

Validating Resource Usage in Least Commitment Scheduling

Nazma Ferdous and Mark Giuliano

Space Telescope Science Institute
3700 San Martin Drive, Baltimore, MD 21218
 {ferdous, giuliano}@stsci.edu

Abstract

Least commitment scheduling is a useful tool for planning astronomical observatories. For many classes of observatories, it is not feasible to create detailed schedules of observations, more than a few weeks or months in advance of execution. For these domains, combining a long range planner with a short term scheduler allows the flexible creation of efficient schedules. The long range planner creates least commitment plan windows that handle proposal constraints and resources. The short term scheduler uses plan windows to drive the creation of efficient telescope uploads. An issue with this approach is that the default method of distributing resource usage evenly across plan windows can create both false positives and false negatives in the resource model. In this paper we present a simple method for verifying plan window resource usage. The method is evaluated in terms of how it impacts the on-going operations of the Hubble Space Telescope.

Introduction

Least commitment scheduling is a useful tool for planning astronomical observatories. For many classes of observatories, it is not feasible to create detailed schedules of observations, more than a few weeks or months in advance. For these domains, combining a long-range planner¹ with a short term scheduler allows the flexible creation of efficient schedules. The long-range planner creates least commitment plan windows that handle proposal constraints and resources. The short term scheduler uses plan windows to drive the creation of efficient telescope uploads. An issue with this approach is that the default method of distributing resource usage evenly across plan windows can create both false positives and false negatives in the resource model. In this paper we present a simple method for verifying plan window resource usage. The method is evaluated in terms of how it impacts the on-going operations of the Hubble Space Telescope (HST).

The SPIKE system is used in the evaluation. SPIKE (Johnston and Miller 1994) is a planning and scheduling

¹ We use the terms ‘long range planner/planning’ to indicate scheduling observations to time windows as opposed to a fixed time.

tool kit that was created for use on the Hubble Space Telescope and is currently being used for multiple orbital and ground based astronomical missions including FUSE (Calvani 2004), Chandra, Subaru (Sasaki 2000), and SIRTF (Kramer 2000).

The remainder of the paper reads as follows. First, we present details of the HST mission including its scheduling constraints and motivation for using a least commitment scheduler. Next, we describe the even distribution method for resource modeling in long range planning and show how it can result in false positives and negatives. We next describe the resource validation engine and present a case study showing how the engine has impacted HST operations. Finally, we discuss limitations of the system and summarize our conclusions.

HST Least Commitment Scheduling

In HST mission operations, planning and scheduling is a two-step process. Long range planning creates coarse-grained least commitment plan windows that are nominally 56 days long. Short term scheduling takes the plan windows as input and creates observation schedules on the granularity of a second.

Both long range planning and short term scheduling create schedules that conform to observation constraint windows. Constraint windows are the physical and user specified restrictions on times an observation can execute. Physical constraints include sun avoidance, moon avoidance, and guide star restrictions. A final physical constraint is due to earth occultation. Astronomers prepare multi-orbit observations, where each orbit requires a specified amount of target visibility. Depending on target position and time of year, target visibility is occulted by the earth for portions of the 96 minute orbit. The target visibility constraint ensures that there is enough visibility for each orbit of an observation at the given time. User specified constraints include the following:

- **o between t_1 and t_2** : observation o has to begin between time t_1 and time t_2 .
- **o_1 after o_2 by d_1 to d_2** : observation o_1 has to begin between $\langle \text{start of } o_2 \rangle + d_1$ and $\langle \text{start of } o_2 \rangle + d_2$.

- **group $o_1 \dots o_n$ within d :** the set of observations o_1, \dots, o_n have to begin within duration d .
- **orientation o_1 from o_2 by a_1 to a_2 :** the difference of orientation angles of the observations o_1 and o_2 has to be between a_1 and a_2 degrees. $a_1 = a_2 = 0$ indicates that o_1 and o_2 need to be executed at the same orientation angle.

Each observation has a set of constraint windows that are a combination of the physical constraints of the observation and the user specified constraints propagated for temporal consistency.

During long range planning each observation is assigned a plan window that is a subset of the observation's constraint windows. A plan window represents a best effort commitment to execute the observation within the time frame of the window. These windows create a pool of candidates that the short term scheduler uses to efficiently utilize scarce telescope resources. Generally, the candidate pool is slightly oversubscribed to ensure that there are enough observations to choose from. Creation of plan windows take into account the following planning criteria and constraints:

- Linked observations need to be planned with respect to link specifications and to maximize link set flexibility (Kramer 1997).
- Plan windows need to avoid the creation of plan window slivers that cross short term calendar boundaries. Having a small fragment of a plan window in a subsequent calendar tightly constrains the possibilities for short term scheduling.
- Plan windows need to distribute scheduling resources across the planning session so that HST resources can be efficiently used by the short term scheduler.

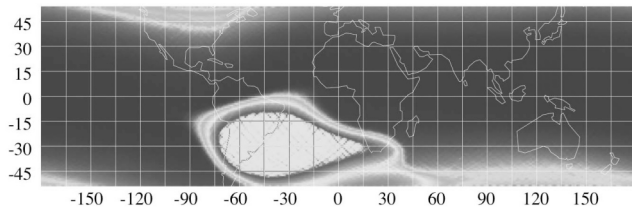


Figure 1. South Atlantic Anomaly (SAA) is shown in lighter shade

There are two main resource types for HST long range planning. The first is just the orbit(s) used for observing the target. There are approximately fifteen 96 minutes orbits in a day. The second resource type occurs due to the South Atlantic Anomaly (SAA). The SAA is a region off the coast of South American that has unusually high radiation as shown in Figure 1.

Depending on the scientific instrument being used, approximately nine of the days' fifteen HST orbits cross the SAA. In each of these orbits, the SAA passage occurs in a slightly different portion of the orbit. These orbits are

called SAA impacted orbits. No observations can occur during an SAA passage. However, we can schedule observations in SAA impacted orbits, if the earth occultation for the observation occurs during the SAA passage. This case is called SAA hiding. Orbits without any SAA crossing are called SAA free orbits, which are relatively scarce and valuable. Nearly 60% of HST observations can only be scheduled in SAA-free orbits, whereas only 33.3% of the orbits are SAA-free. For any given target, SAA hiding only occurs for a small fraction of a year. Therefore, taking advantage of SAA hiding is a major driver of HST schedule efficiency.

Short term scheduling creates second by second schedules for weekly uploads that command the HST telescope. The short term scheduler uses observations plan windows intersecting the week to construct the short term schedule. The idea is that the long range plan distributes plan windows across the time line to provide a slightly oversubscribed candidate pool for the short term scheduler to choose from. Short term scheduling considers the observation constraint windows as well as:

- Slewing overhead between observations.
- Instrument reconfigurations between observations.
- Data upload and download times.
- Maintaining thermal and power control of the telescope.

The overall goal of short term scheduling is to create uploads that maximizes the science return while maintaining the telescope engineering infrastructure.

There are multiple motivations for using a two-phased approach. First, the exact in-track position of HST is not known more than a few weeks in advance. Schedules created with an inaccurate HST ephemeris would not be useful. Second, astronomers often change the details of their observations based on changed science goals or a changed understanding of HST capabilities. Third, detailed observation plans often change in response to the evolution of spacecraft capabilities. The two-phased approach provides a relatively stable plan to the astronomers. Finally, splitting long range planning from short term scheduling provides a separation of concerns. The long range planner can concentrate on the high level distribution of resources and does not need to know the precise details needed to schedule the telescope. Short term scheduling embeds detailed knowledge of telescope capabilities and is not concerned with high-level resource distribution.

Plan Window Resource Model

Resources are tracked at granularity of a day during long range planning. The resources used by an observation are divided equally across all the days in the plan window. Figure 2 gives an example of resource calculation for three observations O_1 (2 orbits long), O_2 (4 orbits long)

and O_3 (3 orbits long). The plan windows of the observations stretches from day 2 to 3, day 2 to 5 and day 3 to 5, respectively. Here, we assume there are two orbits per day. The orbits required for each task are divided equally across the number of days in the plan window. The bottom two rows of the chart show the amount of resources consumed and available for each day.

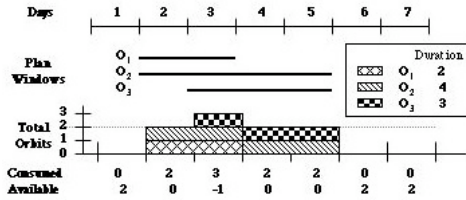


Figure 2. Example of Resource Model

The example shows that day 3 is over subscribed, days 1, 6, 7 are under subscribed, whereas, days 2, 4, and 5 are level subscribed. Although this resource model captures the resource load over time and tracks resource failures, it allows both false positives and false negatives as illustrated below.

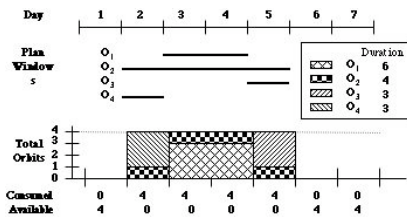


Figure 3. Example of False Positive

Figure 3 shows an example of a false positive situation. The figure shows the resource consumption of four observations across their respective plan windows. Here, the total orbit limit for each day is 4 and the total

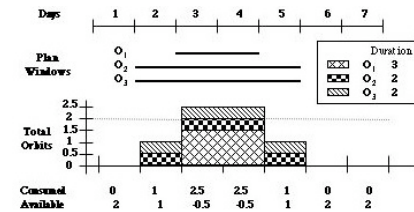


Figure 4. Example of False Negative

consumption shows no oversubscribed days. However, an actual schedule of observations cannot be generated. The problem lies with observation O_2 . There are not enough orbits to schedule it either before or after task O_1 . The intuition behind this false positive is that the resource model allows observations to be broken up during scheduling. However, this is not the case for HST scheduling.

Figure 4 gives an example where the resource model results in a false negative. Each day has 2 total orbits. Day

2 and day 3 are over subscribed. However, a schedule can be obtained by placing O_1 in days 2 and 3, O_2 in day 1, and O_3 in day 4. The false negative occurs as the resource model assumes that the resource consumption is evenly divided across the plan window.

The plan window resource model shows a probabilistic picture of demand for resources, but may result in false positives and false negatives. With the plan window resource model, it is hard to pinpoint which observations are likely to fail due to over commitment or which time of the year HST orbits are underused.

SAA Resource Modeling

Maximizing SAA hiding is crucial for HST scheduling efficiency. SPIKE has a strong preference to plan observations in windows where they can utilize SAA impacted orbits. The scheduler first considers windows that allow SAA hiding. Other windows are considered only if there are no days with SAA hiding or if linked observations prevent SAA impacted orbits from being utilized. The plan window scheduler tracks resources for SAA free orbits and for SAA impacted orbits. SPIKE has an abstract model of a day. Each day has 15 orbits, the first six of which are SAA free. The remaining nine orbits are SAA impacted.

HST observations consist of one or more orbits, each of which must execute in contiguous orbits. On a given day, an observation may hide the SAA for all of its orbits (i.e. full hiding), or for some of its orbits (i.e. partial hiding) or for none of its orbits (i.e. no hiding). Furthermore the observation may fully/partially/not hide the SAA in different orbit combinations. For example, suppose an observation O is 3 orbits long, and can be scheduled anywhere starting from orbit 1 to orbit 8. Table 1 shows the SAA resource usage depending on where the observation is scheduled. Starting in orbits 1-4 the observation consumes the SAA free resource. Starting in orbits 5 and 6 the observation is a partial hider and consumes both resources. Starting in orbits 7 and 8 the observation is a full hider and consumes the SAA impacted resource.

Scheduled at Orbit	SAA Free Resource	SAA Impacted Resource	State of SAA hiding
1 2 3	3	0	No hiding
2 3 4			
3 4 5			
4 5 6			
5 6 7	2	1	Partial hiding
6 7 8	1	2	
7 8 9	0	3	Full hiding
8 9 10	0	3	

Table 1. Possible orbit assignments and corresponding resource usage.

The scheduler consumes resources in a day using SAA impacted orbits first. If an observation has full hiding opportunities for a day then, only SAA impacted resources are used. If an observation can partially hide the SAA but not fully hide the SAA during the day then, both SAA free and SAA impacted resources are used. If an observation cannot schedule in any SAA impacted orbits during a day then, it consumes SAA free resources. In this way, the scheduler consumes the scarce SAA free orbits only if there are no SAA impacted orbits that can be utilized. Note that if an observation can schedule in SAA impacted orbits for a day then it can always schedule in SAA free orbits (i.e. SAA impacts only make schedulability worse). For the example shown in table 1, only SAA impacted resources will be used.

Tracking the total SAA impacted orbit usage does not accurately depict the actual probability of SAA resource contention. Each of the nine SAA impacted orbits (e.g. orbits 7, 8, ..., 15), prevents target viewing in a slightly different portion of the orbit. Targets that may hide the SAA in one of the SAA impacted orbits may not hide the SAA in another. Multiple targets may compete for the same SAA impacted orbits. To get a better picture of SAA usage and contention, SPIKE models each SAA impacted orbit as a separate resource. For days where an observation hides the SAA, resource usage is equally distributed among the SAA impacted orbits.

Orbit Assignment	Probability of scheduling	SAA Impacted Orbit Consumption			
		7	8	9	10
7 8 9	0.5	0.5	0.5	0.5	
8 9 10	0.5		0.5	0.5	0.5
Fractional Consumption		0.5	1	1	0.5
Day Consumption = Fractional consumption / # days		0.1	0.2	0.2	0.1

Table 2. Orbit assignment selected and corresponding resource usage.

Table 2 demonstrates resource calculation for the observation O shown in Table 1. Resources are distributed based on the observation starting in orbit 7 or orbit 8 (i.e. the full hiding orbit assignments). The probability of the two full hiding assignments is equal. Resource usage in each orbit is multiplied by the probability of each assignment being used. The fractional consumption in each orbit, is multiplied by the amount of resource allocated to the day (e.g. 1/(plan window duration)) to calculate resource consumption for the day. In the example given in table 2, we assume observation O has a 5 day plan window.

In addition, SPIKE uses two other resources named “effective SAA hiding” and “effective SAA free”. These resources are calculated from the individual SAA impacted orbit resources. The effective SAA hiding is the sum of resource consumption in the individual SAA

impacted orbit resources that are below or at capacity. The Effective SAA free resource is the sum of the SAA free resource consumption and any resource overflow in the individual SAA impacted resources.

In general, it is harder to hide the SAA in the middle SAA orbits (orbits 10, 11, 12) and easier to hide the SAA on either wing (orbits 7, 8 and 14, 15) because the SAA passages are shorter in the wings. In consequence, the equal distribution of resource usage over all orbit assignments results in false positives and negatives. Moreover, the orbits at the wings of SAA passage are usually in high demand because of the easiness of SAA hiding and the abundance of partially hiding observations. As a result, the “effective SAA hiding” resource is pessimistic with respect to what is achievable.

Another limitation of the SAA resource model is that only fully hiding orbit assignments are considered, even if there is contention for SAA impacted orbits. Long range planning does not allow a choice between fully/partially/not hiding orbit assignment even if there is contention for SAA impacted orbits. This artificially lowers the SAA free resource usage.

Table 3 summarizes the resources tracked in SPIKE.

Resource Name	What is tracked
Total orbits/Day	Total HST orbit usage (15 per day)
SAA free orbits/Day	SAA free orbit usage (6 per day)
SAA impacted orbits/Day	SAA impacted orbit usage (9 per day)
[1..9] individual SAA impacted orbits/Day	Individual SAA impacted orbit usage (1 per day)
Effective SAA Free/Day	The sum of SAA free resource consumed and any overflow in the individual SAA impacted resources.
Effective SAA hiding/Day	The sum of resource consumed in all of the individual SAA impacted resources that is below or at corresponding limit

Table 3. Resources used in HST long range planning

Verifying Resource Usage

To address the above problems in the resource model, we developed a system for validating plan window resource usage. A modified plan window scheduler was developed that schedules observations into particular HST orbits. As in long range planning, a day is modeled as 15 orbits. The first six are SAA free and the remaining nine are SAA impacted. Observations are assigned to particular orbits and the resource usage is calculated accordingly. Only one observation can be assigned to an orbit without violating resource limits. Link constraints are propagated with respect to day boundaries as our model of a day is abstract and does not represent the actual orbits in a day. The scheduler takes the plan windows generated in long range planning as input as well as all other observation

specification data. The scheduler treats the plan window as an additional constraint and folds it into the observation constraints.

Plan Validation Scheduler Algorithms

Before presenting our experience in using the plan validation engine we provide details on the scheduling algorithms used. The plan validation engine uses a 2-step process similar to that used in HST long range planning. The first step is an “initial guess”, where the observations are assigned to the best possible orbits. The next step is the “repair step”, where we attempt to resolve resource conflicts.

Initial Guess

In the initial guess step, all observations are assigned to the best possible orbits available possibly causing resource conflicts. The algorithm sorts the observations in ascending order in terms of difficulty of scheduling. A number of criteria, such as observation length, number of linked observations, scheduling flexibility, are used to determine the order for initial placement. As the observations are scheduled into orbits, they consume the required resources. The resource manager keeps track of the consumed and available resource for each day. A resource conflict occurs when resource consumption exceeds the resource limit. The scheduler tries to place observations without creating new resource conflicts. Otherwise, it places observations in orbits with the least amount of resource contention.

For linked observations, the initial guess uses a backtracking mechanism to minimize resource conflicts. A recursive algorithm is used, which backtracks on a resource conflict to the last observation without a resource conflict. Using a recursive algorithm prevents SPIKE from piling up too many linked observations in the same region.

The plan validation engine assigns observations to particular orbits rather than distributing over all possible orbit assignments. This gives a better sense of effective SAA hiding and points out any contention in individual SAA orbits. Moreover, the plan validation engine can choose between fully/partially/not hiding orbit assignments to resolve resource contention. The orbit assignments are ordered in strict preference of full/partial/no SAA hiding. SPIKE chooses the first assignment that does not have resource contention.

Repair Step: Using Iterative Deepening

In the repair step, the scheduler tries to resolve existing resource conflicts without creating new conflicts. The

plan validation engine uses a systematic search for resolving resource conflicts, which is implemented by an iterative deepening depth-first search repair algorithm. It allows every observations with a resource conflict a chance to get repaired and does not create any new conflicts in that process.

The objective of the repair step is to move observations from oversubscribed resource regions to undersubscribed resource regions in order to have a leveled resource profile. We use a guided search that iteratively increases the number of moves the scheduler needs to make in order to reduce resource conflicts. In the first iteration, the scheduler tries to move observations from oversubscribed to undersubscribed resource regions. In the second iteration, the scheduler first moves observations from leveled resource regions to under subscribed regions so that some room is created which can be utilized by observations from an oversubscribed region. These moves are called intermediate moves. In subsequent iterations, the scheduler increases the number of intermediate moves by one. The schematic for moving observations looks like

```
Iteration 1: Oversubscribed → Undersubscribed
Iteration 2: Oversubscribed → Leveled subscribed → Undersubscribed
Iteration 3: Oversubscribed → Leveled subscribed → Leveled
subscribed → Undersubscribed
Iteration n: Oversubscribed → Leveled subscribed → Leveled
subscribed ... (n-2 moves)→ Undersubscribed
```

At each iteration, the scheduler attempts to repair the schedule for all conflicted observations. The algorithm proceeds until a conflict-free solution is found or the user specified iteration level is reached. The algorithm can be sketched as:

```
Method Iterative-Deepening-Repair
iteration = 1
while (iteration <= max-level)
  while (Oi = next conflicted observation)
    Resolve-Conflict (Oi, iteration, 1)
  end while
  iteration ++

Method Resolve-Conflict (Oi, iteration, current-depth)
if Oi has undersubscribed day in its plan window
  move Oi to the undersubscribed day.
  return t
else if (curr-depth >= iteration)
  return nil
else
  trial-set = Select-Observations-for-Intermediate-Move (Oi)
  while (trial-set <> nil)
    if (Resolve-Conflict ((pop trial-set), iteration, curr-depth++))
      move Oi to an undersubscribed day
      return t
    end while
  return nil
end if

Method Select-Observations-to-Intermediate-Move (Oi)
observations-to-move = nil
for all observations Oj scheduled within plan window of Oi
  if Oj is not resource-conflicted & holds resources for Oi
    push Oj into observations-to-move
end for
return observations-to-move
```

The *Iterative-Deepening-Repair* method tries to resolve all conflicts for an *iteration* level. It starts with *iteration* 1 and subsequently increases the *iteration* level by 1 until *max-level* is reached. The *Resolve-Conflict* method performs a depth first search to find a chain of moves of length less than or equal to the given iteration level, which will ultimately resolve the conflict for the given observation. The *Select-Observations-for-Intermediate-Move* method returns the set of observations that are currently in level subscribed regions but can be moved to create room for an observation. We used several heuristics to order the observation set so that a solution with minimal length can be found. These are discussed in a later section.

An example of the iterative deepening repair algorithm is presented below in Figures 5a-5d. For the purpose of the example, we assume each observation occupies only one orbit and that there are only two orbits in a day. The days are numbered on top of the figures. The observations are represented by small circles and shown on the day they are currently scheduled. Observations A, B, C, D, E and F will be used in the example. The remaining unfilled circles represent observations not critical to the example. The corresponding plan windows for the observations are depicted as bars underneath, and are shown to straddle the days their plan windows intersect.

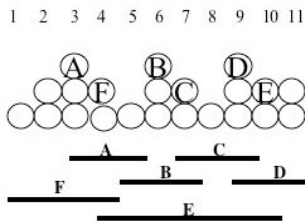


Figure 5(a). Initial schedule of observations

Initially, observations A, B and D are scheduled in days with resource contention as shown in Figure 5(a). Days 1, 5, and 8 are undersubscribed and rests are at level subscription. In the next paragraphs, we demonstrate moves to resolve contention for observations A, B, and D.

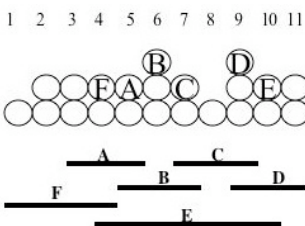


Figure 5(b). Schedule after iteration 1

In the first iteration, we first attempt to resolve contention by moving observations from oversubscribed to undersubscribed regions. For observation A, an

undersubscribed day is found within its plan window and A is moved from day 3 to day 5 as shown in figure 5(b). Similar attempts to resolve conflicts for observation B and D returns unsuccessful.

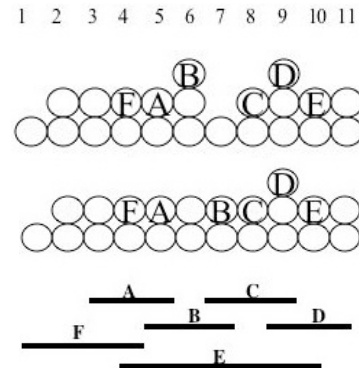


Figure 5 (c). Schedule after iteration 2

In the next iteration, we attempt to resolve a conflict by first moving an observation out of level subscribed regions to make room for an observation in oversubscribed region. For observation B, this can be achieved by moving observation C from a leveled subscribed day 7 to an under subscribed day 8 to create room for observation B, which is then moved to day 7, as shown is figure 5(c). Similar efforts for observation D will be unsuccessful at this point.

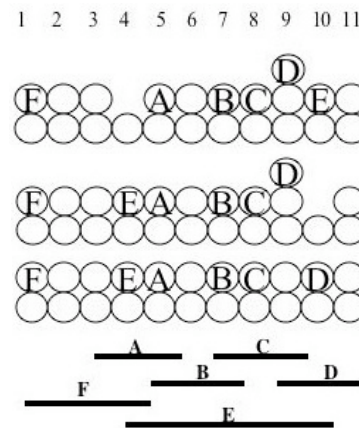


Figure 5(d). Schedule after iteration 3

In the iteration 3, we attempt to resolve the only remaining conflict for observation D by increasing number of moves from level-subscribed regions. To achieve this, observation E is first selected for intermediate move. This, in turns requires observation F to move to undersubscribed day 1 to create room for observation E. Observation E is subsequently moved to day 4 creating room for observation D. After moving observation D to day 10, all resource conflicts are addressed as shown in Figure 5(d).

The algorithm is similar to iterative deepening A* search (Korf 1988). The repair algorithm is also similar to the task-swapping algorithm in (Kramer and Smith 2004). Both offer a more systematic approach for repairing schedules with resource conflicts. The task swapping approach places a new task within its constraint foot print and then recursively searches for assignments to displaced tasks. This may result in a non-minimal chain of moves. In contrast, iterative deepening repair looks ahead for the minimum number of moves needed to resolve the conflict. On the other hand, iterative deepening repair does not repair conflicts by swapping observations. Swapping observations can temporarily create new conflicts, which is not allowed in this scheme. Our iterative deepening approach explores a subset of the search space explored by the task swapping approach. Both approaches use heuristics, such as, maximum flexibility to select candidates to move.

Plan Validation

The goal of the initial guess and repair algorithm is to schedule observations without any resource conflicts. The iterative deepening repair algorithm allows a systematic search for resolving resource conflicts. After running the plan validation engine, the observations with remaining resource conflicts are examined and the reasons for the conflicts are analyzed. As a whole, this process reduces the number of false positives and negatives in the resource profile, and raises flags for observations that need to be closely examined.

Operational Experiences

HST accepts new observations in cycles that are typically 1-1.5 years in length. In each cycle thousands of new observations are chosen by a time allocation committee to be executed by HST. After the detailed requirements for the accepted programs have been defined the cycle-ingest process creates a plan that integrates the existing unexecuted observations with the newly prepared observations. SPIKE loads both the new observations and any unexecuted observations from previous cycles. The existing baseline plan windows for the unexecuted visits are then loaded. The planner then produces a long range plan for the new observations while balancing resource consumption and maximizing various user preferences.

For the year 2006 HST planning cycle, 1724 new observations were selected. The new observations have a total of 2818 orbits, 1441 of which can only be scheduled in SAA-free orbits. The remaining 1377 can be scheduled in SAA-impacted orbits. There were unexecuted observations in the base plan, which requires 958 orbits. Out of these, 587 can only be scheduled in SAA-free

orbits. The total amounts to 3776 orbits, 2028 of which must schedule in SAA-free orbits and 1748 orbits can schedule in SAA impacted orbits. SPIKE takes into account the resources already reserved for old observations when planning for the new ones. The resource limits set for the planner are conservative (10 total orbits, 6 SAA free orbits and 5 SAA hiding orbits per day) to allow for targets of opportunities, slewing and data down loading.

SPIKE was used to create a long range plan and the plan validation engine was used to validate the resource usage. The resulting resource profiles for the long range plan (LRP) and the validation engine (VAL) are presented in figures 6 and 7. The figures show a snapshot of the SPIKE resource GUI. The GUI plots for the total orbits, SAA hiding (i.e. impacted), SAA free, effective SAA, and effective SAA-hiding resources. It is evident from the figures that plan validation engine distributes the resource usage more effectively.

	Total Orbits		SAA Free Orbits		SAA Impacted Orbits	
	LRP	VAL	LRP	VAL	LRP	VAL
Number of Days with Oversubscription	183	98	126	86	226	56
Total Oversubscription	1029	495	404	278	817.4	253
Avg. Oversubscription	5.6	5.05	3.2	3.23	3.6	4.51
Worst Oversubscription	18.3	32	18.6	16	13.6	32
False Positives	5		23		0	
False Negatives	90		63		170	

Table 4. Data from LRP and plan validation tool

Table 4 compares some characteristics of the resource plots produced by long range planning and the plan validation engine. The table shows the number of days with resource oversubscription, the total oversubscription (in orbits), the average oversubscription (in orbits) and the worst oversubscription (in orbits), for three different resources. The last two rows show the number of false positives and negatives in terms of number of days, for each resource. The days when the LRP shows resource oversubscription but the plan validation engine did not are counted as false negatives. On the other hand, there are days when the plan validation engine shows resource oversubscription but the LRP did not. These days are calculated as false positives. For example, the LRP showed 183 days of oversubscription for total orbits, out of which 90 days are identified as false-negatives. That leaves 93 days with real resource contention. The plan validation tool also identified 5 additional days resource contention that LRP failed to identify. That adds up to a total of 98 days with resource contention in plan validation tool.

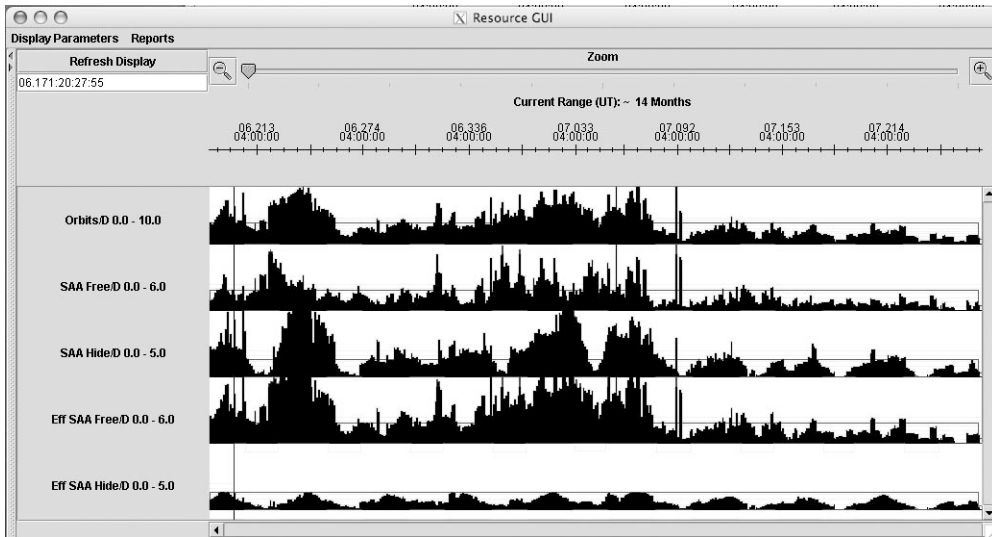


Figure 6. Resource Usage of Long Range Plan (LRP)

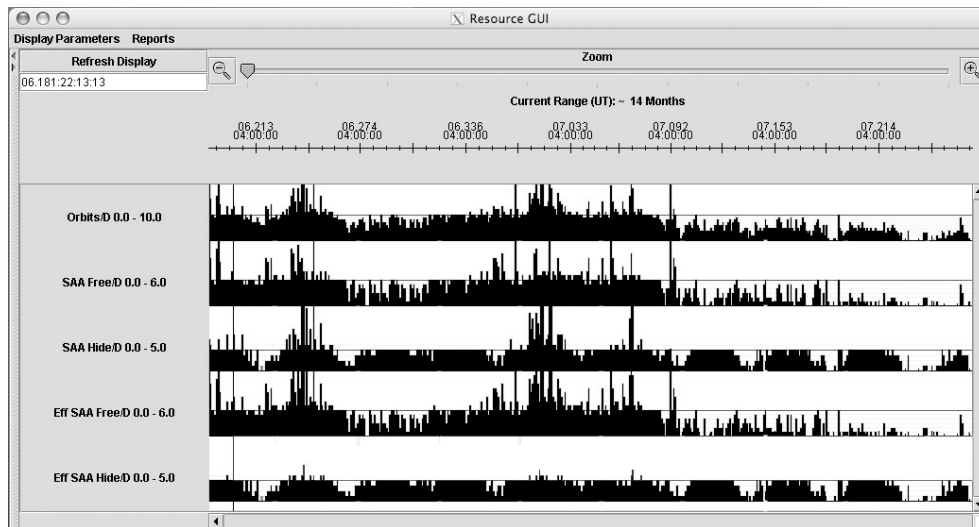


Figure 7. Resource usage of Plan Validation Engine (VAL)

The plan validation tool flattens the resource profile and points out regions with actual resource contention. Users can quickly identify the observations with resource conflict and take appropriate action, such as manually moving the plan window or changing the observations requirements so that undersubscribed regions can be filled in. Another very useful aspect of the plan validation resource graph is the effective SAA hiding plot. In the long range plan, SPIKE produces a worst-case scenario of effective SAA hiding. The validation engine gives better estimates of how much SAA hiding is possible. Moreover, the guided choice between all possible orbit assignments (e.g. full/partial/no SAA hiding) does not artificially lower the SAA free consumption.

Limitations

There are limitations of the iterative deepening repair algorithm presented. The number of possible intermediate moves that need to be evaluated grows fast as the *iteration* level increases. This limitation can be mitigated to a large extent by using heuristics to guide the search. Heuristics often depend on the application and designing effective heuristics require intimate knowledge of the domain. The search depth that can be explored within a reasonable run time depends on the application. With too many observations in an overloaded plan, very few moves are possible. The scheduler often evaluates moves that turn out to be futile.

For the HST plan validation engine, a few heuristics were designed that focus on the observation moves that are likely to yield a result. This reduces the number of moves that need to be examined during each iteration and has greatly improved our runtime. It also allows us to explore deeper into the search tree in a reasonable time. The following heuristics were used to select observations to move during resource repair.

- Target only those observations that can be moved without creating new resource conflicts.
- For intermediate moves, target only those observations that will free up the critical resources needed for an incoming observation. Critical resources are defined as resources that will have a new conflict if the incoming observation is placed before the intermediate move.
- Target observations that have wider suitability and higher probability for being scheduled elsewhere without creating new resource conflict.

The maximum iteration level (e.g. *max-level*) is set as 2 for current HST operations and is used to produce results presented in this paper. Most resource conflicts are resolved within the second level search. Searching at higher iteration levels rarely resolves more conflicts but takes a long time. The remaining resource conflicts are usually not solvable without changes to the underlying observation specifications.

Extending SPIKE for Long Range Planning

Both the plan validation tool and the SPIKE long range planner use the same interface with different values for a few parameters. By mistake operations staff ran the plan validation engine to generate long range windows. The mistake was beneficial as the validation algorithm produced a more resource balanced long range plan than

the current plan window scheduler. We realized the validation engine offers many improvements over the current long range planning algorithm including.

- Resource balancing during the initial guess phase of search.
- Systematic as opposed to stochastic repair of resource conflicts.

Based on this experiment, we extended long range planning to incorporate the iterative-deepening approach making modifications where necessary.

Figure 6 shows the resource profile of the long range plan produced by previous baseline version of SPIKE. There are many days with highly oversubscribed as well as undersubscribed resources. In these cases, LRP users would manually move observation plan windows to undersubscribed regions to achieve a more balanced resource profile. Figure 8 shows the resource profile using the new iterative deepening version of the long range planner. The resource profile of the corresponding plan validation run is shown in figure 9.

Table 5 lists the same characteristics of both resource profiles as in Table 4. The total resource oversubscription improved across the board. The number of days with resource contention increased for total orbits and for SAA impacted orbits. This is because the new SPIKE algorithm tries to create windows with the least resource contention even when the resource limit has to be violated. The planner avoids building up large resource overruns in any region. The average oversubscriptions are much less in the new long range plan. Due to better resource distribution in the input long range plan, the plan validation tool significantly reduced the number of resource conflicted days and total, average and worst oversubscription.

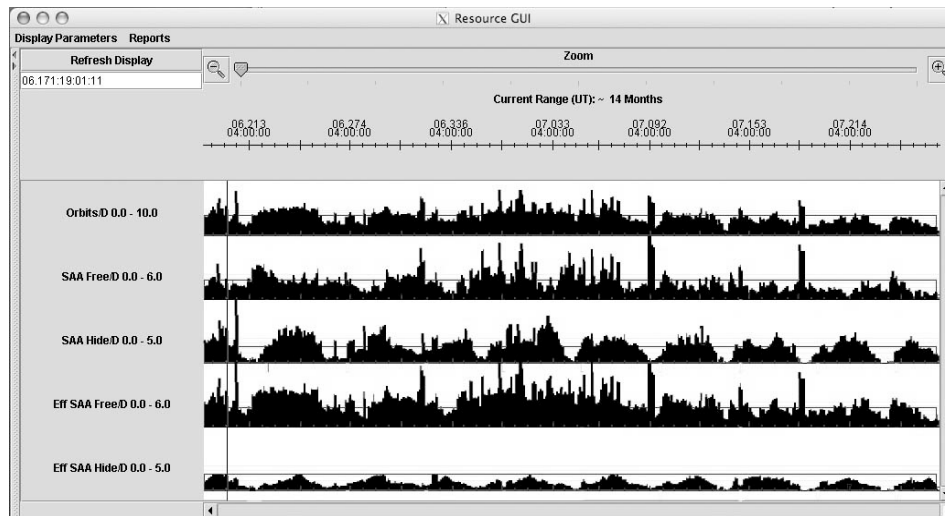


Figure 8. Resource Profile of Long Range Plan (LRP) after extending SPIKE

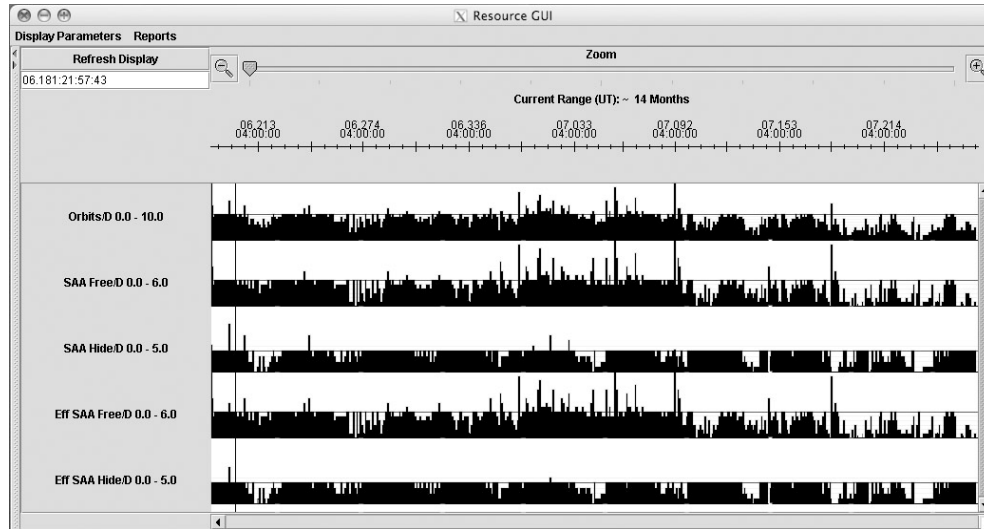


Figure 9. Resource Profile of Plan Validation Engine (VAL) after extending SPIKE

	Total Orbits		SAA Free		SAA Impacted	
	LRP	VAL	LRP	VAL	LRP	VAL
Number of Days with Oversubscription	221	44	157	45	272	6
Total Oversubscription	558	142	380.3	154	449.5	17
Avg. Oversubscription	2.52	3.27	2.42	3.42	1.65	2.83
Worst Oversubscription	9.58	15	10.02	15	10.3	5
False Positives	2		12		0	
False Negatives	179		124		266	

Table 5. Data from LRP and plan validation tool after extending SPIKE.

With fewer false positive/negative to analyze, the new iterative deepening version of SPIKE has reduced operational work and resulted in overall better output. Due to better resource balancing, very few plan windows have to be moved manually. HST operations has only 2-3 weeks to integrate a new cycle worth of observations into the long range plan. The new resource balancing algorithms in SPIKE, have proved tremendously effective for producing high quality results in short time span.

Conclusion

The SPIKE long range planning has been augmented with a plan validation engine and iterative deepening repair strategies. The new process has improved our ability to create and maintain resource balanced long range plans. Beside that, the tools also enable the operations staff to quickly identify problems.

Acknowledgements

The authors would like to thanks the reviewers as well as our colleagues Reiko Rager, Rob Hawkins, Ian Jordan and William Workman for their helpful comments. In

addition, we would like to thank all the past and present SPIKE developers for their efforts.

References

- Calvani, H.M. Berman, A.F. Blair W.P. Caplinger, J.R. England, M.N. Roberts, B.A. Hawkins, R. Ferdous, N. and Krugger A.P., 2004. The evolution of the FUSE Spike Long Range Planning System. *In Proceedings of the 4th International Workshop on Planning and Scheduling for Space. Darmstadt Germany.*
- Johnston, M. and Miller, G. 1994. Spike: Intelligent Scheduling of Hubble Space Telescope Observations. In Zweben M. and Fox M. eds. *Intelligent Scheduling*, 391-422. Morgan-Kaufmann.
- Korf, R. 1988. Optimal Path Finding Algorithms. In *Search in Artificial Intelligence*, 223-267, Springer-Verlag.
- Kramer, L. 2000. Generating a Long Range Plan for a New Class of Astronomical Observatories. *In Proceedings of the 2nd NASA Workshop on Planning and Scheduling for Space.*
- Kramer, L. and Giuliano, M. 1997. Reasoning About and Scheduling Linked HST Observations with SPIKE. *In Proceedings of the NASA Workshop on Planning and Scheduling for Space.* Oxnard, CA.
- Kramer, L. and Smith, S. 2004. Task Swapping: Making Space in schedules for Space. *In Proceedings of the 4th International Workshop on Planning and Scheduling for Space. Darmstadt Germany.*
- Krueger, T. 2004. APT: Hubble Space Telescope's Proposal Preparation Tool, *In Astronomical Data Analysis Software and Systems XIV.*
- Sasaki, T. Kosugi, G. Kawai, J.A. Kusumoto, T. Koura, N. Hawkins, R. Kramer L. Krueger, A.P. Miller G.E., 2000. Observation scheduling scheme for Subaru telescope, in SPIE.