Spike: AI Scheduling for Hubble Space Telescope After 18 Months of Orbital Operations

Mark D. Johnston
Space Telescope Science Institute
3700 San Martin Drive,
Baltimore, MD 21218 USA
johnston@stsci.edu

Abstract

This paper is a progress report on the Spike scheduling system, developed by the Space Telescope Science Institute for long-term scheduling of Hubble Space Telescope observations. Spike is an activity-based scheduler which exploits AI techniques for constraint representation and for scheduling search. The system has been in operational use since shortly after HST launch in April 1990. Spike has been adopted for several other satellite scheduling problems: of particular interest has been the demonstration that the Spike framework is sufficiently flexible to handle both long-term and short-term scheduling, on timescales of years down to minutes or less. We describe the recent progress made in scheduling search techniques, the lessons learned from early HST operations, and the application of Spike to other problem domains. We also describe plans for the future evolution of the system.

1 Introduction

Efficient utilization of expensive space-based observatories is an important goal for NASA and the astronomical community: the cost of facilities like Hubble Space Telescope (HST) is enormous, and the available observing time is much less than the demand from astronomers around the world. The Spike scheduling system was developed by the Space Telescope Science Institute starting in 1987 to help with this problem. The aim of Spike is to allocate observations to timescales of days to a week, observing all scheduling constraints, and maximizing preferences that help ensure that observations are made at optimal times. Spike has been in use operationally for HST since shortly after the observatory was launched in April 1990.

Although developed specifically for HST scheduling, Spike was carefully designed to provide a general framework for similar (activity-based) scheduling problems. In particular, the tasks to be scheduled are defined in the system in general terms, and no assumptions about the scheduling timescale were built in. The mechanisms for describing, combining, and propagating temporal and other constraints and preferences were designed to be general. The success of this approach has been demonstrated by the application of Spike to the scheduling of other satellite observatories: changes to the system are required only in the specific constraints that apply, and not in the framework itself.

In the following we first provide a brief description of the HST scheduling problem and of the Spike scheduling framework. We then discuss some of the experience gained with the system since the start of HST flight operations. This is followed by a description of the changes required to adapt Spike to other satellite scheduling problems. We conclude with some comments on the implementation of Spike, and on our plans for future work.

2 Overview of HST Scheduling

HST scheduling is a large problem: some 10,000 to 30,000 observations per year must be scheduled, each subject to a large number of operational and scientific constraints. Most of the operational constraints arise from the low earth orbital environment of the telescope. With an orbital period of about 96 minutes, potential targets are only visible for a portion of each orbit before they are occulted by the earth. There are constraints due to guide star availability, avoiding the earth's radiation belts, and stray light from the sun, moon, or bright earth. There are also constraints arising from thermal and power considerations, which tend to restrict the allowable attitude of the satellite at different times during the year. Scientific constraints are specified by astronomers when they define the exposures to accomplish their scientific goals. These frequently take the form of minimum exposure times, temporal relationships among exposures (before, after, grouped within some time span, separated by some minimum and/or maximum interval, etc.). Astronomers may also constrain the state of the telescope in other ways, e.g. by requiring exposures when HST is in earth shadow (to exclude scattered earthlight), by specifying the orientation of the telescope, or by configuring one of the six instruments in a particular mode. A recent change to the HST ground systems now permits the scheduling of two instruments for simultaneous operation: this is expected to significantly increase the amount of useful data taken by the telescope.

Because of the design of the telescope and ground system, nearly all HST activities must be scheduled in detail in advance. The detailed schedule specifies what commands will be executed by the onboard computers, and when communications contacts will be available for uplinking commands and downlinking data. Real-time interaction by observers is limited essentially to small pointing corrections to place targets accurately into the proper instrument aperture.

Scheduling HST has been divided into two processes: the first is long-term scheduling, which allocates observations to week-long time segments over a scheduling period of a year or more in duration. This is the responsibility of the Spike system. Individual weeks are then scheduled in detail by the Science Planning and Scheduling System (SPSS),

which orders observations within the week and generates a detailed command sequence for the HST control center at NASA Goddard Space Flight Center. Further details on HST scheduling may be found in [1,2].

3 Spike and HST Long-Term Scheduling

HST observing programs are received at STScI in machine-readable form over national and international computer networks. They are then translated by an expert system called Transformation [3] into a form suitable for scheduling. The Transformation system collects exposures into "scheduling units" which are collections of exposures to be executed contiguously. Transformation makes use of the Spike temporal constraint mechanism to collect and propagate temporal constraints: these are made path-consistent and saved in files along with the scheduling unit definitions. Spike takes the saved scheduling units and derives scheduling constraints and preferences for them, based on operational and scientific factors such as those described above. Spike then determines an allocation of scheduling units to weeks which satisfies all hard constraints and as many soft constraints as possible. Constraints from different sources are combined using a weight-of-evidence mechanism generalized to cover a continuous time domain, as described in detail elsewhere [4]. The result is a set of "suitability functions" which indicates goodness over time for each scheduling unit, and also indicates times when a scheduling unit cannot be scheduled due to violations of strict constraints. Most of the HST-specific scheduling details go into the definition of the suitability functions, which, for long-term scheduling, are defined at a high level of abstraction and relatively coarse time granularity. More details about Spike constraint representation and manipulation may be found in [5].

Spike treats schedule construction as a constrained optimization problem and uses a heuristic repair-based scheduling search technique. An initial guess schedule is constructed, which may have temporal or other constraint violations as well as resource overloads (in fact, given that HST observing time is intentionally oversubscribed by about 30%, it is known ahead of time that there is no feasible schedule that can accommodate all the requested observations). Repair heuristics are applied to the initial guess schedule until a preestablished level of effort has been expended. At that point observations are removed to eliminate remaining constraint violations, until a feasible schedule remains. 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. The heuristic repair method is fast, and typically many runs are made and the best schedule is adopted as a baseline. The Spike algorithm has desirable "anytime" characteristics: 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.

The repair heuristics used by Spike are based on a very successful neural network architecture developed for Spike [6,7] and later refined into a simple symbolic form [8] which has made the neural network obsolete. 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 good initial guesses [9]. We have found that further improvement comes 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. Spike permits different constraints to have different conflict weights, which can be used to cause the repair of the most important constraints first; in practice, however, all constraints have so far been given the same weight. Both hillclimbing and backtracking repair procedures have been tried, but hillclimbing has been shown to be the most cost-effective on problems attempted to date.

The choice of a good initial guess is important for repair-based methods, and to this end we have conducted experiments on different combinations of variable and value selection heuristics to find the "best" combination. Over a thousand combinations of heuristics were tried by making multiple runs on sample scheduling problems. The adopted initial guess heuristic 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 assigned, with ties broken by maximum preference as derived from suitability functions.

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 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 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.

Spike provides support for rescheduling in several ways. Two worth mentioning in particular are 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 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 [10], but much more work remains to be done in this area.

Hillclimbing repair methods like the one used in Spike have much in common with simulated annealing techniques such as described by Zweben et al.[11]. One of the open research issues is which technique has an advantage on which types of problems.

4 The Experience of HST Operations

Shortly after HST was launched it was discovered that the telescope main mirror had been figured incorrectly, resulting in lower resolution than anticipated. This has not only limited the scientific usefulness of HST (although it still remains far superior to any ground-based telescope), it has also greatly disrupted the scheduling process. Observing plans made years in advance of launch have had to be revised, leading to a shortage of ready-to-schedule observing programs and thus reducing the efficiency with which schedules could be generated. This problem still affects ongoing operations, and as a result Spike has only once been used to generate a true long-term schedule. Instead, Spike is used routinely to identify observations to place in the schedule approximately two months into the future. As the characteristics of the telescope and instruments have become better understood, the pool of observing programs has been growing: the second round of open proposal selection will be completed in December 1991, and we anticipate that by the Spring of 1992 a sufficient pool will exist to permit long-range planning as originally expected. NASA is now planning a servicing mission to correct the HST optics in early 1994.

The most significant lesson learned since launch, however, is the impact of high levels of change on the planning and scheduling systems. Instead of the anticipated level of about 10% of proposals changing, the actual rate of change has been closer to 100%. While some of this change is clearly attributable to the discovery of HST's spherical aberration, many other factors have contributed as well: nearly every instrument on the telescope has demonstrated unexpected behavior in one form or another, and each has led to revisions in observing plans to compensate. The net effect is that change is the norm, not the exception, to the extent that stress has been high on the software systems and on the people who operate them. The problem stems from the fact that an observing program may consist of many hundreds of exposures, which can all be at different stages of the scheduling pipeline. If an observing program is changed, users must back up to the beginning of the process for that program, thus work done on the previous version is potentially wasted. Alternatively, a new observing program can be created to describe the changed portions of the original one, but then keeping track of active and obsolete portions of the original is required.

If there is any recommendation to be made to developers of future systems like those for HST, it is to build in the expectation of change from the outset [12]. Even though the initial cost will be higher, the operational costs will be significantly lower.

Spike and the other HST ground systems have been exercised several times on "targets of opportunity" --- programs to be scheduled and executed on an crash basis. Turnaround has been as short as a few days, which is well within the pre-launch expectations. One such target of opportunity program took the pictures of the dramatic storm on Saturn in December 1990, which were subsequently made into a time-lapse movie.

5 Hierarchical and Short-Term Scheduling

Spike has been adopted for scheduling three future astronomical satellite missions:

- the Extreme Ultraviolet Explorer (EUVE), an ultraviolet telescope built and operated by UC Berkeley and Goddard Space Flight Center,
- ASTRO-D, a joint US-Japan X-ray telescope, and
- XTE, the X-ray timing Explorer (MIT/GSFC) to study time-variability of X-ray sources.

The adaptation of Spike for these problems has led to the successful demonstration of 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 and XTE, 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:

- 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 belts
- 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.
- an interface between different hierarchical levels, by which a long-term schedule constrains times for short-term scheduling and conversely
- 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, and the schedule search techniques, apply directly to both long-term and short-term scheduling. Figure 1 illustrates the application of Spike to short-term scheduling for a sample of X-ray targets such as might be observed by ASTRO-D or XTE. Note that several observations are broken to fit around occultations 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 (see, e.g. [13]).

EUVE is unusual in that it makes long (2-3 day) observations, in contrast to HST, XTE, 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 days* of observing

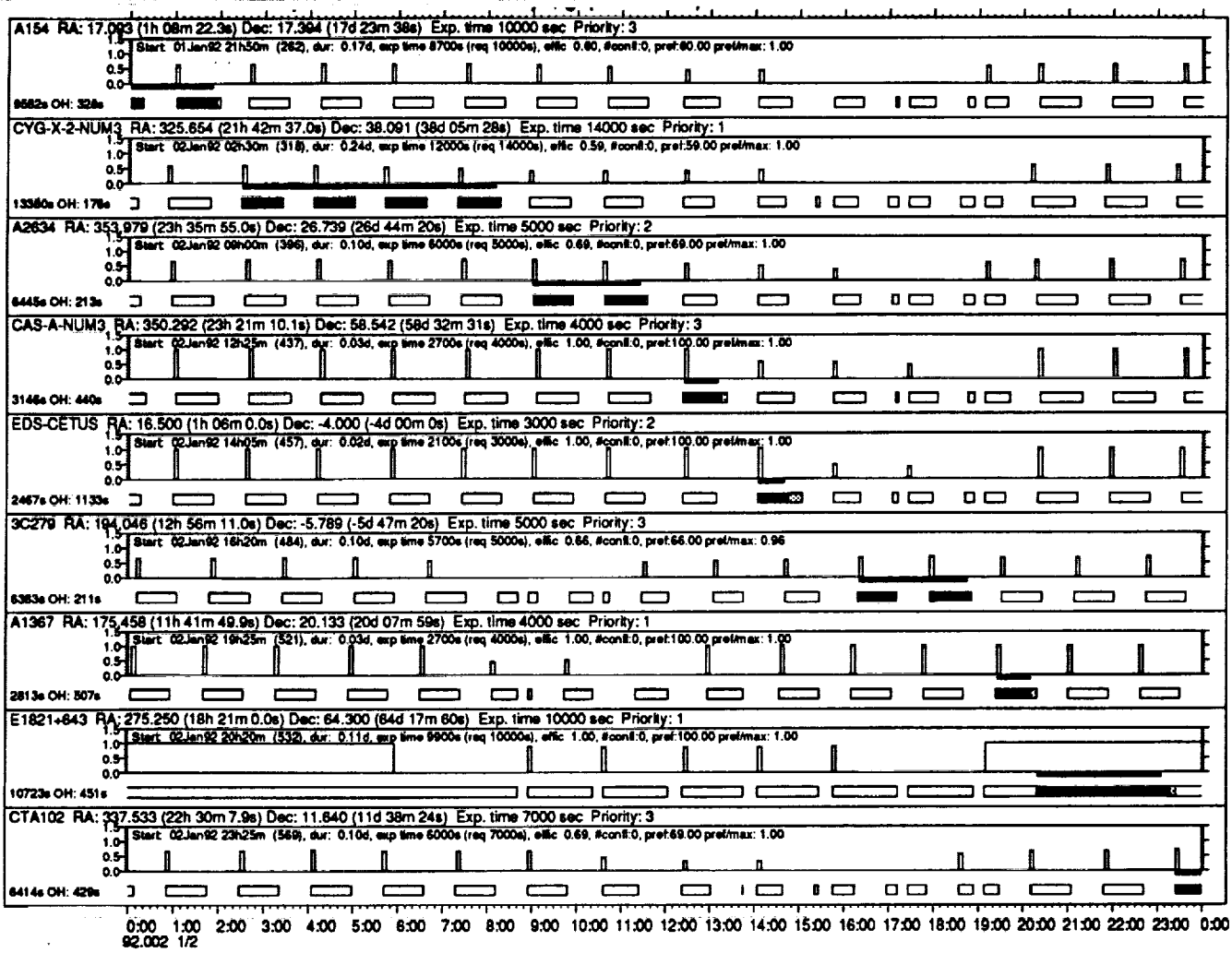

Figure 1: An example of Spike output on short-term scheduling of astronomical observations. Shown is a 24-hour portion of a 7-day schedule. The start-time suitability for each exposure is plotted as the upper graph, with interruptions due to target blockage by the earth and by satellite passage through 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.

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

6 Spike Implementation

The implementation of Spike started in early 1987 and was initially based on Texas Instruments Explorers as the hardware and software environment. The Spike graphical user interface was implemented in KEE CommonWindows (Intellicorps, Inc.), but the remainder of the system (about 40,000 lines of code) used only Common Lisp and the Flavors object system. At HST launch, STScI had a complement of 8 TI Explorers and microExplorers used for Spike operation, development and testing.

Since the initial development of Spike began there has been a great deal of evolution in Lisp hardware and software. A significant amount of effort has gone into modifying the system to keep current with these changes. In late 1991 we are in the process of moving from Explorers to Sun SparcStation IIs as the primary operations and development workstation. All of the Flavors code has been automatically converted to the Common Lisp Object System. The Lisp used on the SparcStation is Allegro Common Lisp from Franz Inc. Allegro CL supports a version of CommonWindows based on X-windows, and so the user interface continues operate on Unix platforms as it did on the Explorers. We are presently investigating the use of alternative window systems, and have prototyped the use of CLX, CLIM, and Motif for the user interface (the latter is based on the publicly available GINA/CLM). We expect to see a complete redesign of the user interface in the next year. Spike can also generate high-resolution Postscript versions of schedules and constraints; one example of this is shown in Figure 1.

Updating Spike for new Lisp language features has not been difficult. There are, however, plans to remove some features that were developed for Spike which have since become part of the language (such as a logical filename mechanism). At present there are no plans to convert any of the system to C or C++.

7 Future Directions

Several significant enhancements to Spike are planned over the next year. One of these, a rewrite of the graphical user interface, has already been mentioned above. Another enhancement deals with tracking the status of HST observing programs and exposures. All scheduled programs pass from the proposal entry system through Spike, while feedback on scheduling and execution status is received by Spike both from SPSS and from the HST data analysis pipeline. This provides information to Spike users which forms the basis for rescheduling decisions. We plan to integrate this data into a relational database, along with additional information from the HST optical disk data archive, which will provide a central source of information on the status of all HST observations.

We are also planning several systematic studies of the Spike scheduling search heuristics to see what further improvements can be made, either in performance or in quality of schedule. These will include the initial guess, repair, and deconflict strategies. We also plan to investigate whether the use of short-term scheduling on the HST observations can improve the quality of the long-term schedule sent to SPSS. There are, however, no plans to have Spike do the final short-term scheduling for HST, due to the extreme cost of integration with the existing telescope and instrument commanding software which generates the command sequences for the spacecraft.

8 Conclusions

The Spike system has performed as planned in the first 18 months of HST operations. The success of Spike helps demonstrate the utility of AI technology in NASA flight operations projects. The flexibility of Spike has been demonstrated by adapting it for several other missions, and by integrating long-term and short-term scheduling at different hierarchical levels of abstraction in the same constraint representation and scheduling search framework.

Acknowledgments: The author wishes to thank Dr. Glenn Miller and the present and former members of the Spike development team (Jeff Sponsler, Shon Vick, Tony Krueger, Dr. Mark Giuliano, and Dr. Michael Lucks) for their efforts which have led to the successful deployment of Spike. The patience and support of the Spike user group, led by Dr. Larry Petro, chief of the STScI Science Planning Branch, has been much appreciated. Stimulating discussions with Dr. Steve Minton, Monte Zweben, Don Rosenthal, and Hans-Martin Adorf are gratefully acknowledged. The EUVE project at UC Berkeley helped with the initial Unix port, and the ASTRO-D and XTE staff at MIT have continued to help push the system in new and fruitful directions. The Space Telescope Science Institute is operated by the Association of Universities for Research in Astronomy for NASA.

References

[1] Miller, G., Rosenthal, D., Cohen, W., and Johnston, M.D. 1987: "Expert System Tools for Hubble Space Telescope Observation Scheduling," in *Proc. 1987 Goddard Conf. on Space Applications of Artificial Intelligence*; reprinted in *Telematics and Informatics* 4, p. 301 (1987).

[2] Miller, G., Johnston, M.D., 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 Conf. on Space Applications of Artificial Intelligence*; reprinted in *Telematics and Informatics* 5, p. 197 (1988)

[3] Gerb, A. 1991: "Transformation Reborn: A New Generation Expert System for Planning HST Operations", in *Proc. 1991 Goddard Conf. on Space Applications of Artificial Intelligence*, ed. J.L. Rash, NASA Conf. Publ. 3110 (Greenbelt: NASA), pp. 45-58; to be reprinted in the Dec 1991 issue of *Telematics and Informatics*.

[4] Johnston, M.D. 1989: "Reasoning with Scheduling Constraints and Preferences," Spike Tech. Report 89-2, Jan. 1989.

[5] Johnston, M.D. 1990: "SPIKE: AI Scheduling for NASA's Hubble Space Telescope", M.D. Johnston, in *Proc. Sixth IEEE Conf. on Artificial Intelligence Applications* (Santa Barbara, March 5-9, 1990), (Los Alamitos, CA: IEEE Computer Society Press), pp. 184-190.

[6] Adorf, H.-M., and Johnston, M.D. 1990: "A Discrete Stochastic 'Neural Network' Algorithm for Constraint Satisfaction Problems", H.-M. Adorf and M.D. Johnston, in *Proc. Int. Joint Conf. on Neural Networks* (IJCNN 90), (San Diego, June 17-21 1990), (Piscataway, NJ: IEEE), Vol. III, pp. 917-924.

[7] Johnston, M.D., and Adorf, H.-M. 1991: "Scheduling with Neural Networks - The Case of Hubble Space Telescope", to appear in *Int. J. Computers and Operations Research.*

[8] Minton, S., Johnston, M.D., Philips, A., and Laird, P. 1990: "Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method", in *Proc. of the Eighth National Conf. on Artificial Intelligence* (Boston July 29-August 3, 1990), (Menlo Park, CA: AAAI Press), pp. 17-24.

[9] Minton, S., Johnston, M.D., Philips, A., and Laird, P. 1991: "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling problems", submitted.

[10] Sponsler, J.L., and Johnston, M.D. 1990: "An Approach to Rescheduling Activities Based On Determination of Priority and Disruptivity", in *Proc. 1990 Goddard Conf. on Space Applications of Artificial Intelligence* (Greenbelt, Maryland, May 1-2, 1990), ed. J. L. Rash, NASA Conf. Pub. 3068 (Greenbelt: NASA), pp. 63-74, reprinted in *Telematics and Informatics*, 7, pp. 243-253.

[11] Zweben, M., Davis, E., Stock, T., Drascher, E., Deale, M., Gargan, R., and Daun, B. 1991: "An Empirical Study of Rescheduling Using Constraint-Based Simulated Annealing", submitted.

[12] Miller, G. and Johnston., M.D. 1991: "A Case Study of Hubble Space Telescope Proposal Processing, Planning and Long-Range Scheduling", in *Proc. AIAA Conf. Computing in Aerospace 8*, Oct 21-24, 1991, Baltimore.

[13] Muscettola, N., Smith, S., Cesta, A., and D'Aloisi, D. 1992: "Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture", to appear in *IEEE Control Systems Systems Magazine* 12 Feb. 1992.