Interleaved Observation Execution and Rescheduling on Earth Observing Systems

Lina Khatib[(*)], Jeremy Frank, David Smith, Robert Morris, Jennifer Dungan
[(*)]Kestrel Technology
NASA Ames Research Center
Mail Stop N269-2
Moffett Field, CA 94035-1000
{lina,frank,de2smith,morris}@email.arc.nasa.gov, jdungan@gaia.arc.nasa.gov

Introduction

Observation scheduling for Earth orbiting satellites solves the following problem: given a set of requests for images of the Earth, a set of instruments for acquiring those images distributed on a collecting of orbiting satellites, and a set of temporal and resource constraints, generate a set of assignments of instruments and viewing times to those requests that satisfy those constraints. Observation scheduling is often construed as a constrained optimization problem (Bensana, Lemaitre, & Verfaillie 1999; Lemaitre *et al.* 2000; Pemberton 2000) with the objective of maximizing the overall utility of the science data acquired. The utility of an image is typically based on the intrinsic importance of acquiring it (for example, its importance in meeting a mission or science campaign objective) as well as the expected value of the data given current viewing conditions (for example, if the image is occluded by clouds, its value is usually diminished).

Currently, science observation scheduling for Earth Observing Systems is done on the ground (Potter & Gasch 1998), for periods covering a day or more. Schedules are uplinked to the satellites and are executed rigorously. An alternative to this scenario is to do some of the decision-making about what images are to be acquired on-board. The principal argument for this capability is that the desirability of making an observation can change dynamically, because of changes in meteorological conditions (e.g. cloud cover), unforeseen events such as fires, floods, or volcanic eruptions, or unexpected changes in satellite or ground station capability. Furthermore, since satellites can only communicate with the ground between 5% to 10% of the time, it may be infeasible to make the desired changes to the schedule on the ground, and uplink the revisions in time for the on-board system to execute them.

Examples of scenarios that motivate an on-board capability for revising schedules include the following. First, if a desired visual scene is completely obscured by clouds, then there is little point in taking it. In this case, satellite resources, such as power and storage space can be better utilized taking another image that is higher quality. Second, if an unexpected but important event occurs (such as a fire, flood, or volcanic eruption), there may be good reason to take images of it, instead of expending satellite resources on some of the lower priority scheduled observations. Finally, if there is unexpected loss of capability, it may be impossible to carry out the schedule of planned observations. For example, if a ground station goes down temporarily, a satellite may not be able to free up enough storage space to continue with the remaining schedule of observations.

This paper describes an approach for interleaving execution of observation schedules with dynamic schedule revision based on changes to the expected utility of the acquired images. We describe the problem in detail, formulate an algorithm for interleaving schedule revision and execution, and discuss refinements to the algorithm based on the need for search efficiency. We summarize with a brief discussion of the tests performed on the system.

On-board Schedule Revision

In formulating any reasonable approach to on-board spacecraft decision-making, it is essential to accommodate expected limitations of on-board memory and processing capabilities, which in turn limits the size or complexity of the problem the on-board system could solve. For this reason, we begin by assuming that there is a separate, ground-based system for generating schedules that has a rich domain model and the ability to perform the search required to generate high quality schedules (an example of such a system, built explicitly to interact with the on-board system described in this paper, is found in (Frank *et al.* 2001)). In addition, while the on-board system has knowledge of only its own satellite, the ground-based system is expected to deal with multiple satellites which allows it to generate schedules that incorporate "global" optimization. During execution, the nominal schedule produced on the ground is to be preferred in the absence of any changes in the actual or expected values of observations. Consequently, the behavior of the on-board scheduler will allow it to revert to the ground schedule in the absence of any utility changes.

The satellite receives updates on the actual value of observations just completed (e.g. as the result of per-

Inputs
1. a complete schedule produced by a ground-based scheduler and uplinked,
2. a set of additional observations that were not scheduled, and
3. the utility of each observation in the schedule and in the set of alternatives.

Setup Procedure
1. Artificially boost the utility values of scheduled observations by the maximum utility of the extra observations
2. Remove observations from the schedule and combine them with the extra observations as requests for the on-board scheduler.

Figure 1: On-board Set-up Procedure.

forming a cloud cover analysis of the acquired data onboard), or updates on the expected values of observations that could be done in the near future (via communication with other satellites, forward looking instruments, or weather forecast update). We also identify two primary constraints that must be adhered to during rescheduling, in addition to the usual temporal constraint dealing with when an observation can be taken, given the orbital profile of the satellite. The first deals with capacity constraints on on-board storage (Solid State Recorder or SSR): the amount of data stored onboard at any time cannot exceed a constant amount. The second is associated with science instruments that are pointable (as opposed to those that are fixed at a nadir-pointing position). The constraint states that between two observations, there must be enough time to slew the instrument so that it is pointing at the angle required by the later observation.

The basic approach to on-board schedule revision is for a system to acquire more observations than it expects to be able to keep, based on on-board storage capacity limitations, incrementally discarding those of lesser value, as necessary, in order to retain observations of higher value. This over-commitment helps ensure that a full complement of useful observations will be collected, even if later scheduled observations turn out to be of low value. The bias towards acquiring observations in the original schedule, discussed earlier, is implemented by artificially raising the utility value of the pre-scheduled observations to guarantee they are higher than any extra observation. These observations are "removed" from the schedule and combined with the extra observations as inputs to an on-board scheduler (Figure 1). After on-board setup is completed, Execution and Schedule Revision is applied as shown in the algorithm in Figure 2. Starting with the first execution time of requests, at any time, we have past time requests (either satisfied or not), current request(s), and future requests. The algorithm decides what observations to schedule out of the set of current requests aiming at highest overall scheduled utility. The loop in the algorithm represents the progress of observation selection and execution while "next" time slot t is moving along the time horizon.

The algorithm is applied to a scheduling horizon starting at a given time slot t, and evaluates the set of requests that can be scheduled starting at t. At time t the SSR has a set of images stored from images acquired before t, and there are cloud cover data analysis algorithms on-board that have been applied to the stored images in order to possibly update their utility.

Execution Steps
For each time slot t:

1. consider the set R of requests that can be scheduled at time t.
2. Apply a lookahead strategy to assign a heuristic value to each request in R.
3. Schedule best "feasible" request(s) in R.

```
Step 3 elaborated:
  While not done
     choose $r \in R$ that has highest heuristic value
     If SSR has sufficient capacity for $r$
        Acquire & Record $r$
        assess the actual utility of $r$
        done = true
     If SSR has insufficient capacity for $r$
        let $W$ be the set of past observations with
          lower utility than $r$ and higher SSR
          allocation than needed
        if NotEmpty($W$)
          let $w$ be a minimum utility in $W$
          discard $w$ for SSR release
          Acquire & Record $r$
          assess the actual utility of $r$
          done = true
        Else remove $r$ from $R$
```

Figure 2: Interleaved Execution and Schedule Revision Algorithm.

Step 1 of the algorithm computes the set R of all requests at the next time slot t that do not conflict with past scheduled requests.

Step 2 computes a heuristic value for each request in R in terms of the overall utility of the schedule that would result from executing it. The heuristic value of a request r is calculated based on the utilities of "some" future requests that can be given that r is executed. executed (i.e. do not violate the pointability constraints[1]), given that r is executed. Details are given in the next section.

[1]SSR constraints are not taken into consideration in determining conflicting observations during the lookahead phase.

```
int FixLookahead($r$,$t$,$h$)
if ($h$=0) return (util($r$))
else
  for each non-conflicting observation $r'$ at $t+1$
      heur_val($r'$) = FixLookahead($r'$,$t+1$,$h-1$)
  return (util($r$) + $max_{r'}$ heur_val($r'$))
```

Figure 3: Fixed Lookahead Approach.

Step 3 applies the greedy strategy for schedule revision. If adding a request does not violate SSR capacity, it is simply added. If there is a violation, required storage space will be made available by discarding, and releasing the SSR storage space of, acquired images whose actual (analyzed) utility is less than the expected utility of the current request. If no such acquired images exists, then the next highest expected utility request in R is considered. The process repeats until either a request is added or there are no more requests left. Note that different requests may produce different amount of data and, as a result, require different amount of SSR storages. This complicates the process of selecting, among acquired images, the best "set" to discard. We attempt for one with minimum utility provided it has sufficient SSR release.

Lookahead Strategies

We have considered two approaches to lookahead; a *fixed* approach and a *variable* approach. For each request time t, the on-board scheduler uses one of the strategies to assign a heuristic value to *each* of the observations requested at t. Then Step 3 of the Schedule Revision Algorithm (figure 2) is preformed for selecting observations.

Fixed Lookahead Approach

The parameters of the fixed lookahead strategy include an observation r, its start time t, and the lookahead horizon h. The output is a heuristic value for r returned as an integer value. The algorithm is recursive and described in Figure 3. Given a lookahead horizon h, the ***heuristic value*** of a request r is the maximal sum of the utilities of requests that can be scheduled with the current request during the interval $[t, t+h]$.[2] More formally, let $s(r)$ be a feasible schedule including r during the time window $[t, t+h]$, and let $ut(s) = \Sigma_{r_i \in s} ut(r_i)$, where $ut(r_i)$ is the expected utility of r_i. Let $S(r)$ be the set of such feasible schedules; then the expected value of r is $max_{s \in S(r)} ut(s)$. Additionally, a "no-observation" option is available at each t in $[t+1, t+h]$. As far as the algorithm is concerned, a "no-observation" is an observation that can be chosen at any time, that is non conflicting with any other observation, and has a utility equal to zero.

[2]For the sake of notation simplicity, we use the interval notation $[t, t+h]$ to refer to time indexes rather than values, and is interpreted as the scheduling horizon starting at time slot t and ending after h time slots of requests. For example, if the current schedule has requests with start times 30, 60, 120, 180, and 240, then $[60, 60+2]$ refers to the set of time slots starting at $\{60, 120, 180\}$

Variable Lookahead Approach

The idea of a variable lookahead strategy emerged to solve the "grass is greener", also known as the "horizon effect", anomaly that was observed using the fixed approach. Briefly, in some cases it is possible, by greedily delaying an observation in favor of a later one with greater utility, that a deeper fixed lookahead makes worse decisions than a shallower one. A variable lookahead approach avoids this anomaly by making the lookahead horizon depend on the point of acquiescence in the search for better schedules. If lookahead for horizons of $i, i+1, \ldots, i+j-1$ specify the same choice for next step, it is highly likely that this is the best choice. i and j are positive integers that represent, respectively, the lookahead horizon at which the acquiescence starts and the length of the acquiescence.

Specifically, in the variable approach, h depends on two other factors: RAL, Required Acquiescence Length, and MLH, Maximum Lookahead Horizon. The value assigned to h corresponds to the lookahead horizon such that the past RAL lookahead horizons have agreed on which $r \in R$ is the one with highest heuristic value. MLH is an upper limit on h, in order to control the size of the lookahead. Again, as in the case for fixed lookahead, "no-observation" is a choice as a non-conflicting observation at any time with utility equal to zero.

The output is an extended set of sequences of non-conflicting observations S' (each of size $h+1$ representing a partial schedule for $[t-h, t+1]$ and associated with a heuristic value). The algorithm is described in figure 4.

To illustrate the entire process, consider the example in Figure 5. A nominal schedule is pictured, along with a set of alternatives. Observations on the same "stack" with respect to the time axis have the same start times. The current utility values for each observation are also displayed, as are scheduled activities for downlinking observations to a ground station, and off time for the instrument due to duty cycle restrictions. During the set-up phase, the on-board scheduler will boost the utilities for the scheduled activities so that they exceed those of any extras, then remove them from the schedule. During the execution phase, given a time t, the algorithm will assign a heuristic value to each observation that can be taken at t (in the figure, this value would be thus assigned to $O1$ and $O5$). Depending on the lookahead strategy, this value would be obtained by summing the utilities of requests on all feasible sequences starting with either $O1$ or $O5$ of lookahead horizon h. For example, with h assigned as in the figure, the set of feasible sequences would include all the observations in the circle.

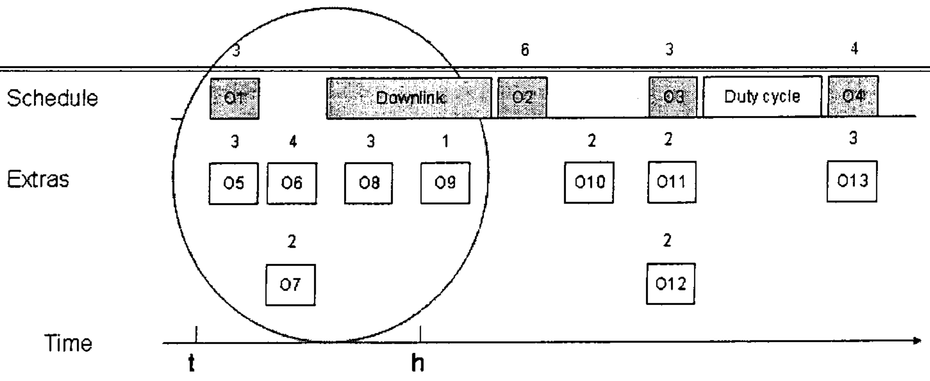

Figure 5: Example of the On-board Scheduling Problem: Timeline showing Initial Nominal Schedule and Set of Alternatives.

Initialization

```
set $S$ to the set of all schedulable observations at $t$
for each $r$ in $S$: heur_val($r$) = util($r$)
call VarLookahead(1,1,$t$,$S$)
```

VarLookahead(a,h,t,S)

```
if ($h>MLH$) or($a=RAL$)
  for each request $r$ at $t$
    heur_val($r$)=$max_s$ heur_val($s$) where
      $s \in S$ and $r$ starts $s$
  exit
else
  for each sequence $s$ in $S$
    for each non-conflicting observation $r$ at $t+1$
      add the sequence $s+r$ to $S'$
      heur_val($s+r$)=heur_val($s$)+util($r$)
  if highest-heur-observation agrees with previous step
    $a' = a+1$
  else
    $a' = 1$
  VarLookahead($a'$,$h+1$,$t+1$,$S'$)
```

Figure 4: Variable Lookahead Approach.

Enhancements

From a theoretical perspective, such lookahead strategies are potentially expensive, involving the evaluation of a set of trees of all possible feasible schedules of length h, each rooted at one of the requests in R. For this reason, we have been devising a number of heuristics to decrease the cost of searching for expected values.

One enhancement for the fixed lookahead implementation involves a sort of alpha-pruning. For each time point, the highest utility of the set of observations that can be taken at this time is known. Consequently, a partial schedule of size less than h has no need to be extended if the extension is guaranteed to not exceed the highest expected utility observed thus far.

A second component of this enhancement involves checking for slewing constraint violations. While looking at future requests during the lookahead phase, if we reach consecutive time slots for observations with a temporal gap larger than the maximum slewing requirements, there is no need to continue the lookahead to its specified horizon. This is because, from this point on, all potential requests will result in the same set of extensions and, therefore, will not affect their comparative expected value if we ignore this part in the calculation. More generally, the lookahead process can be stopped whenever previous choices do not limit any next choice in any way.

A third enhancement is used in conjunction with variable lookahead. At each step, out of all partial schedule sequences that end with the same request, only the one(s) with highest so-far expected value need to be extended (since all of them will have the same possible extensions). More generally, whenever more than one sequence have the same possible future extenstions, all but the one(s) with the highest expected value can be dropped from future consideration.

Experiments

In order to identify the usefulness of on-board rescheduling, we are studying the expected gain in the value of observations collected, over those that would be taken if we just followed the schedule produced on the ground. This, of course, depends on the frequency and nature of the value revisions. So, more generally, we would like to know the net gain in the value of observations collected, as a function of the frequency and nature of value revisions. For the particular algorithm proposed above we would also like to know how this value is affected as SSR capacity changes, the ground schedule bias is decreased, the size of the set of alternatives increases, or as lookahead changes.

We have considered two different value revision scenarios. In the first scenario, we suppose that there is on-board image analysis software, so that the actual value of an image is updated after an image is taken. With the above algorithm this primarily affects which observations will be discarded to make way for future observations. In the second scenario, we suppose that updates of expected value are received for observations to be taken in the future, as might occur if updated cloud cover forcasts were provided to the satellite. Note that this information impacts both which observations are taken and which ones are kept.

Our current experiments involve scheduling horizons of up to 4 hours and problem sizes of up to 200 requests. Before the proposed enhancements, such problem sizes could not be solved in a reasonable time. A reasonable variable lookahead strategy turned out to be $RAL = 3$, which in general outperformed one with $RAL = 2$, while either $RAL = 2$ or $RAL = 3$ outperformed no lookahead ($RAL = 0$) by about 10%. In general, a variable lookahead strategy seems to be faster than, and out perform, fixed lookahead. An intutive explanation for the speed improvement is that, in a variable lookahead, only small lookahead horizon is needed at most steps and the maximum horizon is rarely reached. On the other hand, the reason that better results are obtained with variable lookahead is apparently the avoidance of the "grass is greener", or "horizon effect", anomaly.

Summary

This paper has proposed a new approach to managing the selection of science observations for Earth observing systems. This approach is motivated by expected increasing demand for high quality science data. It has been argued here that communication delays between ground station and satellite, combined with an uncertainty in the conditions under which the data will be acquired, justifies an approach to interleaving scheduling and execution. Constraints on on-board computing resources, as well as the demand for on-line scheduling, necessitates an approach that is simple and fast. Our solution relies on a greedy search through a space of candidate schedules, rooted at a time point in the execution horizon, based on the evaluation of both expected and actual scientific utility of the data.

References

Bensana, E.; Lemaitre, M.; and Verfaillie, G. 1999. Earth observation satellite management. *Constraints* 3(4).

Frank, J.; Jónsson, A.; Morris, R.; and Smith, D. 2001. Planning and scheduling for fleets of earth observing satellites. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space.*

Lemaitre, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.; and Bataille, N. 2000. How to manage the new generation of agile earth observation satellites? In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space.*

Pemberton, J. 2000. Towards scheduling overconstrained remote sensing satellites. In *Proceedings of the 2d International Workshop on Planning and Scheduling for Space.*

Potter, W., and Gasch, J. 1998. A photo album of earth: Scheduling landsat 7 mission daily activities. In *Proceedings of the International Symposium Space Mission Operations and Ground Data Systems.*