

## REFERENCES

- [1] King, J.R., and Spachis, A.S. 1980: "Scheduling: Bibliography and Review," *Int. Journal of Physical Distribution and Materials Management* **10**, p. 105.
- [2] Garey, M., and Johnson, D. 1979: *Computers and Intractability*, (W.H. Freeman & Co.: San Francisco).
- [3] Fox, M., and Smith, S. 1984: "ISIS: A Knowledge-Based System for Factory Scheduling," *Expert Systems* **1**, p. 25.
- [4] Fox, M., Sadeh, N., and Baykan, C. 1989: "Constrained Heuristic Search", in *Proc. 11th IJCAI*, Detroit, Aug. 1989, p. 309
- [5] Miller, G., Rosenthal, D., Cohen, W., and Johnston, M. 1987: "Expert System Tools for Hubble Space Telescope Observation Scheduling," in *Proc. 1987 Goddard Conference on Space Applications of Artificial Intelligence*; reprinted in *Telematics and Infomatics* **4**, p. 301 (1987).
- [6] Johnston, M., 1988: "Automated Telescope Scheduling," in *Proc. Conf. on Coordination of Observational Projects*, Strasbourg, Nov. 1987, (Cambridge Univ. Press: Cambridge), p. 219.
- [7] 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", in *Proc. 1988 Goddard Conference on Space Applications of Artificial Intelligence*.
- [8] "HST Planning Constraints", 1987, Space Telescope Science Institute, Spike Tech. Report 87-1.
- [9] Johnston, M. 1988: "Reasoning with Scheduling Constraints and Preferences," Spike Tech. Report No. 89-2, Jan. 1989.
- [10] Adorf, H.-M., and Johnston, M. 1990: "Stochastic Neural Networks for Constraint Satisfaction Problems", submitted.
- [11] Johnston, M., and Adorf, H.-M. 1989: "Learning in Stochastic Neural Nets for Constraint Satisfaction Problems," in *Proc. NASA Conference on Telerobotics*, Pasadena, CA (Jan. 1989).
- [12] Johnston, M., and Adorf, H.-M. 1990: "Scheduling with Neural Networks", in preparation.
- [13] Hopfield, J., and Tank, D. 1985: "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics* **52**, p. 141.
  
- [14] Sponsler, J., and Johnston, M. 1990: "An Algorithm to Determine the Suitability for Rescheduling Activities on an Existing Timeline", to appear in the *Proceedings of the 1990 Goddard Conference on Space Applications of Artificial Intelligence* (Greenbelt, May 1-2, 1990).
- [15] Johnston, M. 1988: "Automated Observation Scheduling for the VLT", in *Proc. ESO Conference on Very Large Telescopes and their Instrumentation*, Garching, March 1988.
- [16] Johnston, M., and Miller, G. 1988: "AI Approaches to Astronomical Observation Scheduling," in *Proc 3<sup>rd</sup> International Workshop on Data Analysis in Astronomy*, Erice (June 1988) (Plenum Press, NY).
- [17] Johnston, M. 1989a: "Knowledge-Based Telescope Scheduling," in *Knowledge-Based Systems in Astronomy*, ed. A. Heck and F. Murtagh (Springer-Verlag: 1989)

**Acknowledgements:** The authors are grateful to J. Sponsler, S. Vick, and R. Jackson (STScI), D. Rosenthal, and H.-M. Adorf (ST-ECF) for many useful discussions. The support of University of California, Berkeley, for the Spike Unix port is gratefully acknowledged. Space Telescope Science is operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration.

changing and adding constraints. This flexibility has been exercised by conducting several experiments in adapting Spike to schedule observations from other missions, including a one-year period of European International Ultraviolet Explorer (IUE) programs[], the Extreme Ultraviolet Explorer (EUVE), and a ground-based telescope in Chile [15].

Spike was recently ported to Unix (Allegro Common Lisp from Franz, Inc.) with the user interface running under X-windows. This portability offers the important advantage of platform independence and will allow us to take advantage of future improvements in the performance of general-purpose workstations. This will also permit Spike to be used by other observatories which are unable to justify the acquisition of special purpose Lisp machines for scheduling: both the Extreme Ultraviolet Explorer mission (Univ. of Calif. Berkeley) and the X-ray Timing Explorer (MIT) are planning to use Spike for their observatory scheduling.

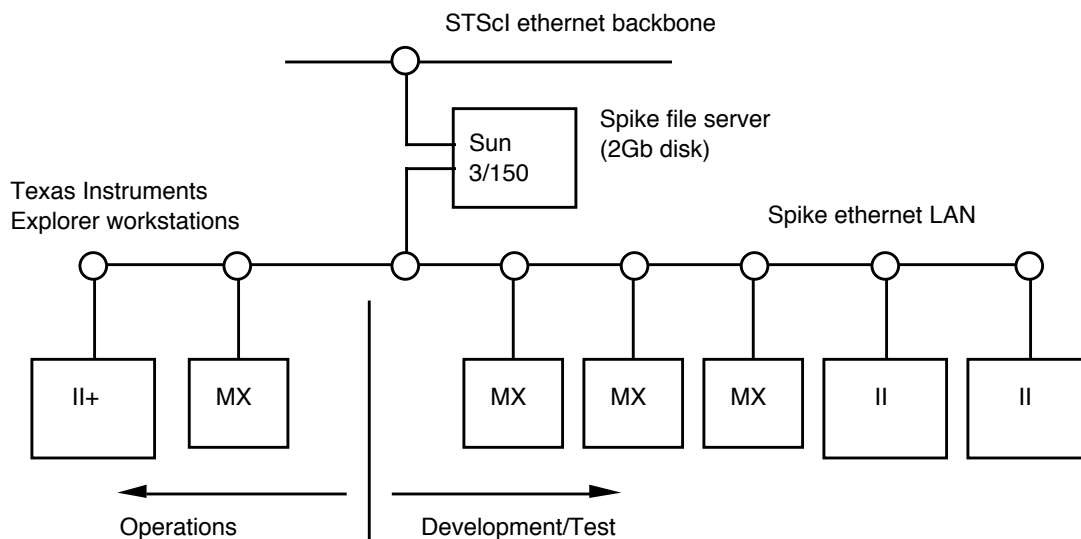


Figure 3: The Spike operations and development hardware configuration. The system is isolated (for data security) from the main STScI backbone by a Sun file server with ~2Gb of local disk storage. The Spike local area network (LAN) includes seven Texas Instruments Explorer lisp machines, ranging from microExplorers (an Explorer co-processor in a Macintosh II chassis) to the very fast Explorer II+.

## 6. CONCLUSIONS

It is clear that software technology and approaches to scheduling have reached a sufficient level of development that intelligent spacecraft scheduling is a realistic goal. The use of AI techniques makes it possible to develop and adapt software, such as the HST Spike scheduling tools, for a variety of telescope scheduling problems. The Spike framework is sufficiently general [16,17] that scheduling problems from other domains could also be handled. The advantages of using these techniques are primarily a rapid software development cycle, a concise but expressive representation of scheduling data, flexibility in the definition and modification of scheduling constraints, powerful facilities for expressing search strategies, and the ability to incorporate a graphics-oriented user interface to help the user understand and modify the evolving schedule.

In addition to the above, preliminary work has been done on several schedule repair strategies that focus on modifying the unexecuted portion of a schedule given the goal of making minimal changes [14]. This will be important when preserving a published schedule must be included as a goal during rescheduling, i.e. the schedule becomes self-constraining.

### **Formulate and attack the problem hierarchically**

A common and important problem-solving strategy is to formulate and solve a simpler higher-level problem, then attack the resulting lower-level subproblems by constraining them with the higher-level solution. In the scheduling domain there are two obvious ways to accomplish this: by scheduling *groups* of related activities at once, and by limiting the *time granularity* of the schedule. Both of these are exploited by Spike.

- The overall scheduling period is divided into intervals and activities to be scheduled are first committed to these intervals. Once a satisfactory set of commitments is found, the intervals are further decomposed and the process repeated. This avoids the need to make early commitments to specific times for activities when scheduling over long periods (months to years).
- Activities that can be clustered together and scheduled as single “meta-activities” are identified by the system before scheduling starts. This reduces the number of individual activities to schedule, as well as reducing the number of constraints that need to be separately considered. This technique alone has been shown to reduce the size of the HST scheduling problem by a significant factor.

### **Provide explicit user visibility and control**

The approach we have taken in Spike is that automated scheduling is fundamentally a support tool for the people who are responsible for making scheduling decisions. In this approach one of the most important characteristics of the scheduler is how it interacts with the user. The user must have *visibility* into all aspects of the scheduling problem and the evolving schedule. The user must also have *control*, i.e. the ability to override any decisions made by the scheduler, and the ability to create and evaluate alternative schedule fragments. A graphical user interface is very important to this concept, as illustrated in Fig. 2. Though this interface the user is free to construct new displays at will, manipulate the schedule (down to the point of clicking on a displayed activity and changing its location in the schedule), and invoking the various automatic scheduling algorithms. The ability to spawn “alternative” schedules is also provided, so that the user can perform “what-if” experiments at any point without committing to choose one of the alternatives.

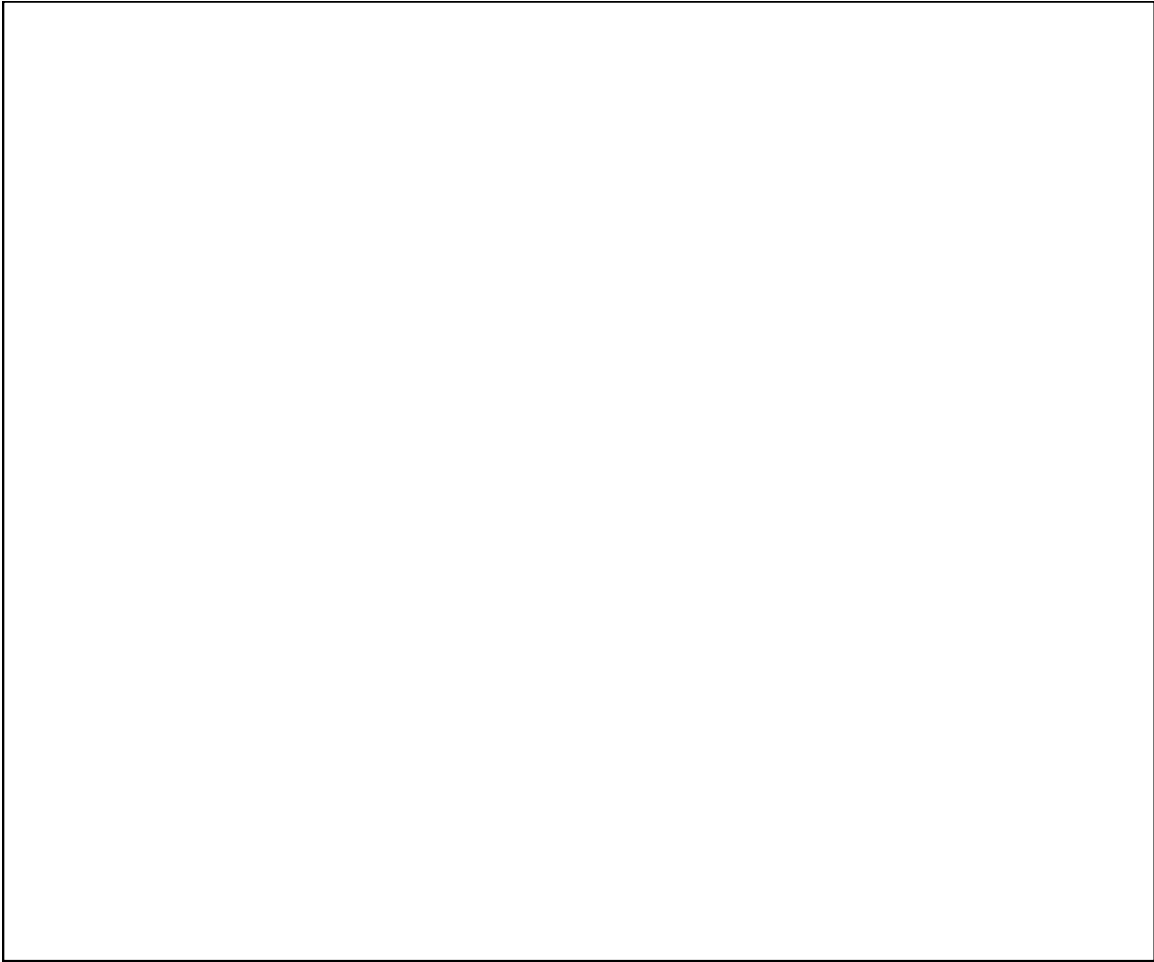
## **5. STATUS AND PLANS**

Spike has been used for Space Telescope ground system testing and is presently starting on the first flight schedules (HST launch is currently scheduled for April 1990). The system is implemented in Common Lisp and (old) Flavors, with a conversion to the Common Lisp Object System in progress and partially completed. The user interface makes use of Common-Windows, and the rulebase is implemented in KEE, both from Intellicorps, Inc. The system has been developed and will be operated on Texas Instruments Explorer workstations, as shown in Fig. 3.

The performance of the system has been shown to be extremely good. A recent test involved scheduling some 12,000 exposures over a 6-month period to 1-week resolution: these activities were clustered into 2,600 “meta-activities”. The neural net search algorithm required less than one hour to generate a schedule. While there is no guarantee that this schedule is optimal, the time required is short enough that several weeks could be devoted to initial schedule optimization. Runtime efficiency could likely be improved by an order of magnitude by re-implementing the core search algorithms in C instead of Lisp.

To date, Spike has concentrated on the long-range scheduling problem of assigning observations to times at a resolution of a few days to a week in schedules of duration one year or more. Some of the constraints appropriate to this problem represent statistical abstractions of the detailed orbital constraints: this is necessary since the precise orbit of HST cannot be accurately predicted more than a few months in advance. The use of Spike on short-term scheduling has been demonstrated and is planned to be developed further in the future. To use Spike on short-term scheduling problems requires only the accurate definition of the appropriate constraints: no limitations on time granularity are built into the system.

Scheduling problems are rarely static: changes come about because of increased experience with the problem and because of on-orbit experience with the operating spacecraft. Spike was designed with this fact in mind, particularly with regard to



**Figure 2:** Example screen from Spike showing the scheduling of a few HST observations. The upper left window represents a six-month scheduling interval and displays the combined degree of preference (suitability function) for scheduling a number of exposures (running vertically up the window). The lower window shows an expanded time view of one specific exposure and the constraints that contribute to its scheduling preference. The text window on the right displays descriptive information about the schedule, activities, and constraints. The user interacts with the system by clicking on various active regions and selecting from pop-up menus. For example, clicking on the time scale at the bottom of each window permits zooming in or out in time, or paging forwards or backwards. The user can create new windows and build new displays dynamically.

---

- **neural networks:** a very different approach makes use of an “artificial neural network” (e.g. [13]) to represent a set of discrete scheduling choices. These networks are conceptually composed of a large number of simple processing elements operating in parallel whose computational power comes from their massive interconnection. We have developed a means of constructing the network directly from the suitability functions which describe the scheduling constraints, thus eliminating the need for an expensive training stage which has limited the utility of neural networks in other applications. This technique is fast and will be the method used for initial construction of the long-range schedule. It is also particularly well suited for reactive rescheduling, since only network biases and connections need to be modified.

## 4. STRATEGIES FOR OPTIMAL SCHEDULING

In this section we describe some of the techniques implemented in Spike that can help deal with the problems described above.

### Separate constraint reasoning from strategic search

This is a statement about the overall architecture of the scheduler (see Fig. 1). The purpose is to separate those aspects of the system that reason about *constraints* from those that reason about (partial or complete) *schedules*. The reason for this separation is that these reasoning processes take place on very different levels. Constraint reasoning is low-level and determines feasible and preferred scheduling times among which choices can be made; strategic reasoning evaluates one or more schedules and actually makes the choices. There can be more than one source of strategic knowledge available to work on one scheduling problem: all, however, make common use of the results of constraint reasoning.

#### Use uncertainty reasoning methods for reasoning about constraints

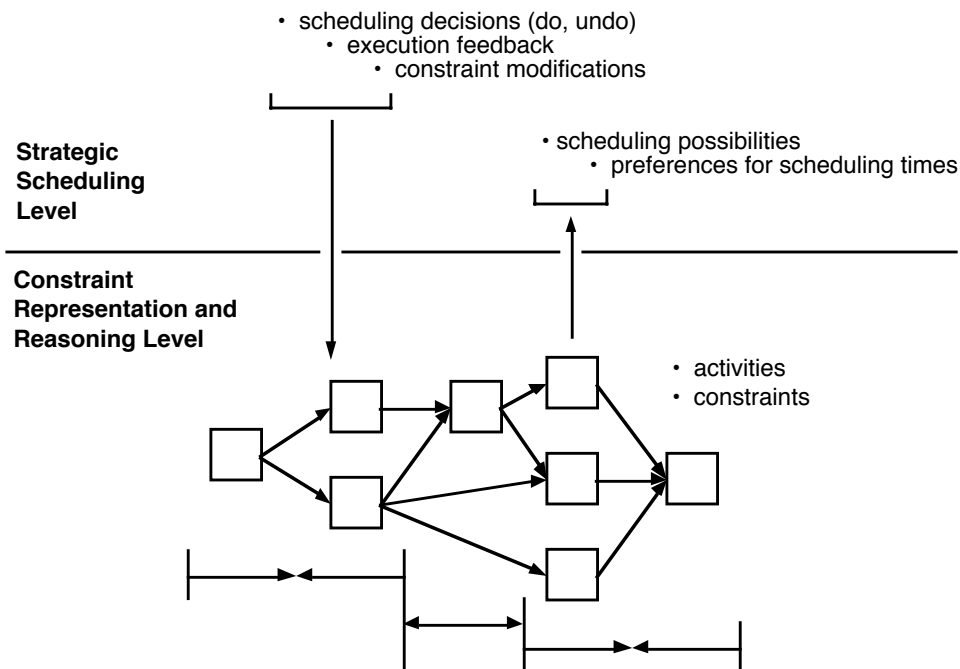
There has accumulated a large body of theoretical and practical results on reasoning with uncertainty in the context of discrete rule-based expert systems. Based on this work we have developed a continuum version of uncertainty reasoning that can efficiently represent a wide variety of scheduling constraints [9]. Our framework is well-suited to the weighing of evidence for and against different scheduling hypotheses, thus providing essential inputs to trade-off decisions. In this framework a *suitability function*  $S_i(t)$  is computed for each activity  $A_i$  to be scheduled, representing the combined degree of preference of scheduling  $A_i$  at time  $t$ . The suitability function includes information about strict constraints (times when  $A_i$  cannot be scheduled due to constraint violations) as well as the combined preferences from multiple sources. The effects of both absolute time constraints and constraints that depend on when other activities are scheduled are included in the suitability functions, examples of which are illustrated in Fig. 2. For several large classes of constraints, suitability functions can be specified declaratively, making easy to add or change the constraint reasoning level without affecting strategic reasoning. The suitability function mechanism allows Spike to capture quantified expert judgements of scheduling factors in a uniform manner.

The implications of constraints are pre-propagated to the greatest extent possible and saved, avoiding repetitive computations during the scheduling search process. This also has the beneficial side-effect of identifying early some types of over-constrained activities which are unschedulable because of irreconcilable constraint conflicts. These usually must be resolved in consultation with the astronomer who requested them. Constraints are also used to identify activities that can be *clustered* into single scheduling entities in order to reduce the number of individual activities that must be considered during the search process.

### Provide multiple control mechanisms for strategic scheduling and search

Based on the constraint-reasoning layer it is possible to implement a variety of strategic search mechanisms, any of which may be invoked depending on the nature and state of the problem. To date we have implemented three such mechanisms in the Spike scheduling tools:

- **procedural search:** this includes standard search techniques such as best-first or most-constrained-first algorithms. These tend to be computationally expensive and often encounter deadends which result in grossly sub-optimal schedules.
- **rule-based heuristic search:** this mechanism includes *search rules* to examine the state of a collection of partial schedules to identify the most “promising”, and *commitment rules* to decide how to extend the schedule by making some scheduling decision [7]. The rules communicate with the constraint-reasoning layer through “frames” or “schema” that hold summary information about the partial schedules. This general approach is well suited to the representation of quite complex scheduling heuristics, but has not proven to be very efficient. At present it is not used in the operational Spike system.



**Figure 1:** A diagram of the overall Spike architecture. The lower level is a constraint representation and reasoning system which contains descriptions of activities to schedule and their constraints. Temporal constraints are indicated schematically, but the system can deal with a wide variety of strict and preferences constraint types. The upper level interacts with the constraint representation level when searching for feasible and optimal schedules. A variety of modular search strategies can be utilized at this level.

HST observations are subject to many constraints from a variety of sources. The telescope may not be pointed too close to the sun, moon, or sunlit earth limb. The orientation of the telescope is constrained by power and thermal considerations. There are a large number of constraints on communications, pointing stability, and operation timings that follow from the telescope and instrument design and from operational procedures. In addition to constraints from these sources, astronomers are free to specify constraints on how and when their exposures are to be taken in order to satisfy their scientific objectives. As a result, most observations are constrained relative to other observations (before, after, or within some interval), and, less frequently, with respect to absolute time. In addition to constraints that specify strict constraints on exposures, there are a large number of subtle constraints that indicate preferences, i.e. conditions that improve the quality or usefulness of the data obtained. Strict and preference constraints must be traded off, so that none of the former are violated while the latter are satisfied to the maximum extent possible.

Astronomers provide their exposure and constraint descriptions to STScI in machine-readable form over a network. These are checked, stored in a database, and extracted to form the scheduling database. A long-range schedule covers one year or more of observing time and will typically include about 10,000-30,000 exposures. Since each exposure usually participates in some tens of constraints, this is a very large-scale scheduling problem indeed. The role of Spike is to integrate the requested exposures into a feasible schedule that satisfies as many preference constraints as possible, makes efficient use of the telescope, and is robust against disruptions. Spike's long-range scheduling is carried out to a time resolution of a few days to a week. The final detailed ordering is presently performed by another system which essentially "compiles" the exposure sequence into telescope and instrument commands. As the exposures are executed, Spike's schedule will be constantly revised to incorporate feedback from the as-flown timeline.

differences.

At Space Telescope Science Institute we initiated the Spike project in early 1987 for the purpose of developing AI scheduling tools [5,6,7] for Hubble Space Telescope (HST). HST scheduling is an extremely demanding task, requiring the scheduling of some tens of thousands of observations per year subject to a large number of proposer-specified and operational constraints [8]. Our overall approach to HST scheduling was inspired by the work of Smith and co-workers on the factory scheduling problem but has drawn on a number of other lines of research as well: as part of the Spike project we have developed a new framework for representing and reasoning with scheduling constraints [9] (based on discrete uncertainty reasoning for rule-based expert systems) and new techniques for searching the space of possible schedules [10-12] (based on recent developments in artificial neural networks).

There are several notable features of spacecraft scheduling that make it a difficult problem: interacting constraints, uncertainty, optimization criteria, and search. Realistic scheduling problems will typically involve a large number of different types of constraints, both strict and preference. Trading-off and balancing conflicting constraints adds greatly to the complexity of scheduling. Uncertainty can enter in a variety of ways, ranging from chaotic (completely unpredictable) scheduling factors to a smooth degradation of confidence in an extrapolated model. Optimization criteria are often complex and situation-dependent: there is no single criterion that can be used to indicate schedule optimality. All of these considerations must be incorporated into a schedule search process which grows exponentially with the size of the problem.

### **3. HST SCHEDULING**

Hubble Space Telescope is a large astronomical observatory to be placed into low-earth orbit by the Space Shuttle in April 1990. It is planned to have a lifetime of at least 15 years. Astronomers from around the world request time on the telescope to perform scientific investigations of various types. To this end they specify the exposures they wish to obtain, along with any relevant constraints, and provide these descriptions to the Space Telescope Science Institute (STScI). STScI is responsible for scheduling the observatory in an integrated fashion, i.e. by combining into a single schedule exposures from many different astronomers. Integrated scheduling is required in order to ensure efficient utilization of valuable telescope time.

# ARTIFICIAL INTELLIGENCE SCHEDULING FOR NASA'S HUBBLE SPACE TELESCOPE

**Mark D. Johnston**

Space Telescope Science Institute  
3700 San Martin Drive  
Baltimore MD 21218  
(301) 338-4742, johnston@stsci.edu

**Glenn Miller**

Astronomy Programs  
Computer Sciences Corporation  
Staff member of Space Telescope Science Institute  
(301) 338-4738, miller@stsci.edu

## ABSTRACT

We have implemented an AI system called Spike for scheduling observations with NASA's Hubble Space Telescope. The system incorporates innovative methodologies for representing and reasoning with scheduling constraints and preferences, and for conducting scheduling search. For the former, a combination of constraint satisfaction techniques and weight-of-evidence combination has been devised to propagate temporal constraints and preferences. For the latter, a "neural network" scheduling algorithm has been found to be highly efficient. The Spike system is operational and is currently being used to generate long-range schedules for Space Telescope operations to follow initial on-orbit checkout.

## 1. INTRODUCTION

To obtain the maximum benefit from expensive space facilities it is important to schedule spacecraft operations in an "optimal" manner. Since truly optimal scheduling is usually intractable, in practice it is therefore necessary to determine the best schedule possible, subject to resource and time constraints on the computational effort that can be invested.

The fundamental requirements of optimal spacecraft scheduling are similar in many ways to those of other scheduling problems, e.g. those encountered in commercial and industrial domains. These problems have been found to be notoriously difficult to solve in practical settings. In this paper we discuss some of these difficulties and how the use of artificial intelligence (AI) techniques can be applied to help overcome them. We describe the progress made at Space Telescope Science Institute (STScI) in developing the Spike AI scheduling system for Hubble Space Telescope (HST), and conclude with a discussion of how the Spike approach can be adapted to other scheduling problems.

## 2. APPROACHES TO SCHEDULING

Computer techniques for optimal scheduling have been investigated for many years for a number of applications (see, e.g., [1] for a comprehensive review and bibliography). Much of this classical work has focused on versions of the idealized "job-shop" scheduling problem. This problem and related ones are NP-complete, meaning that there are no known efficient algorithms for finding optimal solutions (see, e.g., [2]).

The basic problem with many of these approaches is that they require key features of the problem to be abstracted away, so that even an exact solution to the abstracted problem is often of little relevance to the original real problem. Approximate solutions to the abstracted problem suffer from the same limitations. It is clear that classical approaches can be useful for problems which are sufficiently simple: in practice this often means that schedule optimization is driven by a *single* overriding criterion. For the problem of scheduling complex modern space facilities, however, this is not the case: more powerful techniques are required.

Several artificial intelligence research efforts have considered scheduling as a domain where AI techniques can be fruitfully applied. Of particular interest is the factory scheduling work of Fox, Smith, and co-workers (e.g., [3,4]) who have developed a rich constraint representation and versatile reasoning process for attacking realistic factory scheduling problems. While factory scheduling shares a number of common features with spacecraft scheduling, there are some obvious and important