To appear in 10th International Workshop on Planning and Scheduling for Space (IWPSS 2017) 15-16 June 2017

Intermediate Fidelity Solid State Recorder Modeling for NISAR

Michael Trowbridge and **Joshua R. Doubleday**
Jet Propulsion Laboratory, California Institute of Technology
{michael.a.trowbridge,joshua.r.doubleday}@jpl.nasa.gov

Abstract

At 26Tb/day return average, NISAR (National Aeronautics and Space Administration (NASA) and the Indian Space Research Organization (ISRO) Synthetic Aperture Radar) will stress ground station facilities with a variable cadence of data generation observing nearly all of earth's land and ice to produce global time-series maps of cryospheric, solid-Earth and ecosystem phenomena. Between the instrument data generation source and downlink sink lies the primary science data buffer, the SSR (Solid State Recorder).

The SSR has deferred, discrete file deletion events that occur only after a file is played back to all required ground segments (sometimes both NASA and ISRO). A forward dispatch, advancing frontier repropagation playback scheduler is proposed as a model of the SSR's greedy playback scheduling behavior. Schedules based on the existing low fidelity model and new intermediate fidelity model are compared to assess the impact of deferred, discrete deletion on planning. The low fidelity model is found to produce unfeasible schedules that exceed SSR capacity, causing 3.87% of all observations in a 12 day simulation to be removed by repair actions.

Introduction

NISAR (National Aeronautics and Space Administration (NASA) and the Indian Space Research Organization (ISRO) Synthetic Aperture Radar) is a joint India-US science mission that observes solid Earth land mass, ice masses and ecosystems from a 747 km low Earth orbit over a three year mission. NASA's instrument (L-band SAR) is a hosted payload on an ISRO bus, and will sometimes be operated in tandem with ISRO's S-band SAR instrument. The nations will share science data, but their ground segments are not federated. Science data will sometimes be played back twice – once to NASA and once to ISRO – before it is deleted.

The variable playback policy is a significant planning complication. The two instruments record to a common data recorder, which plays back observations from different combinations of the two instruments to different combinations of ISRO and NASA's separate ground segments. It is as much a store-and-forward routing message queue as it is a data buffer (figure 1).

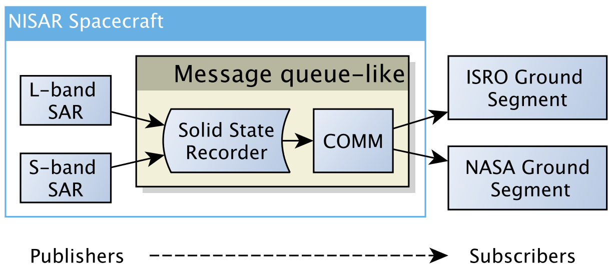

Figure 1: Solid State Recorder modeled as a message queue

Downlink order is nominally by priority, then age, with preemptions possible. We only use our full playback rate if we play two files simultaneously, but hardware prevents us from playing S-band and L-band files at the same time. We can play two L-band or two S-band files at the same time, but this works against the age sort. Simulation shows files lingering on the data recorder as long as 34 hours (figure 2).

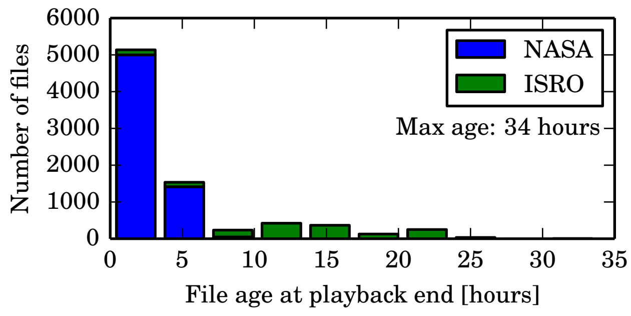

Figure 2: File age at playback end over a 12 day, preemption-free simulation.

Figure 3 shows recorder fill state when deferred deletions occur. Note how the fill level (blue line) does not

Contact author: Michael Trowbridge

drop after any of NASA's playbacks, but does drop, one file at a time, after each of the final (ISRO) playbacks finish. The three abrupt, stair-step decreases between 400 and 600 seconds are the discrete file deletions.

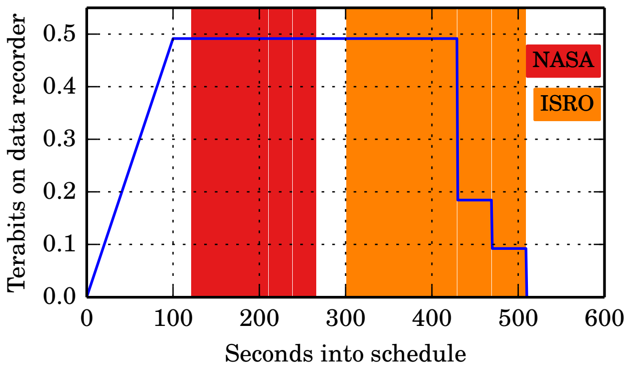

Figure 3: Deferred, discrete file deletions after playback to both ground segments. Vertical orange and red bands are playbacks.

How significant are the effects of these delayed deletions and deferred playbacks? If they can be ignored, we can use a simpler, faster model of a continuous fill/drain integral quantity (Knight and Hu 2009) and add margin to our mission plan. If deferred playbacks and deletes are significant, we may need to adjust our scheduling approach. This paper explores that question by passing a schedule based on a fill/drain integral recorder to a detailed file system model and playback scheduler to identify storage capacity violations. We also describe a forward dispatch playback/delete scheduling algorithm used in this experiment.

Related Work

Cesta et al. defined the related Mars Express Memory Dumping Problem (MEX-MDP) as choosing a sequence of science data buffer playbacks and playback durations such that science data is delivered as soon as possible, subject to data priority (Cesta et al. 2002). The solution space is the sequence of playbacks, where the observational schedule is considered fixed. MEX-MDP solutions include a greedy forward dispatch algorithm, tabu search (Cesta et al. 2002) and an optimal linear programming solution (Righini and Tresoldi 2010). MEX-MDP is similar to NISAR's memory management problem in that science data is separated into different banks, data priority is a constraint and playback preemption is possible (Cesta et al. 2002).

MEX-MDP and NISAR memory management differ in several ways. MEX-MDP has one receiving ground segment, where NISAR has two receiving ground segments and file-specific space-to-ground routing policies. In MEX-MDP, failure to play back data does not prevent future observations because the spacecraft (Mars Express) autonomously overwrites old data in a FIFO, circular buffer manner (Cesta et al. 2002). NISAR, on the other hand, will stop recording observations when recorder capacity is exceeded. The unit of deletion is more granular in MEX-MDP, where NISAR does not piece-wise delete a file; it defers deletions to large, bulk-free events. Lastly, in MEX-MDP, the observation schedule is fixed and the solution space is the playback sequence (Cesta et al. 2002; Rabideau et al. 2016), where in NISAR the observation schedule is the solution space and playback behavior is fixed because the recorder/playback scheduler is an existing, third-party hardware component.

The MEX-MDP formulation was also applied to the Rosetta mission (Rabideau et al. 2016). The max-flow implementation for the Rosetta version of MEX-MDP added an aspect similar to NISAR - data reservations and post-playback frees that are instantaneous, discrete events (Rabideau et al. 2016). Like the original MEX-MDP, Rosetta playback planning treated the observation schedule as fixed and the playback schedule as solution space (Rabideau et al. 2016).

Piece-wise linear integrals with simultaneous fill/drain have been used to model satellite data recorders in ASPEN (Fukunaga et al. 1997; Knight, Donnellan, and Green 2013) and CLASP (Knight and Hu 2009). The Mars EXpress scheduling ARchitecture (MEXAR) also modeled simultaneous fill and playback events (Cesta et al. 2002).

The University of Surrey built and operated more than one store-and-forward message routing satellite in the 1980's and 1990's, noting that they were but one of many such organizations (Ward 1990). These systems had frequent contact with the ground and were primarily designed for reliable message delivery, potentially using more storage and with lower throughput. Our goal for NISAR is different - empty the data recorder as quickly as possible, because the science mission is limited by data throughput (Doubleday 2016).

Joslin and Clements introduced Squeaky Wheel Optimization in 1999 (Joslin and Clements 1999), an algorithm for fast schedule optimization in massively oversubscribed systems based on greedy schedulers. Knight et al. applied it to pushbroom sensor scheduling for Earth observing satellites in the context of NISAR's predecessor, the DESDynI mission (Knight and Hu 2009; Knight, McLaren, and Hu 2012; Doubleday and Knight 2014). Prior versions of CLASP treat each spacecraft's on-board storage as a piece-wise linear fill/drain integral, where playing back to any ground segment allows constant rate draining during the playback. Higher fidelity models were added to CLASP during NISAR phase B preliminary design (Doubleday 2016), but these versions did not model end-to-end playback scheduling with discrete deletion events.

Forward dispatch is an older approach that has been used for job shop scheduling (Fox et al. 1983) and for space systems by ASPEN (Fukunaga et al. 1997) and MEX-MDP[1] (Cesta et al. 2002). It is a greedy algo-

[1]The Greedy (One-pass) Heuristic Solver (Cesta et al.

rithm that moves forward in time from the planning horizon start, choosing its next task based on a criteria like value, utility or deadline (Chien 2009). Forward dispatch is a constructive algorithm that only chooses feasible options at each decision point, guaranteeing that the final schedule is also feasible.

NISAR is similar to the Soil Moisture Active-Passive (SMAP) mission. SMAP was designed to detect soil moisture content from low Earth orbit using synthetic aperture radar (SAR) and radio interferometry payloads (Entekhabi et al. 2010). Its SAR instrument payload, orbit and use of the NASA Near Earth Network (Choi 2012) are similar to NISAR. They both normally buffer their data to an internal data recorder before transmission to the ground (Deems, Swan, and Weiss 2012). SMAP's instrument look up table (LUT) is also included in NISAR design.

NISAR and SMAP schedule observations and delete files differently. SMAP always makes observations when the spacecraft is over a region of interest (Deems, Swan, and Weiss 2012), but NISAR must be more selective to preserve storage space for later observations. The two missions also delete data differently. SMAP retains data until it receives a deletion command from the ground and is capable of ground-directed replays (Choi 2012). NISAR autonomously deletes files after the final playback, with no confirmation of receipt by the ground before file deletion. SMAP's ground segment is more involved in file and playback management than observation scheduling - the reverse of NISAR.

Formulation

Scale is the core challenge of planning for NISAR. CLASP's Squeaky Wheel Optimization implementation is driven by failure to schedule observations. Observations can only fail to schedule if there is a conflicting instrument/orientation reservation or insufficient onboard memory to store the observation. As the search space is combinatorial, fast execution of the scheduling loop is required for meaningful optimization.

The scheduler cannot simply query the instantaneous value of the data recorder. Suppose we plan an observation at $t_2 > t_1$ that consumes 95% of the spacecraft's on-board storage. If we later schedule a request at t_1 that consumes 10% of the on-board storage, the prior commitment for the t_2 observation is broken, because only 90% of the recorder space is available at execution time. The schedule would be infeasible (figure 4).

CLASP uses a fast, linear integral model of the solid state recorder for squeaky wheel optimization. We apply the intermediate fidelity SSR model as a post-processing step and repair the schedule by removing observations that fail to place on the intermediate fidelity SSR model. We use this two-pass approach because the intermediate fidelity model is sensitive to the start time sequence of the observations scheduled and too slow for

2002)

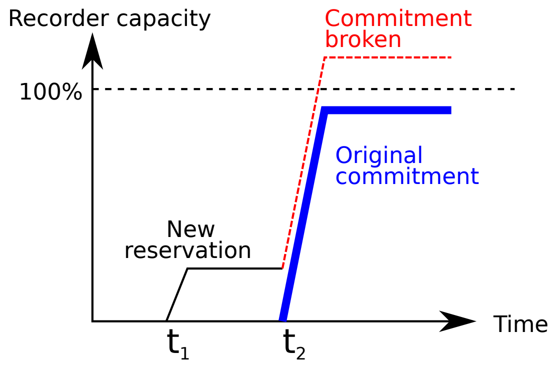

Figure 4: Considering only instantaneous state at t_1 can invalidate a future commitment at t_2.

individual scheduling operations during squeaky wheel optimization (see Discussion).

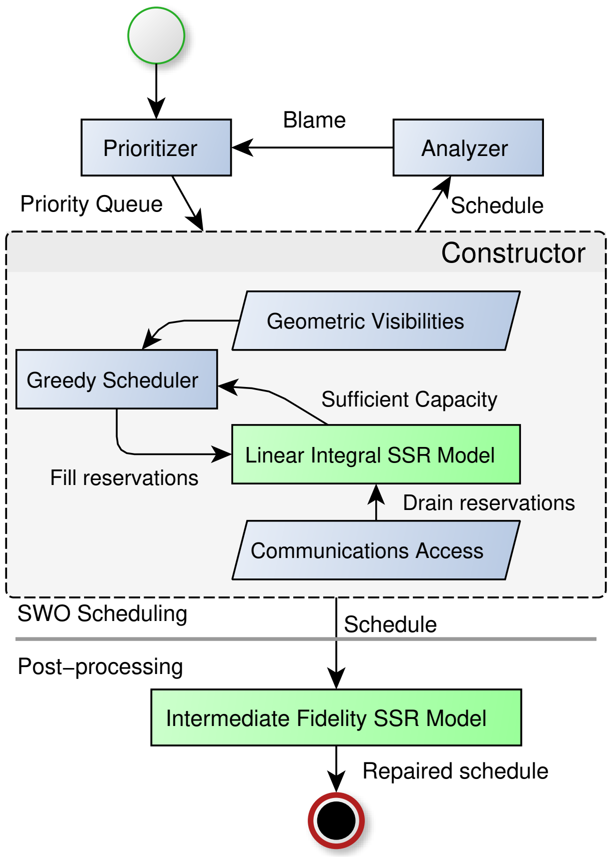

Figure 5: Squeaky wheel scheduler and two-pass SSR model

SWO Scheduling: Linear Integral Model

CLASP addresses future commitments by making a test reservation, propagating the changes to recorder fill (evaluating the integral), checking for violations, then undoing the test reservation. If the test reservation succeeds, we say that scheduling the t_1 observation is feasible. If t_1 is feasible, CLASP commits the on-board storage reservation. If the test reservation found t_1 to be infeasible, the higher level scheduler will attempt alternate different times or move on to different science targets.

This approach leads to significant on-board storage timeline propagation. For speed, CLASP approximates the downlinks as a piece-wise linear negative rate reservations on the data recorder with a lower bound clamp at 0 bits. At the beginning of each scheduling loop, downlinks are scheduled over an empty data recorder. The downlinks are not modified after they are placed at the start of the planning loop. Positive rate reservations for observations are later summed with any rate reservation that happened to be already present (downlink or observation) and numerically integrated (forward Euler).

This model makes several assumptions. It is possible to read and write a file at the exact same time, at the exact same spot. There are no bandwidth limitations into or out of the data recorder. We can read and write to the recorder at single-bit intervals. Each file must only be downlinked once before it is deleted and it doesn't matter which downlink window we use to play the file back. Nonlinear, discrete effects of block size and buffering are ignored, as the piece-wise linear integral is only an approximation of true system behavior.

Post-processing: Intermediate Fidelity Model

Some of the linear integral model assumptions conflict with the NISAR concept of operations. The greatest discrepancy is that some files must be played back to both NASA and ISRO ground segments separately before the file is deleted. Likewise, some observations will only go to NASA or ISRO, and we must wait until the spacecraft has a contact window with the appropriate nation's ground segment before the file can be played back. Furthermore, no available hardware architectures offer unlimited I/O throughput rates and discrete effects are large enough that we question the feasibility of a schedule produced with a monolithic, piece-wise linear data recorder model.

With the file system architecture above the model moves from managing an abstract pool of accumulated data to planning management of specific files. Knowing when a file can be played back is in part determined by knowing which parts of which observations were in it. The rules for deciding this are based on temporal adjacency in the timeline or explicit annotation by a human planner. This creates a circular dependency: file contents $\rightarrow$ schedule contents $\rightarrow$ recorder capacity $\rightarrow$ file contents.

We address the circular dependency by using the monolithic recorder model's schedule as a starting point that may or may not be feasible. We simulate execution of the schedule, assembling files from observations and playing back files when they are ready and the correct nation's ground stations are in view (algorithm 1). When an schedule observation cannot be added to a file because it would exceed the data recorder's capacity, we treat that observation as infeasible and repair the schedule by removing the infeasible observation from the schedule.

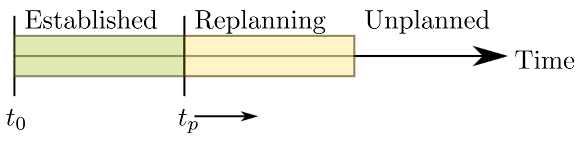

Figure 6: Replanning playbacks along an advancing forward dispatch frontier

Algorithm 1 Forward dispatch playback scheduling

```
$O \leftarrow$ SORT(Observations, by time, ascending)
$t_r \leftarrow t_0$                                   ▷ Failure rollback limit
$failed, success \leftarrow \emptyset$
for all $o \in O$ do
    $t_p \leftarrow t_f$                               ▷ Propagation start
    $r \leftarrow$ AGGREGATESIZE($o$)
    if RESERVE($r$) then                      ▷ Recorder reservation
        $journal \leftarrow \emptyset$
        for all $c \in o$.channels do
            $f \leftarrow$ GETFILEORDOPENNEW($o, c$)
            if $t_p > f$.start then $t_p \leftarrow f$.start
            end if
            $f$.append $(o, c)$
            $journal$.append $(f, o, c)$
        end for
        ROLLBACKTO($t_p$)
        if PROPAGATEDOWNLINKS($t_p$) then
            $success$.append ()
            $t_r \leftarrow t_p$        ▷ Update failure rollback limit
        else
            $failed$.append $(o)$
            ROLLBACK($journal$)
            ROLLBACKTO($t_r$)
            PROPAGATEDOWNLINKS($t_r$)
        end if
    else
        $failed$.append $(o)$
    end if
end for
```

We then replan the playbacks and deletes that were rolled back, including the file containing the newly added observation (algorithm 2). We stop propagating when either the planning horizon ends or the playback queues are empty for each nation's ground segment. This incremental, advancing frontier roll-back and repropagation frees up data recorder space for the next observation that will be planned.

Algorithm 2 Propagating downlinks

```
function PROPAGATEDOWNLINKS($t_p$)
    for all $g \in$ ground segments do
        $t \leftarrow t_p$
        $W \leftarrow$ GATHERDOWNLINKS($g, t_p$)
        repeat
            $W \leftarrow W$.intersectWith($t, W$.end)
            $f \leftarrow g$.queue.front()                ▷ Next file
            $t \leftarrow$ FINDNEXTSTART($g, t, f, W$)
            $finished, t_{end} \leftarrow$ PLAY($g, t, f, W$)
            if $finished$ then
                $g$.queue.pop()
                if PLAYEDTOALLRECIPIENTS($f$) then
                    $recorder$.unreserve($f$.size, $t_{end}$)
                end if
            end if
        until $g$.queue empty or $W$ empty or $t \geq t_f$
    end for
    return $recorder$.hasNoViolations()
end function
```

Algorithmic Traits of the Intermediate Fidelity Model

The theorems that follow assume that a playback schedule can only be infeasible because:

- Communications path is oversubscribed
- Playback outside of communications access
- Data recorder is oversubscribed

Theorem 0.1 (Correctness)**.** *The final forward-dispatch playback schedule produced by the intermediate fidelity solid state recorder model is feasible.*

Lemma 0.1.1. *A forward-dispatch playback sub-schedule produced by the intermediate fidelity solid state recorder model is feasible.*

Proof. Assume that the output schedule is unfeasible. One or more of the following is true:

- Communications path oversubscribed
- Playback outside of communications access
- Data recorder is oversubscribed

As the algorithm is a forward dispatch algorithm, when scheduling playback or delete activity j at $t_j \geq t_p$, we first check to see if the required resources (communications path channels, communications access to ground segment) are available. If they are not, we seek forward in time until they are both available ($t_j := t_{avail}, t_{avail} \geq t_p$). This contradicts the first two possible reasons for playback schedule infeasibility.

If the data recorder is oversubscribed, then intermediate-fi model must have permitted a reservation that either directly exceeded recorder capacity or undid a delete operation that would have kept the data recorder usage within data recorder capacity. Both of these conditions are criteria under which the intermediate-fi model will fail to place an observation, which contradicts that algorithm definition. □

Lemma 0.1.2. *A playback schedule generated by forward dispatch of feasible sub-schedules is feasible.*

Proof. By lemma 0.1.1, sub-schedule i will be feasible, as will sub-schedule $i+1$. As this algorithm is a forward dispatch algorithm, $t_{p,i+1} \geq t_{p,i} \geq t_0$. By induction, the final schedule, which is the combination of each locally modified sub-schedule from earliest to latest, will also be feasible. □

Theorem 0.2 (Boundedness)**.** *The forward dispatch playback scheduling algorithm in the intermediate fidelity solid state recorder will terminate.*

Lemma 0.2.1. *A forward dispatch sub-schedule propagation will terminate.*

Proof. Assume the sub-schedule propagation does not terminate. Each repropagation will terminate when one of the following occurs:

- Time cursor $t \geq t_p$ at or after horizon end t_f
- Playback queues empty (SSR empty)
- Playback opportunities exhausted

Playback queues are depletable work lists and opportunities are depletable resources consumed by playback scheduling. Trivially, these deplete when playbacks are scheduled, but cannot grow within a single downlink repropagation. As this is a forward dispatch algorithm, replanning cursor t advances when insufficient resources are available for the next playback and cannot move backwards. By induction, time will either remain the same or move forward until queues are not empty or horizon end. □

Lemma 0.2.2. *A playback schedule produced by advancing frontier forward dispatch repropagation will terminate.*

Proof. Each input observation can cause at most one repropagation. The number of observations is fixed at schedule time, so at most $2n$ repropagations (a failure followed by a rollback repropagation) will occur (in series). Applying lemma 0.2.1 by induction, the overall algorithm terminates. □

Theorem 0.3 (Complexity)**.** *Producing a forward dispatch playback schedule with the intermediate fidelity SSR model algorithm has complexity $O(n)$*

Proof. At most $2n$ repropagations are possible and each of the n observations are processed in series (see proof for lemma 0.2.2). The amount of repropagation work for each repropagation is bounded by the number of files in the playback queues at each repropagation time t_p, which is bounded by the capacity of the solid state recorder, which is fixed at run-time. □

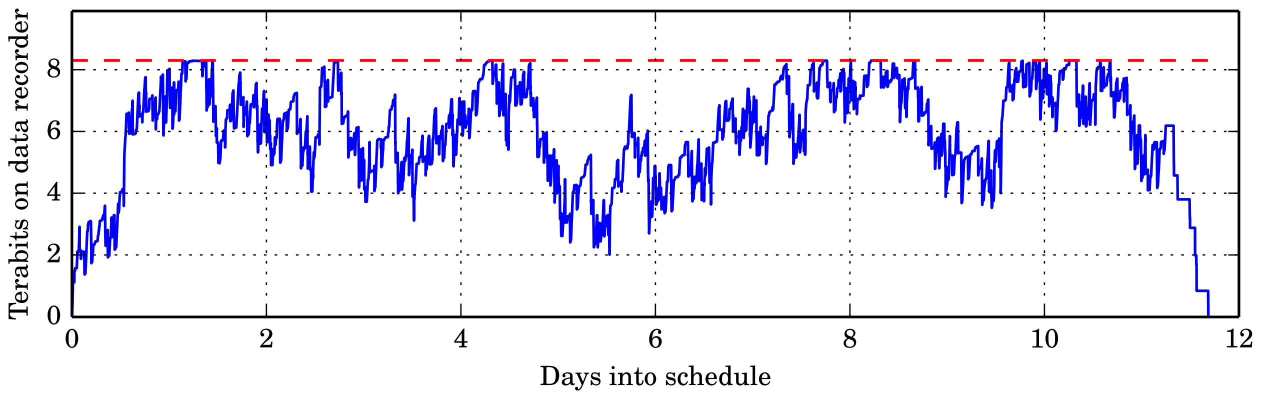

Figure 7: Data recorder fill over a 12 day simulation.

Experiment

To determine the impact of a deferred deletion policy with multiple playback destinations is, we will generate an observation schedule using the low-fidelity data recorder model. This schedule will be constraint-compliant, but based on simplifications described in the Formulation section. We will post-process the schedule using the intermediate-fidelity model and remove any observation that fails to place on the intermediate-fidelity SSR model.

If the effects are insignificant, all observations will successfully place on the intermediate-fidelity model. Significance of deferred deletion effects will be expressed as a percentage of the initial schedule's observations that are removed by intermediate-fidelity model repair actions.

Results

Of the 3902 observations in the initial schedule, 151 (3.87%) failed to place on the intermediate fidelity SSR model. While this is a small percentage, the intermediate fidelity model's repair action does not respect priority, so some of the removed requests could be urgent science requests. Figure 7 shows that extended periods near maximum recorder capacity (dashed red line) start every two days, sometimes for a whole day at a time. It is common enough that normal operating and scheduling policies should address the effects of deferred file deletions.

Discussion

When to use the intermediate-fi model

The forward dispatch planning approach is key to the boundedness and complexity theorems. If the planning cursor does not move semidefinitely forward in time, the worst case run time is $O\left(n^2\right)$ for reverse-dispatch (planning backwards from horizon end). In Squeaky Wheel Optimization, the planning cursor t_p moves in first-fit, scheduler priority order, where $t_{p,i} < t_{p,i-1}$ is possible. Applying the intermediate fidelity model this way invalidates the linear complexity proof in theorem 0.3.

If this model were used inside a squeaky wheel scheduling loop and only one insertion point was tested for each observation, we expect overall complexity between $\Omega\left(n\right)$ and $O\left(n^2\right)$. If multiple insertion points are tested for each observation within one squeaky wheel iteration, overall complexity could be $O\left(n^4\right)$ by analogy to the rod cutting problem. As n will be large and n^2 is expected to be intolerable, we recommend applying the intermediate fidelity model no more than once per squeaky wheel iteration, as a forward-dispatch schedule repair in the Analyze phase (figure 8).

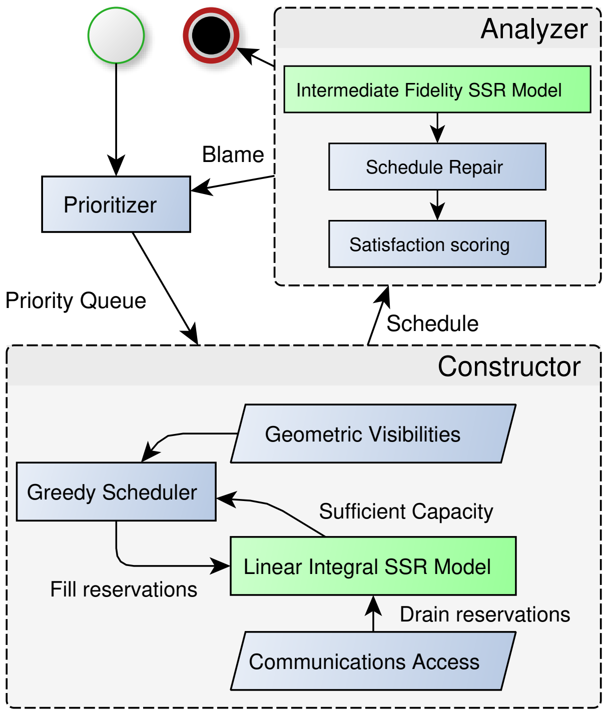

Figure 8: Recommended use of the intermediate fidelity SSR model for Squeaky Wheel Optimization

Preventing the need for repair actions

The intermediate fidelity solid state recorder model described in this paper has only one repair action: remove the most recently placed observation when replanning the playback schedule after that observation results in an over-capacity data recorder constraint violation. This ignores the priority considerations of Squeaky Wheel Optimization. Worse, it partially deprives Squeaky Wheel Optimization of its only feedback mechanism - failure to schedule.

A more appropriate repair would be to sort the requests that have already been placed by priority order (least important to most important), then remove placed requests until the schedule is feasible again. This still has problems because it is locally scoped and may undo squeaky wheel's global optimization. Allowing the flight system to execute this infeasible schedule is also not advised, as this also undoes squeaky wheel's global optimization and causes unintended science data loss during fault-free, routine operations.

The best outcome would be to never need a schedule repair. One option is to add a configuration space to the low fidelity SSR model's maximum capacity, but apply no margin to the intermediate fidelity model's capacity. Choosing the amount of margin to subtract from the low fidelity model's capacity may be difficult, as failure to place on the intermediate model is a discrete phenomenon that we expect to be driven by variable file sizes and the exact circumstances of the communications access windows. One option would be to automate an empirical search for the smallest possible margin that results in 100% success rate on the intermediate fidelity model.

Recommendations for Future Work

Alternative repair actions that provide feedback to the squeaky wheel optimizer should be considered. Other repair strategies that respect priority should also be investigated. Lastly, empirical search for a configuration space margin to apply to the low fidelity model such that its output schedules have a 100% success rate on in intermediate fidelity SSR model should also be explored.

One of this paper's anonymous reviewers recommended replacing the internal storage linear integral model with an instantaneous reservation, instantaneous free system. The start of the reservation would be the start of data recording and the end of the reservation would be the latest possible playback end time, assuming that the data recorder is exactly full after finishing the recording in question. The model would make pessimistic reservations based on the worst case age of the file when it is deleted, assuming no preemption. There was insufficient time to prototype this idea, but it sounds promising. Future work should examine this more closely.

Conclusions

An algorithm to model the nonlinear, discrete effects of deferred file deletions for multiple ground segment playback routing was presented as a linear complexity forward dispatch scheduler. Neglecting deferred, discrete deletion effects found to produce an infeasible observation schedule (exceeding recorder capacity) in a computer simulation experiment. Repairing this infeasible schedule resulted in the loss of 3.87% of the observations, possibly conflicting with the goals of the higher level squeaky wheel optimizer. The simulation suggests that over-capacity conditions may occur as frequently as every other day if deferred deletion effects are ignored.

Future work should focus on improving communication between the intermediate fidelity model and the higher level scheduler, better repair action selection and margining the low fidelity SSR model so that it is far enough away from the recorder's true capacity that deferred deletion effects do not cause the recorder to exceed its capacity. Pessimistic reservations based on worst case file age at deletion time should also be examined.

Acknowledgements

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The authors would like to thank our our reviewers for their constructive feedback. It helped improve our paper immensely.

References

Cesta, A.; Cortellessa, G.; Oddi, A.; and Policella, N. 2002. Mexar: An intelligent support for space mission planning. In *Proceedings of the Workshop on AI Planning and Scheduling for Autonomy in Space Applications, Manchester.*

Chien, S. 2009. Practical planning & scheduling. In *International Conference on Automated Planning and Scheduling (ICAPS).*

Choi, J. 2012. Cost-effective telemetry and command ground systems automation strategy for the soil moisture active passive (smap) mission. In *SpaceOps 2012.* 1275978.

Deems, E.; Swan, C.; and Weiss, B. 2012. End-to-end data flow on the soil moisture active passive (smap) mission. In *2012 IEEE Aerospace Conference*, 1–16.

Doubleday, J., and Knight, R. 2014. Science mission planning for NISAR (formerly DESDynI) with CLASP. In *SpaceOps 2014 Conference*, 1757.

Doubleday, J. R. 2016. Three petabytes or bust: planning science observations for nisar. In *SPIE Asia-Pacific Remote Sensing*, 988105–988105. International Society for Optics and Photonics.

Entekhabi, D.; Njoku, E. G.; O'Neill, P. E.; Kellogg, K. H.; Crow, W. T.; Edelstein, W. N.; Entin, J. K.; Goodman, S. D.; Jackson, T. J.; Johnson, J.; et al. 2010. The soil moisture active passive (smap) mission. *Proceedings of the IEEE* 98(5):704–716.

Fox, M. S.; Allen, B. P.; Smith, S. F.; and Strohm, G. A. 1983. Isis: A constraint-directed reasoning approach to job shop scheduling. Technical report, DTIC Document.

Fukunaga, A.; Rabideau, G.; Chien, S.; and Yan, D. 1997. Aspen: A framework for automated planning and scheduling of spacecraft control and operations. In *Proc. International Symposium on AI, Robotics and Automation in Space.*

Joslin, D. E., and Clements, D. P. 1999. Squeaky wheel optimization. *Journal of Artificial Intelligence Research* 10:353–373.

Knight, R., and Hu, S. 2009. Compressed large-scale activity scheduling and planning (CLASP) applied to DESDynI. In *Proceedings of the Sixth International Workshop in Planning and Scheduling for Space, Pasadena, CA.*

Knight, R.; Donnellan, A.; and Green, J. J. 2013. Mission design evaluation using automated planning for high resolution imaging of dynamic surface processes from the ISS. In *International Workshop on Planning and Scheduling for Space (IWPSS 2013).*

Knight, R.; McLaren, D.; and Hu, S. 2012. Planning coverage campaigns for mission design and analysis: CLASP for the proposed DESDynI mission.

Rabideau, G.; Chien, S.; Nespoli, F.; and Costa, M. 2016. Managing spacecraft memory buffers with overlapping store and dump operations. In Bernardini, S.; Chien, S.; Sohrabi, S.; and Parkinson, S., eds., *Proceedings of the 10th Scheduling And Planning Appications woRKshop (SPARK)*, 69–75. London, UK: Association for the Advancement of Artificial Intelligence.

Righini, G., and Tresoldi, E. 2010. A mathematical programming solution to the mars express memory dumping problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40(3):268–277.

Ward, J. 1990. Store-and-forward message relay using microsatellites: The uosat-3 pacsat communications payload.