Reasoning About and Scheduling Linked HST Observations with SPIKE

Laurence A. Kramer, Mark E. Giuliano

Space Telescope Science Institute (STScI)
Baltimore, MD 21218

1. Abstract

Scientific observations executed by the Hubble Space Telescope (HST) are subject to a number of complex and interacting timing and orientation constraints. We classify these constraints as "absolute" and "relative." The former apply to only one observation, whereas the latter apply to two or more linked observations or "visits." Examples of absolute constraints are **"Schedule Visit A BETWEEN Day 10 and Day 20"** or **"Schedule Visit A at an ORIENT in the range 30 to 40 degrees."** Relative constraints express either how the visits should be linked in time (for example **"Schedule Visit2 AFTER Visit1 by 10 days"** or linked in orientation (for example **"Schedule Visit2 ORIENT FROM Visit1 by 20 degrees"**).

This paper will focus on the implementation of relative constraints, how they are combined with absolute constraints, and the problems encountered and solved in combining relative constraints along the orthogonal dimensions of time and orientation.

We also briefly present the SPIKE Plan Window Scheduler (Giuliano 1997), which creates a Long Range Plan by assigning "plan windows" where the visits might feasibly be scheduled, and which optimizes the placement of windows to satisfy a number of criteria.

Since the assigned plan windows are much larger (nominally eight weeks) than the actual times needed to schedule the visits, we confront the problem of "link set flexibility." This implementation allows a great deal of latitude for a short term scheduler to later assign actual spacecraft execution times to visits. We also describe an enhancement of the Plan Window Scheduler to schedule orientation angles along with plan windows for orient linked visits, while confronting computational constraints.

2. Introduction

The SPIKE (Johnston and Miller 1994) software system is used both to determine where Hubble Space Telescope observations are schedulable and to craft a long range observing plan that is flexible while maximizing telescope usage. Scheduling in this large, complex domain is further complicated by dependencies between individual observations, or ***visits***, in an observing program. Approximately 39% of 2,444 HST visits for Cycle 7 are linked by timing and/or orientation constraints, and ensuring that these observations are scheduled correctly and planned optimally is a major operational concern for the Hubble ground system.

In this paper we describe problems that have arisen in reasoning correctly about linked observations and our solutions to these problems. We go on to investigate issues that arise in creating a long range plan that will assure a short-term scheduler as much flexibility as possible to create one-week calendars.

3. HST Domain

Scheduling observations on the Hubble Space Telescope involves first processing a number of interacting constraints to determine where an observation might possibly schedule, and then creating a plan based on where all observations, individually and as a whole, might *best* schedule. In practice, the SPIKE system is used to determine schedulability and a long range plan (***LRP***) over approximately a one-year time period. Other software systems are used to craft a weekly short-term schedule that will actually "fly" on the spacecraft.

Determining those times that are suitable for scheduling depends not only on constraints due to the nature of the Space Telescope and the celestial target being observed, but also constraints that the Principal Investigator (PI) has introduced into the observing program (or ***proposal***) in order to obtain desired scientific goals. For long range planning purposes, the HST and target are responsible for what we call ***absolute constraints,*** constraints that affect only a single observation. The PI, however, may specify conditions that are responsible for both absolute and ***relative constraints,*** constraints relating one observation to another.

3.1. HST Absolute Constraints

Hubble's low Earth orbit, solar panel power requirements, and scientific instrument sensitivities, combined with the target position over time, are responsible for numerous absolute constraints on observation scheduling. First of all, due to instrument sensitivity, the HST must not be pointed within a certain degree range from bright objects such as the Sun and Moon. This will restrict when targets can be viewed to those times when their pointing is not too close to a bright object. In viewing a target, the HST must be oriented both so that it does not overheat, but also to provide adequate power from the solar panels. This ***orientation constraint*** is for most purposes treated as another absolute ***timing constraint***, i.e. those times for which it is legal to achieve a certain orientation. The distinction between orientation and timing constraints, produces another dimension, orthogonal to that of absolute/relative. We will explore these dimensions in some detail below.

In crafting her program, the PI can introduce various absolute constraints on the HST. For instance, it may be desirable to view a target only between certain dates, although the target's window of visibility might actually be much greater. This is expressed as **[VISIT n] BETWEEN <DATE1> AND <DATE2>**. Other constraints might be **BEFORE** or **AFTER** a certain date. Similarly, the PI might want to restrict the spacecraft orientation (roll angle) relative to an instrument aperture to some absolute degree range (roll range), expressed as **[VISIT n] ORIENT <ANGLE1> TO <ANGLE2>**.

3.2. Representation of Constraints in SPIKE

SPIKE implements constraints through the use of piecewise continuous functions or pcfs (Johnston and Miller 1994), which represent schedulability over time. A pcf can be represented as a list of time intervals and ***suitability*** values, ($time_1$ $suitability_1$ $time_2$ $suitability_2$ $time_3$ ...). For the purposes of this paper, however, we will discuss SPIKE constraint representation and manipulation in terms of ***non-zero intervals***, i.e., those time intervals that have non-zero suitability for scheduling, and we will use the notation **(($time_1$ $time_2$) ($time_3$ $time_4$) ...)** to represent the suitable intervals for a given constraint as well as overall schedulability for a visit. Unless there is a need to be more precise, we will use the terms *suitability*, *non-zero intervals*, *constraint windows* and *schedulability* interchangeably.

Each $time_n$ will be denoted by an integral "logical" date, as opposed to a real calendar date. For instance, the constraint that Visit1 be scheduled between November 1 and November 10 or

Visit1 BETWEEN 305 and 314

will be expressed as the non-zero intervals **((305 314))**. {1}

Now suppose suitability for Visit1 due to Sun and Moon avoidance is expressed as the non-zero intervals **((200 312))**. {2}

Then the ***absolute suitability*** for Visit1 is **((305 312))**, {3}

which is derived by intersecting suitable intervals from {1} and {2}.

Similarly, an absolute orientation constraint may be converted to a time suitability function. For instance, the constraint

Visit1 ORIENT 30deg. TO 40deg.

may be equivalent to the non-zero intervals **((302 311))**. {4}

This says that orients in the range 30 to 40 degrees can be achieved between day 302 and day 311. Intersecting {4} with {3}, we get an overall suitability function, or schedulability of **((305 311))**. {Visit1 Suitability}

3.3. HST Relative Constraints

Relative constraints express relationships between two or more visits. PIs have at their disposal a rich language for expressing both timing and orientation constraints between distinct observations. Continuing our previous example, suppose there is a second observation, Visit2, which should be scheduled after Visit1. Visit2 has combined schedulability due to absolute constraints of **((308 314))**. {Visit2 Absolute Suitability}

In addition the PI desires the following constraints:

Visit2 AFTER Visit1 by 5 to 10 Days {Rel. Constraint 1}
Visit2 SAME ORIENT AS Visit1 {Rel. Constraint 2}

The former expresses a ***relative timing constraint***, and the latter a ***relative orientation constraint***.

4. Combining Absolute Constraints and Relative Constraints

In order to derive the suitability for visits linked by relative constraints, we express as non-zero intervals the effects of each relative constraint on all the visits linked by that constraint, and intersect the relative suitability intervals with the absolute constraint suitability intervals for each visit. Considering first Relative Constraint 1 above, since Visit1 is schedulable between 305 and 311, Visit2 may be scheduled between 310 and 321 (*at least* five days after the earliest date and *at most* ten days after the latest date). Thus, Visit2's ***relative suitability function*** is

((310 321)). {Visit2 Relative Suitability}

Similarly Visit1 may be scheduled *at most* ten days earlier than Visit2's earliest date and *at least* five days before

Visit2's latest date, producing a relative suitability for Visit1 of

((298 309)). {Visit1 Relative Suitability}

Now relative suitability intervals can be combined directly with the absolute suitabilities to produce suitability functions for Visit1 of

((305 309)) {Revised Visit1 Suitability}

and for Visit2

((310 314)). {Revised Visit2 Suitability}

In the general case, there may be more than two visits linked by a relative constraint, and a number of relative constraints which interact with each other. This could necessitate several iterations of constraint propagation until suitability functions of the linked visits stabilize to a final value.

It is important to note that once relative constraints are introduced between visits, the strength of what the suitability function expresses for a visit diminishes. For all visits not linked by relative constraints we can state that they will be schedulable at *every* time with non-zero suitability in their suitability function. For visits linked by relative constraints we can only state that a suitability function represents *potential* schedulability, and once a visit in the link set becomes scheduled there *will exist* time intervals with non-zero suitability at which other visits in the link set will be schedulable.

This is easy to see from the example above. Initially, 310 is a perfectly legal date to schedule Visit2, but once Visit1 is scheduled on 306, the interval up to 311 becomes unsuitable for Visit2 (due to the After constraint). In SPIKE we handle this problem by introducing "***execution time constraints***" after a visit is placed on a short term schedule, which have the effect of collapsing a visit's suitability function to its actually scheduled time.

4.1. Propagating Relative Suitability

We state a property of *potential schedulability* to which any algorithm for propagating relative suitability must adhere:

Property of Potential Schedulability

Define a ***link set***, L, as the transitive closure of all visits in a proposal, P, mutually reachable through relative constraints (both timing and orient). Visits in a proposal that are not reachable through relative constraints form singleton link sets.

Then, For every visit V in L and for every point in time where V has non-zero suitability, there must exist a point in time having non-zero suitability for every visit linked to V, these time points representing a fully feasible schedule for L. In other words, suitability functions for a link set must contain no *a priori* unschedulable time intervals.

The method for propagating relative suitability for visits in a proposal can be expressed algorithmically as follows:

Algorithm for Propagation of Relative Suitability

```
For each visit V
    set suitability(V) = absolute-suitability(V)
End for
set Changed = True
While (Changed)
    Changed = False
    For each visit V
        set S = suitability(V)
        For each rel. constraint C which affects V
            set R = Apply C to V
            set S = Intersection (S, R)
        End for
        If S <> suitability(V)
            set suitability(V) = S
            set Changed = True
    End for
End while
```

4.2. Propagating Relative Orient Constraints

In the discussion above we have glossed over how relative *orient* constraints are propagated. As with absolute orient constraints, a good approach might be to somehow convert the relative orient constraint roll range to a relative timing suitability function and then propagate this suitability as we have described.

Consider Relative Constraint 2. It states that Visits 1 and 2 must be scheduled at the same orientation. In other words, if Visit1 is scheduled at 59 degrees, Visit2 must be scheduled at 59 degrees. If Visit2 is scheduled at 120 degrees, so must Visit1 (there is no preferred "first" visit to an orientation constraint).

To compute a relative suitability function for this constraint for Visit2, recall that Visit1 is schedulable between days 305 and 309. Assume that over this time period orientations in the roll range from 34 to 38 degrees are allowed. We then consider during which time periods Visit2 can achieve a roll in the [34 38] range, say days 308 to 312. Then, Visit2's initial relative suitability function due to Relative Constraint 2 and Visit1 is

((308 312)). {Visit2 Relative Orient Suitability}

This suitability will be combined with Visit2's absolute suitability to produce a new suitability. A relative suitability function for Visit1 based on roll ranges that Visit2 can achieve is similarly computed. Suitabilities continue to be propagated in this fashion until they become stable (they no longer shrink or become zero).

4.3. A Problem Reasoning About Relative Orient Constraints

SPIKE was originally designed to use this method (Sponsler 1990) of propagating relative orient constraints, but problems with converting back and forth between roll ranges and time suitability functions were uncovered and a different propagation methodology has been developed.

In order to illustrate the problem, we extend the preceding example. We now have three visits in the link set with several constraints:

Visit1 ORIENT 30deg. TO 40deg. {Absolute Constraint}
Visit2 SAME ORIENT AS Visit1 {Rel. Constraint 1}
Visit3 ORIENT 10deg. FROM Visit2 {Rel. Constraint 2}

Suppose all the absolute constraints on Visit1 imply a schedulability period of day 302 to day 311 for that visit. Converting this to an orient range and propagating to Visit2, we compute that Visit2 is schedulable between day 308 and day 316. Now, to propagate constraints to Visit3, we must convert Visit2's suitable time interval to a roll range and then apply an "orient-from" operator to this roll range, finally converting it to a time interval for Visit3. Suppose the available roll range for Visit2 in the day 308 to day 316 time frame is [30 41]. Then Visit3 must be scheduled 10 degrees from this, or in the [40 51] range. We convert this to a time interval for Visit3, say day 340 to day 350.

While the schedulability intervals for Visits 1 and 2 can be guaranteed to be good, it is quite possible that the interval for Visit3 may contain false positives for scheduling opportunities. By this we mean that even before any visits in the link set are scheduled there may exist illegal time intervals in Visit3's suitability function.

Suppose both Visit1 and Visit2 are scheduled somewhere within their legal time intervals and the short term scheduling engine attempts to schedule Visit3 on day 350, but finds that it can only schedule in the roll range [51 55]. To schedule anywhere in that range will violate the orient constraint of 10 degrees from Visit2 (whose range is [30 40]), and thus the Property of Potential Schedulability (day 350 is a priori unschedulable). How is this possible?

The error has cropped up when Visit2's suitable time frame is converted to the roll range of [30 41]. In doing so, information has been lost about the absolute orient constraint on Visit1 of [30 40], which Visit2 must implicitly honor through the same orient constraint. Note that this is not an error in computation, but a true flaw in our methodology.

While it is true that the time frame day 308 to day 316 for Visit2 is the full extent at which a [30 40] roll range can be achieved, it may be *possible* to achieve a roll angle of 41 during that interval, as well. A sound algorithm should exclude 41 from the roll range of Visit2 and 51 from the roll range of Visit3.

4.4. A Better Algorithm for Propagating Relative Orient Constraints

Our solution to this problem is that while propagating relative orient constraints, roll range information must be preserved throughout the propagation, and not converted to final time suitabilities until the propagation has stabilized. To implement this we revise the Algorithm for Propagation of Relative Suitability roughly as follows:

Revised Algorithm for Relative Suitability Propagation

Define a Total Roll Restriction RR for a Visit V to be the set of roll angles from [0 360) at which it is feasible to schedule V.

```
For each visit V
    set suitability(V) = absolute-suitability(V)
    set RR(V) = [0 360)
End for
set Changed = True
While (Changed)
    Changed = False
    For each visit V
        set S = suitability(V)
        set Roll = RR(V)
        For each rel. constraint C which affects V
            set values R, RR_r = Apply C to V
            set S = Intersection (S, R)
            set Roll = Intersection (Roll, RR_r)
        End for
        If S <> suitability(V) or Roll <> RR(V)
            set RR(V) = Roll.
            set suitability(V) = S
            set Changed = True
    End for
End while
```

Roll restrictions that are intersected to produce a total roll restriction on a visit include restrictions due to legal roll angles for the visit due to its absolute suitability, restrictions due to absolute orient constraints, the actual angle at which a visit is scheduled (analogous to an execution time constraint), affects from other linked visits through relative orientation constraints, etc.

Reworking our prior example, we illustrate this new algorithm: Suppose the absolute timing constraints on Visit1 imply a schedulability period of day 302 to day 311 for that visit and a possible roll restriction of [23 42]. We calculate a preliminary total roll restriction for Visit1 of [30 40] by intersecting with the absolute orient constraint roll range. Propagating to Visit2, its roll restriction must also be [30 40], by the Same Orient constraint. Assume that this is feasible, otherwise Visit1's roll restriction would be further constrained. Based on this roll restriction, Visit2 is schedulable between day 308 and day 316. To propagate constraints to Visit3 apply an "orient-from" operator to Vist2's roll restriction, producing a roll restriction of [40 50] for Visit3. We convert this to a time interval for Visit3, day 340 to day 349.

Notice that the end result of this process drops the problematic day 350 from Visit3's suitability function and ensures that all visits will be scheduled where both their timing and orient constraints will be strictly obeyed.

4.5. Yet Another A Problem Reasoning About Relative Orient Constraints

Implementation of the improved algorithm for propagating relative orient constraints has greatly reduced the incidence of false positives in computing visit schedulability. However, it has recently come to our attention that this solution is not completely correct. Consider the following simple example:

Visit2 AFTER Visit1 by 80 to 90 Days {Rel. Constraint 1}
Visit2 ORIENT 180deg. From Visit1 {Rel. Constraint 2}

Suppose there are no other constraints on either visit and the planning interval we are considering is for a full year. Both Visits initially have unlimited schedulability, which is only slightly constrained after considering the After constraint. Visit1 will have a suitability function of

((1 285)), {Visit1 Suitability}

while Visit2's suitability will be truncated at the other end:

((81 365)). {Visit2 Suitability}

Assume that due to these large time intervals neither Visit1 nor Visit2's roll range is restricted beyond the full [0 360) range. Applying any Orient From offset to a full roll range returns a full (unrestricted) roll restriction of [0 360). Therefore both visits' suitability functions remain unchanged due to the Orient From constraint.

Unfortunately this solution is not correct. It turns out that if Visit1 is scheduled on day 1, Visit2 is not schedulable on day 81 as we might expect. This is due to the fact that for most celestial targets, HST's legal nominal roll range only varies by about a degree a day, and even allowing for as much as 30 degrees off-nominal, it would be impossible to span 180 degrees (required by the Orient From constraint) in 80 days.

Again, the Property of Potential Schedulability has been violated, as our suitability functions contain days that have no schedulability. What's worse, in this case *all* times in the suitability functions are unsuitable as it is impossible (at least for this example) to schedule a visit 80 to 90 days after another and 180 degrees from it!

How could such an egregious error be missed? In practice, such underconstrained proposals where the orient and timing links line up so perversely are very rarely encountered. We designed our new algorithm specifically to handle orient information, though, so what is its flaw?

The error occurs in propagating roll ranges as global entities which apply to a visit without regard to the time interval. In actuality a roll range for a visit is only "good" for a restricted period of time. For instance Visit1 may have an allowable roll range of [20 50] on day 1, an allowable range of [21 51] on day 2, and so on. By day 10 the roll range might be [29 59]. In the time interval ((1 10)) the allowed range applicable over the entire interval would then be [29 50].

This example, although typical, is very arbitrary. For some targets the roll range will remain relatively constant over time, and then change radically in a one-day interval. In theory then, it is virtually impossible to craft a combined relative timing and orient link propagator that will not violate the Property of Potential Schedulability.

4.6. A (Best Effort) Solution To The Orient Propagation Problem

In practice though, we can come arbitrarily close to a sound propagation algorithm by modifying our current algorithm to build up a suitability function in small time increments. In other words, we run the same algorithm but one day (or one hour) at a time and aggregate a suitability function from these iterations. Clearly there is a time slice at which this becomes *computationally* infeasible, but initial prototypes have shown that a one-day sampling period should be both tractable and correct for almost all real world HST observation programs.

5. The SPIKE Plan Window Concept

We have gone into some detail presenting problems and solutions for reasoning with linked observations in an HST observing program. Now, we briefly present the SPIKE Plan Window concept and the SPIKE Plan Window Scheduler, which generates a Long Range Plan (LRP) for a cycle's (typically one to two years) worth of proposals. We then go on to confront the issue of "link set flexibility."

In constructing a Long Range Plan we encounter the conflicting goals of producing a plan that should be as stable as possible over time, while allowing frequent revision of proposals necessitating changes in where they can schedule (See (Giuliano, 1997) for a full discussion of these issues). Historically, the SPIKE system generated an LRP where visits were assigned to one-week windows, allowing a short-term scheduler to assign an actual time within the week.

This method proved to be somewhat inflexible as replanning took place over time, often ending up with weeks where there were too many visits to schedule, and other weeks that were undersubscribed. If a visit missed its one-week window, often it could only be rescheduled a full year later.

To address these problems we implemented the SPIKE Plan Window Scheduler. Instead of scheduling visits to a fixed week in the plan, it schedules them to an eight-week "***plan window***," any week in which is suitable for scheduling (of course, many visits are so constrained as to have constraint windows less than eight weeks in duration). This implementation has proved in practice to lead to a more stable while flexible LRP. Each week the short-term scheduler has a pool of visits from which to select and craft an efficient schedule. If a visit cannot be placed on the current calendar, it can usually be placed on a later calendar within its assigned plan window.

5.1. The SPIKE Plan Window Scheduler

The SPIKE Plan Window Scheduler works roughly as follows:

Plan Window Scheduler Algorithm

Given an input LRP (null for the first iteration) and a set of link sets (including singletons) to schedule, Do for each link set L:

1. Iterate (one day at a time) over time for the entire planning period and the link set's constraint windows, assigning plan windows to each visit in the link set.
2. Score each set of plan window assignments based on various scheduling, planning, and resource criteria.
3. Make a final assignment of plan windows, those with the highest score, to L.

If there are any oversubscribed regions in the plan so generated:

1. Execute a stochastic repair algorithm which selects link sets to reschedule.
2. Attempt to reschedule these link sets subject to resource and other criteria.

Save the resulting plan as the new LRP.

In practice a new LRP is generated on a daily basis. Generally, if a link set has been assigned plan windows in today's LRP, and no changes are made to that link set, it will retain its original plan windows in tomorrow's and succeeding LRPs.

5.2. Assigning Plan Windows *and* Angles

We have described in general how SPIKE assigns plan windows for link sets, but have neglected the subject of assigning orientation angles. For visits that are unaffected by orientation constraints, an orientation is typically assigned (outside of SPIKE) at the nominal orientation. For orient linked visits, though, this has up until very recently been a tedious and error prone manual process.

An enhancement to the SPIKE Plan Window Scheduler has been to schedule orientation angles for orient link sets. We discuss our implementation and some time complexity challenges we have faced.

Recall that as an output of the Revised Algorithm for Relative Suitability Propagation we compute a Total Roll Restriction RR for each orient linked Visit V. To schedule "optimal" orientation angles and plan windows we implement the following:

Algorithm for Assigning Plan Windows and Orient Angles

For a link set L having relative orient links Define $\text{Visit}_{\text{first}}$ to be that visit in the link set with the most highly constrained roll restriction, RR. Call this RR_{first}.

1. Select an angle, A, from RR_{first} and set RR_{first} = [A A]. Propagate this restriction through the link set, constraining each linked visit.
2. For each Visit_i orient linked to $\text{Visit}_{\text{first}}$, select and propagate an angle from its constrained RR_i.
3. Execute the Plan Window Scheduler Algorithm as previously outlined (iterate through time, testing plan windows, assigning the most highly score plan windows, *and in addition*, orient angles).
4. Until all valid combinations of angles from (RR_{first}, RR_i) have been exhausted, return to step (1) and repeat.

Note that this algorithm can be quite time consuming. Basically, to schedule each link set L with relative orient links takes $N * T_L$, where N is the number of valid combinations of angles for L and T_L is the time it would have taken just to assign plan windows to L.

What is a bound for N? The worst case value for N can be computed as follows:

Let m be the number of orient linked visits in L, then an upper bound on N is 360^m (assuming sampling at 1-degree increments).

In practice N is typically much smaller than this value, both because each RR_i is generally much smaller than [0 360), and also because the RR_i are not independent of each other. For example, suppose we have a link set L with m visits linked by a Same Orient constraint, then we can bound N as 360, no matter how great m is. In this case, selecting an angle from RR_{first} constrains each angle in RR_i to be identical, thus limiting the number of distinct combinations to be 360.

Since most link sets happen to be simple Same Orient link sets, and since most roll ranges are far more constrained than a full 360-degree range, a conservative average case time estimate for N is 180. In other words, for link sets with orient links, assigning plan windows *and* angles takes about 180 time longer than assigning plan windows alone.

As we previously mentioned, the Plan Window Scheduler is run nightly to generate a new LRP, so the enhancement of scheduling angles and plan windows must not take so long as to cause the LRP run to be longer than about six hours.

In order to reduce the time necessary for selecting angles, we have introduced a sampling algorithm, which significantly reduces the average case time for scheduling plan windows and angles:

Grid Search Algorithm for Sampling Angles

1. For each $Visit_i$ in L, set the grid size $GS_i = 5 * (\text{ceiling } (\text{size } (RR_i) / 90))$. I.e., the grid size increases by five, for each 90 degrees of roll range.
2. Execute the Algorithm for Assigning Plan Windows and Orient Angles, sampling angles at intervals of GS_i for each $Visit_i$. After the "grid" has been fully sampled, select the two highest scored angles, and search in both directions in integral increments from these angles, terminating the search when half the grid size or a worse angle has been reached for each angle and each direction.

Given our conservative average case scenario of $RR_i = 180$, GS_i will be equal to 10, and thus the number of samples will be approximately 18. Thus we have achieved an order of magnitude speed up. Preliminary tests with sampling angles shows that the domain is smooth enough that we sacrifice no accuracy by sampling compared to the brute force approach, and miss no local maxima.

5.3. Link Set Flexibility

When SPIKE creates plan windows for linked visits it needs to ensure that the windows allow flexible scheduling of all visits in the set. In general if a visit has at least eight contiguous weeks of suitability, it makes sense to assign it an eight week plan window, thus maximizing scheduling opportunities. We have uncovered a counterintuitive result, however, that extending the plan window for one visit in a link set may actually greatly *decrease* scheduling flexibility for other visits in the link set. Consider the following example:

Visit1 is suitable days 1 to 30. Visit2 is suitable days 21 to 50. In addition we have the constraint

Visit2 AFTER Visit1 by 20 to 30 Days.

If Visit1 is scheduled on day 1 then Visit2 is schedulable days 21 to 31. In this case the full link tolerance is available. In contrast if Visit1 is scheduled on day 30 then Visit2 is only schedulable on day 50. In this case only 10% of the link tolerance is available.

SPIKE should not include day 30 in the plan window for Visit1. In the above example creating a window for Visit1 from 1 to 20 will ensure that the minimum size window for Visit2 is 10 days long.

Our solution to the flexibility problem is that SPIKE should create plan windows which maximize the guaranteed minimum window size of the entire link set. A technical definition of the concept is developed below.

The discussion given is independent of the type of link set (e.g. timing, or relative orient). The algorithm measures flexibility in terms of days, which is sufficient for a Long Range Plan. However, the algorithm could be modified to measure flexibility in finer units if desired. In general SPIKE chooses plan windows which optimize a set of criteria out of which link set flexibility is one criterion. In practice, then, flexibility may be somewhat compromised to benefit other planning and scheduling criteria.

Algorithm For Ensuring Link Set Flexibility

Let L be a link set (i.e. the transitive closure of timing and relative orient links in a proposal).

Let Pw be a set of plan windows for the link set L, V is a visit not equal to the first visit in L, and D is the day where the first visit of the link set is scheduled.

Then define Actual(L,V,D) as the raw number of days that V can be scheduled.

Define Actual_Flex(L,D) = Min for all V in L {Actual(L,V,D)}

Define First(L,Pw) to be the window for the first visit in the link set.

Define GMWS(L,Pw) as the guaranteed maximum window size for link set L given that it has plan windows Pw.

We can now give a formalism for computing the guaranteed maximum window size:

GMWS(L,Pw) = Min (size(First(L,Pw)), Min for all D in First(L,Pw) of { Actual_Flex(L,D) })

The GMWS measure can be used as an evaluation criterion for selecting plan windows. In a world with no computation costs we would generate all possible plan windows for a link set and then evaluate them subject to this criterion. However, we cannot possibly generate all possible combinations of windows.

A practical approach is to prune times from the plan window for the first visit in a link set which do not maximize GMWS. Given a candidate set of plan windows for a visit determine the subset of the assignment for the first window in the visit which maximizes the guaranteed minimum size window for all the visits in the link set.

Define Rest(L,Pw) to be the plan windows for the visits other than the first visit in the link set L.

Prune(L,Pw) = Determine the subset S of First(L,Pw) which maximizes { GMWS(L, S union Rest(L,Pw) }

The scheduler would use the flexibility code in two ways:

1. Given a candidate window starting in a day optimize the flexibility using the prune operator.

2. Use the GMWS measure as a criterion to compare different plan windows.

Two additional issues need to be addressed.

1. Below a certain link tolerance SPIKE does not have the constraint accuracy to measure flexibility. For example, SPIKE could not meaningfully measure flexibility for the link **Visit2 After Visit1 by 50 days plus or minus 12 hours.**

2. For chain links we may want to repeat the flexibility procedure when the first visit becomes executed. After the first visit becomes executed the second visit becomes the new "first" visit.

6. Summary

Over the past seven years the SPIKE system has been used operationally in the planning and scheduling process for the Hubble Space Telescope. In refining the system, we have tackled thorny issues related to reasoning about and scheduling linked observations. Recent advances include a better approach to combining constraints along the orthogonal dimensions of time and orientation, and creating a more flexible Long Range Plan.

Acknowledgment

We would like to thank Wayne M. Kinzel of STScI for his contributions to and clarifying discussions of this work.

References

Giuliano, Mark 1997. *Achieving Stable Observing Schedules in an Unstable World.* ADASS '97, Sonthofen, Germany.

Johnston M.; Miller G. 1994. *Spike: Intelligent Scheduling of Hubble Space Telescope Observations.* in Zweben M., and Fox M. eds. *Intelligent Scheduling* (San Francisco: Morgan-Kaufmann), ISBN 1-55860-260-7 (1994), pp 391-422.

Sponsler, J.L. 1990. *The ORIENTATION Constraints.* SPIKE Technical Report Number 1989-21, Revision B, STScI.