N91-22775

Long Range Science Scheduling for the Hubble Space Telescope

Glenn Miller and Mark Johnston
Space Telescope Science Institute[1]
3700 San Martin Dr.
Baltimore, MD 21218

Abstract

Observations with NASA's Hubble Space Telescope (HST) are scheduled with the assistance of a long-range scheduling system (Spike) that was developed using artificial intelligence techniques. In earlier papers, we have described the system architecture and the constraint representation and propagation mechanisms. In this paper we describe the development of high-level automated scheduling tools, including tools based on constraint satisfaction techniques and neural networks. The performance of these tools in scheduling HST observations is discussed.

1. Introduction

Launched by the Space Shuttle in April 1990, NASA's Hubble Space Telescope (HST) has begun its mission of scientific exploration. Despite a problem with the primary mirror, the HST has already taken data of unprecedented quality for many objects, including Pluto, a gravitational lens, a star cluster and a supernova.

Scheduling the HST is an especially challenging problem since a year's observing program consists of tens of thousands of exposures which are coupled by numerous constraints. Constraints which must be considered include proposer specified constraints (e.g. precedence, timing, restrictions on spacecraft orientation), orbital viewing constraints (e.g. Earth occultations and Sun avoidance), and spacecraft power and communications constraints. The Space Telescope Science Institute (STScI) is responsible for conducting the science operations of the HST, including planning and scheduling observations.

A detailed description of the HST planning and scheduling problem, including a discussion of the individual scheduling constraints is given in Miller, et al. (1987). In a subsequent paper (Miller, et al. 1988), we described the initial development of the Spike planning system including the method of representing constraints and propagating scheduling decisions. This paper continues the series by describing high-level automated scheduling tools and the operational experience of using these tools to produce the science schedules for the HST. Section 2 describes the Spike system and its role in the HST planning and scheduling process. Section 3 discusses the automated scheduling tools. The experience of using these tools for HST testing and operations is given in Section 4.

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

2. Overview of the Spike System

An astronomer wishing to observe with the HST submits a scientific observing proposal. Based on the recommendations of a peer review committee, the Director of the STScI selects which proposals are awarded observing time (refer to Miller, et al. 1987 for a description of the proposal selection process). Proposals are assigned either high or supplemental priority. Unless prevented by unforeseen technical problems, all high priority observations will be executed and constitute about 70% of the estimated available observing time. The supplemental proposals form a pool used to fill out the remainder of the schedule and the choice of a particular supplemental proposal is likely to be based on scheduling and operational considerations. The supplemental pool oversubscribes the available time, so there is only a moderate chance that a particular supplemental program will actually be executed.

The scheduling process begins with the submission of approved observing proposals (see Figure 1). Astronomers submit machine-readable proposals to the STScI using the Remote Proposal Submission System (Jackson, et al. 1987). A year's scheduling pool of about 300 proposals comprises tens of thousands of exposures on a few thousand targets.

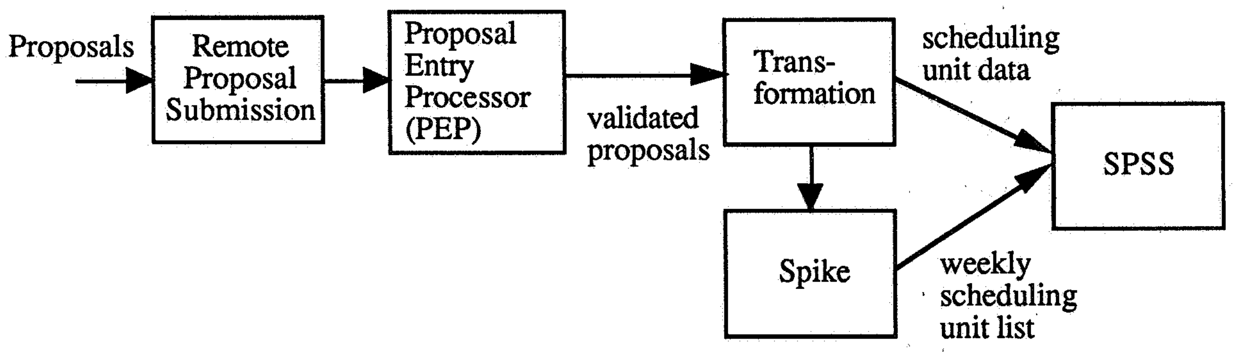

Proposals
Remote Proposal Submission
Proposal Entry Processor (PEP)
validated proposals
Trans-formation
scheduling unit data
SPSS
Spike
weekly scheduling unit list

Figure 1 - Overview of the HST Proposal Processing, Planning and Scheduling Systems. The Remote Proposal Submission System provides proposal preparation tools and accepts proposals via computer. The Proposal Entry Processor (PEP) handles the proposal data. Transformation is used to populate the SPSS system with data. Spike produces a long term plan which is periodically sent to SPSS for short-term scheduling. SPSS returns the results of short-term scheduling to Spike so that the long-term plan can be kept current.

A proposal includes target specifications (position, brightness, etc.) and a list of exposures (target, instrument, operating mode, exposure time, etc.). In order to express scientific constraints on the exposures, a proposal can specify a wide range of properties and interrelationships. For example, exposures may be designated as acquisition or calibration exposures. Some exposures must be executed at particular times or at specific spacecraft roll angles. Ordering and grouping of exposures may be specified as well, and these links may couple exposures separated by many weeks or months. Exposures requiring low background light conditions are identified for execution when HST is in the Earth's shadow. To provide flexibility and a choice of alternative observation plans, exposures may be marked as conditional on some external event (e.g. a ground-based observation) or conditional on the results of other exposures in the proposal. The proposer notifies the STScI when the conditions have been satisfied.

At the STScI, proposal information is contained in the Proposal Entry Processor (PEP) System (Jackson, et al. 1988), which provides tools for entry, editing, evaluation and selection of proposals. If questions or problems are encountered, STScI staff assist the proposer in modifying the proposal.

In order to ensure that the requirements of the proposers and the overall goals of the HST observatory are met, it is necessary to develop a long-range plan. Key considerations for this plan include:

- plan must cover a long time interval (multi-year)
- planning is far in advance of execution, and many constraints can not be predicted in detail in advance
- plan must incorporate a large number of exposures (tens of thousands)
- constraints can couple exposures separated by long time intervals (months to years)
- replanning will be required

A hierarchical approach to scheduling was developed, with the Spike system performing long-range scheduling, and the Science Planning and Scheduling System (SPSS, developed by TRW) performing short-range scheduling. Spike is used to generate multi-year schedules with observations allocated to week-long segments. SPSS does detailed scheduling of observations within a weekly segment.

To be scheduled, the proposal must be cast into a form which can be understood by the SPSS. A PEP proposal expresses the astronomical observations in terms familiar to an astronomer, while the input to SPSS must conform to the design of SPSS, HST and fundamental orbital considerations. This process is called Transformation and is performed by an expert system described by Gerb (1991, see also Rosenthal, et al. 1986). In SPSS, the fundamental data structure is called the **scheduling unit**. Transformation assigns PEP exposures to scheduling units. Transformation performs more than a simple translation of the proposal from PEP to SPSS formats. Transformation chooses implementation scenarios based on the proposer's requirements, orbital conditions and spacecraft constraints.

The Spike system performs long-range scheduling, using Transformation results as input. Spike provides several important features for HST scheduling:

- Constraint representation and propagation mechanism which includes the ability to express human value judgments as well as constraints which can never be violated
- Proposal evaluation tools which allow planners to display observations and constraints on a high-resolution graphics workstation
- Automated and manual scheduling tools
- Window-oriented interface to Spike commands
- Automated tools to track information sent to and received from SPSS

Details of Spike are given in Miller, et al. (1988), Johnston (1990), and Miller and Johnston (1990). We summarize some of the most important features of the Spike system below.

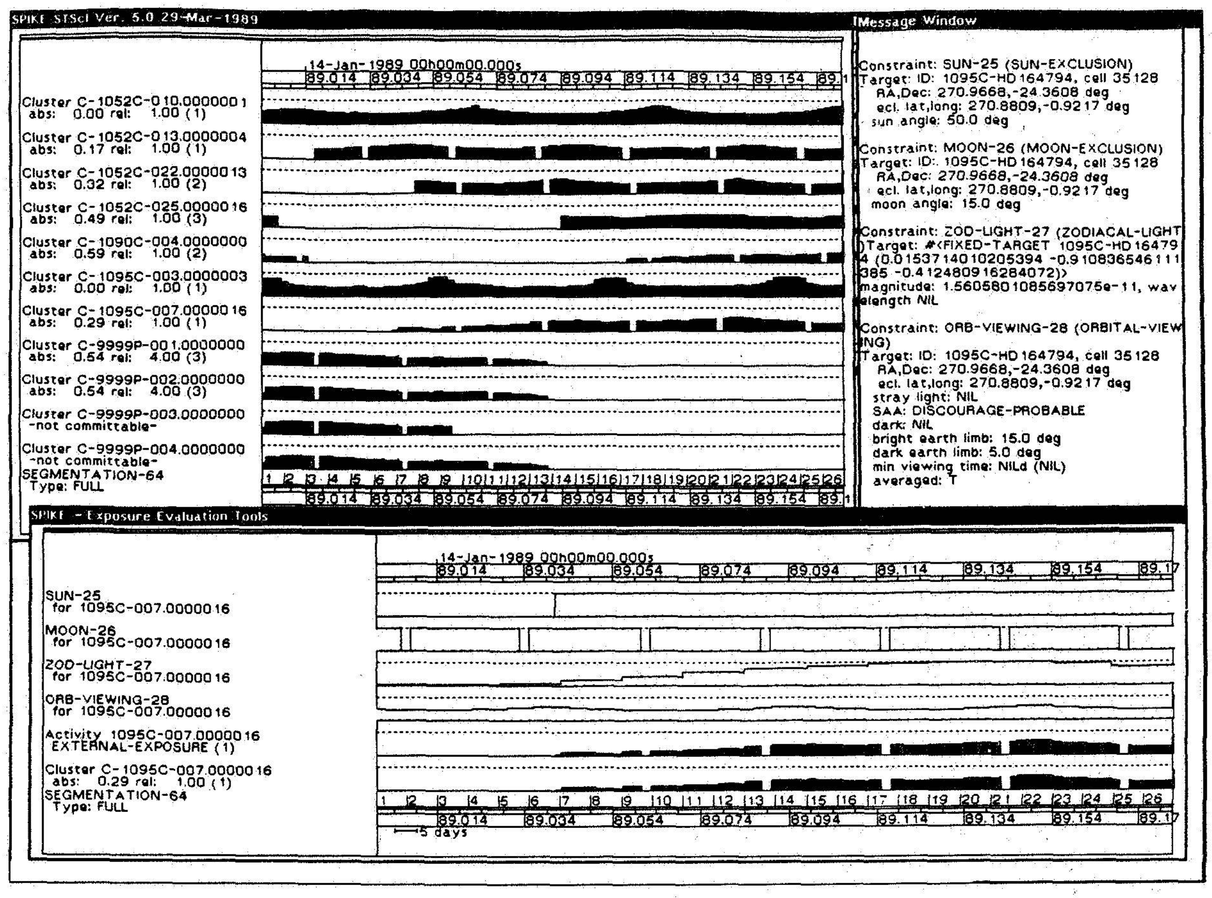
A scheduling **cluster** is the lowest level entity which can be scheduled in Spike. Clusters consist of one or more activities. For the evaluation of single proposals, clusters correspond to scheduling units and the activities are the individual PEP exposures in the scheduling unit. Using graphic tools, a planner can click on a cluster (scheduling unit) and examine its constituent exposures. For scheduling many proposals, the PEP exposure information is discarded and each cluster consists of a single activity which is a scheduling unit. On average, a scheduling unit consists of about 5 PEP exposures. This results in a compression of the amount of activities considered by Spike and a resultant increase in performance.

A **constraint** is any factor that describes when it is possible or desirable to plan an activity. This includes strict constraints ("never point the HST closer than 40^o to the Sun") and preference constraints ("an deviation of spacecraft roll of up to 30^o is sometimes allowable but should be avoided unless necessary"). **Absolute constraints** limit the opportunities for a particular activity and are independent of when other activities are planned (e.g. Sun avoidance, guide star availability). **Relative constraints** describes the relationship between two or more activities and change as activities are fixed in the scheduling process (e.g. if activity B must follow activity A by 90 days or more, then fixing a time for A's execution affects B's allowed times). Constraints are represented in Spike by a **suitability** function. This gives the desirability of starting an activity at a particular time. A suitability of 0 means that a start time is forbidden, while a positive suitability indicates that the activity can begin at that time. A suitability of 1 is defined as the nominal suitability, while suitabilities < 1 (but > 0) indicate less desirable, but allowed, start times. This method of constraint representation allows Spike to represent all constraints in a consistent manner and to provide an efficient means for propagating constraints and the effects of scheduling decisions.

In Spike, a long range plan is called a **planning session.** Within a planning session, time is divided into intervals called **segments.** Segments are simply a convenient means to discretize time, creating time "bins". For long-term planning, this allows a significant reduction in the time dimension of the problem without being artificially limiting. A scheduling cluster can be **committed** to a segment, that is, restricted to start during the time interval of the segment. For HST scheduling, the planning session is several years long and there are typically 52 1 week segments per year. The commitments in these segments are periodically transmitted to SPSS for short-term scheduling and subsequent execution. A planning session can have one or more **resource constraints.** These express limitations on resources consumed by clusters committed to a segment, e.g. total exposure time, data volume and real time interaction.

A Spike user can manually commit scheduling units to segments or invoke automated scheduling tools. Figure 2 shows the proposal evaluation tools display. Long term scheduling is complete when all high priority scheduling units have been allocated to weekly segments and supplemental scheduling units have been allocated to oversubscribe the resources to about 20% above capacity.

Several months prior to execution, the list of scheduling units for a week are transmitted to SPSS. After short term scheduling, SPSS returns to Spike the scheduling status of each scheduling unit, i.e. whether or not it was scheduled, the start time of the observation, and the spacecraft roll angle. Generally, all high priority observations will be scheduled and only supplementals will need to be rescheduled by Spike, however it is expected that some rescheduling of highs will be encountered. After execution of the observations by the HST, data is processed by the Post Observation Data Processing System (PODPS) which provides Spike with the definitive list of observations which were executed. Spike compares this to SPSS-supplied schedule and notes to the user any differences.


SPIKE STScI Ver. 5.0 29-Mar-1989
Message Window
SPIKE - Exposure Evaluation Tools
14-Jan-1989 00h00m00.000s
89.014
89.034
89.054
89.074
89.094
89.114
89.134
89.154
5 days

```
Cluster C-1052C-010.0000001
 abs:   0.00 rel:   1.00 (1)
Cluster C-1052C-013.0000004
 abs:   0.17 rel:   1.00 (1)
Cluster C-1052C-022.0000013
 abs:   0.32 rel:   1.00 (2)
Cluster C-1052C-025.0000016
 abs:   0.49 rel:   1.00 (3)
Cluster C-1090C-004.0000000
 abs:   0.59 rel:   1.00 (2)
Cluster C-1095C-003.0000003
 abs:   0.00 rel:   1.00 (1)
Cluster C-1095C-007.0000016
 abs:   0.29 rel:   1.00 (1)
Cluster C-9999P-001.0000000
 abs:   0.54 rel:   4.00 (3)
Cluster C-9999P-002.0000000
 abs:   0.54 rel:   4.00 (3)
Cluster C-9999P-003.0000000
 -not committable-
Cluster C-9999P-004.0000000
 -not committable-
SEGMENTATION-64
 Type: FULL
```

```
Constraint: SUN-25 (SUN-EXCLUSION)
Target: ID: 1095C-HD164794, cell 35128
   RA,Dec: 270.9668,-24.3608 deg
   ecl. lat,long: 270.8809,-0.9217 deg
  sun angle: 50.0 deg

Constraint: MOON-26 (MOON-EXCLUSION)
Target: ID:. 1095C-HD164794, cell 35128
   RA,Dec: 270.9668,-24.3608 deg
   ecl. lat,long: 270.8809,-0.9217 deg
  moon angle: 15.0 deg

Constraint: ZOD-LIGHT-27 (ZODIACAL-LIGHT
)Target: #<FIXED-TARGET 1095C-HD16479
4 (0.01537140 10205394 -0.910836546111
385 -0.412480916284072)>
magnitude: 1.5605801085697075e-11, wav
elength NIL

Constraint: ORB-VIEWING-28 (ORBITAL-VIEW
ING)
Target: ID: 1095C-HD164794, cell 35128
   RA,Dec: 270.9668,-24.3608 deg
   ecl. lat,long: 270.8809,-0.9217 deg
  stray light: NIL
  SAA: DISCOURAGE-PROBABLE
  dark: NIL
  bright earth limb: 15.0 deg
  dark earth limb: 5.0 deg
  min viewing time: NILd (NIL)
  averaged: T
```

```
SUN-25
 for 1095C-007.0000016
MOON-26
 for 1095C-007.0000016
ZOD-LIGHT-27
 for 1095C-007.0000016
ORB-VIEWING-28
 for 1095C-007.0000016
Activity 1095C-007.0000016
 EXTERNAL-EXPOSURE (1)
Cluster C-1095C-007.0000016
 abs:   0.29 rel:   1.00 (1)
SEGMENTATION-64
 Type: FULL
```

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. The user can create new windows and build new displays dynamically.

ORIGINAL PAGE IS
OF POOR QUALITY

3. Automated Scheduling

The following sections describe the automated scheduling strategies which have been most frequently used in testing and operations: procedural strategies, neural network scheduler and a scheduler motivated by work on constraint satisfaction problems (CSP). We have prototyped other approaches as well. Miller, et al. (1988) describes a rule-based expert system scheduler, while Sponsler (1989) describes a scheduler which uses genetic algorithms to select from a population of neural network schedules.

In any discussion of scheduling techniques it is important to define how the quality of a schedule is measured. One metric which is readily calculable in Spike is "summed suitability" which is the sum over all clusters of the maximum suitability in allowable segments. Before any commitments are made, the summed suitability gives an upper limit to the best possible schedule. Often it is the case that no schedule can actually be this good since the commitment of a cluster to its best segment can force another cluster to be scheduled at a less than ideal time. After commitments are made, the summed suitability measures how well the schedule comes to satisfying the preferences of each observation. Another scheduling metric is the number of clusters which become unschedulable during the scheduling process.

3.1. Procedural Scheduler

The Spike system provides a set of procedural scheduling strategies. These work on the following pattern: From all available clusters and their allowable segments, select one cluster and segment according to some criterion (described below). Commit this cluster to the segment and propagate the effects. Repeat this process until no more clusters can be committed. This is an implementation of the so-called "greedy" algorithm (it is "greedy" since it makes a choice on the basis of what is best for one cluster without regard to how other clusters might be affected). As an example, consider the most suitable cluster-segment strategy: The selection criterion is the average cluster suitability in a segment. Each cluster is queried to find its highest average suitability and the segment in which this occurs. The cluster with the highest suitability of all clusters is committed (with ties broken by taking the first cluster examined). This commitment will usually limit the scheduling choices for the remaining clusters, either by constraints between clusters or by resource constraints. This process is repeated until all clusters are scheduled or all remaining clusters are unschedulable.

Other strategies include the most absolute constrained cluster and the most relative-absolute constrained cluster. In the former, the cluster of choice at each step is the one with the smallest total interval duration of non-zero suitability (i.e. the one with the fewest scheduling choices). In the latter, the cluster selected is the one with the largest number of links to other clusters and (in the event that two or more clusters have the same number of links) the smallest total non-zero suitability interval.

The advantage of these strategies is the speed of execution. The main disadvantage, which is well known, is that such simple strategies can lead to very poor schedules (e.g. a large number of unschedulable clusters and clusters scheduled at times of low suitability).

Spike provides a means to control the clusters examined by these strategies. The user can define a focus set which restricts the attention of the procedural strategies to a subset of all clusters. The focus set can be a user-specified set of clusters, or can be defined dynamically according to certain criteria, e.g. priority, absolute or relative constrainedness.

3.2. Neural Network Scheduler

In order to provide a more intelligent and flexible approach to scheduling than the procedural strategies, a scheduler based on neural networks was implemented. Neural nets are motivated by models of how biological nervous systems work and have been applied to a variety of problems including pattern recognition, classification, memory and speech understanding (see Tank and Hopfield 1987 for a general introduction to neural nets).

In Spike, a modified Hopfield neural net is used (Adorf and Johnston 1990). The network can be visualized as a rectangular matrix, where the columns represent segments (time) and the rows represent scheduling clusters (see Figure 3). A particular matrix element (neuron) represents the potential commitment of a particular cluster to a particular segment.

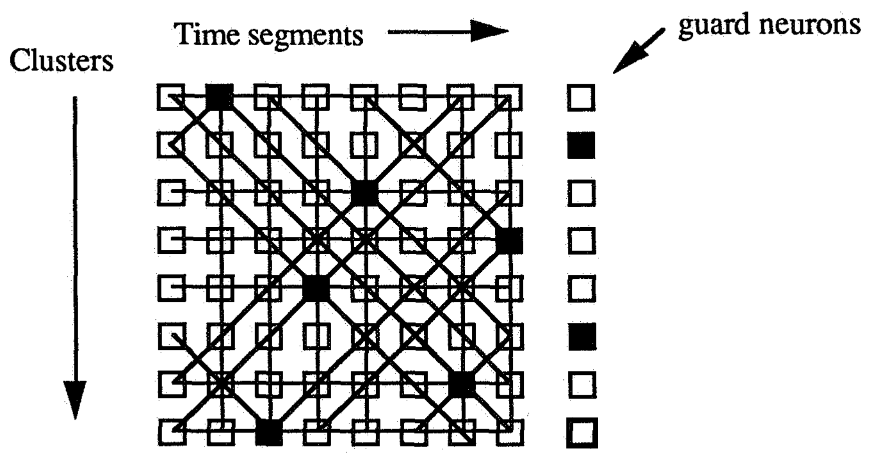


Figure 3 - Neural Network Scheduler.

The output of a neuron is determined by the sum of all inputs. If the sum of the inputs ≤ 0 then the output is zero (the neuron is in the "off" state), otherwise the output is 1 (the neuron is "on" and signifies the commitment of a cluster to a segment). Inputs come from the following:

absolute suitability: This term expresses the absolute suitability of a cluster as a function of time. Suitabilities as represented in Spike are combined by multiplication. Inputs in the neural network are combined by addition, so that the weight in the network is simply the logarithm of the average suitability in the segment. This is a constant value and does not change as commitments are made. Zero suitability (or any suitability below a user-specified tolerance) is converted to an inhibitory (negative) weight that cannot be overcome by any amount of positive contributions. This preserves the fundamental constraint representation of Spike suitabilities.

relative constraints: For all pairs of clusters A and B which are related, all neurons in A's row are connected to all neurons in B's row. The weight of the connection is determined by the suitability of one cluster given the commitment of the other to a particular segment. As with the previous term, the weight is the logarithm of the suitability, with a large inhibitory value used for suitabilities near zero.

resource constraints: When a cluster is committed to a segment, the amount of resources consumed in that segment is updated and any uncommitted clusters which require more than the available resource are given a large inhibitory input.

guard neuron: This is discussed below.

The Spike planning session contains the information necessary to determine the absolute and relative suitability terms, thus the neural network does not need to undergo a "training" phase where it "learns" the effect of one cluster on another and adjusts the weight between neurons. (One step of Spike's constraint propagation is to explicitly express all constraints between pairs of clusters. For example, if the proposer specifies a constraint between A and B and between B and C, Spike derives the induced constraint between A and C. We have found that the increase in performance of the scheduling system is well worth the additional computation and computer memory.) The Spike suitability information is saved to a file so that the computation is performed only once and the network can be rapidly initialized from these files.

The network searches for a solution as follows: A row is selected at random and within that row the neuron which is most out of line with its input is selected (i.e. the neuron is on but has an inhibitory total input or the neuron is off but has positive total input). The neuron's state is changed (flipped) to be consistent with the input. The corresponding guard neuron is examined and changed if necessary. This process is repeated until it converges (i.e. the state of all neurons is consistent with their inputs) or a user-defined iteration limit is exceeded.

A Hopfield neural net is guaranteed to converge to a solution, but unfortunately this includes solutions in which not all clusters are committed to segments (in fact a schedule with zero commitments is a solution). The guard neurons are a novel feature of the Spike neural network scheduler and their purpose is to drive the network to committing all clusters. The guard neurons work as follows: For each row (cluster) there is a guard neuron which is connected to all other neurons on that row. When all neurons in the row are off (the cluster is not committed anywhere), the guard neuron is on and applies a positive input to all other neurons on the row, which tends to force them on. When any one neuron in the row is on (the cluster is committed to a segment), the guard neuron applies a large negative input to all other neurons on that row. The addition of the guard neurons, while necessary, breaks the symmetry of the Hopfield model so that convergence is no longer guaranteed. The software monitors the number of steps in the convergence process (i.e. neuron state changes) and halts when a user-specified limit is exceeded. In practice the network runs fast enough that large numbers of trial schedules can be evaluated and compared.

3.3. Constraint Satisfaction Formulation

The neural network system described in the previous section was very effective at finding solutions to scheduling and other problems. Investigations into the source of this good performance (Minton, et al. 1990) led to the development of a scheduler which is based on constraint satisfaction problems. The CSP scheduler improved on the performance of the neural network scheduler while allowing a greater control of the scheduling strategies.

A constraint satisfaction problem (CSP) can be described as follows: Consider N variables v_1, v_2, ..., v_N. Each of these variables has a domain of allowed values. The domain is a set of discrete (not continuous) values. There is a set of constraints which limit the allowed values for a variable v_i based on the values of other variables. The problem is to assign a consistent set of values for all variables such that there are no conflicts between constraints and value assignments (i.e. all constraints are satisfied). In the case of HST long-range scheduling, each scheduling unit is represented by a variable and the values allowed for the variables are the segment (week) in which the scheduling unit is to be executed.

The concept of conflicts is central to the CSP scheduler. A conflict occurs when a value for a variable is prohibited by some constraint. A value may have zero conflicts (in which case is consistent with the constraints and current assignments of other variables) or it may have 1 or more conflicts (possibly many more!). The number of conflicts can never be negative. Minton, et al. (1990) found that the key to the neural network's performance was in the way it chose which neuron to update - it chooses the neuron which is most inconsistent with its input and assigns it a value which minimizes the number of other variables that will be in conflict. The CSP scheduler records the number of conflicts on all values of each variable.

The CSP scheduler operates in two distinct phases:

1. Generate an initial guess at a solution (i.e. assign values to all variables). Since this is a guess at a solution, there can be conflicts on the values of some variables.

2. Repair this guess until a solution is found, i.e. until all variables are assigned and none of the assigned values have conflicts.

To be useful, the initial guess must be fairly easy to calculate and must be relatively close to the true solution so that the repair method can find the solution. If it is too computationally intensive to generate the guess, the perhaps some other search technique should be used. If the initial guess is too far from a solution, then the repair method may not find a solution quickly. A useful initial guess method that has been used is to select a variable at random and assign it a value which will result in the minimum number of conflicts with other variable values. A typical repair method selects a variable with the largest number of conflicts on its assigned value and reassigns it a value which has the minimum number of conflicts.

4. Experience

This section describes the results of using the Spike system to support the science operations of the HST. Section 4.1 concentrates on the experience gained from scheduling, while Section 4.2 discusses the software engineering aspects of this project.

4.1. Planning and Scheduling

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 consist 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 is planned to end in the summer of 1991.

The Spike system was first used to support HST scheduling for Cycle 0. The timeline for SV observations was established by NASA. The STScI Science Planning Branch used Spike to verify this timeline and to schedule GTO observations during weeks when time was available. Spike (and Transformation) was able to uncover a number of scheduling problems with proposals. Often these problems were due to an inadvertent specification by the proposer of inconsistent requirements in the PEP proposal. Although the PEP system performs syntactic checking on proposal information, Spike and Transformation are the first systems which can detect problems related to planning and scheduling. (In particular, accurate instrument overhead times and orbital viewing conditions are calculated by Transformation and Spike and these can reveal problems with the proposal.) We are currently investigating how to incorporate such checks in the PEP Remote Proposal Submission System software which is used by proposers to prepare the proposals. Not only will this provide proposers with immediate feedback of certain classes of problems, but it will lessen the number of delays in the scheduling process due to proposal changes.

Scheduling of the GTO proposals in Cycle 0 was mostly manual. Planners would display individual proposals on the timeline displays and choose times of high suitability for observations. The various procedural commitment strategies were also used in conjunction manual commitments.

The neural network and CSP schedulers will be used for Cycle 1 scheduling. Currently they have been used only in system tests. These tests have been been extensive in that they have included nearly all of the existing proposal pool. Running on a TI Explorer, a multi-year schedule with 1000 scheduling units (corresponding to about 5000 PEP exposures) can be created in less than one hour.

4.2. Software Engineering

The Spike system (as well as Transformation) has been developed using "artificial intelligence" techniques and is implemented in Common Lisp. The Lisp environment provides powerful tools to the developer and is clearly capable of supporting large, computation-intensive applications such as HST scheduling.

Careful attention has been paid to the Common Lisp standard and portability. This has allowed some or all of Spike to be run on the TI Explorer, TI MicroExplorer, Sun, Macintosh, Symbolics, and Vax. No source code changes are required to run on different machines, and machine-specific compiler directives are rare (mostly in code accessing the file system). A practical and important benefit of portability has been the ability to incorporate new hardware into our system. Initial development was on the Explorer and MicroExplorer, but within the last year we have added Sun Sparcstations (running Allegro Common Lisp from Franz, Inc.) to development and operations. The Allegro Lisp environment, though not as mature as the TI workstations, is acceptable and the performance of the Sparcstation is impressive. Currently there are 9 Lisp workstations used to support operations, testing and development. A Sun 3 is used as a file server and gateway to other networks. Communications between machines and file access is via TCP/IP for Explorers and NFS for the Sparcstations. We implemented a batch processing utility which allows individual workstations to operate from a central queue. Not only does this allow for efficient use of the operational workstations, but it allows for the development workstations to be pressed into service at times of peak demand.

Common Lisp and the Common Lisp Object System (CLOS) have standardized major portions of the Lisp environment, but there has not yet emerged a high-level standard for windowing systems (X-windows is a standard, but is at a low level). This is a serious obstacle to the development of portable code. This has been recognized by the Lisp community and a number of standards are competing for acceptance.

Parts of the Spike system were prototyped in expert system shells, including KEE (from IntelliCorp) and ART (from Inference), but the current system only uses IntelliCorp's Common Windows subsystem. Why the expert system shells are not used more seems to be the result of two factors. First, Common Lisp now provides some of the features of these shells, in particular, CLOS provides object-oriented programming. Second, Spike does not make use of the paradigms supported by the shells, e.g. forward and backward chaining rules, truth-maintenance and hypothetical reasoning.

Since many requirements of the problem originally ill-defined, a rapid prototyping methodology is an important component of our approach. Users and developers worked very closely on development and operational issues.

5. Summary

In this paper we have described the Spike scheduling system which supports long-term science scheduling of the HST. This system is capable of producing long-term schedules consisting of many observations. Three automated schedulers were presented: procedural, neural network and constraint satisfaction based. The latter two schedulers provide a sophisticated and efficient approach to searching for optimal schedules. Lessons learned from HST scheduling and software development were presented and can be applied to other scheduling problems.

We would like to thank the following people for their contributions to the Spike system: Jeff Sponsler, Shon Vick, Anthony Krueger, Michael Lucks, Andrew Gerb, and Mike Rose.

References

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 (IJCNN 90), Piscataway, NJ: IEEE, Vol. III, pp. 917-924.

Gerb, A. 1991, "Transformation Reborn: A New Generation Expert System for Planning HST Operations", Proceedings of the 1991 Goddard Conference on Space Applications of Artificial Intelligence, 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 Science Operations", Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence, ed. J. Rash and P. Hughes, NASA Conference Publication 3009.

Johnston, M., and Miller. G. 1990, "Artificial Intelligence Scheduling for NASA's Hubble Space Telescope", Proceedings of the Fifth Annual Expert Systems in Government Conference,, ed. B. Silverman, V. Huang and S. Post Los Alamitos, IEEE Computer Society Press, pp. 33-39.

Johnston, M. 1990, "SPIKE: AI Scheduling for NASA's Hubble Space Telescope", Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications" Los Alamitos, CA: IEEE Computer Society Press, pp. 184-190.

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", Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence, ed. J. Rash and P. Hughes, NASA Conference Publication 3009, reprinted in Telematics and Informatics, 5, pp. 197-212.

Miller, G., Rosenthal, D., Cohen, W. and Johnston, M. 1987, "Expert Systems Tools for Hubble Space Telescope Observation Scheduling" in Proceedings of the 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics, reprinted in Telematics and Informatics, 4, 301-311.

Minton, S., Johnston, M., Philips, A and Laird, P 1190, "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, pp. 17-24.

Rosenthal, D., Monger, P., Miller, G. and Johnston, M. 1986, “An Expert System for Hubble Space Telescope Ground Support”, Proceedings of the 1986 Goddard Conference on Space Applications of Artificial Intelligence and Robotics.

Sponsler, J. 1989, "Genetic Algorithms Applied to the Scheduling of the Hubble Space Telescope", Proceedings of the 1989 Goddard Conference on Space Applications of Artificial Intelligence, ed. J. Rash, NASA Conference Publication 3033, reprinted in Telematics and Informatics, 6, pp. 181-190.

Tank, D. and Hopfield, J. 1987 “Collective Computation in Neuronlike Circuits”, Scientific American, December 1987, pp 104-114.