SPIKE: ARTIFICIAL INTELLIGENCE SCHEDULING FOR HUBBLE SPACE TELESCOPE

Mark Johnston*, Glenn Miller†, Jeff Sponsler*, Shon Vick*, and Robert Jackson†

Space Telescope Science Institute
3700 San Martin Drive
Baltimore MD 21218

Abstract

The problem of optimal spacecraft scheduling is both important and difficult. Efficient utilization of spacecraft resources is essential, but the accompanying scheduling problems are often computationally intractable and are difficult to approximate because of the presence of numerous interacting constraints. We have applied artificial intelligence techniques to the scheduling of the NASA/ESA Hubble Space Telescope (HST). This presents a particularly challenging problem since a yearlong observing program can contain some tens of thousands of exposures which are subject to a large number of scientific, operational, spacecraft, and environmental constraints. We have developed new techniques for machine reasoning about scheduling constraints and goals, especially in cases where uncertainty is an important scheduling consideration and where resolving conflicts among conflicting preferences is essential. These technique have been utilized in a set of workstation-based scheduling tools (Spike) for HST. Graphical displays of activities, constraints, and schedules are an important feature of the system. High-level scheduling strategies using both rule-based and neural network approaches have been developed. While the specific constraints we have implemented are those most relevant to HST, the framework we have developed is far more general and could easily handle other kinds of scheduling problems. This paper describes the concept and implementation of the Spike system and some experiments in adapting Spike to other spacecraft scheduling domains.

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 computationally intractable, it is therefore necessary to determine the best possible schedule given the 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 describe the source of some of these difficulties and how the use of advanced software technology ("artificial intelligence") can be applied to help overcome them. We describe the progress made at Space Telescope Science Institute (STScI) in developing AI tools for

---

* Space Telescope Science Institute (Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration)

† Astronomy Program, Computer Sciences Corporation, Staff member of the Space Telescope Science Institute

the Hubble Space Telescope (HST), and conclude with a discussion of how these tools can be adapted for other spacecraft scheduling problems.

HUBBLE SPACE TELESCOPE SCHEDULING

In this section we give a brief description of the problem of scheduling Hubble Space Telescope. Further discussion can be found in [Miller et al. 1987, Johnston 1988, and Johnston et al. 1987].

Astronomers from around the world submit proposals to obtain observations with HST. Following an annual peer review and selection process[1], accepted proposals are prepared and provided by the successful proposers to STScI. These contain the detailed specification of each observing program. In addition to such obvious aspects as which instrument to use and which astronomical targets to observe, astronomers can (and do) specify other constraints on how their observations are to be taken. These include a wide variety of relative timing constraints (precedence, minimum and maximum time separation, interruptability conditions, repetitions). Some exposures must be taken at precise times or within specified time windows. Others may require special observing conditions (e.g. they must be taken while HST is in the Earth's shadow to minimize scattered light background). Still others may be conditional on results obtained after analysis of precursor exposures taken by HST, or, in some cases, by other observatories (on the ground or in space).

Constraints specified by the astronomer must be combined with constraints due to many other sources. HST is limited in how close to bright sources of light (e.g. the Sun, Moon, and sunlit earth limb) it can be pointed. Sensitive observations require minimizing subtle sources of stray and scattered background light. The low-earth orbit of HST (~95^m period) means that targets are typically occulted by the earth after no more than ~40^m of observing time. Radiation belts interfere with observations to reduce even further the available viewing time per orbit. For pointing stability, a pair of suitable guide stars must be found for placement in Fine Guidance Sensors: in some parts of the sky these stars are sparse. There are thermal and power constraints that appear in the form of off-nominal roll limitations and recovery-time requirements. This list of constraints is far from exhaustive, and, in general, each exposure is subject to some tens of constraints.

A one-year observing schedule for HST will generally contain ten to thirty thousand exposures constrained as described above. It is clear that a scheduling problem of this magnitude is a very large one indeed.

SPIKE

Computer techniques for optimal scheduling have been investigated for many years by a number of researchers (see, e.g. [King and Spachis 1980] for a comprehensive review and bibliography). Much of this classical work has focussed on versions of the idealized "job-shop" scheduling problem. This problem and related ones are NP-complete, meaning that there are no efficient algorithms for finding solutions (see, e.g., [Garey and Johnson 1979]).

The basic problem with these classical results is that they require key features of the problem to be abstracted away, so that even "exact" solutions to the abstracted problem are 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

[1]The first such solicitation and proposal selection was completed in mid-1989.

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 needed that can handle the complexities of real-world problems.

There are four notable features of spacecraft scheduling that make it a difficult problem: *interacting constraints*, *uncertainty*, *optimization criteria*, and *search*. Realistic scheduling problems typically involve a large number of different types of constraints, both strict and preference. These constraints often have a very large range of timescales compared to classical scheduling domains, ranging from seconds or minutes to months or years. Trading off and balancing constraints adds greatly to the complexity of scheduling. Uncertainty can enter in a variety of ways, ranging from chaotic (i.e. completely unpredictable) scheduling factors to the smooth degradation of confidence in the results of an extrapolated model. Optimization criteria are often complex and situation-dependent: there is usually no single criterion that can be used to indicate schedule optimality. At various times, the schedule may be optimized with respect to operational efficiency, schedule robustness, or speed of recovery to an original schedule following a disruption. The process of constructing schedules by searching among alternatives is computationally expensive, with exhaustive search usually out of the question.

Spike is an activity-oriented scheduling system developed at Space Telescope Science Institute for scheduling HST. The Spike project was initiated in early 1987: the system has so far been used during ground test activities and is currently beginning work on the first flight schedule (HST launch is now scheduled for April 1990). Spike's current focus is the long-range scheduling problem, i.e. that of scheduling over a year or more to a resolution of a few days. The system is not limited to this problem and was designed to schedule at arbitrary time resolution. The basic architecture of the system is illustrated schematically in Fig. 1.

The overall approach adopted in Spike was inspired by advances in artificial intelligence research (see., e.g. [Fox and Smith 1984] and [Smith, Fox, and Ow 1986] for a discussion of these techniques as applied to factory scheduling problems). Spike incorporates several novel features, however, including an innovative constraint representation and reasoning mechanism [Johnston 1989b] and a new type of search technique motivated by research on artificial neural networks [Adorf and Johnston 1989, Johnston and Adorf 1989].

Spike's constraint representation provides not only for *strict* constraints but can also represent in a natural way *preference* constraints that indicate conditions that are desired but not required in the final schedule. Constraints are propagated in a manner that informs the scheduler (whether human or an automatic search process) what the allowed scheduling opportunities are for all tasks, as well as a measure of the degree of preference of those opportunities.

In addition to a powerful constraint representation mechanism, Spike employs several other strategies to reduce the size of the problem as much as possible:

(1) 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).

(2) Implications of constraints are pre-propagated to the greatest extent possible and saved, avoiding repetitive computations during the scheduling search process. This also identifies many types of overconstrained activities that are unschedulable because of irreconcilable constraint conflicts.

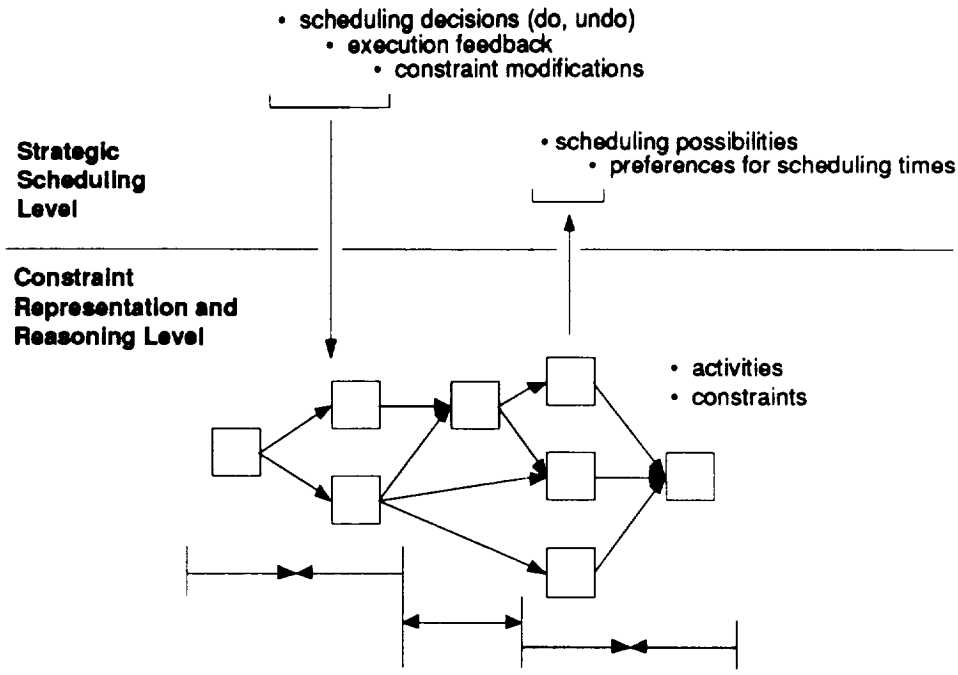

Figure 1: A diagram of the overall Spike architecture. The lower level is a constraint representation and reasoning system which contains descriptions of activities to schedule and their constraints. Temporal constraints are indicated schematically, but the system can deal with a wide variety of strict 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.

(3) Activities that can be clumped 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.

(4) The set of activities to schedule can be partitioned into disjoint sets with associated precedence. The resources consumed by scheduling one set can be cascaded to other sets, thus permitting a significant reduction in the number of activities that must be considered at one time.

Uncertainty is always a serious problem with predictive scheduling. Spike's constraint mechanism provides several ways to deal with this problem. The most important is to define constraints that represent the *probability* of success as preference constraints, or, alternatively, constraints that represent the *maximum acceptable risk* as strict constraints. This permits some constraints to be treated in a statistical fashion. This technique is used to deal with the fact that the in-track position of HST in its orbit cannot be accurately predicted more than a few months into the future.

Although Spike was developed as an automatic scheduling system, it is fundamentally a support tool for the people who are responsible for making scheduling decisions. Thus one of the most important characteristics of the scheduler is how it interacts with users. 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 automatic system, and the ability to create and evaluate alternative schedules. These features are provided by the system. The user interface makes extensive use of a bitmapped graphics window system and mouse to

facilitate two-way interaction with the user. An example screen from the running system is shown in Fig. 2.

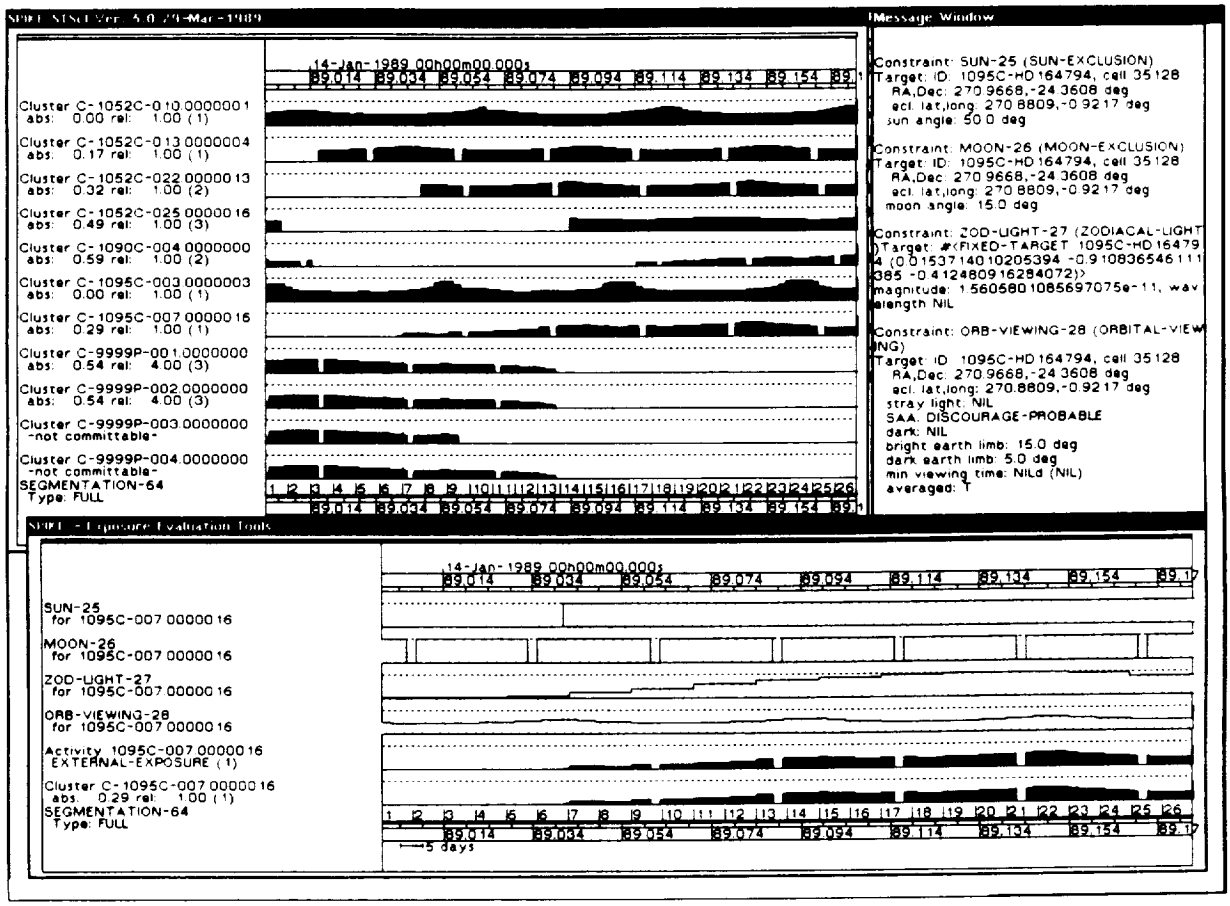

Figure 2: Example screen from Spike showing the scheduling of some HST observations. The upper left window represents a six-month scheduling interval and displays the combined degree of preference for scheduling a number of exposures (running vertically up the window). The lower window shows an expanded 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.

Spike was developed on Texas Instruments Explorer workstations and is implemented in Common Lisp, (old) Flavors, and the Common Lisp Object System (CLOS). Commonwindows (from Intellicorp, Inc.) is used as the windowing system for the user interface. The operations hardware configuration is shown in Fig. 3. There are two operations workstations for scheduling and for evaluating programs for scheduling feasibility. These are networked with the development cluster of five workstations. Both share the same file server which is a Sun workstation with 2Gb of local storage. This server also acts as a gateway to the STScI ethernet backbone and provides secure access to the scheduling data.

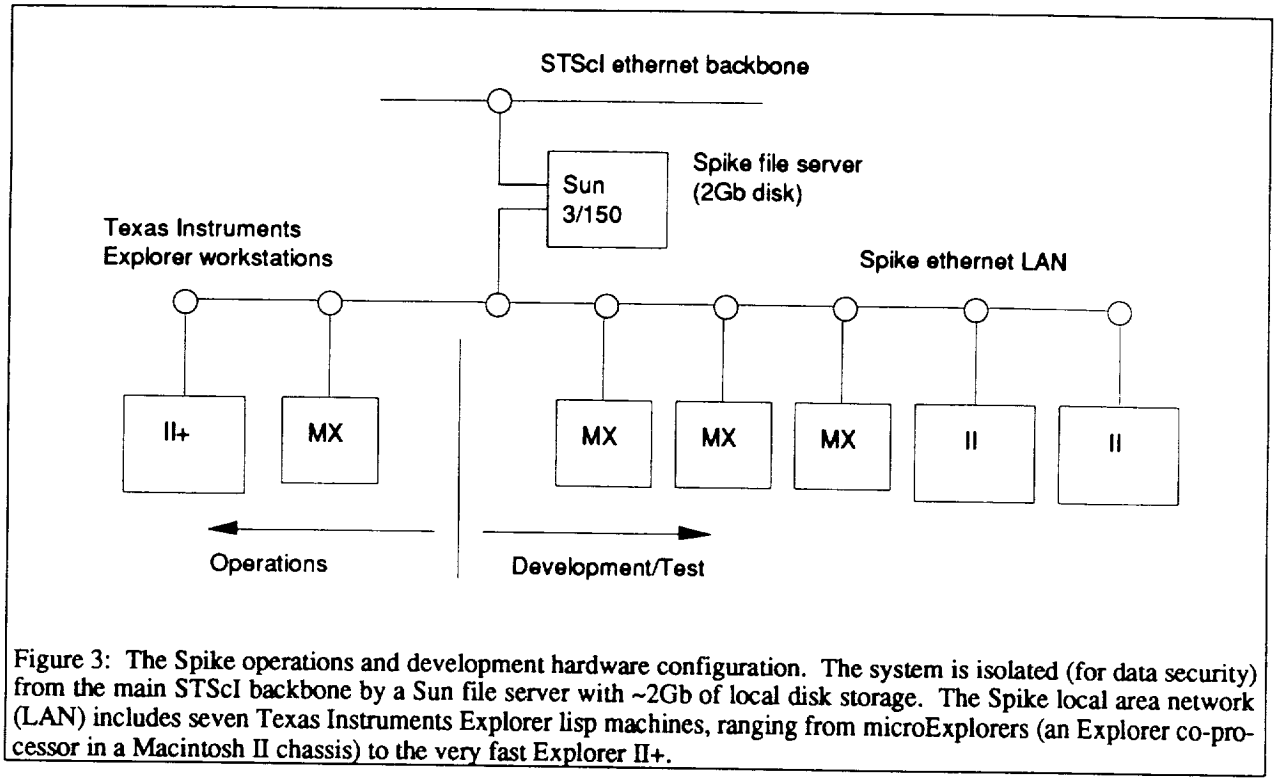

Figure 3: The Spike operations and development hardware configuration. The system is isolated (for data security) from the main STScI backbone by a Sun file server with ~2Gb of local disk storage. The Spike local area network (LAN) includes seven Texas Instruments Explorer lisp machines, ranging from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

Spike has recently been ported to Allegro Common Lisp (from Franz, Inc.) running on Sun workstations. The user interface in this configuration is displayed in an X-windows environment. This permits running the computational kernel of Spike on one machine while the user interface runs on another. Although the present computational power of general-purpose workstations does not yet match that of the fastest Lisp machines, it can be expected that the demonstrated portability to Unix workstations will prove useful for HST and other applications.

Spike is currently operational at STScI and is in the first stages of constructing the initial schedule to follow launch and vehical and instrument checkout.

EXTENSIONS

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 changing and adding constraints.

This flexibility has been exercised by conducting several experiments in adapting Spike to schedule observations from other missions. The results of one of these experiments are shown in Fig. 4. This represents the scheduling of a one-year period of European International Ultraviolet Explorer (IUE) programs to a resolution of one week. Spike was adapted to represent the IUE scheduling constraints with minimal effort. Similar experiments have been conducted for the Extreme Ultraviolet Explorer (EUVE) as well as for a ground-based telescope in Chile [Johnston 1988a]. Spike has been chosen by the EUVE project to perform science scheduling for that mission; the Unix version of the system is presently running at the University of California, Berkeley, for that purpose.

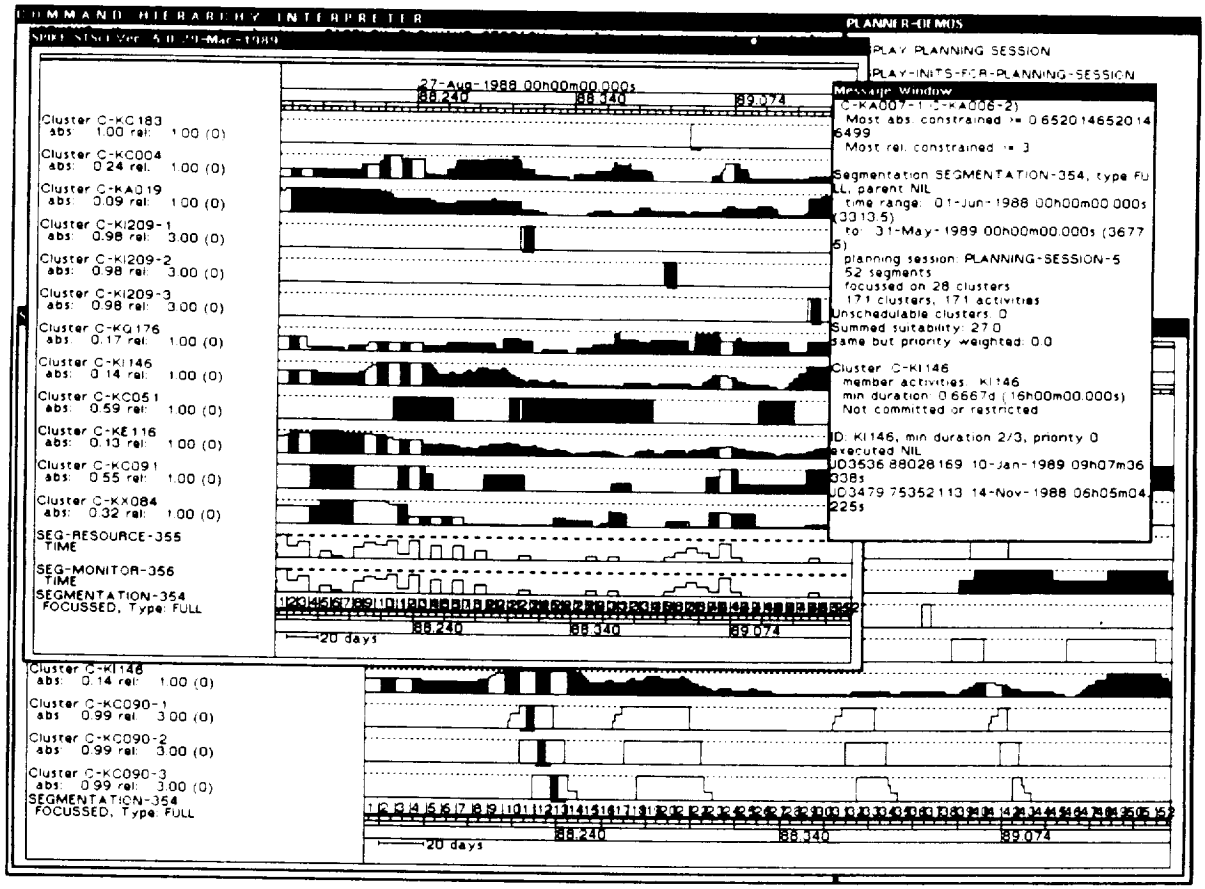

Figure 4: Display of the results of the experimental adaptation of Spike to schedule IUE programs. Shown here is a snapshot of a partial schedule. The top left window covers a time period of a year: 4-hour observing periods are being allocated to weeks. The function plotted is the scheduling degree of preference as in Fig. 2, but here the restrictions of some programs to specific times or sets of times is indicated by the outlined portions of the curves. The bottom plot in the same window shows the amount of time allocated per week (solid line) compared to that available (dashed line).

SUMMARY AND CONCLUSIONS

It is clear that the software technology and approaches to scheduling embodied in Spike 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 Spike for a variety of spacecraft scheduling problems. Spike embodies innovative approaches in several areas, including constraint representation and reasoning and optimal scheduling search methods for large-scale problems.

Future plans for Spike include upgrading all of the system to be compatible with the new ANSI standard Common Lisp Object System (CLOS) and further research on scheduling search methods for large problems. Experience from early scheduling for HST operations (in progress now) will be valuable in further augmenting both the user interface and the computational core of the system.

REFERENCES

Adorf, H.-M., and Johnston, M. 1989: "Stochastic Neural Networks for Constraint Satisfaction Problems", submitted.

Fox, M, and Smith, S. 1984: "ISIS: A Knowledge-Based System for Factory Scheduling," Expert Systems **1**, p. 25.

Garey, M., and Johnson, D. 1979: *Computers and Intractability*, (W.H. Freeman & Co.).

Johnston et al. 1987: "HST Planning Constraints", 1987, Space Telescope Science Institute, Spike Tech. Report 87-1.

Johnston, M. 1988a: "Automated Observation Scheduling for the VLT", in *Proc. ESO Conference on Very Large Telescopes and their Instrumentation*, Garching, March 1988.

Johnston, M. 1988b: "Automated Telescope Scheduling," in *Proc. Conf. on Coordination of Observational Projects*, Strasbourg, Nov. 1987.

Johnston, M. 1989a: "Knowledge-Based Telescope Scheduling," in *Knowledge-Based Systems in Astronomy*, ed. A. Heck and F. Murtagh (Springer-Verlag: 1989)

Johnston, M. 1989b: "Reasoning with Scheduling Constraints and Preferences," Spike Tech. Report 89-2 (Jan. 1989).

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).

Johnston, M., and Miller, G. 1988: "AI Approaches to Astronomical Observation Scheduling," in *Proc 3rd International Workshop on Data Analysis in Astronomy*, Erice (June 1988) (Plenum Press, NY).

King, J.R., and Spachis, A.S. 1980: "Scheduling: Bibliography and Review," Int. Journal of Physical Distribution and Materials Management **10**, p. 105.

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.*

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).

Smith, S., Fox, M., and Ow, P. 1986: "Constructing and Maintaining Detailed Construction Plans," AI Magazine, Fall 1986, p. 45