Commentary on Paper 026: A Max-Flow Approach for Improving Robustness in a Spacecraft Downlink Schedule

Gregg Rabideau

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
gregg.rabideau@jpl.nasa.gov

Abstract. In this paper I will discuss the paper titled "A Max-Flow Approach for Improving Robustness" and authored by Angelo Oddi and Nicola Policella from the Planning & Scheduling Team, Italian National Research Council. The authors describe a solution to the MARS EXPRESS Memory Dumping Problem (MEX-MDP). Their solution involves reducing the problem to a Max-Flow problem and making use of standard algorithms to find solutions. They also describe an iterative procedure for increasing the robustness of solutions. My comments will focus mostly on how these techniques might be adapted by a new project and how they might be integrated into a mission operations system.

1 Introduction

The paper offers a novel and promising approach to solving a very common problem in spacecraft operations. Specifically, the authors address the problem of generating a spacecraft operations schedule to downlink on-board data to Earth. The operations for collecting the data are assumed to be fixed and include both requested science data and continuous engineering (or housekeeping) data. Several typical constraints are considered during scheduling, including on-board memory capacity and communication bandwidth. The problem is first redefined as a well-known computer science problem – the Max-Flow Problem. By doing this, the problem can be solved using well-studied and highly optimized algorithms. The authors also tackle the added complexity of execution uncertainty. During operations, the actual amount of data generated on-board may be more than expected (due to unknown compression factors) and the actual amount of data downlinked may be less than expected (due to unknown downlink rates). To address this problem, the authors present an iterative algorithm for increasing the robustness of the downlink schedule. Robustness is roughly defined by the maximum percent usage of any on-board memory bank (packet store) at any point in time. The smaller the maximum, the more robust the schedule, and the more amenable it is to execution uncertainty. The authors present a simple and clever algorithm that works to reduce this maximum value.

2 Using Max-Flow

The MARS EXPRESS Memory Dumping Problem (MEX-MDP) is solved by first mapping the specific problem to the more general Max-Flow graph (or network) problem. The Max-Flow problem seems like the perfect match for MEX-MDP and the mapping described in the paper is very intuitive. Activities, resources, and constraints from the MEX-MDP become nodes and edges in the Max-Flow graph. One interesting detail is the introduction of a "tuning" parameter for the maximal level of a packet store. This parameter, a value between 0 and the store capacity, will be used later in the iterative-leveling algorithm. The remaining details of this transformation are clearly stated in the paper. However, it is important to note that many of these details can be hidden from the end user by automatically generating the graph structure from a simple problem description. In other words, the user would only need to specify how many data stores are available, their capacities, etc.

Once this transformation is complete, one of the existing graph algorithms can be used to find a solution. The run-time of the chosen algorithm scales with the number of packet stores and time points, where the number of timepoints is roughly proportional to the number of store operations, memory dumps, and downlink rate changes. A particular solution to the Max-Flow problem directly corresponds back to the original MEX-MDP. In other words, the result will tell you if there is enough memory and bandwidth to downlink all data being collected. If a solution exists, the downlink schedule can be extracted from the result. In general, it seems that the paper assumes that the problem is under-constrained and it is not clear what would be done if no solution is found.

3 Uncertainty, Robustness, and the Iterative-Leveling Algorithm

In the MEX-MDP, there is uncertainty in the amount of data stored (due to on-board data compression) and the amount of data downlinked (due to downlink rates). This means that the actual amount of data stored in memory at any time may be more than the predicted amount (which is most likely based on average compression and downlink rates). The schedule should be robust to accommodate small differences that arise from these uncertainties. If the memory use is close to the capacity, then there is a greater chance of losing data. To address the difficult problem of execution uncertainty, the authors present the Iterative-Leveling algorithm to increase schedule robustness. On each iteration of the algorithm, the critical packet store is identified, the maximum level parameter is decreased, and the Max-Flow algorithm is run again. The critical packet store is the one that is (at any time) closest to its maximum level parameter. The maximum level parameter was introduced to provide a way to "tune" the Max-Flow algorithm. When the maximum level is set to the memory capacity, the Max-Flow algorithm generates a *consistent* but possibly sub-optimal solution. By decreasing the value, the algorithm generates a new solution that *improves* on the worst part of the previous solution.

4 System Integration

It is interesting to think about how these algorithms could be used in conjunction with a full planning system as part of a mission operations system or as a flight software component. First, the authors have focused on a sub-problem to a larger spacecraft planning problem. We would like to solve the full planning problem while taking advantage of the techniques and results of the authors. Specifically, the MEX-MDP uses robustness to define schedule quality. More generally, schedule quality can be thought of as some measure of data return. Schedule robustness is a key factor but other factors must be considered. For example, what if the maximum memory levels were reduced so much that more Payload Operations Requests could be added? Should they be? Adding them could return more data, but will also decrease robustness and increase the likelihood of losing data. As the authors discuss in Future Work, many similar trade-offs exist in the larger planning problem.

To focus on a specific problem, the paper assumes that the operations for collecting data have been provided as input and do not change. In practice, to solve the larger problem we need to generate the data collect schedule as well as the downlink schedule. The author's approach to the sub-problem seems well suited for use as a sub-component of larger problem solvers. For example, similar to the Iterative-Leveling algorithm, one could imagine another iterative algorithm that adds or removes Payload Operations Requests and runs the Max-Flow algorithm to generate a downlink schedule for the new problem. Performance may become a problem, however. While the polynomial time of Max-Flow is fast in theory, when used on large practical problems, and as only one step in a repeating process, it may prove to be too expensive.

While what the authors present is new research, it is based on proven algorithms and could be part of a flight software system. Flight qualified CPUs are typically much slower than desktop computers, sometimes making even polynomial time algorithms too slow to be used for critical operations. Generating the first downlink schedule with Max-Flow is critical, but only needs to be done once. Iterative-Leveling, on the other hand, is a non-critical optimization algorithm. It is a natural any-time algorithm that always has a solution, and can be run in the background to continuously make the schedule more robust when CPU is available. In addition to addressing execution uncertainty, it is interesting to note that the Iterative-Leveling algorithm could also make the schedule robust to changes in Payload Operations Requests. For example, a request submitted at the last minute is more likely to fit into a robust schedule. If an on-board planner is used to schedule Payload Operations Requests, then Iterative-Leveling makes the downlink schedule more robust to unexpected re-scheduling of the store operations due to execution failures.

5 Conclusion

The authors present a unique and effective approach for generating and optimizing data downlink schedules. This is a very common problem for space mission operations. The algorithms presented are general and seem likely to be easily adapted to new domains. I look forward to future work in this area of planning and scheduling.