# Knowledge Based Tools for Hubble Space Telescope Planning and Scheduling: Constraints and Strategies

Glenn Miller[1]
Astronomy Programs, Computer Sciences Corporation

Mark Johnston, Shon Vick and Jeff Sponsler
Space Telescope Science Institute[2]

Kelly Lindenmayer[1]
Astronomy Programs, Computer Sciences Corporation

3700 San Martin Dr.
Baltimore, MD 21218

## Abstract

The Hubble Space Telescope (HST) presents an especially challenging scheduling problem since a year's observing program encompasses tens of thousands of exposures facing numerous coupled constraints. This paper discusses recent progress in the development of planning and scheduling tools which augment the existing HST ground system. General methods for representing activities, constraints and constraint satisfaction, and time segmentation have been implemented in a scheduling testbed. The testbed permits planners to evaluate optimal scheduling time intervals, calculate resource usage and to generate long and medium range plans. Graphical displays of activities, constraints and plans are an important feature of the system. High-level scheduling strategies using rule based and neural net approaches have been implemented.

---

[1] Staff member of the Space Telescope Science Insitute

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

## 1. Introduction

NASA's Hubble Space Telescope (HST) will provide important new capabilities for astronomical observation. HST will orbit the Earth above the distorting effects of the atmosphere, allowing unprecedented angular resolution, sensitivity and wavelength coverage using six scientific instruments (for more details, refer to Hall 1982). The Space Telescope Science Institute (STScI) is responsible for conducting the science operations of the HST, including planning and scheduling observations. Planning and scheduling HST observations is a particularly challenging problem for several reasons: A year's observing program will consist of a large number of activities (about 30,000 exposures of approximately 3,000 celestial targets). Many constraints must be satisfied, including proposer specified constraints (e.g. timing, precedence), orbital viewing constraints (e.g. Earth, Sun and Moon occultations), and spacecraft power and communications constraints. Detailed schedules must be prepared days to weeks in advance in order to obtain communications contacts and to generate spacecraft computer command loads. Schedules must be repaired to account for disruptions due to unpredictable astronomical events (e.g. a supernova) and due to spacecraft anomalies.

In this paper we describe the Science Planning Interactive Knowledge Environment (Spike) System planning and scheduling tools which are being developed at the STScI to augment existing ground system scheduling capabilities. Initial work has resulted in the development of an "Exposure Evaluation Tools" testbed which provides both manual and automated tools for long term scheduling.

The next section provides a brief overview of the HST planning and scheduling problem. The reader should consult Miller, et al. (1987) for more details. Section 3 describes the problems of long term planning for HST. Section 4 focuses on Spike's Exposure Evaluation Tools and how they provide the necessary functions for long term scheduling. Automated strategies for scheduling are the topic of section 5. The development methodology, including the use of artificial intelligence techniques and rapid prototyping are discussed in section 6. The interfaces between the Spike system and other ground system components are described in section 7. The last section discusses some planned extensions of the current system and the application of these tools to other scheduling problems.

## 2. Overview of HST Planning and Scheduling

An astronomer wishing to observe with the HST submits a scientific observing proposal. Based on the the recommendations of a peer review committee, the Director of the STScI selects which proposals are awarded observing time and assigns each proposal to one of three priority categories: high, medium and supplemental. Unless precluded by unforeseen technical problems, all high and medium proposals will be executed. The difference between the high and medium categories is that medium priority observations may be rescheduled to accommodate rescheduling of a high priority observation. High and medium priority proposals will consume about 70% of the estimated 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.

At this point, the scheduling process begins. A year's scheduling pool of about 300 proposals comprises tens of thousands of exposures on a few thousand targets. Proposal information is contained in the Proposal Entry Processor (PEP) System (Jackson, et al. 1988), which provides tools for entry, editing, evaluation, selection and transformation of proposals. 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 inter-relationships. 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.

In addition to the constraints imposed by the proposer's scientific program, there are a large number of other constraints which must be considered. Many orbital factors exert a strong influence on scheduling: targets are occulted (blocked) by the Earth for up to $40^m$ each $95^m$ orbit. Observations cannot be taken during HST's passage through the South Atlantic Anomaly (SAA), which may last $20^m$. The telescope cannot point too closely to the Sun, Moon or bright Earth limb. The roll orientation of the spacecraft is constrained in order to maintain correct power and thermal balance. Communications with HST is via the Tracking and Data Relay Satellites (TDRS) and links will be available for only part of an orbit (this also limits the amount of real time interactions with the HST and instruments). Slews are relatively slow ($90^o$ in $\sim 15^m$), so efficient ordering of telescope pointing is important. Available electrical power limits the number of instruments that can be in "standby" or "operate" modes, and cycling between instrument can take several hours. (Refer to Miller, et al. 1987 for details.)

As a consequence of these and other factors, the operation of HST is almost entirely pre-planned. Long range plans must ensure the overall feasibility of the program. Short term plans must be consistent with the long range plan. Changes to the schedules caused by unexpected astronomical events (e.g. a supernova), instrument anomalies, changes in TDRS schedules, etc. must be accommodated and factored into the long term plan and to related exposures.

Currently, HST planning and scheduling is supported by the Science Operations Ground System (SOGS) Science Planning and Scheduling System (SPSS), which was developed by TRW. Initial population of the SPSS scheduling data structures is via the PEP Transformation Subsystem, an expert system which takes the astronomer-oriented proposal from the PEP system and creates the detailed implementation parameters required by SPSS (Rosenthal, et al. 1986). While SPSS has been successfully used to generate detailed schedules of a few days duration, there are several factors that severely limit its use on the long-range planning problem: SPSS scheduling algorithms only examine a few possible times to schedule exposures, and can therefore easily miss good scheduling opportunities. SPSS always considers detailed orbital events and conditions, even when they are uncertain or unpredictable. This makes it computationally very expensive to construct and evaluate long-range plans. A significant number of scheduling constraints are not considered by SPSS, and, because of the design and implementation of the system, they are difficult to add to the software. Coordination of related exposures which fall into different short term plans is essentially a labor intensive, manual process. Even for short term plans, the throughput of the overall system (people plus software) remains a concern.

Work towards enhancing the scheduling capabilities of the HST ground system is directed along two lines: 1. increasing the performance, reliability, maintainability and functionality of the existing SPSS software, and 2. creating new tools which augment the existing software. The latter work, the Spike system, is the subject of this paper. These two efforts are being carefully coordinated. The current focus in the Spike system is the development of long term planning tools, the first phase of which are the Exposure Evaluation Tools.

## 3. Long Term Planning

For HST science operations there are several key considerations for long term planning:

- 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
- constraints can couple exposures separated by long time intervals
- replanning will be required

Multi-year planning is an essential part of HST science operations. The basic observing cycle is one year long, and it is necessary to consider observations in preceding and succeeding cycles: Priority observations from the preceding cycle may, for various reasons, not be executed and will be "carried over" into the current cycle. Although most proposals will be completed in one cycle, some proposals are for multi-year observation programs, and the effect of these on future observing cycles must be considered. Additionally, in the first few years of HST operations, the Science Verification (SV) proposals from the Instrument Teams are mixed with General Observer (GO) proposals from the scientific community. Long term planning will be vital to ensure that short term schedules are consistent with the overall SV and GO objectives. Long term planning is also necessary to evaluate the effects of changes in the spacecraft, e.g. replacement of scientific instruments, slow decay of electrical power, etc.

The position of HST within its orbit can be predicted accurately about 3 months in advance. Beyond this, the in-track error grows so large that timing of events which depend on HST position (e.g. occultation by the Earth, SAA entry and exit) cannot be accurately predicted. A primary cause of this is fluctuations in the atmospheric density at HST's altitude due to fluctuations in solar activity. Fortunately, the orientation of plane of HST's orbit can be forecast with reasonable accuracy about one year in advance. This allows an average treatment of certain constraints. It should be noted that due to the large number of exposures in a long range plan, even if it were possible to predict HST orbital events with infinite accuracy, it would not be desirable to do so. A simpler approach for some constraints reduces the amount of computation while supplying the needed accuracy for long term plans. We consider the long term planning problem to range from several years to approximately 3 months before execution of the exposure.

South Atlantic Anomaly (SAA) passage serves to illustrate how constraints dependent upon the HST position can be meaningfully treated in an average sense for long term planning. The exact times of SAA entry and exit depend on the orbital location of HST. However, the fraction of time per day a target is unocculted by the Earth while the HST is outside the SAA depends only on the orientation of the plane of the orbit, and therefore can be estimated in advance (Sherrill 1987). In other words, for a particular day in the future, we cannot know the absolute time of SAA-free observing opportunities, but the number and duration of such opportunities can be known and used in a long term plan. The effects of scattered light from the Earth, Sun and Moon on faint targets can be treated in a similar manner.

Another important category of constraints are those which constrain the long term plan itself. This includes constraints on the consumption of various limited resources such as time, data volume, power, TDRS contacts and real-time operations. The efficiency of a plan is also another important metric in construction a long term plan.

The notion of hierarchical planning from a coarse to a fine level of detail has been explored in many scheduling problems (see, for example, Smith, Fox and Ow 1986) and will be useful for HST scheduling. Long term plans will span a few years, with perhaps a two month time resolution (two months is the precession period of the HST orbital pole, so this interval will encompass a complete

cycle of orbit dependent scheduling opportunities such as SAA passage and continuous viewing). As the long term plan is populated, subplans will be scheduled with perhaps a week's time resolution. Sufficiently detailed plans will then be sent to SPSS for detailed scheduling.

Replanning of two distinct types will be necessary. In going to a more detailed level (say a two month plan to a one week plan), it may be found that the high level plan was overly optimistic and that some observations allocated to a particular week cannot be executed due to some constraint. In this case, information from the lower level plan must be used to modify the higher level plans and the effects propagated (perhaps to other two month plans due to linkages between observations). The second type of replanning occurs when an observation fails to execute properly. For example, one of the stars in a guide star pair may be a binary star, which prevents tracking by the HST. Not only will it be necessary to replan that observation, it will be necessary to examine the effects on other observations which attempt to use that guide star. When an instrument shows some unexplained behavior, it will be necessary to suspend normal observations, schedule the necessary diagnostic operations and replan the suspended observations for some point in the future. Observations of targets of opportunities (e.g. a comet or supernova) will also cause schedule disruption and replanning.

Given this view of long term scheduling for the HST, the next section presents the Spike Exposure Evaluation Tools, demonstrating how they provide the required capabilities.

## 4. Spike Exposure Evaluation Tools

The Exposure Evaluation Tools were designed with the following features:

- uniform representation and manipulation of scheduling constraints
- express human value judgments and tradeoffs ("shades of gray" are handled as well as go/no-go criteria)
- easy to modify relative importance of constraints, and to add new constraints
- the interaction of constraints and tradeoff options are visible to human planners
- provide a means to track resource usage
- effective user interface
- ability to build automated scheduling tools on top of exposure evaluation tools

### 4.1 Activities, Clusters, Constraints and Suitabilities

An **activity** is the lowest level scheduling entity and is used to represent exposures (or possibly groups of exposures) and other planned actions such as instrument configurations, slews, communications contacts, etc. Exposures have a very few properties: duration, constraint list, priority and a flag to mark if the exposure is executed. As the duration of an activity may depend on its time of execution and relationship to other activities (e.g. slew time, exposure time, instrument configuration), the duration of an exposure is implemented as a function, not a constant.

Activities are grouped into **scheduling clusters**. These are the lowest level entities which can be scheduled. Scheduling clusters can represent SPSS scheduling units, branching sequences, logical components of an observing proposal, etc. To preserve scheduling flexibility, ordering of activities within scheduling clusters is not required. The default is one activity per scheduling cluster. Clustering reduces the number of entities to be considered by the scheduler.

A **constraint** is any factor that effects when it is possible or desirable to plan activities. This includes such **strict** constraints as "never point the HST closer than $40^{o}$ to the Sun" and **preference** constraints such as "its preferable to execute the observation when 3 independent pairs of guide stars are available, but one pair is acceptable". There are two categories of

constraints: activity and segment constraints. **Activity constraints** limit the opportunities for a particular activity. An **absolute activity constraint** is independent of when any other activities are planned (e.g. Sun avoidance, orbital dark time, guide star availability, roll). A **relative activity constraint** relates two or more activities and depends on the suitabilities of other activities (e.g. precedence constraints, minimum and maximum time separation between exposures). Absolute activity constraints are fixed when planning begins, while relative constraints change as activities are fixed in the scheduling process. Segment constraints represent limitations on the overall plan and are defined later.

A fundamental component of any scheduling system is the representation of constraints. Constraints are represented in the Spike system as **suitability functions** (see Figure 1). A suitability function gives the desirability of starting an activity at a particular time. A suitability of zero means that a start at that 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, with suitabilities greater than 1 indicating a more favorable starting time and suitabilities less than 1 indicating a less favorable starting time. A suitability function can be considered as a generalization of binary planning windows. The fact that suitabilities are not limited to a binary "yes/no" allows a powerful and natural way to express preferences and allowable tolerances on constraints. A suitability can be interpreted as the degree of constraint satisfaction. The formal properties of suitability functions are discussed in more detail in Johnston (1988b).

In the Spike software, constraint suitabilities are implemented as non-negative piecewise-constant functions (PCFs). This class of functions was chosen since it allows a particularly efficient representation of constraints and propagation of the effects of constraints. PCFs are closed under the operations of addition and multiplication, i.e. the sum or product of two PCFs is another PCF.

An important feature of the Spike software is that new constraints can be readily added to the system, or invoked as planning proceeds from long range to medium range to short range planning. Human judgment is required to establish the shapes of the suitability functions and the scale factors.

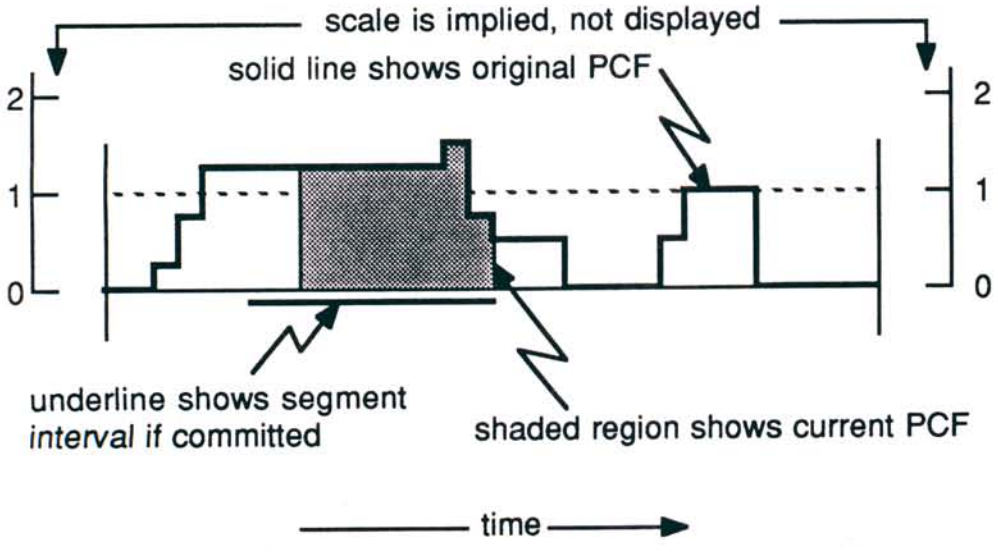


Figure 1 - Suitability function.

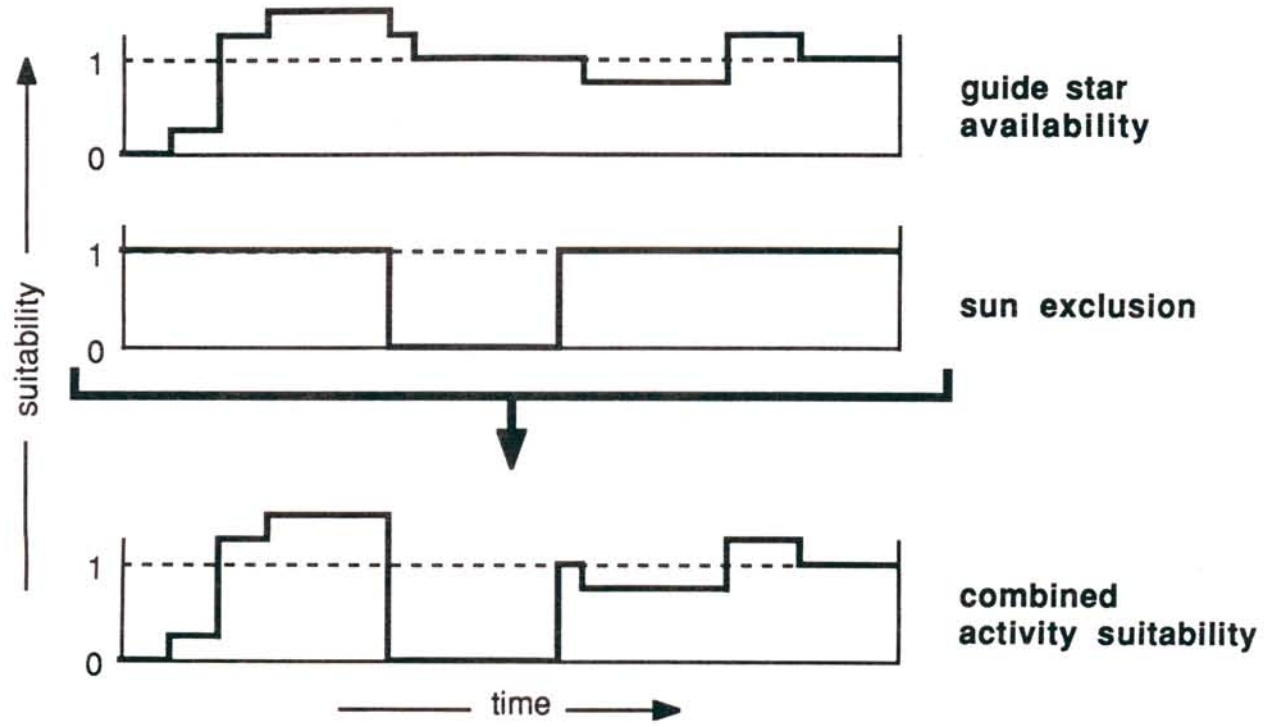


Figure 2 - Combination of two constraints on an activity to form the overall suitability.

The use of suitability functions allows a natural and expressive means to combine the effects of constraints. The overall suitability of an activity is the product of the suitability functions of all constraints (both relative and absolute) attached to that activity. In Figure 2 it can be seen that the strict constraint of Sun avoidance excludes certain times, while the guide star constraint "modulates" the suitability outside the Sun exclusion.

The suitability of a scheduling cluster is the geometric mean of the suitability of all component activities. This requires that all activities have non-zero suitability at that time (since the activities within a scheduling cluster are not necessarily ordered, it must be possible for all to begin at that time).

Constraints are used to restrict the times when it is possible to schedule an activity. The manner in which constraints are propagated in Spike is illustrated in the following example: Let A and B be activities with absolute suitabilities $S_A^{abs}$ and $S_B^{abs}$ derived from their absolute constraints. Let $C_1$ and $C_2$ be relative constraints where $C_1$ expresses a constraint of activity A on activity B and $C_2$ expresses a constraint of B on A. Given initial values for the suitabilities of activities A and B to be $S_A = S_A^{abs}$ and $S_B = S_B^{abs}$, the constraint processing algorithm evaluates the effect of A on B via $C_1$ (which depends on $S_A$) and stores the result in a temporary suitability $S_B^{temp}$. The suitability of activity B is then updated via $S_B := S_B^{abs} * S_B^{temp}$. Recalling the way suitabilities are defined, any times of zero suitability in $S_B^{temp}$ will result in zero suitability in $S_B$, other times will result in a larger or smaller suitability depending on the preference expressed in the constraint. The effect of activity B on activity A via constraint $C_2$ is next calculated in a similar fashion. This iteration continues until there are no more changes in suitability. It can be seen from this that mutual effects of constraints will be propagated onto both activities and that this method is quite

general: no specific assumptions as to the nature or the number of the constraints is required. Constraint propagation is described in more detail in Johnston (1988b).

A **dependency cluster** is a set of activities which are related by relative constraints. They reduce the computational complexity of constraint propagation since the constraint propagation algorithm need only consider clusters in the same dependency cluster.

### 4.2 Segmentations and Commitments

In Spike, a long range plan is called a **segmentation.** Within a segmentation, time is divided into intervals called **segments**. Segments are simply a convenient means to discretize time, creating time "bins" or "buckets". For long term planning, this allows a significant reduction in the time dimension of the problem without being artificially limiting. The duration of a segment is subject only to the restriction that the time spanned by the segment is greater than an orbital timescale and the duration of activities, i.e. the duration of a segment > 1 day. Segments need not have equal duration, nor need they be contiguous.

A scheduling cluster may be **committed** to a segment, that is, restricted to start during the time interval of the segment, so long as its suitability is non-zero somewhere in the segment. When a cluster is committed to a segment, the cluster suitability function is set to zero outside the segment and the effect of this is propagated automatically. Thus a commitment may further restrict other clusters related via constraints such as precedence or time separation. Note that a cluster may have times of zero suitability within a segment due to the operation of some strict constraint such as Sun avoidance. Any limitations more stringent than that imposed by the segment boundaries are preserved and propagated, thus preserving the maximum scheduling information.

Commitment only requires that the cluster *begin* within the segment; it does not require that the cluster *end* within the segment (which would be unnecessarily restrictive). Multiple clusters may be committed to segments (subject to constraints) and no ordering of clusters within a segment is imposed by the commitment.

A segment may have one or more **segment constraints**. These express limitations segment resources consumed by clusters committed to that segment (e.g.total time, data volume, communications contacts and real time interaction). Given a segment with many clusters already committed, the segment constraint may allow a cluster of short duration to be committed, but prevent the commitment of a cluster of long duration exposures.

### 4.3 User Interface

Given a problem as complex as HST scheduling, an important aspect of the Spike system is a user interface which facilitates human-machine interaction and rapid comprehension of scheduling constraints, dependencies and commitments.

The main system interface for the Spike software is the Command Hierarchy Interpreter (CHI). This is a general-purpose utility for accessing Spike commands. Commands are organized hierarchically and are accessed by using the mouse and menus. The user of CHI need not ever remember function names or spellings. Context-sensitive help is provided as well as a simple command dependency checker (e.g. command A must be selected before command B).

Within the Exposure Evaluation Tools, the focus of the user interface is the window-oriented planning environment (Figure 3). Time is displayed horizontally, while the suitability function for clusters, activities and constraints can be displayed vertically (refer to Figure 1). This display gives a powerful means to understanding the schedule. The foreground window in Figure 3 displays the suitability functions for 5 related exposures (an acquisition, two science exposures and two

calibration exposures). The background screen displays components of the suitability function for one of the exposures. Functions to access clusters, activities, constraints and the timeline are activiated via the mouse and popup menus.

Most of the interaction involves using the mouse and various popup menus: The mouse can be used to zoom in on times of interest. Segmentations, segments, clusters, activities and constraints are all mousable, providing various options which invoke commands. For instance, selecting a constraint's description tells the user the type of constraint, various parameters and the affected activities (e.g. a proposer-specified time separation constraint of 30±10 days between activities A1 and A2).

The user can create multiple planning windows; each showing a different view of the segmentation so far, for example, different time intervals or different sets of scheduling clusters. A change to the plan effected from one window will be reflected in the global database and thus may show up in other windows. The user can also spawn a new, independent segmentation. This is useful for exploring the effects of different commitments in parallel.

Another subsystem that has been incorporated into the Spike system is the Lisp Object State Saver (LOSS, Sponsler 1988). This tool saves memory-resident data structures to an ASCII file. The file can later be reloaded into Lisp memory and and the recreated data structures restored to a logically equivalent pre-save state. In Spike, it is used to save partial schedules; a planner using Spike can take a LOSS "snapshot" of memory for use at some later time.

### 4.4 Operations Concept

In this section we give a brief outline of how the Exposure Evaluation tools can be used in HST science operations. Long term planning begins with the pool of approved proposals in the Pep system. These proposals have also been validated to remove syntactic and semantic errors. Processing of proposals through Pep Transformation, which defines SPSS scheduling data structures, can also be performed at this point, but is not essential for long term planning.

A planner can use the entire accepted proposal pool or any subset (e.g. just high priority and time critical proposals). The relevant proposal information is extracted from the Pep database and transferred to a Spike workstation (see section 7). Next, the planner creates a segmentation timeline, with the time interval of the plan divided into a number of time segments or "buckets". The planner then issues a command to load a set of proposals into Spike. The planner can invoke a command which checks proposals for inherently unschedulable conditions. Such conditions include circular precedence constraints (execute exposure A before B, B before C and C before A) and inconsistent constraints (a proposer specified time window that occurs while the target is too near the Sun for observation). Proposals with inherently unschedulable exposures can be modified, according to policy, to correctly implement the proposer's science objectives.

Exposures are then grouped to form scheduling clusters. One clustering option is to group exposures according to the results of Pep Transformation (either preserving the strict ordering imposed by the SPSS data structures or allowing a more flexible scheme where activities inside SPSS scheduling units are not ordered). In either case, estimates of the amount of resources consumed by a cluster (e.g. time, communications contacts, real time usage, etc.) are calculated. Clustering exposures on the basis of other criteria such as target proximity, instrument usage or relationship to other exposures in the proposal will be explored in future work.

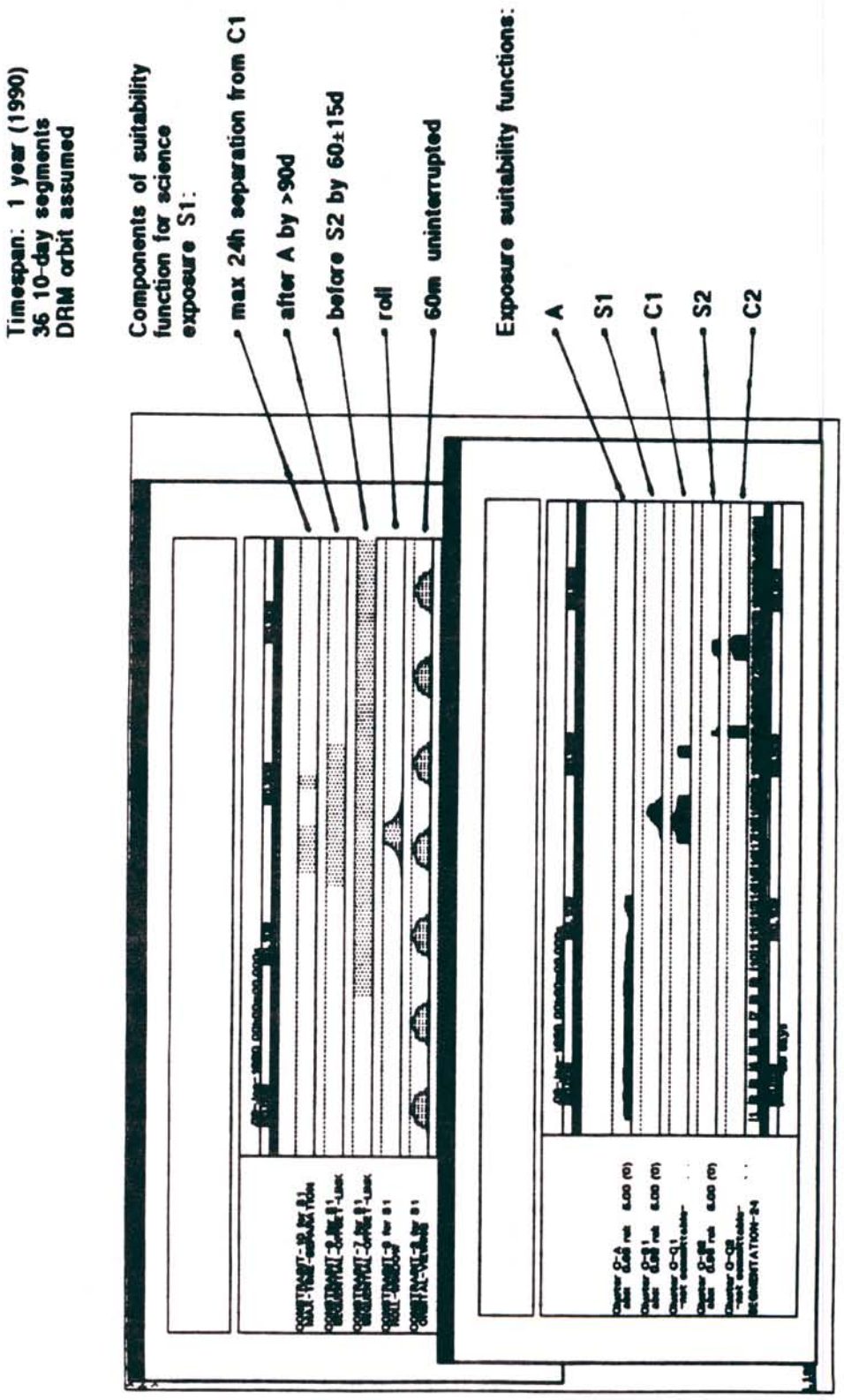


Figure 3 - Exposure Evaluation Tools Timeline Display.

At this point, long term planning can begin, with the goal being to commit each cluster to a segment in what will typically be a year-long or multi-year plan. Several options are available to the planner. Clusters can be committed manually via the user interface, or automatic scheduling software can be invoked (see section 5). The timeline display provides visual information on the commitments and clusters which become unschedulable as the result of other commitments. The schedule is iterated until a suitable solution is found. During the commitment process, the planner can create new segmentations in order to explore different scheduling choices. The state of the system can be saved to a file for use in later planning sessions.

Hierarchical planning fits naturally in this scheme. The planner can create a new segmentation which covers a smaller time interval with finer time resolution in the segments. The commitments from the higher-level plan can be transferred to the more detailed plan for further commitment. Tools to automate hierarchical planning will be developed in future work. When clusters are committed to segments of approximately a week's duration, the commitments can be transferred to SPSS for detailed scheduling.

## 5. Strategies

Given the large number of observations to be executed by HST in a year, automated tools to schedule the bulk of the observations are clearly necessary. This section discusses three types of automated commitment strategies which have been prototyped: procedural, rulebased and artificial neural networks.

### 5.1 Procedural Strategies

Two simple procedural strategies are available in the Spike software: most suitable cluster-segment, and most absolute constrained cluster. The first strategy finds the cluster-segment pair with the highest suitability of all clusters in all segments and commits that cluster to that segment. The effect of this commitment is propagated through the relative and segment constraints to find the effects on other clusters. This will usually limit the scheduling choices for these remaining clusters, i.e. the suitability at some times will be smaller (perhaps zero) due to the commitment. Suitabilities are recalculated and the highest cluster-segment suitability of the non-committed clusters is identified. The process is repeated until all clusters are committed or all remaining clusters are unschedulable. This strategy is one implementation of the so-called "greedy" algorithm.

The second algorithm operates in a similar manner, with the cluster which has the smallest time span of non-zero suitability (i.e. most constrained) being committed to the segment with highest suitability at each step.

The advantage of these strategies is the speed of execution. The disadvantage, as is well known in scheduling problems, is that such simple strategies can lead to grossly sub-optimal schedules and an unacceptable number of unschedulable clusters. More flexible strategies which take into account both resource and cluster bottlenecks are required to solve a problem as complex as HST scheduling (Smith, Fox and Ow 1986).

### 5.2 Rule Based Strategies

In order to provide a more flexible and intelligent approach to scheduling, a rulebased scheduler has been implemented. The rulebase is a **control** layer directing the Exposure Evaluation tools which serves as the **representation** layer. Essentially, the rulebase replaces a human planner making commitments via the window interface. The rulebase chooses what commitments to make and analyzes the results of commitments, while the Exposure Evaluation tools spawn alternative schedules, execute the commitments and propagate constraints. The rulebase system was implemented in KEE (a product of IntelliCorp).

Communications between the two layers is through a schema in the rulebase. For each segmentation (partial schedule) in the representation layer, there exists a corresponding schema in the control layer. The schema contains a number of slots which hold summary information about the segmentation, e.g. segmentation name, most suitable cluster and segment, most absolute constrained cluster, highest priority cluster, unschedulable clusters, etc. All reasoning about commitments uses this summary information. When a commitment is made the representation layer creates a new segmentation and the control layer creates a new schema. A method is executed which populates the schema with summary information. Both the schemas and the segmentations can be thought of as a tree where each node is a partial schedule.

The search strategy conducts a "best-first" search based on the A* algorithm. Two classes of rules were used: search and commitment. The search rule class identifies which one partial schedule is most promising (in some sense, see below). The commitment class of rules decides for the most promising node, what is the best commitment to make (e.g. highest priority cluster, most absolute constrained cluster).

Definition of the "best" or "most promising" partial schedule is of central importance to this strategy. Work to date has primarily used the sum over all clusters of the highest suitability in any segment. This allows the rulebase to identify poor solutions (since the suitability of some clusters becomes low), but does not provide enough discrimination between schedules before poor solutions are generated. This is largely due to the fact that the summed suitability neglects the effects of segment constraints until a cluster becomes totally unschedulable. In other words, it is focusing too closely on scheduling clusters in suitable segments without considering the effects on the diminishing resources of the segments themselves. Smith and Ow (1985) have addressed the similar problem of task versus resource based views in factory scheduling. We have also made preliminary investigations of using the rulebase to identify resource bottlenecks in order to schedule critical clusters first.

A typical commitment rule might be:

```
For the best partial schedule
if    there is a high priority exposure
      and it is the most absolutely constrained exposure
      and this rule hasn't been applied to this partial schedule
then  commit the exposure to its most suitable segment
      populate the schema with segmentation information
      add this partial schedule to the list of open schedules
```

This example is paraphrased from the Kee code, however implementing a natural language tool to take the above and translate to executable code is not a difficult task. In the event that more than one commitment rule is instantiated for the current best node, the rule with the highest weight is chosen. More sophisticated strategies for choosing among competing commitment rules are under investigation.

### 5.3 Artificial Neural Network Strategies

An artificial neural net has been implemented as another approach to automated scheduling. Neural nets are based on models of how biological "computers" work and have been applied to a variety of problems including pattern recognition, classification, memory, and speech understanding (see Tank and Hopfield 1987 for an introduction to neural nets).

In SPIKE scheduling, a version of the Hopfield neural net model is used. The network can be considered as a rectangular matrix, where the columns represent segments (time) and the rows represent scheduling clusters. A particular network element (neuron) represents the potential

commitment of a particular cluster to a particular segment. When the network has relaxed (found a solution), a neuron in the "on" state signifies a commitment of that cluster to that segment. For each neuron, a bias term expresses the absolute suitability of that cluster-segment combination, with the bias term proportional to the suitability in that cluster (or a large inhibitory bias if the suitability in that segment is zero). Several factors determine the connections strengths between neurons: All neurons in the same column (same segment) are connected to express the segment constraints. All neurons in the same row (same cluster) are connected so that a cluster committed to a segment will inhibit the commitment of that same cluster to another segment. The effect of the commitment of cluster A1 to segment S1 on the commitment of cluster A2 to segment S2 is implemented as a connection between the corresponding neurons. The strength of the connection (weight) is determined using suitabilities derived from the relative constraints relating A1 and A2. Additionally, there is a set of "guard neurons", one for each row (cluster) which apply a bias to force each cluster to be committed to some segment. Without the guard neurons, the network could converge to solutions where many clusters were not committed to any segment. However, the addition of guard neurons also breaks the symmetry of the basic Hopfield model so that the network is no longer guaranteed to always converge to a solution. With the network connections established, the neural network tends to frequently find "good" solutions (most suitable cluster/segment commitments) very quickly. When the network fails to converge it can be stopped after a fixed number of neuron state changes and then restarted. In this way is is possible to quickly generate and evaluate a number of partial or full schedules. The neural network approach appears particularly promising for replanning: the network does not need to be rebuilt since changing only the neuron biases is sufficient to indicate which activities have been executed. It also offers the potential for implementation on parallel hardware, which would provide even more dramatic performance improvements.

## 6. Development Methodology

The software described in this paper has been developing using "artificial intelligence" tools, including Lisp, object oriented programming, expert systems and artificial neural networks. The use of a Lisp software development environment on a workstation (Texas Instruments Explorer and Apple MacIntosh) provided an unparalleled development and implementation environment.

A rapid prototyping methodology is a key component of our approach, since many requirements of the problem are not understood in detail - the HST scheduling problem is orders of magnitude more difficult than that found at any other observatory. Examples of areas requiring substantial investigation include the relative importance between various constraints, the choice of scheduling strategies in various situations, metrics for schedule quality, replanning from schedule disruptions and display tools. Our development approach is to quickly provide the users an initial set of tools with fundamental scheduling capability. This testbed will allow users and developers to explore the problems together, better understand the requirements and implement improvements quickly and efficiently (see Agresti 1986 for more on rapid prototyping).

Most of the system is implemented in Common Lisp. Currently we are using the Flavors object system but we expect to migrate to the Common Lisp Object System when it becomes available. The graphics and user interface utilities are based on IntelliCorp's Common Windows. The use of expert system shells such as KEE (from IntelliCorp) or ART (from Inference) to control scheduling decisions is under investigation. Careful attention has been paid to standardization and portability: portions of the system have been used on TI Explorer, Symbolics, Vax, MacIntosh and Sun computers. Although development takes place on Explorers and MacIntoshes, we are investigating the utility of other machines for operational use including general purpose workstations (such as Sun computers) or hybrid machines (such as the Macintosh II-micro Explorer).

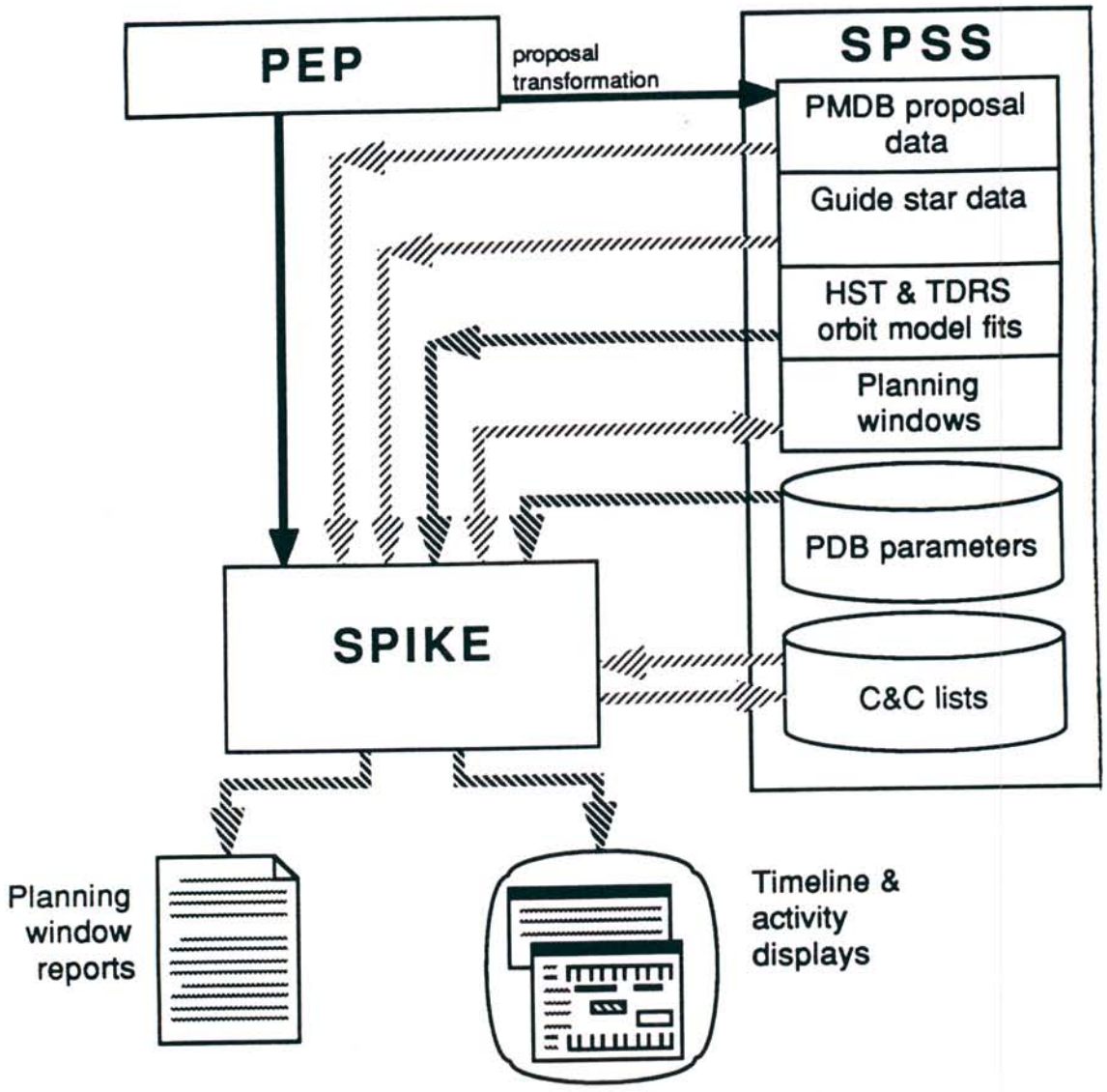


Figure 4 - Spike Interfaces. Solid arrows show existing interfaces, dark hatched arrows show interfaces to be implemented for Exposure Evaluation Tools, while light hatched arrows show interfaces planned for later phases. Interfaces to SPSS (e.g. Guide Star Selection System) are not shown.

## 7. Interfaces

The Spike system has interfaces to two other systems: Pep and SPSS (see Figure 4). Currently, the Pep interface has been partially implemented. Interfaces to the SPSS system will be added as development continues.

As noted earlier, the Pep system contains proposal information including the proposer's target list, exposures and their interrelationships, and the initial SPSS data structures generated by the Pep Transformation software. Migration of this information from the Pep relational database to the Spike system is a two step process: First, a general purpose database report writer is used to extract the proposal information into a file, formatted as calls to Lisp functions. In the second step, a Lisp program reads the database report file and creates a second file, again writing calls to Lisp functions. When loaded into the Spike system this file creates data structures such as targets, constraints, activities and clusters. The first step is simply an extraction of the information from a relational database, while the second step performs the mapping between the Pep and Spike data structures. For example, binary links between exposures are stored as tuples in a relation in the Pep database. These links are mapped to different Spike constraints depending on the type of the exposure link (acquisition, precedence, conditional, etc.).

Files are transferred from Pep to Spike using Decnet or TCP/IP and can be initiated from either system. Currently, extraction of Pep information must be initiated on a computer running the Pep system. The second processing step can be performed on either a Pep or Spike computer. We plan to implement remote procedure calls so that all processing can be initiated from the Spike workstation.

Initially, a unidirectional interface from SPSS to Spike will be implemented in order to provide Spike with HST and TDRS orbit models and HST Project Database parameters relevant to long range scheduling. The interface will be extended to provide bi-directional transmission of scheduling windows and planning data structures, e.g. transmission of medium and short range plans from Spike to SPSS for detailed scheduling and the feedback of these results into the Spike plans. The interface will be patterned on the Pep-Spike interface as the relevant SPSS information resides either in a relational database or in disk files.

## 8. Discussion

The Spike Exposure Evaluation Tools provide a powerful means for long range scheduling of HST observations. Essential aspects in reducing this to a tractable problem include the representation and propagation of constraints, clustering exposures to reduce the number of entities scheduled, discretization of time to reduce the number of places examined and a user interface which displays the relevant information in a succinct manner.

The problems faced in scheduling HST observations are common to other telescopes and the Spike system has been designed with this in mind: The concept of suitability functions as a means of expressing constraints and constraint satisfaction is a general one. Likewise, propagation of the effects of constraints is not tied to the nature of the constraint or scheduling problem. Indeed, this approach should be applicable to many classes of scheduling problems outside the realm of telescope scheduling.

Traditionally, little emphasis has been given by observatories to integrated, efficient scheduling programs: For the most part, ground-based telescopes are scheduled manually (perhaps with limited software assistance), often by granting blocks of time (days or weeks) to observers who execute the observations. More sophisticated scheduling techniques can be found in some space-based telescopes. These schemes can limit the scientific efficiency of an observatory: repeated

observations of short duration over a long timespan are very difficult to accommodate in the block time scheduling. An observation requiring outstanding atmospheric or orbital conditions may find useless time that would be more than adequate for other, less demanding observations. Simultaneous observations of a target from several observatories only exacerbates the problem. Given the high oversubscription rate of all major astronomical facilities, an increase in efficiency can be very important. Concurrent with the increased abilities of scheduling software, there is a growing awareness of the need for automated planning to increase scientific productivity (Johnston 1988a). Plans for the University of Texas-Penn State 8m Spectroscopic Survey Telescope (SST), the European Very Large Telescope (VLT) and Space Station Science and Applications Information System (SAIS) call for increased scientific return through sophisticated scheduling and reactive replanning. We have made preliminary investigations in scheduling other observatories with the Spike software (Johnston 1988c).

We would like to thank Hans-Martin Adorf (ST European Coordinating Facility), Robert Jackson (STScI), Don Rosenthal (NASA Ames), Tom Sherrill (Lockheed), Dave Skillman (NASA Goddard), Steve Smith (Carnegie Mellon University) and Monte Zweben (NASA Ames) for stimulating conversations on planning and scheduling problems. We also express appreciation to Texas Instruments for the loan of a TI Explorer.

## References

Agresti, W. 1986, New Paradigms for Software Development, IEEE Computer Society Press.

Hall, D., ed. 1982, The Space Telescope Observatory, NASA CP-2244.

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.

Johnston, M. 1988a, "Automated Telescope Scheduling" in Coordination of Observational Projects, Cambridge University Press.

Johnston, M. 1988b, "Reasoning with Scheduling Constraints and Preferences", in preparation.

Johnston, M. 1988c, "Automated Observation Scheduling for the VLT", in preparation.

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.

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

Sherrill, T. 1987, Lockheed Engineering Memorandum, private communication.

Smith, S., Fox, M. and Ow, P. 1986, "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems", AI Magazine, Fall 1986, p45.

Smith, S. and Ow, P. 1985, "The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks", Proceedings of the Ninth International Joint Conference on Artificial Intelligence.

Sponsler, J. 1988, "Lisp Object State Saver (LOSS): A Facility Used to Save Partial Schedules for the Hubble Space Telescope" in Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence.

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