SPIKE: Intelligent Scheduling of Hubble Space Telescope Observations

Mark D. Johnston and Glenn E. Miller
Space Telescope Science Institute[*]
3700 San Martin Dr.
Baltimore, MD 21218

1.0 Introduction

This paper describes the SPIKE system, a general framework for scheduling which has been developed by the Space Telescope Science Institute for NASA's Hubble Space Telescope (HST). Efficient use of astronomical observatories is very important to the scientific community: the demand for research-grade telescopes far exceeds the supply. The need for efficient scheduling is especially keen for space-based facilities due to their very high cost, limited numbers, and unique scientific potential. The SPIKE scheduler was developed for Hubble Space Telescope but was designed for generality and flexibility: it has since been adapted for several other astronomical scheduling problems, as well as to problems unrelated to astronomy. While the general approach taken in SPIKE was motivated by the paradigm of scheduling as constraint-directed search (e.g. Fox 1987, Smith et al. 1986, Fox et al. 1989), SPIKE incorporates novel approaches to both the quantitative representation and propagation of hard constraints and "soft" preferences, and to the use of scheduling search strategies based on multistart stochastic repair.

In the following we first provide a brief overview of the HST scheduling problem, then we discuss the theoretical and conceptual foundations of SPIKE: Section 2.0 describes the SPIKE representation of constraints as "suitability functions", Section 3.0 casts the HST problem as a constraint satisfaction problem (CSP), and Section 4.0 describes the multistart stochastic repair search strategy that is currently the primary scheduling search technique in SPIKE. In Section 5.0 we discuss the HST science ground system as a whole and the role played therein by SPIKE. Section 6.0 describes our operational experience and some of the lessons learned from the past two years of HST operations. Section 7.0 provides a brief overview of the adaptation of SPIKE to other space- and ground-based observatories.

[*]Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1.1 HST Scheduling

Launched in April 1990, the HST has provided important new capabilities for astronomical research due to its unsurpassed combination of wavelength coverage and angular resolution. Despite a manufacturing flaw in the primary mirror, HST is actively engaged in a full research program that has produced a large number of exciting results. HST will be serviced by the Space Shuttle in late 1993 to compensate for the mirror's figure and to replace the main camera with a second-generation instrument. These changes will bring the optical quality of the telescope up to original expectations.

The HST scheduling problem ranks among the largest and most complex scheduling problems faced on a continuing basis: some 10,000 to 30,000 observations are scheduled per year and each is subject to a large number of operational and scientific constraints. Proposers can specify a variety of constraints on exposures in order to express scientific goals: these include relative timing requirements such as precedence, minimum and maximum time separations, ordering, interruptibility and repetition. Some observations must be executed within a certain absolute time interval. Proposers may constrain the orientation of an instrument's aperture relative to the target or require an observation to be made while the HST is in the Earth's shadow. In order to provide flexibility, proposers can mark exposures as “conditional” or “select”: “conditional” exposures are contingent upon the results obtained from another exposure in the observing program, or, in some cases, upon the results obtained from ground-based observations. These exposures are not scheduled until the proposer has notified the STScI that the condition has been satisfied. The “select” capability allows the proposer to identify alternative sets of exposures from which one or more will be picked for execution. As with “conditional” exposures, exposures contained in “select” sets will be placed on a timeline only after the proposer informs the STScI of a decision.

The HST spacecraft and its associated ground system components introduce a number of scheduling constraints as well. The observatory is in a low earth orbit (590 km) with the result that the Earth typically blocks the line of sight to a target for slightly less than half of each 95 minute orbit. Targets within a few degrees of the orbital poles are not occulted by the Earth and are suitable for uninterrupted observations. Due to precession of the orbit, the orbital poles move around the sky with a 56 day period, with the result that a target is available for extended observation for no more than about 3 days in each precessional period. When passing over South Atlantic Ocean, the HST encounters a portion of the Earth's radiation belt (called the South Atlantic Anomaly, or SAA) during which instrument operations must be suspended. Sources of bright light such as the Sun, Moon and illuminated Earth must be avoided. Thermal and power restrictions limit how the telescope can be oriented, in order to keep adequate sunlight on the solar panels and off the surfaces which radiate heat.

The primary resource constraint for HST scheduling is the amount of observing time available. Other resources which must not be exceeded include the amount of data which can be processed by the communications and ground systems, onboard tape recorder storage, and onboard command computer storage.

Although the HST operates largely in a preplanned mode (with schedules fixed about two months in advance of execution), disruptions to the schedule occur for a variety of reasons. The most welcome disruptions are so-called “targets of opportunity”, which are rare, important astronomical events requiring immediate attention (e.g. a supernova). Other schedule disruptions result from spacecraft anomalies, loss of communications contacts, and changes in observing programs.

Figure 1 illustrates schematically the range of constraint timescales for HST. The interaction of so many constraints on varying timescales makes it impossible to identify any one dominant scheduling factor. Many of the constraints are periodic with different periods and phases. As a consequence there are generally several opportunities during a year to make a particular observation and a prime goal of HST scheduling is to make an optimal choice among these opportunities for as many observations as possible. This effort is complicated by the fact that the majority of the requested exposures have timing, grouping, repetition, or ordering constraints that couple very strongly with the time-dependent constraints of Figure 1. More extensive discussions of the HST scheduling problem may be found in Miller et al. (1987, 1988) and Johnston (1988a, 1989b, 1990).

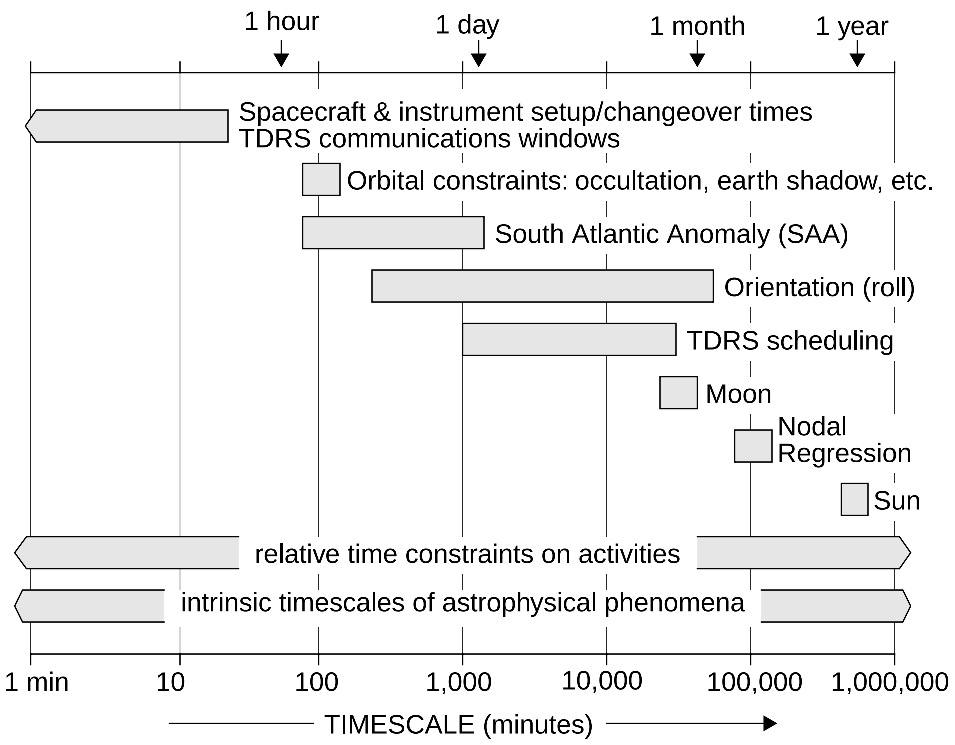

1 hour
1 day
1 month
1 year
Spacecraft & instrument setup/changeover times
TDRS communications windows
Orbital constraints: occultation, earth shadow, etc.
South Atlantic Anomaly (SAA)
Orientation (roll)
TDRS scheduling
Moon
Nodal Regression
Sun
relative time constraints on activities
intrinsic timescales of astrophysical phenomena
1 min
10
100
1,000
10,000
100,000
1,000,000
TIMESCALE (minutes)

FIGURE 1. The range of timescales for HST scheduling constraints, covering more than six orders of magnitude.

2.0 Representation and Combination of Constraints and Preferences

Constraints convey two types of information to the scheduler:

Feasibility constraints specify conditions or times when activities may or may not be scheduled. We interchangeably use the terms “strict” or “hard” constraint for this type, as they may not be violated under normal circumstances. A mechanism for relaxing (that is, violating) strict constraints is discussed in Section 3.0. A few examples in the HST scheduling context are:

- provide a minimum of two months between an observation and a repeat observation on the same target
- never schedule an observation when the Sun is within 50° of the target
- don’t roll the spacecraft more than 30° from its nominal orientation

Preference constraints specify quality judgments on scheduling conditions which are preferred but not required. These may be based on based on objective or subjective factors. In HST scheduling, for example, it is desirable to schedule at times which:

- minimize scattered light from the bright limb of the Earth
- maximize the chance of successfully acquiring guide stars
- place an observation as close to nominal roll as possible

It is important that both feasibility and preference information be considered simultaneously during schedule construction. Ignoring feasibility constraints can obviously lead to unimplementable schedules, but disregarding preference constraints (in order to simplify the problem) can lead to unacceptably suboptimal schedules. For this reason the concept of suitability functions (Section 2.1) was developed by merging ideas from two well-studied frameworks, namely constraint satisfaction problems (CSPs) for expressing and manipulating feasibility constraints, and evidential reasoning techniques as a means to combine preference constraints.

Consider scheduling some activity A_i, given that other activities A_j are already scheduled at timest_j. A human scheduler would assess the opportunities for scheduling A_i at various times by considering the effects of the activities A_j on A_i via the constraints. Constraints can take a variety of forms, but can be generally be cast into statements of the following type:

Given that activities $A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_N$ are scheduled at times $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_N$, the degree of preference for scheduling activity A_i at time$t$ due to constraintC_α is $W_{i\alpha}(t_1, \ldots, t_{i-1}, t_i{=}t, t_{i+1}, \ldots, t_N) \equiv W_{i\alpha}(t;t_{j\neq i})$

The degrees of preference can be assigned over some numerical range based on a judgment of the importance of the constraint, with larger values of $W_{i\alpha}$ corresponding to greater preference. $W_{i\alpha}$ can represent both deterministic constraints and intrinsically unpredictable constraints, e.g. $W_{i\alpha}$ can also be formulated in terms of (a function of) the probability that some desirable condition will hold.

2.1 Combination of Preferences: Suitability Functions

In general there will be a number of constraints acting on a task, so it is necessary to combine the degrees of preference $W_{i\alpha}$ from all applicable constraints. This combination process is formally similar to that employed in a number of rule-based expert systems which assess evidence for and against various conclusions (e.g. Shortliffe 1976, Hart et al. 1978). While this approach to uncertainty reasoning is known to have its limitations — In particular, the knowledge base should form a tree so that no evidence is counted twice via alternative paths of reasoning (Pearl 1988) — it is adequate for many scheduling problems and has the advantage of being computationally tractable.

However, the techniques used in rule-based systems for evaluating evidence for or against *discrete* conclusions cannot be applied directly to scheduling, since a *continuum* of scheduling conclusions must be considered (e.g., schedule A_i at t_i and A_j at t_j, etc.). What is required instead is a continuum version of uncertainty reasoning, formulated in a way which efficiently expresses the variety of constraints that typically appear in these problems and which retains information about choices that affect schedule optimality. Central to this formulation is a way to combine evidence from two or more independent constraints $\{W_{i\alpha}, W_{i\beta}, \ldots\}$.

Two conditions for the combination of evidence are reasonable: the combination function for $W_{i\alpha}$ should be a continuous, monotonically increasing function of its arguments, and it should be associative, i.e. it should not matter in what order the evidence is considered. With these assumptions, the preferences $W_{i\alpha}$ together with the combination operator form an Abelian group isomorphic to the additive group of real numbers on $(-\infty, \infty)$, a result which has been independently discovered by a number of researchers (e.g., Cox 1946; Good 1960, 1968; Hájek 1985, Cheng and Kashyap 1988). Thus with no loss of generality we take the $W_{i\alpha}$ to be real-valued functions that combine simply by addition.

It is common in scheduling problems to have constraints that specify times when an activity is *not* permitted to be scheduled. These are particularly important since they allow the scheduler to eliminate blocks of time from further consideration. In terms of the preferences, these times should have highly unfavorable values, e.g. $W_{i\alpha}(t;t_{j\neq i}) = -w_0$, where $w_0 > 0$ is sufficiently large to indicate *overwhelming* evidence against scheduling activity A_i at time t. We can transform the additive $W_{i\alpha}$ into a multiplicative form $B_{i\alpha}$ by defining:

$$\begin{aligned} B_{i\alpha}(t;t_{j\neq i}) &= \exp[W_{i\alpha}(t;t_{j\neq i})] && W_{i\alpha}(t;t_{j\neq i}) > -w_0 \\ &= 0 && W_{i\alpha}(t;t_{j\neq i}) \leq -w_0 \end{aligned} \qquad \text{(EQ 1)}$$

Except where $B_{i\alpha}(t;t_{j\neq i}) = 0$, the use of the exponential function makes the *additive* combination of the weights $W_{i\alpha}$ equivalent to the *multiplicative* combination of $B_{i\alpha}$. When $B_{i\alpha}(t;t_{j\neq i}) = 0$, multiplicative combination provides precisely the desired behavior, i.e. if there is overwhelming evidence against scheduling A_i at t from *any* source, then no amount of evidence from other sources can counteract this. We have found that the multiplicative formulation is particularly convenient for representing practical constraints defined by scheduling experts. In the HST domain we have further adopted the convention that a value $W_{i\alpha} = 0 \Leftrightarrow B_{i\alpha} = 1$ represents the absence of evidence either for or against a scheduling decision. In practice, the $B_{i\alpha}$ are defined by analysis of the constraints and preferences in consultation with telescope scheduling experts.

It is computationally infeasible to work with the full N-dimensional form of the $B_{i\alpha}(t;t_{j \neq i})$ in any practical scheduling problem. The approach adopted in SPIKE consists of projecting the $B_{i\alpha}$ into functions of one time variable only:

$$S_{i\alpha}(t) = \max\{B_{i\alpha}(t;t_{j \neq i}) | t_{j \neq i}\} \qquad \text{(EQ 2)}$$

where the maximum operator ranges only over times t_j where activities A_j are permitted to be scheduled (based on the current state of the schedule, i.e. accounting both for times excluded by constraints and for times excluded by decision of the scheduler). $S_{i\alpha}(t)$ is zero only when, due solely to the constraint C_α, *no* possible choices for scheduling activity $A_{j \neq i}$ will permit A_i to be scheduled at time t; otherwise its value is the best (most preferable) value of $B_{i\alpha}$ that can be obtained by scheduling A_j at t_j, given any other possible schedule of the other activities. The former property of$S_{i\alpha}(t)$ ensures that no times are excluded prematurely unless provably in violation of a strict constraint. The latter provides an important indicator of optimal scheduling choices to the scheduling agent by always indicating the *best* that can be achieved, regardless of future scheduling decisions. (It is conceivable that a function other than maximum, e.g. some averaging function or even minimum, could be useful in some scheduling problems.) We call $S_{i\alpha}(t)$ the *suitability function* for activity A_i due to constraint C_α. The *total suitability function* for an activity $S_i(t)$ is the product of the suitability functions from each of its constraints, multiplied by a restriction operator $R_i(t)$ indicating any scheduling decisions made so far in constructing the schedule $R_i(t) \equiv 0$ for excluded times, 1 otherwise):

$$S_i(t) = R_i(t) \prod_\alpha S_{i\alpha}(t) \qquad \text{(EQ 3)}$$

$S_i(t) = 0$ if activity A_i is excluded from being scheduled at time t, either because a strict constraint would be violated or because of prior scheduling decisions. These equations implicitly determine the suitability function for an activity and are solved by an iteration procedure corresponding to the propagation of constraints.

The suitability function concept can be illustrated by a simple example: consider a preference constraint of the form:

> "Schedule A_j as soon as possible after the end of A_i, but starting no sooner than x minutes afterwards and ending no later than $x + y$ minutes afterwards."

We can represent this by choosing $B_{j\alpha}(t;t_i) = \varphi(t - t_i)$ where φ is a function indicating the judgment (objective or subjective) of how much better or worse it is to delay scheduling A_j after A_i. Given the suitability function $S_i(t)$ of A_i it is straightforward to construct

$S_{j\alpha}(t)$ due to this constraint as illustrated in Figure 2. Panel (a) shows what the $B_{j\alpha}(t;t_i)$ could look like for a plausible choice of φ. Panel (b) shows what the suitability function $S_i(t)$ might be at some stage in the scheduling: in this case there are two disjoint candidate intervals where A_i could be scheduled. The last panel (c) shows the resulting suitability $S_{j\alpha}(t)$ for task A_j.

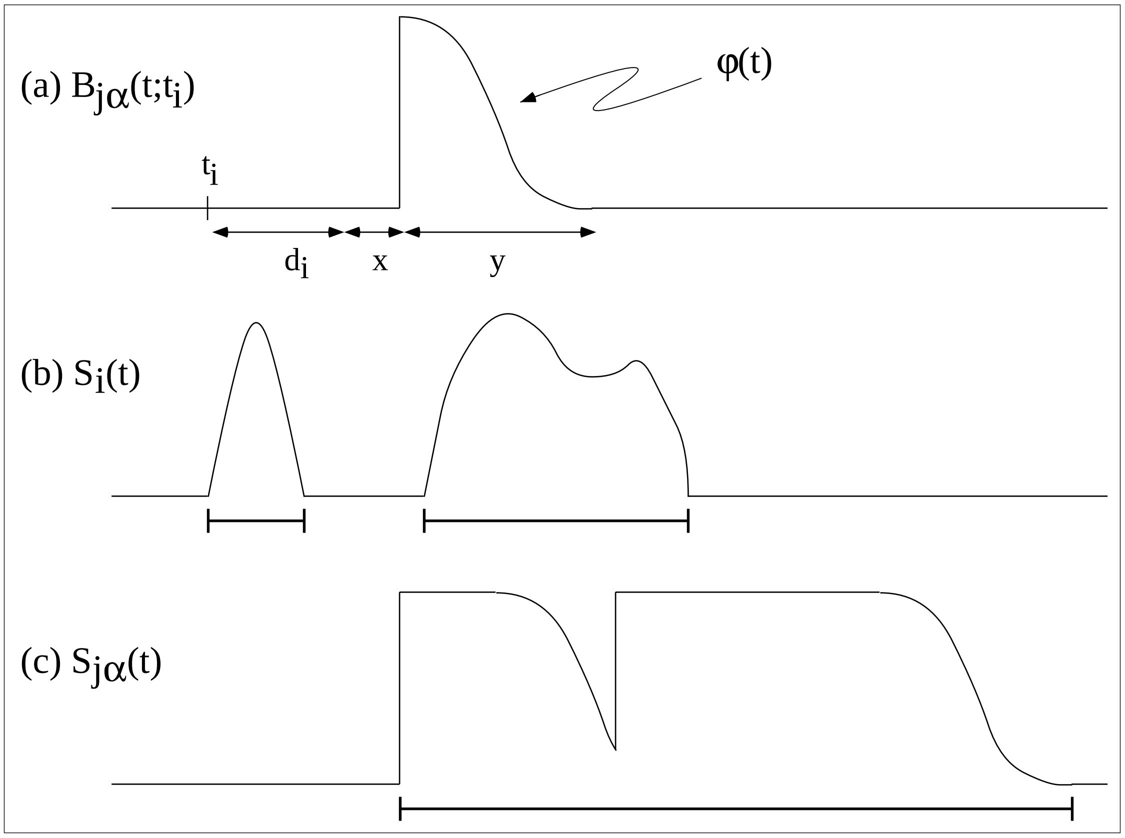


FIGURE 2. Illustration of suitability functions for the case of a binary preference constraint: (a) preference expressing that a task A_j should be scheduled as soon as possible after x minutes from the completion of another task A_i (of duration d_i) and in no case later than $x + y$ minutes; (b) hypothetical suitability of A_i at some stage in the scheduling process; (c) the resulting suitability of A_j. The intervals where each function is non-zero are indicated by bars under the time axes in (b) and (c).

Suitability functions provide a simple but effective framework for capturing metric-time scheduling constraints, both strict and preference. All of the conventional binary temporal interval relationships (before, after, during, etc.: see Allen 1983) are easily represented by appropriate suitability functions, along with a large class of far more general temporal couplings (Shapiro 1980). Like Rit’s (1986) formulation of “constrained occurrences” for binary interval constraints, suitability functions can represent and propagate disjunctions; more generally, however, they also handle constraints of higher order than binary, and can incorporate preferences as well. The combination of preferences is analogous to other similar constraint evaluation methods (e.g. Fox and Smith 1984, Smith et al. 1986), but differs in that the combination of evidence for or against a scheduling decision is required to be monotonic and associative. It is also worth noting that, while there is a resemblance between suitabilty functions and the propagated preferences of Sadeh and Fox (1988), the latter method is based on a probabilistic model of start time distributions. Such a probabilistic characterization of the *results* of scheduling as an input to the scheduling agent differs from the suitability function perspective, which maintains a distinction between the likelihood of different decisions by the scheduler and the characterization of preferences vs. time. However, this does not rule out the use of similar models that attempt to estimate resource demand and contention (e.g. Sadeh 1991): these can play a useful role as suitability components that reflect resource or capacity limits.

2.2 Consistency Methods

Consistency methods have long been known to improve search efficiency for discrete CSPs (see, e.g., Dechter 1986, Dechter and Pearl 1988, Dechter and Meiri 1989 and references therein) and have proven to be useful in SPIKE as well. Consistency techniques make explicit the information that is implicit in the constraints. We have found the following techniques to be useful in speeding scheduling search:

Node-consistency refers to the removal from consideration of domain values which cannot be part of any solution, where this determination is made based on unary constraints. In our formulation this is explicitly represented in the suitability functions.

Arc-consistency refers to removing values from the domains of variables to satisfy binary constraints. This technique is best illustrated by example: suppose A_j is constrained to follow A_i with a minimum end-to-start separation of Δt, that both activities have unit durations and are restricted to be scheduled in the interval $[t_A, t_B]$. Then the interval $[t_B - \Delta t - 1, t_B]$ is excluded for A_i, and the interval $[t_A, t_A + \Delta t + 1]$ is excluded for A_j. To introduce arc-consistency into the network, we restrict activities to fall within the overall scheduling interval $[t_A, t_B]$, and then propagate constraints (Equation 3).

Path-consistency refers to the inference of additional (binary) constraints based on those explicitly stated. Again an example makes the principle obvious. Suppose A_i must precede A_j which must in turn precede A_k: by explicitly representing the constraint "A_i precedes A_k" we can immediately represent the implication of a decision on scheduling A_i which would otherwise require a further decision about A_j. For simple precedence, the additional constraints inferred by path consistency are just those derived from the transitive closure of the precedence relationship. However, for more general binary constraints which depend on time differences only (e.g. "group $\{A_1 A_2 A_3\}$ within 24 hours"), it is possible to generalize this calculation and derive much more informative constraints (Johnston and Adorf 1992).

We have found a substantial benefit in pre-computing and storing for later access the results of node-, arc-, and path-consistency. We have also found that explicit representation of inferred constraints makes the conflict-count repair heuristics (Section 4.2) more effective. However, in other scheduling domains, the significant pre-processing computational cost must be traded against the potential speed-up during search. To help control this, the constraint propagation code used in SPIKE includes a “time-out” capability which can be used to limit the amount of computation devoted to path-consistency.

2.3 Computational Aspects

The implementation of suitability functions on digital hardware requires that continuous suitabilities be discretized, either in time, value, or both. In SPIKE we have avoided any fundamental discretization in time for two important reasons: (a) it introduces an artificial time granularity into the problem, and (b) many important constraints yield suitability functions which have long intervals of constant value and would therefore be inefficiently represented by a large number of identical values for many discrete time points. Instead, we adopted the discretization of suitability function values which allows suitability functions to be represented as *piecewise constant functions* (PCFs). These can be conveniently and compactly be expressed as a list of times and values, e.g. $(t_1\ s_1\ t_2\ s_2 \ldots t_n\ s_n)$ where the suitability has a value of s_1 from t_1 up to t_2, then a value of s_2, etc. Suitability values are not restricted to a fixed set. Arbitrary values are allowed and are only required to be constant over appropriate intervals (which can be different for different constraints). In this way, the basic constraint representation mechanism places no arbitrary restriction on the timescale or suitability value that can be represented.

The choice of PCFs has other advantages as well. They are closed under all of the common operations required for manipulation of suitability functions such as multiplication or maximum (in contrast to other representations such as piecewise linear functions). The cost of storing and combining PCFs is proportional to the number of intervals with distinct values, not to the size of the scheduling interval.

Although the PCF representation of suitabilities does not require discretization of suitability values, in practice this may be useful as small differences in suitability (e.g. a few percent) may not be significant and “collapsing” these differences in suitability can decrease the storage required for the suitability function and increase the speed of constraint propagation.

3.0 HST Scheduling as a Constraint Satisfaction Problem

A constraint satisfaction problem (CSP) consists of as set of variables, each with a domain of discrete values, and a set of constraints which limits the allowed values for each variable based on the assigned values of other variables. The problem is to assign a consistent set of values for all variables such that there are no constraint violations. To cast a scheduling problem into the form of a CSP, we identify each activity to be scheduled with a variable, and we partition the scheduling time range for each activity into intervals which are identified with the domain of the corresponding variable. A CSP as usually stated considers only strict constraints and ignores preferences: we place the further condition on the problem that the preferences should be maximized in the solution state. CSPs on discrete domains arise in a variety of applications and methods for solving them have been widely studied: for a recent survey see Kumar (1992) and references therein.

SPIKE incorporates a general "toolkit" for representing and manipulating constraint satisfaction problems, which we will briefly describe in this section. This toolkit is used in the SPIKE scheduling search algorithm, which will be discussed in Section 4.0.

3.1 The SPIKE CSP toolkit

The SPIKE CSP toolkit is implemented as a set of object classes and associated methods, the most important of which are the abstract classes for:

- **CSP**, representing the collection of variables, constraints, etc. required to represent a particular type of constraint satisfaction problem, and
- **variable**, representing variables, their assigned values, conflicts, preferences, etc.

These two classes must be specialized for each particular kind of CSP implemented.

The state information maintained by CSP and variable instances can be manipulated with an extensive library of methods. These provide capabilities which have proven extremely useful in a practical scheduling environment. Each variable instance maintains constraint conflicts and preferences data for each of its domain values, as well as the variable's current assigned value.

Some of the more important capabilities provided by the CSP toolkit include:

- **inconsistent assignments:** at any point, variables are permitted to have assigned values which are inconsistent with other assigned values. This is important for two reasons: it allows information about constraint linkages among activities to be exploited during scheduling search (Section 4.0), and it allows for search through a relaxed version of the problem. This is an especially important feature when there is no solution to the CSP as originally posed, and the best one can hope is to solve a relaxed problem where some constraints are violated (cf. Freuder 1989).
- **locked and ignored variables:** a variable can be forced to have a specified assigned value, then be ignored thereafter, or can be ignored without an assigned value. This allows the scheduler to partition the activities to schedule into an "active" set and "inactive" set — either with or without assigned values — and thus easily control the focus of the search process. Ignored variables are motivated by scheduling problems (like HST's) where there are *conditional* activities. These start out as ignored until some triggering condition is true, e.g. the commitment of some other observation, or notification from the proposer to schedule the observation.
- **removal and restoration of domain values:** Any domain value for a variable can be excluded from the problem, then restored at any time. This makes it possible to easily search for solutions that contain only high-preference values, or to dynamically exclude scheduling times for reasons external to the formal constraint specification.

- **constraint weights:** constraints can have any desired weight value, which is the amount the conflict count is incremented for each violation. The greater the weight, the more important the constraint. For HST, temporal constraints have higher weight than resource constraints.
- **constraint caching:** the toolkit provides a mechanism for caching the impacts of the current set of assigned values. This has two major benefits: it allows the system to quickly retract any current assigned value, and it allows for an "explanation" of the conflicts on any particular domain value (since the cache records the conflict source constraint as well as the relevant domain values and conflict weights). The cache mechanism can be turned on or off as appropriate, based on the trade-off between time to compute constraint violations versus the space to record them.
- **assignment history and snapshots:** the toolkit provides an assignment history mechanism, with facilities to mark the current state and back up to any marker. There are also facilities for saving the current set of assigned values in various forms, then re-applying them to the problem. The history mechanism can be turned off if desired, to improve runtime performance.
- **capacity constraints:** in addition to temporal constraints between variables, the toolkit implements a general class of capacity (or resource) constraints. The mapping of domain values to capacity "bins" is completely customizable and may be many-to-one or one-to-many.

3.2 Time sampling

As noted above, the scheduling interval for an activity is discretized as part of the translation into the CSP variable domain. It is therefore necessary to consider how to discretize the representation of time (unless there exists some natural time discretization in terms of which the constraints can be defined). As a general rule, the sampling interval must be less than the timescale for significant changes in the scheduling constraints. If this condition is satisfied, then one has to decide upon a suitable *sampling procedure* defining how to treat those strict constraints that would prevent the scheduling of an activity over some, but not all, of a given interval. The basic choice is whether to exclude the entire interval or not:

(a) if the entire interval is excluded, then there is a risk that feasible solutions may be missed;

(b) if the interval is *not* excluded, then the scheduler may find what appears to be a feasible configuration, but which turns out not to be feasible when the timing is examined in detail.

In HST scheduling we generally chose option (b) for the initial implementation and moved toward option (a) for specific constraints as experience was gained with operations. The choice must be determined for each problem type based on the characteristics of the constraints and the difficulty of dealing with the consequences. In some problems there will be a natural time unit in terms of which constraints are defined, so that no sampling error will occur. A further discussion of sampling and discretization is given in Johnston and Adorf (1992).

4.0 Scheduling Search in SPIKE

SPIKE treats schedule construction as a constrained optimization problem and uses a heuristic repair-based scheduling search technique called *multistart stochastic repair*. This technique consists of the following steps:

1. **Trial assignment:** make a trial assignment ("initial guess") of activities to times, based on heuristics to be discussed further below. Such a schedule will generally have temporal or other constraint violations, as well as resource capacity overloads;
2. **Repair:** apply heuristic repair techniques to try to eliminate constraint violations, until either a pre-established level of effort has been expended or there are no conflicts left;
3. **Deconflict:** eliminate conflicts by removing any activities with constraint violations, or by relaxing constraints, until a feasible schedule remains.

The heuristics employed in SPIKE are stochastic, so there is benefit in repeating the three steps above as often as there is time. The general strategy is to select the best of many runs, possibly trying different initial guess and repair heuristics. However, the SPIKE algorithm has desirable "anytime" characteristics (cf. Zweben et al. 1990): at any point in the processing after the initial guess has been constructed, a feasible schedule can be produced simply by removing any remaining activities with constraint violations, as described further below.

Two additional factors play a major role in SPIKE's search process:

optimization: the trial assignment and repair heuristics pay careful attention to suitability function values and attempt to optimize the total suitability of the resulting schedule, and

oversubscription: in general, it is known that more HST observations are intended to be in the pool to schedule than can actually fit into the timeline, so there can be no solution with all activities scheduled. Thus the deconflict step assumes a high degree of importance, since it defines the relaxed problem that is being solved.

4.1 Trial Assignment Heuristics

The choice of a good trial assignment can be important for repair-based methods, and to this end we have conducted extensive experiments on different combinations of variable and value selection heuristics to identify the most powerful combinations. Over a thousand combinations of heuristics were tried by making multiple runs on sample scheduling problems. Several heuristics were identified on this basis: one of the most successful selects most-constrained activities to assign first, where the number of min-conflicts times is used as the measure of degree of constraint. Min-conflict times are then assigned, with ties broken by maximum preference derived from suitability functions, or by earliest time. Several other heuristics are also employed in some settings, considering e.g. the number of temporally related activities, task priority, maximum suitability (preference), whether related tasks have assigned values, etc. The number of temporally related activities was found to be particularly effective on the 60 CSP scheduling problems defined by Sadeh (1991) and discussed further by Muscettola (1992) and Johnston and Minton (1993). It is worth noting that the SPIKE trial assignment heuristics are quite simple and are based on easy-to-calculate measures of degree of constraint and constraint connectivity. This is in contrast to the much more elaborate analyses which characterize the approaches taken e.g. by Sadeh and Fox (1988), Sadeh (1991), Sycara et al. (1991), and Muscettola (1992). Further research into the cost-effectiveness of lookahead is clearly warranted.

4.2 Repair Heuristics

The repair heuristics used by SPIKE are based on a successful neural network architecture developed for SPIKE (Johnston and Adorf 1989, 1992, Adorf and Johnston 1990) and later refined into a simple symbolic form (Minton et al. 1990, 1992; Johnston and Minton 1993) which has since superseded the neural network. The SPIKE repair heuristics make highly effective use of *conflict count* information, i.e. the number of constraint violations on scheduled activities or on potential schedule times. Min-conflicts time selection is one such repair heuristic, in which activities are moved to times when the number of conflicts is minimized. Both theoretical analysis and numerical experiments have shown that min-conflicts can be very effective in repairing reasonable trial assignments. We have found that further improvement can come from the use of a max-conflicts activity selection heuristic, which selects activities for repair which have the largest number of conflicts on their current assigned time. (This heuristic is also important when constraints have different weights: it then tends to select for repair those activities which violate the most important constraints, i.e. those with the largest weights.)

Both hill climbing and backtracking repair procedures have been tried, but hill climbing has been shown to be the most cost-effective on problems attempted to date. Typically only a relatively small number of repair steps is allowed, e.g. kN where N is the number of activities to schedule and k is usually 2 but is kept in the range 1-5. This helps deal with the problems of “cycles” which can afflict hill-climbing procedures, where the repair process repeatedly attempts to place the same set of activities at mutually inconsistent times. While cycles are sometimes observed and there has been some work done to identify and avoid them, they have turned out not to be a significant problem in practice.

4.3 Deconflict

SPIKE currently uses a rather simple technique to remove conflicting activities from an oversubscribed schedule: activities to be removed are selected based on lower priority, higher numbers of constraint conflicts, and lower preference time assignments. If there remain gaps when all conflicting activities have been deleted, then a simple best-first pass through the remaining unscheduled activities is used to fill them. This final phase of “schedule deconflicting” has been little studied and is an area which could benefit from further effort.

4.4 Schedule Quality Measures

There are several important measures of schedule quality employed, including the number of observations on the schedule, the total observing time scheduled, and the summed degree of preference of the scheduled observations. Although some applications of SPIKE use minimum makespan or related measures, these have not important for HST scheduling: the time boundaries of the schedule are essentially fixed, and the goal is to maximize the quantity and quality of the observations scheduled within them.

One particularly interesting measure plays a role when the activities to be scheduled have durations $d(t)$ that vary as a function of time: in this case the total *gap* time in the schedule serves as one component of a quality measure, since it indicates how much time could potentially be used if there were appropriate activities available. Note that, in this case, the total summed activity duration scheduled can be highly misleading — it is possible to construct a very inefficient schedule which rates highly by this measure simply by placing activities at times when they are very inefficient (i.e. $d(t) » d_{\mathrm{min}}$) but tend to fill up the schedule. So the appropriate quality measure is the non-intuitive sum of total *minimum* activity duration, plus the total gap time.

4.5 Rescheduling

SPIKE provides support for rescheduling in variety of ways. Two worth mentioning in particular are provided by the CSP toolkit (Section 3.0): *task locking* and *conflict-cause analysis*. Tasks or sets of tasks can be locked in place on the schedule, and will thereafter not be considered during search or repair (unless, of course, the user unlocks them). These tasks represent fixed points on the schedule. Conflict-cause analysis permits the user to force a task onto the schedule, then display what constraints are violated and by which other tasks. The conflicting tasks can be unassigned if desired, either individually or as a group, and returned to the pool of unscheduled tasks. This helps with the most common rescheduling case, where a specific activity (e.g. a target of opportunity) must be placed on the schedule, thereby disrupting at least some tasks which are already scheduled. A limited study of minimal-change rescheduling has been conducted (Sponsler and Johnston 1990), but much more work remains to be done in this area. Most of the other SPIKE support for rescheduling makes use of facilities provided in the user interface which allow the scheduler to freely manipulate the timeline (Section 5.3).

5.0 SPIKE and HST Science Scheduling

The framework described in the preceding sections has been integrated into an observatory science planning and scheduling system for the Hubble Space Telescope. SPIKE has been used since 1988, first in pre-launch readiness tests and then in science operations since HST launch in 1990. Figure 3 shows the high-level scheduling flow for the HST ground system which is described in this section and is covered in more detail in Miller (1989), Adorf (1990), and Miller and Johnston (1991).

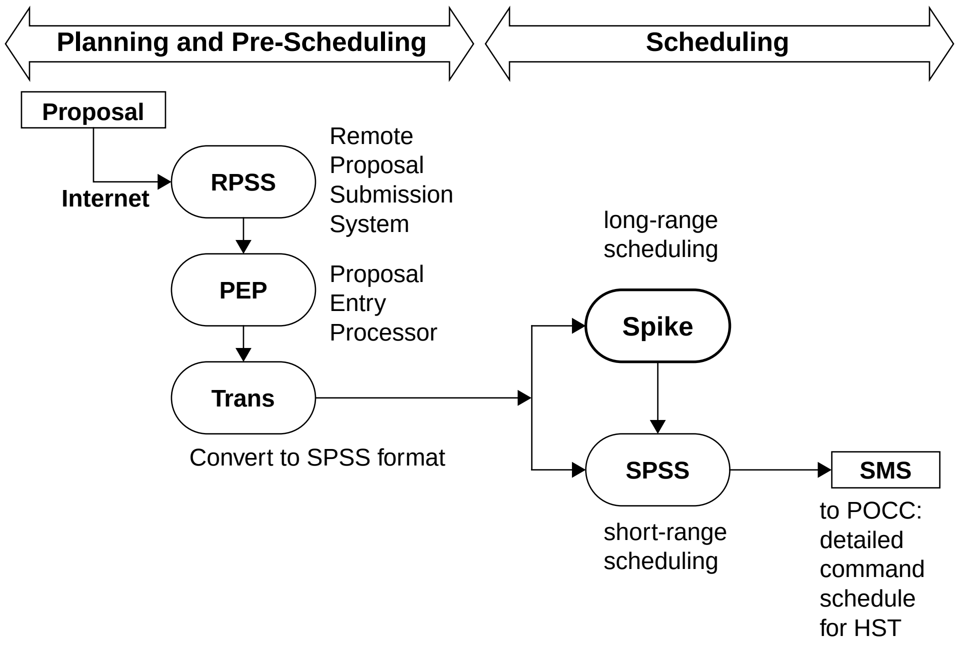


FIGURE 3. The processing flow for HST scheduling. Proposals are received electronically over the Internet and processed through a proposal database. The “Transformation” system converts the astronomer’s observing plan into a set of tasks to schedule. SPIKE does the long-range scheduling, then passes off 1-week segments to SPSS for short-term scheduling and instrument command request generation. The main output product, the Science Mission Specification (SMS) is a detailed time-tagged command list for the HST’s onboard computers.

- **Proposal Preparation:** An astronomer who plans to use HST must create an observing proposal which specifies the observations to be made. This proposal is the primary input to the planning and scheduling process. The Remote Proposal Submission System (RPSS) and Proposal Entry Processor (PEP) handle observing proposals, including electronic submission by astronomers (Section 5.1).
- **Planning:** The Transformation (Trans) expert system converts the proposal from a high-level specification into detailed task descriptions for scheduling (Section 5.2).
- **Long-term Scheduling:** Since the HST scheduling problem covers such a wide range of timescales and tens of thousands of tasks, a two-tiered approach to scheduling was adopted. A long-term plan (Section 5.3) spans approximately one year and allocates tasks (as defined by Transformation) to specific weeks or parts of a week. From the long-term plan, week-long segments are extracted for short-term scheduling. Long-range scheduling is done with the SPIKE system, which was developed at the STScI.
- **Short-term Scheduling:** Short-term scheduling (Section 5.4) with the Science Planning and Scheduling System (SPSS) performs the final sequencing of groups of observations within a week, generates the detailed command list, and transmits the results as the Science Mission Specification (SMS) to the HST Payload Operations Control Center. SPSS was originally developed by TRW and now maintained by the STScI.

Section 5.5 considers some of the implementation issues that were faced during development of the integrated HST ground system.

5.1 Proposal Preparation

The selection of successful proposers for HST is based on a relatively simple ("Phase I") description of the scientific intent of the program and the observatory resources required to accomplish it. Once the observations are approved by a peer review process, each proposer must prepare a detailed ("Phase II") definition of exactly what exposures must be taken. Since the HST and its ground system are very complex, and the astronomer does not have real-time control over the telescope, essentially all observations are scheduled and taken by the staff of the Space Telescope Science Institute (STScI) in Baltimore.

The Phase II proposal contains information on the astronomical objects, individual exposures, instrument parameters, and the relationships (i.e. constraints) among exposures: Table 1 lists the kinds of constraints that proposers may specify, in the syntax that they actually use. Proposals are submitted electronically in an ASCII file, through the Remote Proposal Submission System (RPSS). The RPSS software validates the contents of a proposal file and can detect a wide range of problems, including typographical errors (e.g. a misspelled filter name), values out of range (e.g. a target declination exceeding 90°), and missing or inconsistent information (e.g. an exposure referencing an undefined target). A dedicated RPSS computer is available to the astronomical community over the Internet and the Space Physics Astrophysics Network (SPAN). The RPSS software has been distributed to approximately 100 astronomical institutions around the world, and is run by most proposers at their home institutions before they send their proposals to STScI.

STScI designed RPSS and the HST proposal language with the following goals in mind:

- Oriented towards the astronomical community — easy to understand, and concise and logical in the amount and sequence of data requested,
- Able to accommodate both simple and sophisticated observations from novice or experienced HST users, and
- Formulated in declarative terms, i.e. the astronomer can specify what data should be collected without becoming needlessly encumbered by instrument, telescope and ground system particulars.

RPSS was the first system of its kind for a major scientific installation — it has been in use since February 1986. The ability to locally validate and electronically submit a proposal is an extremely valuable tool, both for proposers and for STScI. Proposers can detect and correct a large class of errors and are assured that typographical errors are not introduced by data entry personnel. The STScI can process proposals more rapidly and avoid the costs of manual data entry. A further discussion of the STScI proposal handling software can be found in Jackson et al. (1988) and Adorf (1990).

TABLE 1. The syntax of constraint specifications used by HST observers to define scheduling constraints on their exposures. Any number of the these “special scheduling requirements” may be applied to exposure “lines”, referring to their origin on paper forms. Keywords may be abbreviated to the form shown in uppercase letters (e.g. “ACQUISITION” can be abbreviated as “ACQ”). A *line-list* is a list of exposure line numbers, e.g. “1-5,10,12,15-20”. Square brackets indicate optional syntactic elements, and the “/*line-list*” shorthand applies the specified constraint to all listed lines. Most of the phases provide an English-like description of the constraint, all of which must be properly handled by the HST scheduling software.

HST Observing Proposals: Scheduling Constraint Syntax
EARLY ACQuisition FOR *line-list*
ONBOARD ACQuisition FOR *line-list*
INTeractive ACQuisition FOR *line-list*
GUIDing TOLerance *angle* [/*line-list*]
ORIENTation *angle* +/- *angle* [/*line-list*]
ORIENTation *angle* +/- *angle* FROM *line* [/*line-list*]
ORIENTation *angle* +/- *angle* FROM NOMINAL [/*line-list*]
POSition TARGet *x-val*, *y-val* [/*line-list*]
SAME ORIENTation FOR *line-list* AS *line*
SAME POSition FOR *line-list* AS *line*
CALIBration FOR *line-list* [NO SLEW]
SPATIAL SCAN [/*line-list*]
TARGet OF OPPortunity [/*line-list*]
CRITical OBServation [/*line-list*]
RT ANALYSIS [FOR *line-list*]
REQuires UPLINK [/*line-list*]
AFTER *date*|*line* [BY *time* [+/- *range*]]
AT *date* +/- *range*
BEFORE *date*|*line* [BY *time* [+/- *range*]]
DARK TIME [/*line-list*]
DECision TIME *time*
GROUP *line-list* WITHIN *time*
GROUP *line-list* NO GAP
GROUP *line-list* NON-INTerruptible
NON-INTerruptible [/*line-list*]
PERIOD *time* +/- *error*
PHASE *phase* +/- *range* [OF REF *line*]
REQuires DATA FROM *line-list* [/*line-list*]
REQuires UPDATE [/*line-list*]
SEQuential *line-list*
SEQuential *line-list* NO GAP
SEQuential *line-list* WITHIN *time*
SEQuential *line-list* NON-INT
ZERO-PHASE *date* +/- *error*
REPEAT *line-list* EVERY *time* +/- *range* for *number* MORE TIMES
CONDitional [ON *line-list*] IF *condition-text* [/*line-list*]
SELECT *number* OF *line-list* OR *line-list* ...

5.2 Transformation

Given the high-level, astronomically-oriented description of the observations found in the Phase II proposal, the next step is essentially one of *planning*. It is the role of the Transformation system to convert the declarative proposal from the astronomer into a set of aggregated exposures and constraints called *scheduling units* (SUs), with all of the details required to enable the execution of the exposures on the spacecraft.

Transformation performs several planning tasks, including: determining the order to execute observations (when not explicitly specified by the proposer), breaking exposures into pieces to better match target visibility conditions, grouping observations to minimize overhead operations, choosing specific implementation scenarios, supplying values of instrumental settings which were defaulted by the proposer. Transformation also detects certain errors which may be present in the proposal including: conflicting timing requirements among exposures, loops in precedence constraints, and inconsistencies in instrument parameter settings. Transformation makes use of suitability function framework (Section 2.1) and the SPIKE temporal constraint mechanism to collect and propagate temporal constraints and to achieve path consistency (Section 2.2).

The input to Transformation is a file generated from the Proposal Entry Processor (PEP) database which is essentially a parsed version of the astronomer's Phase II proposal. Transformation produces output files which specify the structure of the scheduling units and the nature of any constraints that act on them. These files then become the inputs to SPIKE and SPSS.

Transformation was initially conceived and implemented as a rulebased expert system implemented in OPS5 (Rosenthal et al. 1986) but was re-implemented in Common Lisp when the complexity of the rulebased system grew too great (Gerb 1991a). It remains an expert system in that it models a large body of expertise developed by the astronomers who operate the HST scheduling systems.

5.3 Long-Term Scheduling

Long-term scheduling begins with a set of observing proposals from Transformation, specified as a set of scheduling units (SUs) and their constraints. An initial pre-processing step calculates absolute time-dependent constraints related to orbit-by-orbit target visibility, and the implications of special orientation requirements; the results of these compute-intensive calculations are cached in disk files.

Using SPIKE, SUs can be committed to time intervals either manually or with the automatic (CSP) scheduler (Section 4.0). A graphical user interface (e.g. Figure 4) is available to view and manipulate the schedule, or to run the automatic scheduler. Scheduling decisions (when final) are recorded in a database, then reported to files and transmitted to SPSS. Once SPSS has completed short-term sequencing, SPIKE software analyzes the calendar to determine what observations were scheduled and what factors could affect the future schedule (e.g. timing, special orientations, or contingent SUs). SPIKE also analyzes a report from the data archive to verify that the data was received and processed. Observations which are not scheduled by SPSS, which are lost in transmission, or which are marked as poor quality, are considered candidates for rescheduling.

SPIKE provides a number of tools to support the scheduling process, including high-resolution PostScript plots of observing constraints and interactive X-window tools for analysis of complex spacecraft orientation constraints and for viewing the component exposures and constraints from individual proposals. The latter can be analyzed in detail, which is particularly important when it is necessary to examine the effects of individual constraints on potential scheduling decisions. This facility has also proven to be very useful for uncovering and fixing problems with proposals.

SPIKE's central position in the HST scheduling process has led us to develop an integrated tracking system called ASSIST to monitor and report the status of all observations in the scheduling pipeline. Prior to the development of ASSIST, users of the various systems (PEP, Transformation, SPSS, etc.) each maintained separate tracking systems. Since proposals consist of many observations which are executed at different times, finding the status of a proposal required substantial work. ASSIST provides a central repository for data from the various stages of scheduling, including proposal preparation, long-range scheduling, short-range scheduling, and archiving.

Certain combinations of HST instruments can operate in tandem, and for some astronomical targets such "parallel" observations can yield very interesting data. SPIKE's Parallel Observation Matching System (POMS) analyzes the weekly schedule and finds appropriate matches from a pool of parallel proposals. POMS is an expert system which includes a sophisticated knowledge base and matching strategies for identifying and ranking the quality of matches (Lucks 1992). Multiple ranked candidate parallel matches are sent to SPSS, along with each weekly schedule.

5.4 Short-term Scheduling

Short-term scheduling operates within a week at a time, working from a list of SUs from the long-term plan and generating a week-long sequence of activities called a *calendar*. After the activity sequence is defined, high-level spacecraft instructions are attached to the calendar activities. The output of the process is a Science Mission Specification (SMS) and can be thought of as the "assembly language" which controls the HST. The STScI delivers the SMS to the Payload Operations Control Center (POCC) at NASA's Goddard Spaceflight Center, where it is checked for errors and constraint violations which would affect the health or safety of the HST and its instruments. Based on the SMS, the POCC prepares the binary command loads for the two onboard computers which control the observatory. The POCC also takes requests for Tracking and Data Relay Satellite (TDRS) links from the SMS and passes them on to the TDRS Network Control Center. Some requests will not be granted due to higher priority users (e.g. Shuttle or other satellites). The POCC notifies the STScI of this and the timeline is modified, usually by making use of one of the onboard tape recorders to hold data for later playback, but occasionally by rescheduling the observation.

Execution of the observations is monitored by both the STScI and the POCC (by monitoring real-time transmissions or by playback of recorded science and engineering telemetry). The STScI analyzes the data to ensure that scheduled observations were completed successfully. It then calibrates the data and delivers it to the proposer for scientific analysis. Since HST observations are an important astronomical resource, an archive of all HST data is maintained. The proposer is normally granted exclusive access to the data for a proprietary period (usually 1 year), after which the data becomes available to the scientific community at large.

5.5 Implementation

SPIKE is an operational application of artificial intelligence technology and in this section we consider some of the implementation issues. The development of SPIKE started in early 1987 using Texas Instruments Explorer Lisp machines. The SPIKE graphical user interface was implemented in KEE CommonWindows (Intellicorp), but the remainder of the system used only Common Lisp and the Flavors object system. At HST launch, STScI had a complement of 8 microExplorers and Explorers used for SPIKE operation, development and testing.

Since 1987 there has been a great deal of evolution in Lisp hardware and software. We have continued to modify SPIKE to keep pace with these changes. All of the Flavors code has been converted to the Common Lisp Object System (CLOS). Between late 1990 and mid-1992 we transitioned from Explorers to Sun SparcStations as the primary operations and development platform: there are currently a total of 22 SparcStation 2s used for Transformation and SPIKE. The Lisp used on the SparcStations is Allegro Common Lisp from Franz, Inc., which supports a version of CommonWindows based on X-windows. Thus the user interface continued to operate on the SparcStations as it did on the Explorers (and allowed us to operate for some time with a mix of Explorers and SparcStations). After substantial investigation of alternative window systems, we recently reimplemented the user interface tools using the Common Lisp Interface Manager (CLIM). Updating SPIKE for new Lisp language features has not been difficult, and there are currently no plans to convert any of the system to C or C++.

A common feature of PEP, Transformation and SPIKE development was that each system had to be developed in a short time (about 6 months for the initial system, with substantial extensions continuing over several years) and with a small staff (2-3 people initially). It was also impossible to specify in advance a complete set of requirements for these systems since many important factors were unknown. These considerations led to the use of a rapid prototyping software development methodology instead of a more classical “waterfall” approach (requirements definition, design, implementation and test). A tool-oriented approach was also encouraged, i.e. the development of general software routines which could be used for other applications later in development.

The most significant advantage of rapid prototyping to the HST was that it allowed PEP, Transformation and SPIKE to be implemented in time to support testing and operations — in an environment with changing requirements there is no choice but to use an adaptive software development methodology. Perhaps the most serious complication of this approach is that once the prototype is used operationally, it becomes increasingly difficult to make large changes to it. (In an ideal rapid prototyping situation, a prototype can be discarded after evaluation). Once operational, it is necessary to ensure that each version of the system is upwardly compatible with the previous version. A corollary to this is that pressure on the users to do operational work can prevent them from further participation in the software development process, e.g. critiquing initial requirements and evaluating prototype software. This can lead to a divergence between the needs of the users and the products of the developers, which must be carefully guarded against. We have found two techniques are useful to keep developers in touch with the needs of users. One is for developers to perform *full-scale* end-to-end tests with *real* data to uncover problems and bottlenecks. (This is in advance of any testing which may be performed by an test group affiliated with a software configuration management effort.) The other is to allow developers to apprentice in the user group (typically for a month or so) in order to gain firsthand knowledge of the operational environment and requirements.

The incorporation of realistic test data proved to be quite important. Due to delays in the launch of HST, we had several hundred observing proposals which were constantly used to test prototype systems and to make development decisions. Had such extensive data not been available, the creation of a substantial body of simulated test data would have been required.

It is interesting to note that some of the most useful software tools were developed as quick-response reactions: a user would informally ask for a tool to handle some problem which was previously unrecognized or thought to be of low priority, and one of the software developers would then provide the tool in a very short time. For example, a number of functions used to fine-tune the long-range plan were developed in this way. One such function identified for a particular week candidate activities scheduled in later weeks which could be moved earlier to compensate for activities which had to be removed from the schedule at the last minute. Also, the ability to store and restore scheduling commitments in files was first implemented informally and was subsequently used to track scheduling decisions until SPIKE's ASSIST database tracking system was completed. A number of useful graphical displays and plots were also developed in this manner, all of which highlights the importance of good communications between developers and users, and a certain flexibility in the development schedule to permit timely response to user requests.

While following a non-classical development methodology, we nonetheless paid careful attention to such classic risk management factors as configuration control and testing. Developers use code management tools (i.e. Unix rcs) to prevent uncoordinated changes. Building, testing and installing the software is automated with software tools to the greatest extent possible to help detect problems before delivery to the users and to minimize the chance for human errors — indeed, Transformation even incorporates a separate expert system to perform its own testing (Gerb 1991b). Procedures and tools to deliver software changes on a very short timescale (days to hours) are also essential.

6.0 Operational Experience

HST science operations is divided into "cycles" in which proposals are solicited from the astronomical community, selected, scheduled and executed. In the long term, cycles will consist of about 1 year of HST observing. Early HST operations consisted of two special phases: "Orbital Verification" (OV), which assessed the basic capabilities of the telescope and instruments, and "Cycle 0" observations, which contain a mix of Science Verification (SV) and Guaranteed Time Observer (GTO) observations. OV ended in November 1990, and Cycle 0 ended in June 1991. The Cycle 1 observing era mixed GTO observations with the "General Observer" (GO) proposals from the astronomical community at large. Cycle 1 ended in July 1992, and is being followed by the Cycle 2 observations, etc.

The SPIKE system was first used to support HST scheduling for Cycle 0. The timeline for SV observations was established by NASA. The STScI used SPIKE to verify this timeline and to schedule GTO observations during weeks when time was available. Scheduling of these proposals in Cycle 0 used SPIKE in an interactive mode: planners would display individual proposals on timeline displays and choose times of high suitability for observations. Various automatic scheduling tools were sometimes used in conjunction with making manual commitments. Scheduling of early Cycle 1 proposals has also been largely interactive with little use of the high-level, automated schedulers. The main value of SPIKE in this mode was the identification of problems with proposals and the assignment of observations to feasible, but not necessarily optimal, weeks.

When used on actual GTO and GO proposals, Transformation and SPIKE reported large numbers of diagnostic messages. Initially, many problems were due to an incomplete understanding of how to best present complex observations to SPSS. The frequency of problems allowed us to effectively prioritize work in determining requirements and implementing the code. A significant number of problems uncovered by SPIKE and Transformation were due to an inadvertent specification by the proposer of inconsistent requirements in the proposal. Although the PEP system performs *syntactic* checking on proposal information, Transformation and SPIKE are the first systems that can detect problems related to planning and scheduling. (In particular, accurate instrument overhead times and orbital viewing conditions are calculated by Transformation and these can reveal problems with the proposal.) We are currently investigating how to incorporate such checks in PEP and RPSS. Not only will this provide proposers with immediate feedback on certain classes of problems, but it will also reduce the delays in the scheduling process due to late proposal modifications.

We originally anticipated that long term schedules covering 6-12 months duration would be maintained beginning with Cycle 0. However true long-term planning began late in Cycle 1 with schedules of approximately 3 months duration and year-long schedules were first generated operationally beginning with Cycle 2. This was not due to any inherent limitation in the software (test schedules of a year duration had been generated well before launch). It was primarily due to a much larger than anticipated rate of change of proposals. Prior to launch, $\leq$ 10% of the proposals were expected to change after submission, whereas the actual rate is nearly 100%. Many proposals have been revised several times. A substantial portion of this can be attributed to the spherical aberration of the HST mirror and other unexpected behaviors of the instruments and spacecraft. Another factor is that the HST and major elements of the ground system are designed for fully pre-planned observations with little capability to inject changes late in the scheduling process. A change to a proposal can therefore often require a repeat of the entire scheduling process, wasting much if not all of the earlier work. Tracking multiple revisions of proposals and their scheduling and execution status also requires substantial effort. Our recommendation to developers of future systems with requirements similar to HST would be to *build in the expectation of change* from the outset and to carefully examine factors in the design which are sensitive to a high rate of change. We realize that such flexibility will increase the initial cost of a system, but it can significantly reduce the lifecycle maintenance and operational costs. A further discussion of the HST experience can be found in Miller and Johnston (1991).

7.0 Application of SPIKE to Other Astronomical Scheduling Problems

SPIKE has been adapted to schedule a variety of astronomical scheduling problems (see Table 2). Of these, two are in or near flight operations (in addition to HST), while several others are in the prototype or planning phases. The experience of customizing SPIKE for other types of problems has been actively sought during SPIKE development: each case provided feedback on the approach, and led to improvements from one version to the next.

The adaptation of SPIKE for these problems demonstrates the flexibility of the SPIKE scheduling framework. As indicated above, SPIKE was designed so that new tasks and constraints can be defined without changing the basic framework. For ASTRO-D (Isobe et al. 1993) and XTE (Morgan 1992), SPIKE is operated in a hierarchical manner, with long-term scheduling first allocating observations to weeks much as they are for the HST problem (and with similar types of long-term constraints and preferences). Then each week is scheduled in detail, subject to the detailed minute-by-minute constraints of low earth orbit operation. The major changes required to implement short-term scheduling were:

1. A new type of task that can have variable duration depending on when it is scheduled, and which can be interrupted and resumed when targets are occulted by the Earth or the satellite is in the radiation belt (i.e. task preemption).
2. New classes of short-term scheduling constraints which more precisely model target occultation, star tracker occultation, ground station passes, entry into high radiation regions, maneuver and setup times between targets, etc.
3. An interface between different hierarchical levels, by which a long-term schedule constrains times for short-term scheduling and conversely.
4. A post-processor which examines short-term schedules for opportunities to extend task durations and thus utilize any remaining small gaps in the schedule to increase efficiency.

All of the general constraint combination and propagation mechanisms (Section 2.0), and the CSP toolkit (Section 3.0) and scheduling search techniques (Section 4.0), apply

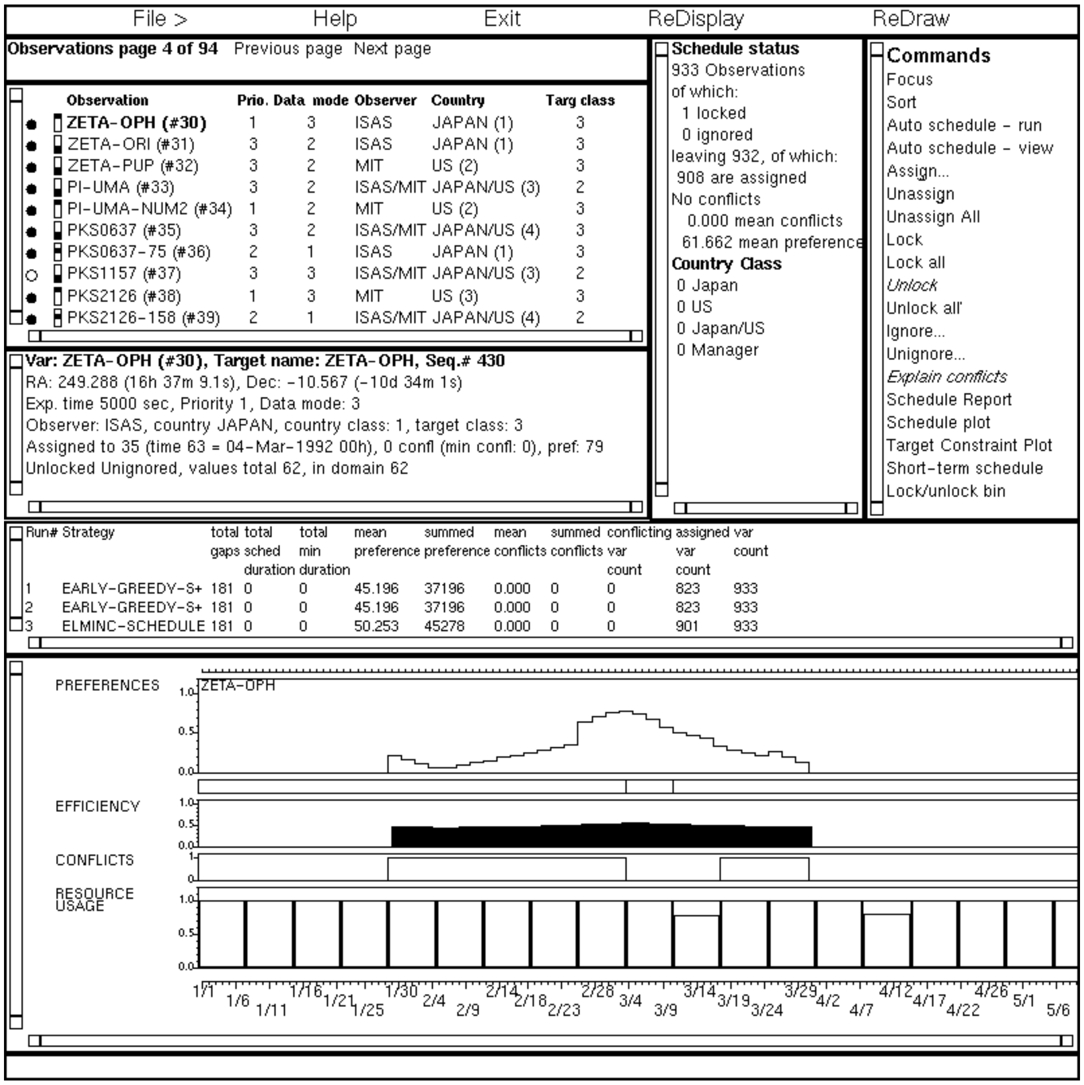
TABLE 2. Adaptations of SPIKE to various astronomical scheduling problems

Mission	Status, scheduling mode, and location
HST	Hubble Space Telescope. Spike operational since Oct 89, HST launch Apr 90. Used for HST long-term scheduling at Space Telescope Science Institute, Baltimore.
EUVE	Extreme Ultraviolet Explorer. Spike operational since Apr 91, EUVE launch in Jun 92. Used for one-year scheduling of pointed observations. Run by Center for Extreme Ultraviolet Astrophysics, Univ. Calif., Berkeley.
ASTRO-D	Operational since Nov 92, flight operations will begin following launch in Feb 93. Spike will be used for long-term and short-term scheduling. Joint Japan/US X-ray telescope mission run from the Institute of Space and Astronautical Sciences in Japan (Isobe et al. 1993)
XTE	X-ray Timing Explorer. Planned for use following launch in 1994. Spike would schedule both long-term and short-term. XTE will be run from the GSFC XTE Science Operations Center (Morgan 1992).
AXAF	Advanced X-Ray Astronomy Facility. Prototype developed 1990 as part of successful science operations center proposal. Under consideration for both science and mission scheduling.
ROSAT	Roentgen Satellite. Prototype developed 1992 for feasibility evaluation for operational X-ray satellite. Dual long-term/short-term scheduling mode. German Space Operations Center, Munich.
IUE	International Ultraviolet Explorer. Prototype developed 1988 for evaluation of Spike framework. Scheduling mode was 6-months of European half-shifts, optimized for target coverage (Johnston 1988b).
Ground-based	Prototypes developed in 1988–1992 to demonstrate feasibility. Dual mode: long-term scheduling for night allocation, short-term scheduling for exposure scheduling within a night. Telescopes included ESO (European Southern Observatory) and CFHT (Canada-France-Hawaii Telescope), as well as hypothetical Automatic Photometric Telescope (APT). Feasibility of coordinated ground- and space-based scheduling has also been demonstrated (Johnston 1988c).

directly to both long-term and short-term scheduling. Figure 4 shows the SPIKE CLIM user interface displaying an ASTRO-D long-term schedule. Figure 5 shows a portion of the high-resolution PostScript plot output for a SPIKE short-term schedule for ASTRO-D. Only one day of a 7-day schedule is shown. Note that several observations are broken to fit around earth blockages or radiation belt passages and so are taken in multiple segments.

Most of the effort required to apply SPIKE to the new problems was limited to the specific domain modelling necessary, which typically involves computation related to the geometry of the satellite, Sun, target, and Earth. These problems can be expected to differ from one satellite to another, and it is not surprising that different models are required. Some of the modelling includes state constraints, although SPIKE does not perform explicit planning (cf. Muscettola et al. 1992).

EUVE is unusual in that it makes long (2-3 day) observations, in contrast to HST and ASTRO-D which typically make numerous short (15-40 minute) observations. As a consequence, EUVE is schedulable over year-long intervals without breaking the schedule into hierarchical levels. One of the more interesting results from a comparison of search algorithms for scheduling EUVE was that the SPIKE repair-based methods gained an extra *20*



Observations page 4 of 94 Previous page Next page

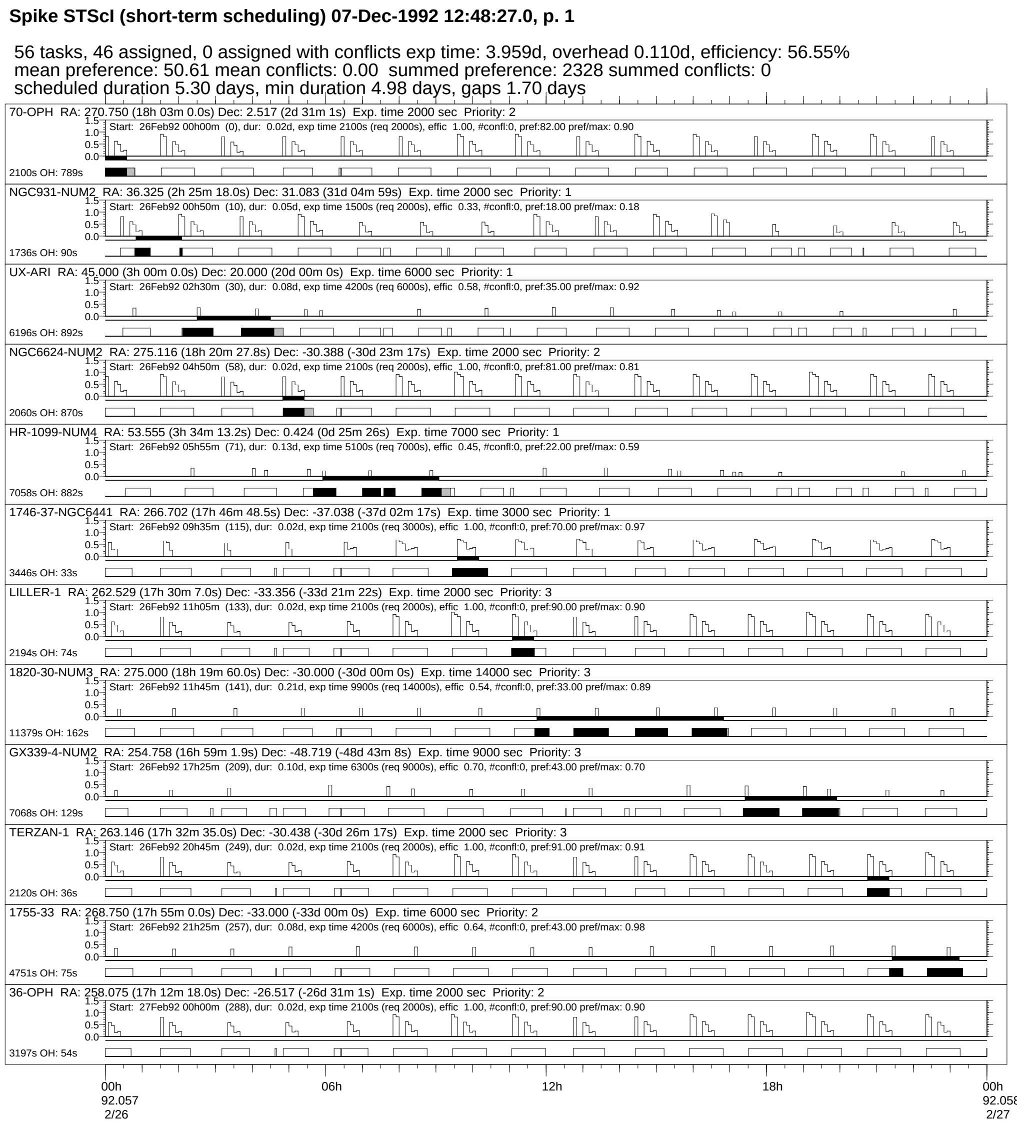
	Observation	Prio.	Data mode	Observer	Country	Targ class
●	**ZETA-OPH (#30)**	1	3	ISAS	JAPAN (1)	3
●	ZETA-ORI (#31)	3	2	ISAS	JAPAN (1)	3
●	ZETA-PUP (#32)	3	2	MIT	US (2)	3
●	PI-UMA (#33)	3	2	ISAS/MIT	JAPAN/US (3)	2
●	PI-UMA-NUM2 (#34)	1	2	MIT	US (2)	3
●	PKS0637 (#35)	3	2	ISAS/MIT	JAPAN/US (4)	3
●	PKS0637-75 (#36)	2	1	ISAS	JAPAN (1)	3
○	PKS1157 (#37)	3	3	ISAS/MIT	JAPAN/US (3)	2
●	PKS2126 (#38)	1	3	MIT	US (3)	3
●	PKS2126-158 (#39)	2	1	ISAS/MIT	JAPAN/US (4)	2

Schedule status
933 Observations
of which:
1 locked
0 ignored
leaving 932, of which:
908 are assigned
No conflicts
0.000 mean conflicts
61.662 mean preference
Country Class
0 Japan
0 US
0 Japan/US
0 Manager

Var: ZETA-OPH (#30), Target name: ZETA-OPH, Seq.# 430
RA: 249.288 (16h 37m 9.1s), Dec: -10.567 (-10d 34m 1s)
Exp. time 5000 sec, Priority 1, Data mode: 3
Observer: ISAS, country JAPAN, country class: 1, target class: 3
Assigned to 35 (time 63 = 04-Mar-1992 00h), 0 confl (min confl: 0), pref: 79
Unlocked Unignored, values total 62, in domain 62

Run#	Strategy	total gaps	total sched duration	total min duration	mean preference	summed preference	mean conflicts	summed conflicts	conflicting var count	assigned var count	var count
1	EARLY-GREEDY-S+	181	0	0	45.196	37196	0.000	0	0	823	933
2	EARLY-GREEDY-S+	181	0	0	45.196	37196	0.000	0	0	823	933
3	ELMINC-SCHEDULE	181	0	0	50.253	45278	0.000	0	0	901	933

FIGURE 4. ASTRO-D long-term schedule generated with Spike: a view of the CLIM user interface on a six month schedule. The different panes provide visibility into the list of available observations (upper left), frequently-used commands (upper right), and provide a graphical view of the a single observation, including preferences (i.e. suitability function values), current assigned value, observing efficiency, constraint conflicts, and total resource utilization (lower pane), all displayed here over about a 5-month period. Many of the objects on the screen are CLIM “presentations” and are thus mousable by the user.



```
Spike STScI (short-term scheduling) 07-Dec-1992 12:48:27.0, p. 1

 56 tasks, 46 assigned, 0 assigned with conflicts exp time: 3.959d, overhead 0.110d, efficiency: 56.55%
 mean preference: 50.61 mean conflicts: 0.00  summed preference: 2328 summed conflicts: 0
 scheduled duration 5.30 days, min duration 4.98 days, gaps 1.70 days

70-OPH  RA: 270.750 (18h 03m 0.0s) Dec: 2.517 (2d 31m 1s)  Exp. time 2000 sec  Priority: 2
  Start:  26Feb92 00h00m  (0), dur:  0.02d, exp time 2100s (req 2000s), effic  1.00, #confl:0, pref:82.00 pref/max: 0.90
2100s OH: 789s

NGC931-NUM2  RA: 36.325 (2h 25m 18.0s) Dec: 31.083 (31d 04m 59s)  Exp. time 2000 sec  Priority: 1
  Start:  26Feb92 00h50m  (10), dur:  0.05d, exp time 1500s (req 2000s), effic  0.33, #confl:0, pref:18.00 pref/max: 0.18
1736s OH: 90s

UX-ARI  RA: 45.000 (3h 00m 0.0s) Dec: 20.000 (20d 00m 0s)  Exp. time 6000 sec  Priority: 1
  Start:  26Feb92 02h30m  (30), dur:  0.08d, exp time 4200s (req 6000s), effic  0.58, #confl:0, pref:35.00 pref/max: 0.92
6196s OH: 892s

NGC6624-NUM2  RA: 275.116 (18h 20m 27.8s) Dec: -30.388 (-30d 23m 17s)  Exp. time 2000 sec  Priority: 2
  Start:  26Feb92 04h50m  (58), dur:  0.02d, exp time 2100s (req 2000s), effic  1.00, #confl:0, pref:81.00 pref/max: 0.81
2060s OH: 870s

HR-1099-NUM4  RA: 53.555 (3h 34m 13.2s) Dec: 0.424 (0d 25m 26s)  Exp. time 7000 sec  Priority: 1
  Start:  26Feb92 05h55m  (71), dur:  0.13d, exp time 5100s (req 7000s), effic  0.45, #confl:0, pref:22.00 pref/max: 0.59
7058s OH: 882s

1746-37-NGC6441  RA: 266.702 (17h 46m 48.5s) Dec: -37.038 (-37d 02m 17s)  Exp. time 3000 sec  Priority: 1
  Start:  26Feb92 09h35m  (115), dur:  0.02d, exp time 2100s (req 3000s), effic  1.00, #confl:0, pref:70.00 pref/max: 0.97
3446s OH: 33s

LILLER-1  RA: 262.529 (17h 30m 7.0s) Dec: -33.356 (-33d 21m 22s)  Exp. time 2000 sec  Priority: 3
  Start:  26Feb92 11h05m  (133), dur:  0.02d, exp time 2100s (req 2000s), effic  1.00, #confl:0, pref:90.00 pref/max: 0.90
2194s OH: 74s

1820-30-NUM3  RA: 275.000 (18h 19m 60.0s) Dec: -30.000 (-30d 00m 0s)  Exp. time 14000 sec  Priority: 3
  Start:  26Feb92 11h45m  (141), dur:  0.21d, exp time 9900s (req 14000s), effic  0.54, #confl:0, pref:33.00 pref/max: 0.89
11379s OH: 162s

GX339-4-NUM2  RA: 254.758 (16h 59m 1.9s) Dec: -48.719 (-48d 43m 8s)  Exp. time 9000 sec  Priority: 3
  Start:  26Feb92 17h25m  (209), dur:  0.10d, exp time 6300s (req 9000s), effic  0.70, #confl:0, pref:43.00 pref/max: 0.70
7068s OH: 129s

TERZAN-1  RA: 263.146 (17h 32m 35.0s) Dec: -30.438 (-30d 26m 17s)  Exp. time 2000 sec  Priority: 3
  Start:  26Feb92 20h45m  (249), dur:  0.02d, exp time 2100s (req 2000s), effic  1.00, #confl:0, pref:91.00 pref/max: 0.91
2120s OH: 36s

1755-33  RA: 268.750 (17h 55m 0.0s) Dec: -33.000 (-33d 00m 0s)  Exp. time 6000 sec  Priority: 2
  Start:  26Feb92 21h25m  (257), dur:  0.08d, exp time 4200s (req 6000s), effic  0.64, #confl:0, pref:43.00 pref/max: 0.98
4751s OH: 75s

36-OPH  RA: 258.075 (17h 12m 18.0s) Dec: -26.517 (-26d 31m 1s)  Exp. time 2000 sec  Priority: 2
  Start:  27Feb92 00h00m  (288), dur:  0.02d, exp time 2100s (req 2000s), effic  1.00, #confl:0, pref:90.00 pref/max: 0.90
3197s OH: 54s
```

FIGURE 5. An example of Spike output on short-term scheduling of astronomical observations. Shown is a 24-hour portion of a 7-day schedule. The start-time suitability for each exposure is plotted as the upper graph, with interruptions due to target blockage by the earth and by satellite passage through high-radiation regions. The available exposure intervals are shown below as open bars, which are filled in to indicate the actual scheduled times. Some of the observations can be fit within one orbit; others must be interrupted and thus span several orbits.

days of observing time in a year, when compared to the best incremental scheduling approach.

8.0 Summary

The SPIKE scheduling system has supported NASA's Hubble Space Telescope since launch and is integrated with a large and complex spacecraft ground system. The concept of suitability functions used in SPIKE makes it possible to efficiently represent the many factors which are important in real-world scheduling decisions. A powerful multistart stochastic repair technique is used to generate schedules. SPIKE's flexibility has been demonstrated by adapting it for several other spacecraft missions and ground-based observatories and by integrating long- and short-term scheduling at different levels of abstraction in the same constraint representation and search framework.

References

Adorf, H.-M., 1990, "The Processing of HST Observing Programs", Space Telescope European Coordinating Facility Newsletter 13, 12—15.

Adorf, H.-M., and Johnston, M. 1990, "A Discrete Stochastic 'Neural Network' Algorithm for Constraint Satisfaction Problems", Proceedings of the International Joint Conference on Neural Networks, (Piscataway, NJ: IEEE), Vol III, 917—924.

Allen, J. F. 1983. "Maintaining Knowledge About Temporal Intervals." Commun. Assoc. Comput. Machin. 26: 832—843.

Cheng, Y., Kashyap, R.L. 1988. "An Axiomatic Approach for Combining Evidence from a Variety of Sources." Journal of Intelligent and Robotic Systems 1: 17—33.

Cox, R. T. 1946. "Probability, Frequency and Reasonable Expectation." American J. Phys. 14: 1—13.

Dechter, R. 1986. "Learning while searching in constraint satisfaction problems." Fifth National Conf. Artificial Intelligence (AAAI 86), Philadelphia, Pennsylvania, 178—183.

Dechter, R. and Meiri, I. 1989. "Experimental evaluation of preprocessing techniques in constraint satisfaction problems." Proceedings of the Eleventh International Joint Conf. on Artificial Intelligence, Detroit, Michigan (Morgan Kaufmann), 271—277.

Dechter, R. and Pearl, J. 1988. "Network-Based Heuristics for Constraint Satisfaction Problems." Artificial Intelligence 34: 1—38.

Fox, M. 1987. Constraint-Directed Search: A Case Study of Job Shop Scheduling. (Morgan Kaufmann: San Mateo, CA).

Fox, M. and Smith, S. 1984. "ISIS: A Knowledge-Based System for Factory Scheduling." Expert Sys. 1: 45.

Fox, M., Sadeh, N. and Baykan, C. 1989. "Constrained Heuristic Search." Proc. 11th International Joint Conf. on Artificial Intelligence, Detroit, MI 309—315.

Freuder, E. 1989. "Partial Constraint Satisfaction." Proc. 11th International Joint Conf. on Artificial Intelligence, Detroit, MI 278—283.

Gerb, A. 1991a. "Transformation Reborn: A New Generation Expert System for Planning HST Operations", Proceedings of the 1991 Goddard Conference on Space Applications of Artificial Intelligence, NASA CP 3110, 283-295.

Gerb, A. 1991b. "The 'Looker': Using an Expert System to Test an Expert System." Proc. 1991 World Congress on Expert Systems, Orlando, Florida (Pergamon), 1005—1012.

Good, I. J. 1960. "Weight of Evidence, Corroboration, Explanatory Power, Information and the Utility of Experiments" J. Royal Statist. Soc. 22: 319—331.

Good, I. J. 1968. "Corrigendum: Weight of Evidence, Corroboration, Explanatory Power, ..." J. Royal Statist. Soc. 30: 203—203.

Hájek, P. 1985. “Combining functions for certainty degrees in consulting systems.” Int. J. Man-Machine Studies 22: 59—76.

Hart, P., Duda, P. and Einaudi, M. 1978. “PROSPECTOR — a Computer-Based Consultation System for Mineral Exploration.” Math. Geol. 10: 589.

Isobe, T., Johnston, M. D., Morgan, E. and Clark, G. 1993. “The Application of SPIKE to ASTRO-D Mission Planning.” Proc. 2nd Astron. Data Analysis Software and Systems (ADASS), Boston, MA (in press).

Jackson, R., Johnston, M., Miller, G., Lindenmayer, K., Monger, P., Vick, S., Lerner, R. and Richon, J., 1988, “The Proposal Entry Processor: Telescience Applications for Hubble Space Telescope Operations”, Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence, NASA CP 3009, 197—212.

Johnston, M. 1988a. “Automated Telescope Scheduling.” in Coordination of Observational Projects in Astronomy, ed. C. Jaschek and C. Sterken (Cambridge Univ. Press: Cambridge), 219—226.

Johnston, M. 1988b. “Artificial Intelligence Approaches to Spacecraft Scheduling.” Proc. ESA Workshop on Artificial Intelligence Applications for Space Projects, ESTEC (Noordwijk, Holland) 5—9.

Johnston, M. 1988c. “Automated Observation Scheduling for the VLT.” Proc. ESO Conf. on Very Large Telescopes and their Instrumentation, ESO (Garching, Germany) 1273—1282.

Johnston, M. 1989a. “Reasoning with Scheduling Constraints and Preferences.” Space Telescope Science Institute SPIKE Tech. Report 1989-2.

Johnston, M. 1989b. “Knowledge-Based Telescope Scheduling.” in Knowledge-Based Systems in Astronomy, ed. A. Heck and F. Murtagh (Springer-Verlag: Heidelberg), 33—49.

Johnston, M. 1990. “SPIKE: AI Scheduling for NASA’s Hubble Space Telescope.” Proc. 6th IEEE Conf. on AI Applications, Santa Barbara, CA 184—190.

Johnston, M. D. and H.-M. Adorf 1989. “Learning in Stochastic Neural Networks for Constraint Satisfaction Problems.” NASA Conf. on Space Telerobotics, 31 Jan — 2 Feb 1989, Pasadena, CA.

Johnston, M., and Adorf, H.-M. 1992, “Scheduling with Neural Networks — The Case of Hubble Space Telescope”, Computers and Operations Research 19, 209—240.

Johnston, M., and Minton, S. 1993: “Analyzing a Heuristic Strategy for Constraint-Satisfaction and Scheduling.” Current volume.

Kumar, V. 1992. “Algorithms for Constraint Satisfaction Problems: A Survey.” Artificial Intelligence Magazine 13: 32—44.

Lucks, M. 1992. “Detecting Opportunities for Parallel Observations on the Hubble Space Telescope.” Goddard Conference on Space Applications of Artificial Intelligence, Goddard Space Flight Center, Greenbelt, Maryland.

Miller, G., Rosenthal, D., Cohen, W. and Johnston, M. 1987. “Expert Systems Tools for Hubble Space Telescope Observation Scheduling.” Telematics and Informatics 4: 301—311.

Miller, G., Johnston, M., Vick, S., Sponsler, J. and Lindenmayer, K. 1988. “Knowledge-Based Tools for Hubble Space Telescope Planning and Scheduling: Constraints and Strategies.” Telematics and Informatics 5: 197—212.

Miller, G., 1989, "Artificial Intelligence Applications for Hubble Space Telescope Operations", in Knowledge Based Systems in Astronomy, ed. F. Murtagh and A. Heck, (Berlin: Springer Verlag), 5—32

Miller, G. and Johnston, M., 1991, “A Case Study of Hubble Space Telescope Proposal Processing, Planning and Long-Range Scheduling”, Proc. Conf. Computing in Aerospace 8 (AIAA), 1—13.

Minton, S., Johnston, M., Philips, A. and Laird, P., 1990, “Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method”, Proceedings of the Eighth National Conference on Artificial Intelligence, (Menlo Park, CA: AAAI Press), 17-24.

Minton, S., Johnston, M., Philips, A. and Laird, P. 1992. “Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling.” Artificial Intelligence ?: ?

Morgan, E. 1992. “Evaluation of SPIKE for XTE.” MIT Center for Space Research Tech. Report

Muscettola, N. 1992. “Scheduling by Iterative Partition of Bottleneck Conflicts.” Carnegie Mellon University Tech. Report CMU-RI-TR-92-05.

Muscettola, N., Smith, S. F., Cesta, A. and D’Aloisi, D. 1992. “Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture.” IEEE Control Systems Magazine 12(2).

Pearl, J. 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. (Morgan Kaufmann: San Mateo, CA).

Rit, J. F. 1986. “Propagating Temporal Constraints for Scheduling.” Proc. 5th National Conf. on Artificial Intelligence, 383—386.

Rosenthal, D., Monger, P., Miller, G. and Johnston, M. 1986. “An Expert System for the Ground Support of Hubble Space Telescope.” Proc. 1986 Goddard Conf. on Space Applications of Artificial Intelligence, Goddard Space Flight Center, Greenbelt, Maryland

Sadeh, N. 1991. “Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling.” CMU Tech. Report CMU-CS-91-102.

Sadeh, N. and Fox, M. 1988. “Preference Propagation in Temporal/Capacity Constraint Graphs.” Carnegie Mellon University Tech. Report CMU-RI-TR-89-2.

Shapiro, R. D. 1980. “Scheduling coupled tasks.” Nav. Res. Logist. Q. 27: 489—479.

Shortliffe, E. 1976. Computer-Based Medical Consultation: MYCIN. (American Elsevier: New York).

Smith, S., Fox, M. and Ow, P. S. 1986. “Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems.” Artificial Intelligence Magazine 7(4): 45—61.

Sponsler, J., and Johnston, M., 1990, “An Approach to Rescheduling Activities Based on Determination of Priority and Disruptivity”, Telematics and Informatics 7, 243-253.

Sycara, K., Roth, S., Sadeh, N. and Fox, M. 1991. “Resource Allocation in Distributed Factory Scheduling.” IEEE Expert 6(1): 29—40.

Zweben, M., Davis, E. and Gargan, R. 1990. “Anytime Rescheduling.” Proc. DARPA Workshop on Innovative Approaches to Planning and Scheduling,