Heuristic Onboard Pointing Re-scheduling for an Earth Observing Spacecraft

Steve Chien, Martina Troesch,
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA
steve.chien@jpl.nasa.gov

Abstract

Prior space missions have not routinely used onboard decision-making. The Autonomous Sciencecraft (ASE), flying onboard the Earth Observing One spacecraft, has been flying autonomous agent software for the last decade that enables it to analyze acquired imagery on board and use that analysis to determine future imaging. However ASE takes approximately one hour to analyze and respond.

This paper describes a scheduling prototype for the Earth Observing Autonomy (EOA) project to increase the responsiveness of spacecraft flight software for onboard decision-making as well as to increase the capabilities of flight software. Specifically, we target onboard image analysis and response within a single orbital overflight at low Earth orbit (about eight minutes). We focus on the re-scheduling of the future image acquisitions in the context of an existing set of requests along with new requests based on onboard analysis of just acquired imagery. We describe a greedy, constructive, scheduler with $O(n^2)$ performance and present preliminary results on its performance.

Introduction

The Earth Observing Autonomy (EOA) project targets the development of a spacecraft autonomy capability to enable a wide range of Earth Observing, pointing spacecraft (e.g., Earth Observing One [Ungar et al. 2003], The Spot constellation [Wikipedia Spot 2015], Orbview Class spacecraft (such as Worldview-3) [Ball, 2015, Wikipedia Worldview-3 2015] to image, analyze the image, and re-image based on that analysis within a single overflight, imposing a responsiveness constraint of 5-8 minutes. This would represent a dramatic improvement over the current state of the art, ASE [Chien et al. 2005], which responds within roughly 1 hour.

We describe a software prototype of the EOA capability that includes several autonomy components:

1. *Onboard science processing algorithms.* Science analysis algorithms process onboard image data to detect science events and suggest reactions to maximize science return. Specifically we investigate the use of the Mixture –tuned Match Filter (MTMF) [Boardman and Kruse 2011] for onboard spectral analysis of acquired imagery. However ASE has already demonstrated the utility of thermal analysis for volcanoes and wildfires [Davies et al. 2006], spectral analysis for flooding [Ip et al. 2006], spectral analysis for cryosphere study [Doggett et al. 2006], as well as spectral unmixing for mineralogical analysis [Thompson et al. 2012].
2. *Onboard planning and scheduling software.* The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) [Chien et al. 2000] combined with the Eagle Eye Mission Planning Software [Knight et al. 2013] system generates a baseline mission operations plans from observation requests. This baseline plan is subject to considerable modification onboard in response to data analysis from step 1. The model-based planning algorithms enable rapid response to a wide range of operations scenarios based on models of spacecraft constraints. However, in this paper we focus on a greedy, constructive, non-backtracking scheduler designed specifically for this application.
3. *Robust execution software.* The JPL core flight software [Weiss 2013] (CFS) expands the CASPER mission plans to low-level spacecraft commands and includes a powerful and expressive sequencing engine. The CFS sequencing engine monitors the execution of the plan and has the flexibility and knowledge to perform improvements in execution as well as procedural responses to execution anomalies.

One challenge to spacecraft autonomy is *limited computing resources*. An average spacecraft CPU offers 200 MIPS and 128 MB RAM – far less than a typical laptop computer. For the EOA prototype, we baseline a Rad 750 or Leon processor for all of the autonomy capability.

EOA demonstrates an integrated autonomous mission response capability using onboard science analysis, replanning, and robust execution. EOA performs intelligent science data analysis, and spacecraft retargeting. This capability can reduce data downlinked in cases where onboard analysis determines the data not of interest (e.g. search for active volcanos and return only images that contain active volcanos). This capability can also enable an increase in science return. In many cases, a mission is not limited by observation time, but rather by downlink volume. In these cases, if the spacecraft can acquire imagery searching for a specific signature and not return the data if the signatures not found, then search can be made much more efficient. Specifically, the spacecraft can search for active volcanoes a large amount of the time, and only pay the downlink cost proportional to the number of images with active volcanoes rather than the total number of images acquired searching for active volcanoes. In cases where phenomena may be short-lived, onboard detection may enable additional data to be acquired, gathering more science data on the scarce phenomena (e.g. when detecting an active volcano, add requests to image it more frequently and in greater detail).

The execution flow of the EOA software is shown in Figure 1. As the spacecraft overflys targets, it images them. As the imagery is acquired, it is processed onboard the spacecraft. Based on the operations policies of the missions, this analysis may result in new image requests. These image requests are folded into the prior image requests and a new schedule is constructed that may acquire the new image and may change other images acquired (such as pre-empting a less valuable target). Spacecraft execution then continues.

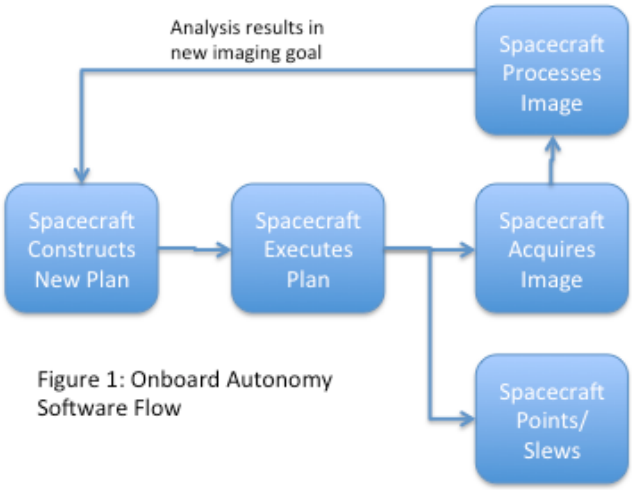


Figure 1: Onboard Autonomy Software Flow

These capabilities enable radically different missions with significant onboard decision-making allowing new ways to conduct science from space. The paradigm shift toward highly autonomous spacecraft will enable future space missions to achieve significantly greater science returns with reduced risk and reduced operations cost.

In this paper, as the meeting topic is planning and scheduling, we focus on the rescheduling portion of the overall responsiveness of the mission. We begin by describing the overall on board response scenario to show the overall mission timeline and the context of rescheduling.

Autonomous Science Scenario

Our onboard planning capability is designed to support an EOA mission scenario. While the EOA software is designed to support a wide range of spacecraft without any modification, in this section, we describe a scenario with a Worldview-3 like spacecraft [Ball 2015, Wikipedia 2015] to image science targets, process and analyze onboard image data, and re-plan operations based on science results.

For this demonstration we assume several baseline mission parameters.

Parameter	Value
Orbit	950 km Sun synchonous
Initial Science Images	30-40° lookahead from nadir
Response image	Nadir to 20° lookahead
Spacecraft slew rate	4.5° per second, instantaneous start and stop, no settle time
Imaging time	Dwell of 1s per image
Image request granule "footprint"	0.5 km along track x 4 km across track

In Figure 1 we highlight some of the geometry characteristics of the EOA scenario. As the spacecraft orbits the earth, it has several viewing windows. The first viewing window is the initial science image window which covers from 31 to 38° in front of the spacecraft. The second viewing window is the response image which covers from 0° lookahead (nadir) to 28° lookahead. As the spacecraft flies over the earth it is imaging in large number of locations in the initial science window. As it acquires this imagery, software analyzes the imagery onboard the spacecraft. This analysis indicates the possible need to take follow-up imagery (in the response imaging window). For example, in the initial science window we might search for the thermal signature of a volcanic eruption or wildfire. In the response window we might further image to precisely determine the extent of the lava flow and the exact temperature map of the flow.

The goal of the scheduler is to accommodate as many of the initial science and response imaging requests but is guided by the priority of the requests and restricted by the pointing and slewing capabilities of the spacecraft (as well

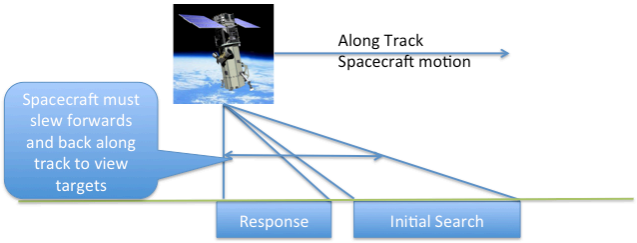


Figure 2: spacecraft must slew along track.
Distance between initial search and response windows large compared to distance between two requests in same window.

as any other operations constraints). As shown in Figure 2, from a side view, the speacecraft must slew forwards and backwards looking a variable amount ahead to view the image targets. At the same time, the spacecraft is moving forwards due to orbital motion (at approximately 7.5km per second). Because processing the images requires some time, the initial search window is significantly ahead of the response window. This enables initial search-ed images to be processed/analyzed in time to allow for scheduling of followup imagery in the response window. The response window does not extend behind the spacecraft in order to maintain consistent lighting conditions.

This slewing forwards and backwards along the spacecraft motion track is complicated by two things. First, the angle at which the spacecraft must look forward to view the target is a non linear function of when the spacecraft wishes to view the target. Specifically, at nadir, for the Earth, in a 950km orbit, 1 degree of lookahead corresponds to 16.6 km ahead of nadir in the ground track. However, at 37° of lookahead, 1° of further lookahead (e.g. to 38° lookahead) corresponds to 30.7 km ahead in the ground track. The second issue is that typically the slew rate of the spacecraft is not linear, there is a ramp up acceleration of the spacecraft to some maximum slew rate, a portion of the slew at the maximum rate, then a ramp down as the spacecraft arrives at the desired position.

Figure 2, Case 1 shows these two factors from the spacecraft pointing perspective. In this example the spacecraft is looking ahead and wants to view a target further ahead beyond the current look angle. The spaceraft could simply wait until the target comes into view, or it can slew ahead to meet the target. The blue line shows the track of a fixed point on the ground in terms of the look angle from the spacecraft as the spacecraft approaches the point. This line indicates that at time 0 the target is at 42° lookahead. The red line shows the angular position of the spacecraft reachable from the starting point of nadir as a function of time. The intersection of these two lines shows the earliest possible time that the spacecraft can view the target. The graph indicates that if the spacecraft begins slewing it will be able to reach the target but that the target will be at 38° lookahead when it is reached. In this case the motion of the spacecraft is helping us to meet the target earlier.

The right side of Figure 2 shows a different case, Case 2. In Case 2, the spacecraft is pointing at 38° lookahead, and

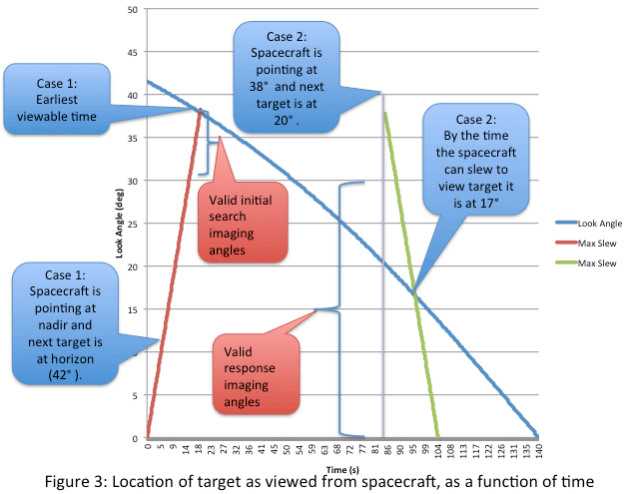


Figure 3: Location of target as viewed from spacecraft, as a function of time

wants to next view a target currently at 20° lookahead. In this case the spacecraft motion is carrying the target (relative to the spacecraft) away from the current spacecraft pointing and the slew must catch up. The graph shows that the by the time that the spacecraft can view the target it will be at 17° lookahead.

Figure 4 is a view from above the spacecraft looking down on the Earth. As the spacecraft moves along track (from left to right in Figure 4), the spacecraft must also slew across track (up and down in the Figure) as well as forwards and backwards along the ground track (left and right in the Figure) to image targets.

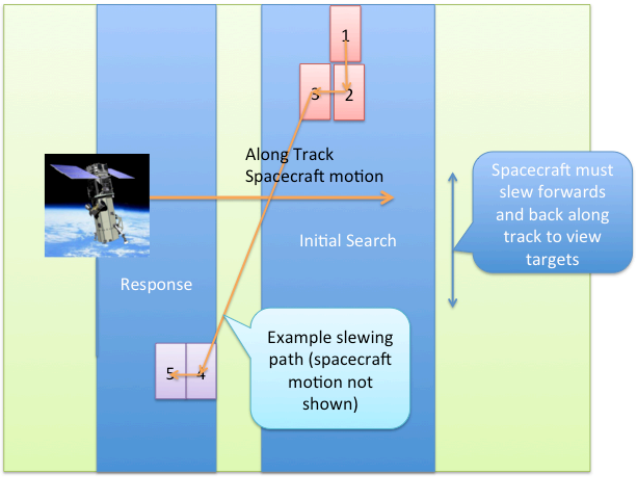


Figure 4: spacecraft must also slew across track.
Spacecraft moves along track as while slewing and imaging.

This scheduling problem is a challenging one for several reasons.

1. The spacecraft has limited ability to slew from one target to the next (e.g. each slew takes up valuable time).
2. Targets are distributed across the ground track of the spacecraft so that the amount of time required to image a target depends on the preceding and following (temporally) targets in the schedule.
3. Because the initial viewing and response doing windows are separated angularly, slewing back and forth between these windows can be wasteful of time.
4. Image analysis takes time. During this time spacecraft is moving towards the target(s). This is the reason why the initial image analysis and response image analysis windows are not overlapping, to allow the onboard software time to analyze the images.
5. Generating the schedule also takes time (the focus of this paper).
6. When calculating a start time to schedule an observation of a target, the spacecraft intercepts the target. The spacecraft must slew to a given position (of the target), reaching that position at the exact time that the target is in that position relative to the spacecraft. This requires an accurate model of the spacecraft slew time as well as the ability to project where relative to the spacecraft any target will be at any point in time.
7. In addition to pointing, the scheduler must consider other resources such as power, thermal, data volume (e.g. [Chien et al. 2010, Chien et al. 2012]). However in this paper we focus on the pointing and slewing aspect of the problem as the state and resource management aspect of the problem has been considered elsewhere.

In order to simplify the scheduling problem we first transform the image request locations from a <latitude, longitude, altitude> coordinate frame of reference to an <along track, across track> frame of reference (in this process using a model of the spacecraft orbit). From this <along the track, across track frame of reference>, combined with the spacecraft orbit, the set of valid times to view any target in the initial viewing window or response window is easily computed.

R = $\{r_1,\ldots r_n\}$ sorted from highest priority to lowest priority

```
achieved_requests = {}
best_solution = nil
for adding_request ∈{r1…ra}
 call schedule( achieved_requests ∪ {adding_request});
 if success then
   achieved_requests ← achieved_requests ∪ {adding_request}
   best_solution ← solution returned by schedule
```

```
Schedule(request_set = {r1…ra})
Sort request set by earliest start time
  (e.g. request with earliest start time is first in set)
current_solution = {}
for current_request ∈{r1…ra}
   attempt to add current_request to current_solution
   by scheduling it at the earliest possible time that
   it will fit into the schedule
   if cannot add return FAIL
   else (success} continue
return current_solution
```

This scheduling algorithm represents a greedy outer loop where we try to add requests in priority order. The inner loop is given a set of requests, and attempts to schedule them sweeping forward in time considering earliest possible start time requests first.

Figure 5 shows the inner loop of the scheduler. In figure 5a the two headed arrows indicate the earliest and latest possible times each image can be acquired. The longer intervals are response images and the shorter intervals are initial search requests. In Figure 5a the requests are sorted by earliest possible start time. Figure 5b shows the requests being scheduled. The software tries to add each request in the earliest start time sorted order, adding the request to the schedule as early as possible. The orange blocks indicate the slew time and the blue blocks indicate the imaging time. The imaging time is roughly constant but the slewing time is higher if the preceding image was of a different type (initial, response), this is because the spacecraft is generally slewing a greater distance (up to 0° → 38° lookahead) as opposed to from one initial search to another (maximum slew from 31° → 38°) or from one response to another (from 0° → 28°).

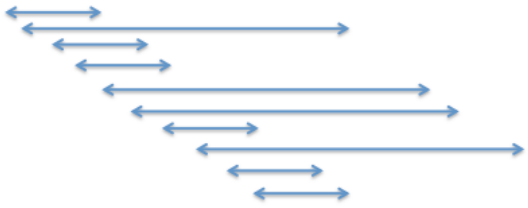


Figure 5a: Requests with possible scheduling times sorted by earliest possible start time

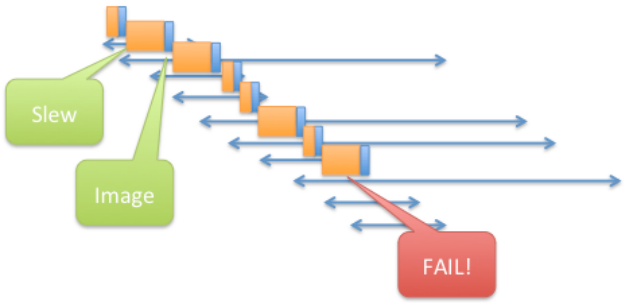


Figure 5b: Scheduling at earliest possible time until failing.

As the inner loop of the scheduler is attempting to insert the next request, it must repeatedly solve the problem of slewing forwards and backwards along track to view targets either ahead of or behind the current position. The scheduler solves this problem of the intersection of the earliest possible slew position curve intersecting the target position angle relative to the spacecraft using Newtons Method [Wikipedia Newtons 2015]. In computing this observation time the software must take the latest of: the spacecraft along track slew intersecting the target, the mission mode constraints (e.g. initial search allowed look angles, response image allowed look angles), as well as the across track slew time to view the target. Solving for the across track slew time is simpler than the along track problem – while the earths curvature does make the angular position a nonlinear function of the ground distance, there is no across track motion to compensate for and indeed this conversion from ground distance to angular distance can be pre-processed. In practice for our scenario slew times can range from a fraction of a second (for adjacent tiles) to 5-10 seconds.

While we currently use a simple constant slew rate model for our current prototype, a more realistic model would have:

1. an acceleration/decceleration profile,
2. a maximum rate,
3. a different model for different axes of the spacecraft,
4. a settling time for the spacecraft to stabilize after a slew in which the settling time depends on the parameters of the image being acquired as well as the velocity and acceleration profile of the slew

In our software architecture we treat the slew computation as a black box so that we can easily insert a high(er) fidelity model.

Estimated computational complexity of the scheduling algorithm

Our analysis of the above observation scheduling algorithm indicates several factors in its computational complexity.

Since we schedule from scratch each iteration we will always make R passes through the outer loop where R is the number of scheduling requests.

Each of the R passes through the outer loop makes a single call to the "schedule" function. The schedule function performs a computation to attempt to add a slew and image to the schedule each iteration. This effort to add a slew and image requires computation in worst case on the order of the number of items currently in the schedule. While the number of items currently in the schedule is certainly no worse than R above (total number of requests) if the number of requests that can actually fit into the schedule S_{max} is much smaller than R this will be a much lower bound.

For example, if the entire search window corresponds to 200s of observation time, and the minimum observation length is 2s $S_{max} < 100$ so if R is >> 100 it does not matter, the complexity is of order S_{max}. So overall the computational complexity of the scheduler is $RS_{max}C$ where R is the total number of requests and S_{max} is the number of requests that will actually fit in the schedule and C is the computation required to evaluate the feasibility of inserting a single request into the schedule.

Note that this algorithm does not take advantage of the fact that the set of changes to the request set is small compared to the size of the request set. An obvious optimization would be to only reschedule the portion of the schedule of lesser priority than the highest priority new request.

Empirical evaluation of the scheduling algorithm

In order to verify our analysis of the computational complexity of the algorithm, we also performed a limited empirical analysis of the algorithm. For this analysis we generated a synthetic data set using the following parameters.

Parameter	Value
Initial request probability	1-5%
Probability of a response image given a search request performed	25, 50, 75%
Scheduling horizon	45° lookahead

Figure 6: Preliminary run information

The empirical data is shown at the end of the paper. Graphs 1 and 2 show that the scheduler can only completely achieve a relatively small percentage of initial search requests (a few percent). Already if 2% of all possible tiles are requested for search with no responses many of the search requests are not being satisfied.

When response requests are included this further drives down the percentage of search requests that can be scheduled because response requests are higher priority and they preclude search requests. Graphs 3 and 4 show that as response images are added to the scheduling problem (at higher priority than search requests) the scheduler is able to accommodate fewer search requests. Graph 3 shows where 25% of searches yield a response

and Graph 4 shows where 75% of searches yield a response.

Graphs 5 and 6 show the CPU time required for the scheduler in VxSIM. The run-time data indicates that the scheduler is extremely fast, taking only a fraction of a second in the software simulation. While the flight processor is expected to be significantly slower, the scheduling algorithm is not optimized in any way. One obvious optimization is that the scheduler is solving the problem from scratch each invocation when the majority of the inputs have not changed. Clearly an incremental rescheduler offers great potential for efficiency gains.

Discussion

The Autonomous Sciencecraft (ASE) has been flying the CASPER continuous planner on board the Earth Observing One (EO-1)spacecraft for the past decade. However, the response time for CASPER on EO-1 is tens of minutes-in part due to the meager computation on board the spacecraft: 3 MIPS and no floating point computation in hardware for the RAD 3000 CPU on board. Additionally, the planning problem for EO-1 does not involve significant geometric issues. The spacecraft generally only images using its push broom imager and a fixed angle relative to nadir, therefore there is no flexibility in the imaging time for any target. The problem is rather one of which combination of images should be acquired. The same issue of computing which combination of images and slews is feasible is challenging (and solved on the ground). The EO-1 pointing problem is complex because EO-1 only has three reaction wheels, therefore as further observations are required momentum builds up on the reaction wheels that restricts later pointings of the spacecraft due to maximum rates that the reaction wheels can achieve. While this momentum can be relieved using a magnetic torque bar, this is a very slow process so observations are quite constrained by builtup angular momentum (for further details see [Chien et al. 2010]).

The CLASP [Rabideau et al. 2010, Doubleday et al, 2014] and Eagle eye [Knight et al. 2013] planners solve geometric coverage planning problems from a ground context. These systems can incorporate more complex geometric constraints but also have better computational resources and less time constraints.

AEOS [LeMaitre et al. 2002] is another project to perform automatic observation planning on the ground. AEOS solves a much more complicated and expressive problem in which the spacecraft slews while imaging to cover target polygons and the direction of the slew can be optimized to cover the polygon as efficiently as possible. In contrast we assume a framing imager and that the alignment of the imager is in a fixed aspect ratio with respect to a long track and across track to simplify the problem. We do this because our onboard computation capabilities are necessarily limited and also our response time for the scheduler correspondingly constrained.

In the future we plan on further maturing this work, refining the scheduling algorithms as well as bringing the work into a relevant hardware testbed.

Summary

We describe an overall software architecture for onboard imaging, image analysis, operations scheduling, and re-imaging within a realistic flight software operating system and flight hardware performance environment. This prototype demonstrated the feasibility of performing such functions autonomously within a low earth-orbiting environment (roughly 5-8 minutes overflight time). Future efforts will further mature this concept and software by bringing the prototype into a relevant flight hardware testbed.

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

Ball Aerospace, The WorldView spacecraft series, http://www.ballaerospace.com/page.jsp?page=294, *retrieved 18 Feb 2015.*

J.W.Boardman and F.A.Kruse,"Analysis of imaging spectrometer data using N-dimensional geometry and a mixture-tuned matched filtering (MTMF) approach," *TGARS*, vol. 49, no. 11, pp. 4138–4152, 2011.

DigitalGlobe, Worldview-3: DigitalGlobe, http://worldview3.digitalglobe.com**,** *retrieved 18 Feb 2015.*

S. Chien; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; Shulman, S.; Boyer, D. 2005. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. Journal of Aerospace Computing, Information, and Communication 2(4): 191–216.

S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling*, Breckenridge, CO, April 2000.

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 2012, Stockholm, Sweden, June 2012.

Davies, A. G., S. Chien, V. Baker, T. Doggett, J. Dohm, R. Greeley, F. Ip, R. Castano, B. Cichy, R. Lee, G. Rabideau, D. Tran and R. Sherwood (2006) Monitoring Active Volcanism with the Autonomous Sciencecraft Experiment (ASE). Remote Sensing of Environment, Vol. 101, Issue 4, pp. 427-446.

T. Doggett, R. Greeley, A. G. Davies, S. Chien, B. Cichy, R. Castano, K. Williams, V. Baker, J. Dohm and F. Ip (2006) Autonomous On-Board Detection of Cryospheric Change. Remote Sensing of Environment, Vol. 101, Issue 4, pp. 447-462.

F. Ip, J. M. Dohm, V. R. Baker, T. Doggett, A. G. Davies, R. Castano, S. Chien, B. Cichy, R. Greeley, and R. Sherwood (2006) Development and Testing of the Autonomous Spacecraft Experiment (ASE) floodwater classifiers: Real-time Smart Reconnaissance of Transient Flooding. Remote Sensing of Environment, Vol. 101, Issue 4, pp. 463-481.

R. Knight, A. Donnellan, J. Green, Mission Design Evaluation Using Automated Planning for High Resolution Imaging of Dynamic Surface Processes from the ISS, *International Workshop on Planning and Scheduling for Space (IWPSS 2013).* Moffett Field, CA. March 2013.

Michel Lemaître,, Gérard Verfaillie, Frank Jouhaud, Jean-Michel Lachiver, and Nicolas Bataille. "Selecting and scheduling observations of agile satellites." *Aerospace Science and Technology* 6, no. 5 (2002): 367-381.

D. R. Thompson, B. Bornstein, S. Chien, S. Schaffer, D. Tran, B. Bue, R. Castano, D. Gleeson, A. Noell, Autonomous Spectral Discovery and Mapping Onboard the EO-1 spacecraft, IEEE Transactions on Geoscience and Remote Sensing. 2012.

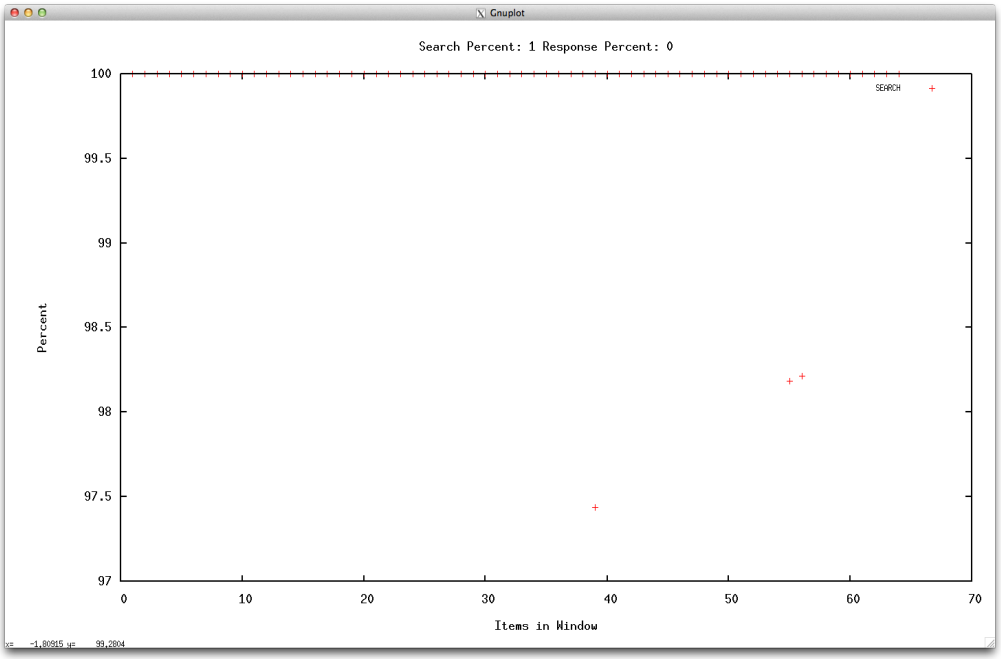
Stephen G. Ungar, Jay S. Pearlman, Jeffrey A. Mendenhall, and Dennis Reuter. "Overview of the earth observing one (EO-1) mission." *Geoscience and Remote Sensing, IEEE Transactions on* 41, no. 6 (2003): 1149-1159.

K. Weiss, "An Introduction to the JPL Flight Software Product Line", 2013 Workshop on Spacecraft Flight Software (FSW-13), Pasadena, CA, December 2013.

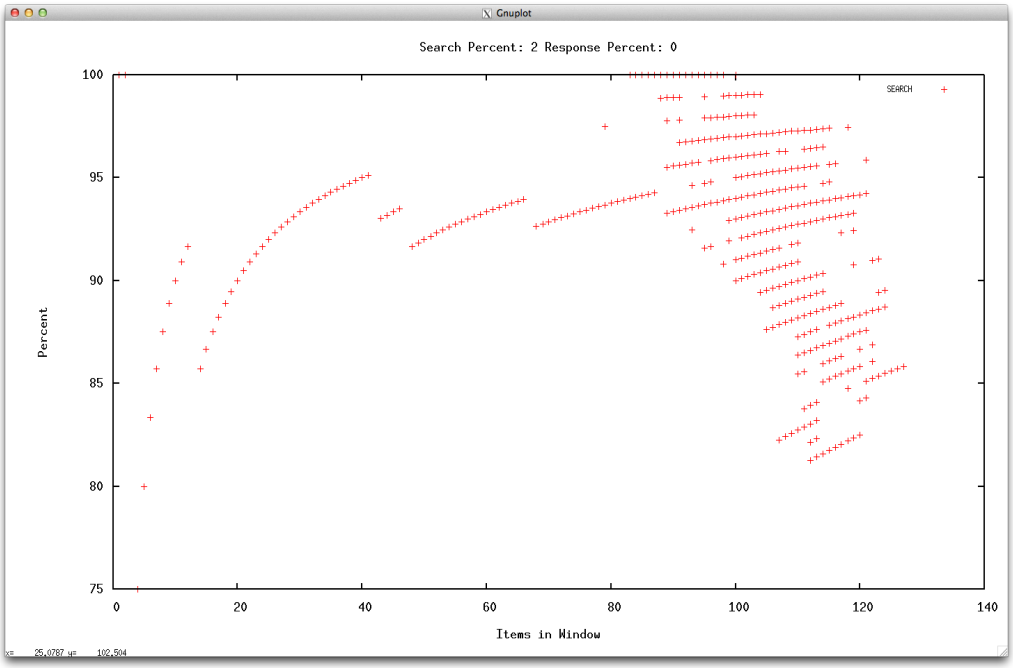
Wikipedia, Newtons Method, http://en.wikipedia.org/wiki/Newton%27s_method **,** *retrieved 10 Apr 2015.*

Wikipedia, Spot(satellite), http://en.wikipedia.org/wiki/SPOT_%28satellite%29**,** *retrieved 10 Apr 2015.*

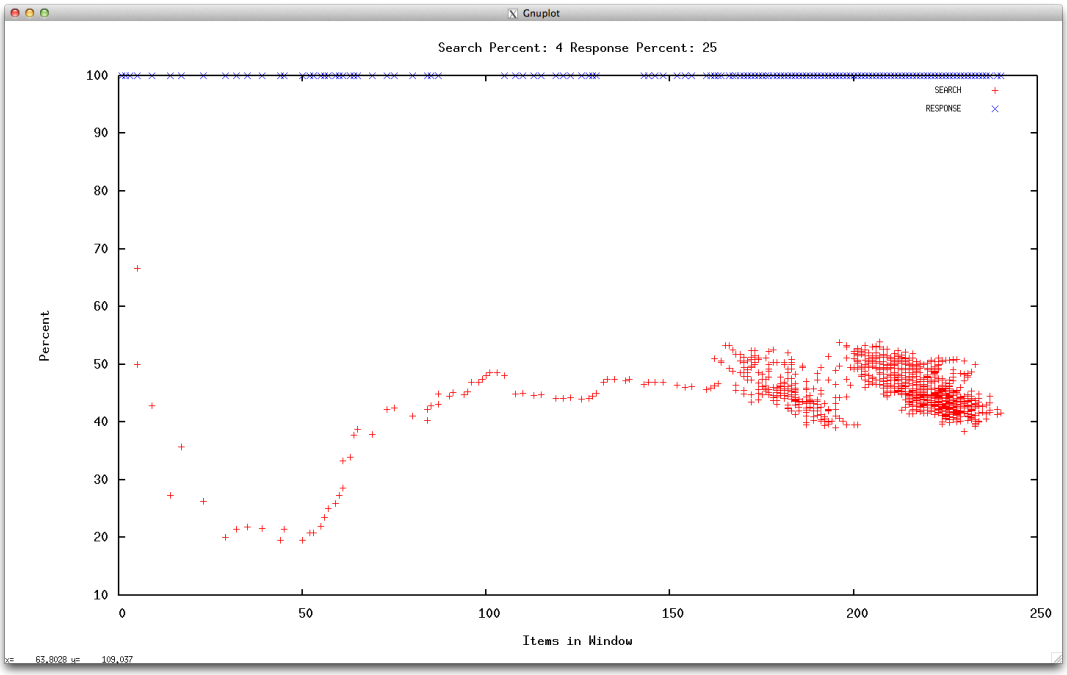
Wikipedia, Worldview-3, http://en.wikipedia.org/wiki/**WorldView-3,** *retrieved 18 Feb 2015.*



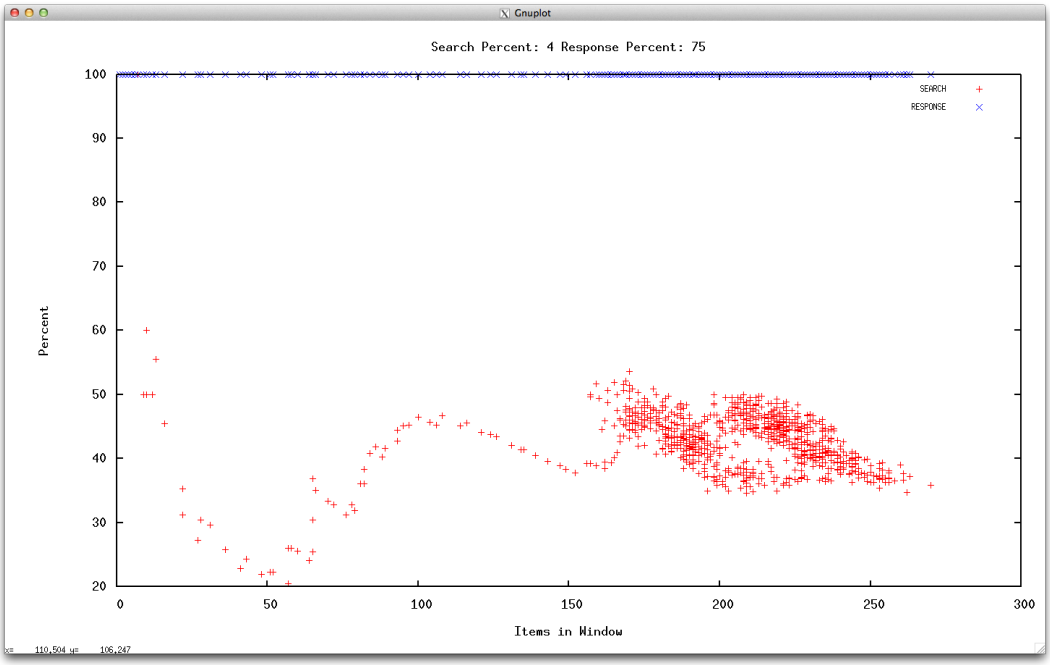
Graph 1: 1% search 0% response – almost all search requests scheduled
X axis is number of requests presented to scheduler



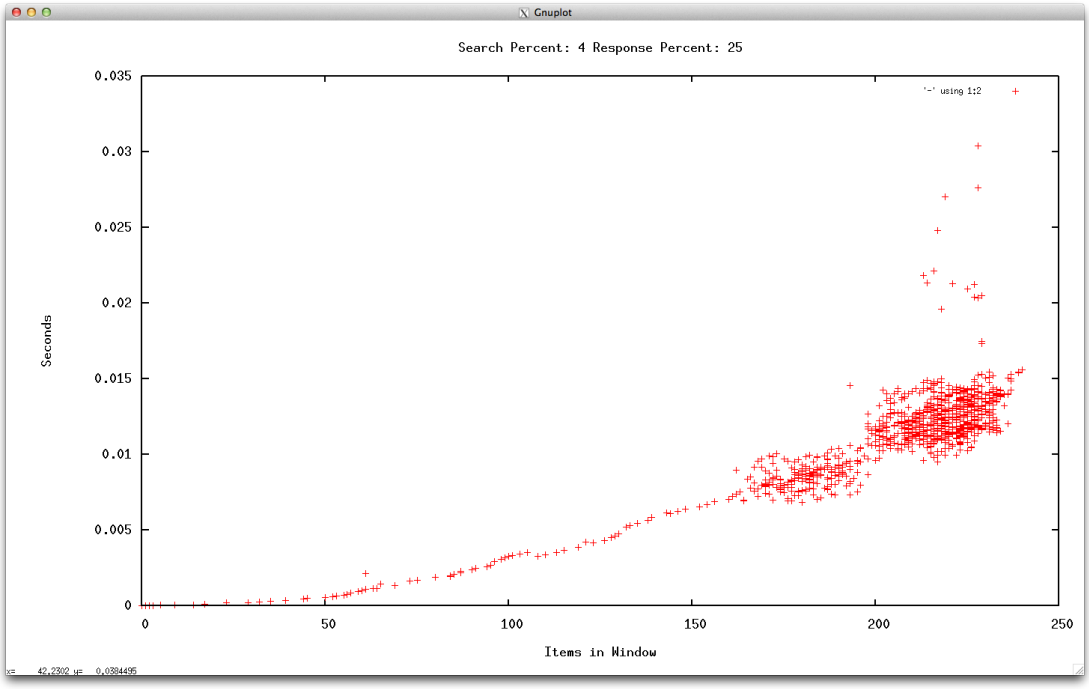
Graph 2: 2% search 0% response – some search requests already are not achieved.
X axis is number of requests presented to scheduler
Y axis is %-age of search requests scheduled



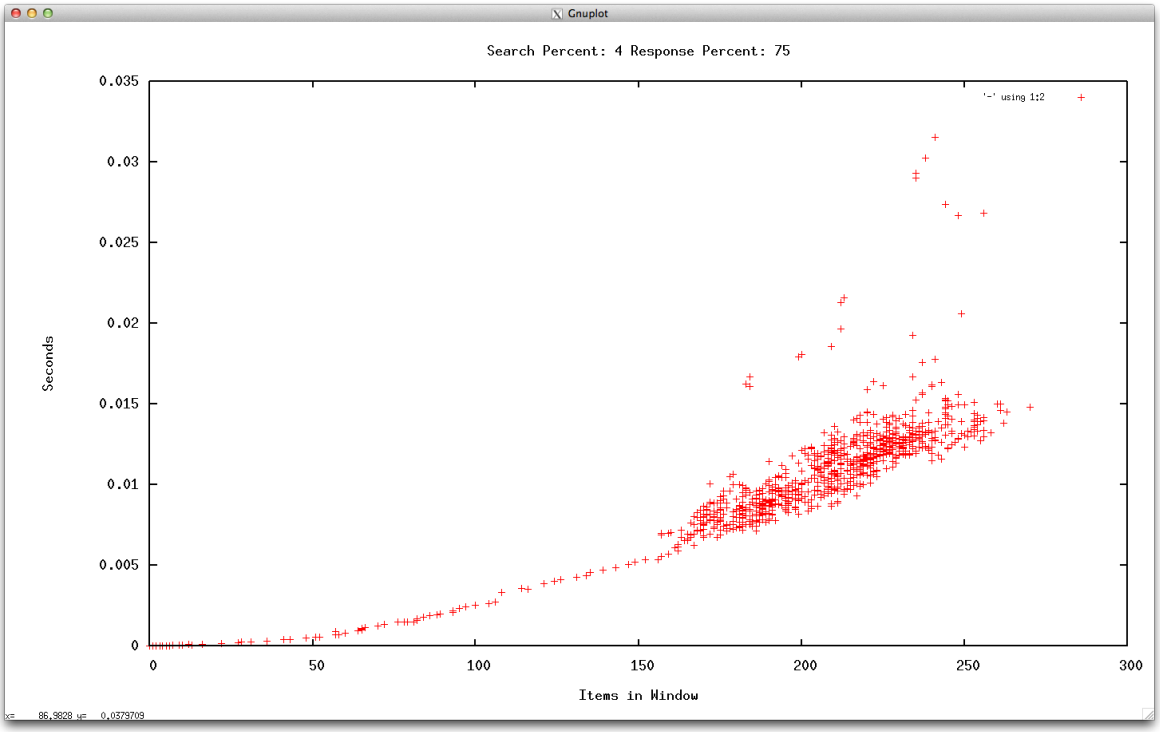
Graph 3: 4% search 25% response – response images are higher priority so all are getting scheduled. Search images are pre-empted and s/c loses time slewing between search and response windows.
X axis is number of requests presented to scheduler
Y axis is %-age of search (red) or response (blue) requests scheduled



Graph 4: 4% search 75% response – more response images are pre-empting search images.
X axis is number of requests presented to scheduler
Y axis is %-age of search (red) or response (blue) requests scheduled



Graph 5: 4% search 25% response
X axis is number of requests (search + response) presented to scheduler
Y axis is CPU time to construct schedule



Graph 6: 4% search 75% response
X axis is number of requests (search + response) presented to scheduler
Y axis is CPU time to construct schedule