) periods during the comet escort phase of the Rosetta mission. The data collected during each MTP is roughly the same as the downlink available. This is because the fill rate data we use was taken from an archive of the tactical planning process, where the strategic selection of observations has already completed. In addition, the data collected from these observations is typically about 2x to 3x the total capacity of all buffers.

First, in order to evaluate the performance over a range of constrainedness, we varied the downlink rates from 80% to 120% of the true rate used in operations. Then, we look at how each heuristic compares when selecting either dump duration (Figure 2) or dump priority (Figure 3).

When selecting dump duration, the relative performance of the heuristics does not change as the problem becomes more or less constrained. As expected, selecting random durations results in schedules that are the most brittle (resulting volumes were greater than 200% and therefore do not appear on the graph in Figure 2). Surprisingly, using the max-flow

	No Leveling	Leveling
Downlink count	14.6	92.6
Max flow	337.5	2190.0
Percent full	11.0	77.3
Random	11.1	76.8

Table 1: Average run-times (seconds)

	No Leveling	Leveling
Downlink count	64.2%	55.6%
Max flow	85.4%	57.9%
Percent full	76.4%	77.1%
Random	162.9%	175.9%

Table 2: Average robustness for the actual (100%) downlink rate

values does not produce the most robust solutions. The "downlink count" heuristic, which selects duration based on the number of downlinks before the first overflow, consistency outperforms max-flow.

Next we look at the performance of heuristically assigning buffer priorities without changing dump durations (Figure 3). Recall that this will simply control the order in which buffers are downlinked. Because max-flow solutions contain volume assignments, we first convert flow values to priorities based on the flow volume as a percent of capacity. Using the actual downlink rate (100%), when fill and downlink volumes are about the same, max-flow outperforms all other methods by at least 10%. But when over- (<100%) or under-constrained (>100%), we see that the "downlink count" heuristic outperforms max-flow by about 10%.

Results from max-flow were surprising since we had expected a global solution to consistently outperform any of the local heuristics. First, we must consider that max-flow was not designed to assign priorities, which could explain the results in Figure 3. For under-constrained problems, one possibility is that max-flow is relying more heavily on iterative leveling to keep the peaks low. For over-constrained problems, max-flow is more likely to fail to find a solution, which could contribute to the drop in robustness. In all cases, max-flow models the filling and downlinking events as instantaneous, while in reality, these activities have durations and even overlap. This modeling inaccuracy may be producing suboptimal solutions.

In addition to comparing the robustness of these various methods, we are also interested in run-times in order to determine whether the benefits outweigh the costs. Table 1 reports the run times (real CPU time in seconds) of max-flow and each of the local heuristics, with and without iterative leveling. Table 2 reports the robustness (max peak volume percent) obtained, averaged from both priority and duration assignment, using actual (100%) downlink rates. As

expected, compared to using local heuristics, the cost of running max-flow is high (roughly 20x slower). Moreover, solutions are not much better, and in some cases worse than those produced by the best local heuristics. Next, we consider the cost-benefit tradeoff of using iterative leveling. As shown in the complexity analysis, the cost is a multiplier of lg100, or about 6.6x. This is consistent with the observed data in Table 1. In Table 2, we see that iterative leveling, as expected, is more important for max-flow where no peak minimization is attempted in a single iteration.

In our experience, the best combination is the use of iterative leveling with a local heuristic. This allows the tool to be more responsive during tactical planning, but also enables its use as part of the strategic planning process, where it may need to be run hundreds of times to generate a single MTP schedule.

8 Discussion

The downlink scheduling algorithms described in this paper were originally deployed as part of the Rosetta early science planning operations tool that also scheduled science activities [Chien et al. 2015]. Later in operations the data downlink scheduling software was modularized and extracted to also be used in mid to late tactical science planning. It is this separated scheduling software and associated algorithms that we describe in this paper. In some form or other this downlink scheduling software has been in use to schedule over 18 Medium Term Plans (each ~1 month of Rosetta Orbiter operations).

Automated downlink planning is in operational use for the Mars Express mission [Cesta et al. 2007]. However, they model observations and downlinks as non-overlapping (or equivalently instantaneous) data producers and consumers. In many space missions, including Rosetta, this assumption does not hold. 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. 2015]. 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 this paper).

9 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 overlapping data creation and downlinks. We describe heuristic solutions to this new problem that are in operational use for ESA's Rosetta mission and show that they outperform prior max-flow methods that do not handle overlapping effects on Rosetta mission data.

Acknowledgments

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

[Cesta et al., 2007] 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.

[Chien et al., 2005] 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.

[Chien et al., 2010] 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.

[Chien et al., 2012] 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.

[Chien et al., 2015] S. Chien, G. Rabideau, D. Tran, J. Doubleday, D. Chao, F. Nespoli, M. P. Ayucar, M. Costa, C. Vallat, B. Geiger, N. Altobelli, M. Fernandez, F. Vallejo, R. Andres, M. Kueppers, Using Constraint-based Search to Schedule Science Campaigns for Rosetta

Orbiter, Invited Talk, Proc. International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 2015.

[Costa et al., 2016] M. Costa, M. Perez-Ayucar, M. Almeida, M. Ashman, R. Hoofs, S. Chien, J. Beteta, M. Kueppers, “Rosetta: Rapid Science Operations for a Dynamic Comet, Space Operations Symposium, Daejon, Korea, 2016.

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

[Knight et al., 2014] R. Knight, C. Chouinard, G. Jones, D. Tran, Leveraging Multiple Artificial Intelligence Techniques to Improve the Responsiveness in Operations Planning: ASPEN for Orbital Express, AI Magazine, Vol 35, No 4, 2014.

[Oddi and Policella, 2004] A. Oddi and N. Policella, A Max-Flow Approach for Improving Robustness in a Spacecraft Downlink Schedule, International Workshop on Planning and Scheduling for Space (IWPSS-04). Darmstadt, Germany. June 2004.

[Perez-Ayucar et al., 2016] M. Perez-Ayucar, M. Almeida, M. Ashman, S. Chien, M. Costa, J. Garcia, R. Hoofs, M. Kueppers, D. Merritt, J. Marin, F. Nespoli, G. Rabideau, E. Sanchez, “Science Data Volume management for the Rosetta Spacecraft, Space Operations Symposium, Daejon, Korea, 2016.

[Pralet et al., 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.

[Rabideau et al., 2015] G. Rabideau, F. Nespoli, S. Chien. Heuristic Scheduling of Space Mission Downlinks: A Case study from the Rosetta Mission. International Workshop on Planning and Scheduling for Space (IWPSS-15). Buenos Aires, Argentina, 2015.

[Righini and Tresoldi, 2010] Righini G, Tresoldi E. A mathematical programming solution to the Mars Express memory dumping problem. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on. 2010 May;40(3):268-77.

[Rosetta Mission, 2015] http://rosetta.esa.int/, retrieved 16 November 2015.

[Simonin et al., 2012] 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.