TELEMATICS and INFORMATICS Vol. 7, Nos. 3/4, pp. 243–253, 1990
Copyright © 1990 Pergamon Press plc. Printed in the USA
0736-5853/90 $3.00 + .00

AN APPROACH TO RESCHEDULING ACTIVITIES BASED ON DETERMINATION OF PRIORITY AND DISRUPTIVITY

Jeffrey L. Sponsler
Mark D. Johnston

Abstract – A constraint-based scheduling system called SPIKE is being used to create long-term schedules for the Hubble Space Telescope. Feedback from the spacecraft or from other ground support systems may invalidate some scheduling decisions and those activities concerned must be reconsidered. A function *rescheduling priority* is defined, which for a given activity performs a heuristic analysis and produces a relative numerical value that is used to rank all such entries in the order that they should be rescheduled. A function *disruptivity* is also defined that is used to place a relative numeric value on how much a preexisting schedule would be changed to reschedule an activity. Using these functions, two algorithms (a *stochastic neural network approach* and an *exhaustive search approach*) are proposed to find the best place to reschedule an activity. Prototypes have been implemented and preliminary testing reveals that the exhaustive technique produces only marginally better results at much greater computational cost.

INTRODUCTION

Scheduling is an intellectual activity that humans do on a daily basis. Often this activity is accomplished without the awareness that reasonable (but not necessarily optimal) solutions are formulated for a generally hard problem. One may argue that the number of activities to be scheduled in a day is small, the constraints to be imposed are simple, and thus the problem is tractable. On the other hand, it should be recalled that a fine-grained massively parallel architecture fine-tuned over epochs is at work. Rescheduling, as distinct from scheduling, is perhaps an equally important activity, due to the fact that schedules are rarely executed precisely as planned and therefore must be revised dynamically. It is the focus of this report.

Description of the Hubble Space Telescope

NASA's Hubble Space Telescope (HST) is an observatory that was placed into orbit by the Space Shuttle *Discovery* in April 1990. It has six scientific instruments and will provide greatly improved resolution and sensitivity because it is above the earth's atmosphere. The Space Telescope Science Institute (STScI) is responsible for managing the ground-based scientific operations of HST. Proposals for observation of astronomical

Jeffrey L. Sponsler and Mark D. Johnston are with the Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, Maryland 21218. The Institute is operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration.

objects are submitted by astronomers (professional and amateur) and are processed by a series of software programs. An expert system called TRANSFORMATION processes proposal data and produces data structures organized by rules. An artifical intelligence (AI) system called SPIKE is used to create long-term schedules (for periods of one year or more). SPIKE feeds the data from one week of such a schedule to a system called SPSS that is used to create a finely detailed schedule. From this is derived specific spacecraft commands. For more details about HST, see (Hall, 1982).

SPSS and TRANSFORMATION

The Science Planning and Scheduling System (SPSS) is the short-term scheduling software for HST. SPSS operates on entities known as *exposures, alignments, observation sets*, and *scheduling units* (SUs). The term *exposure* is defined simply as an observation of some object by a science instrument (SI). An *alignment* is a specification for pointing the spacecraft. Generally this pointing may start at one point and end at another but in practice this usually is a single point. One or more exposures may be assigned to the same alignment. An *observation set* is composed of alignments where all exposures have the same *guide stars* (reference stars used to maintain exact pointing of HST). An SU is composed of one or more *observation sets* that conform to certain requirements (e.g., there is a *sequential no-gap* specification between bordering exposures of different observation sets).

The TRANSFORMATION system has been developed at STScI to generate this data organization. It uses heuristics obtained by *operations astronomers* who have in the past manually generated the SPSS data structures. The input is a proposal file prepared and submitted remotely by an astronomer, its output is used to populate the SPSS database, as well as to provide the SPIKE system with the data needed to generate its schedules.

The SPIKE scheduling system

The SPIKE scheduling environment consists of the core SPIKE constraint-based system, a user interface, and a neural network-based algorithm used to search for optimal solutions to the ST long-term scheduling problem (Miller, 1989). Descriptions of these subsystems follow.

The constraint-based scheduling system. The SPIKE system has been created as a general-purpose scheduler and so specific references to other systems and even to spacecraft are abstracted. The system operates on *activities, constraints*, and *scheduling clusters* (groups of activities). The mapping of terms according to the pattern *SPSS term/SPIKE term* includes: exposure/activity, constraint/constraint, scheduling unit/scheduling cluster (often the term cluster is used). It is the case that within the body of this paper these analogous terms will be used interchangeably.

SPIKE processes information from TRANSFORMATION about targets (e.g., "crab nebula"), exposures (e.g., "crab nebula using planetary camera"), constraints (e.g., "A before B"), and the proposal data organization that SPSS requires.

The *suitability function* is a means for representing scheduling constraints and preferences (Johnston, 1990). The approach is numeric and provides a powerful way to represent the concept of "goodness over time." The SPIKE approach is extensible and the constraint knowledge is represented explicitly as objects (Flavors instances) with associated methods.

In the SPIKE domain, scheduling is treated as a **constraint satisfaction problem.** Constraints may be either *absolute time constraints* ("execute the exposure only if the sun is not in the target path"), *relative time constraints* ("execute exposure A before exposure B"), or *resource constraints.* Such problems are known to be NP-complete (Garey, 1979) and so the exhaustive traversal of the entire search space is not computationally tractable if the number of scheduling clusters is large.

The term *dependency cluster* is defined as follows: Let S be a set of activities that are in a dependency cluster. An activity A is a member of S if one can traverse relative time constraint links to all other activities in S. Informally, the dependency cluster contains activities that directly or indirectly affect (via relative constraints) the other activities in the cluster.

Using the SPIKE scheduling tools, one may make a scheduling decision (i.e., a *commitment*) that restricts the times when a scheduling cluster may be scheduled. The scheduling system will propagate changes based on the relative constraints to other clusters that contain activities so linked. In general, the suitabilities of other activities within a dependency cluster will shrink, reflecting the notion that available scheduling windows are smaller.

SPIKE also keeps track of *resource constraints* such as available data storage, available exposure time, available TDRSS down-link time, and so on. Each resource is represented as a suitability function that will reflect lower suitability as the resource is consumed in a given time segment.

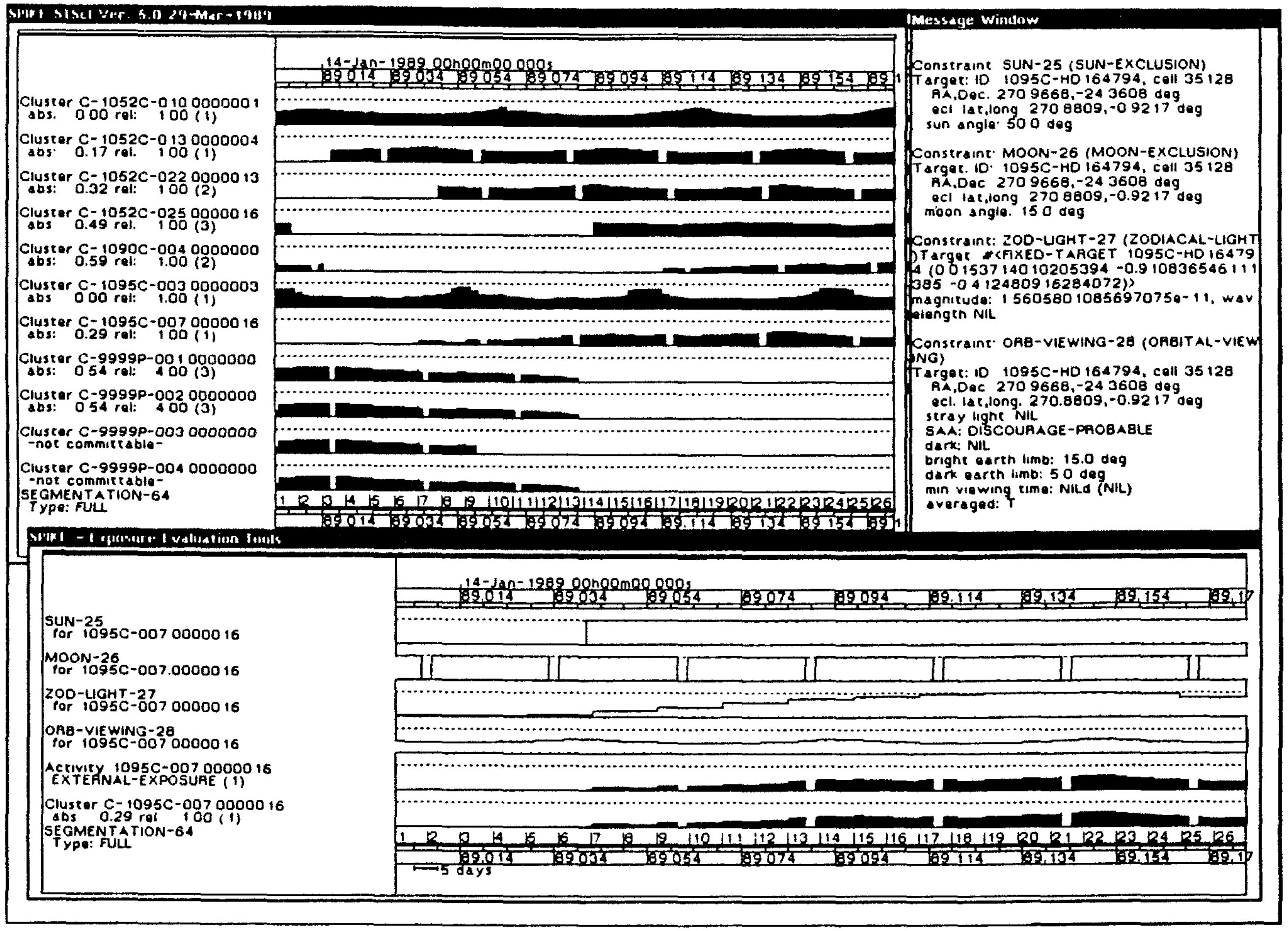
The mode of SPIKE usage considered in this paper is long-range scheduling. In this mode, the overall scheduling interval is divided into discrete units called *segments.* The length of a segment is arbitrary but expected to be one week during normal operations.

A long-range schedule will consist of a number of time segments each of which will have a set of scheduling clusters that have been committed there. The commitments are to week-long segments and do not specify precise times. Periodically, the information about one segment will be communicated to SPSS, which will then build a more detailed schedule. The logic of this organization is based on the notion that SPIKE can attempt to optimize a year-long schedule. SPSS will then have far fewer SUs upon which to work and will operate in much less time. Scheduling the SUs in that time range should be successful (based on SPIKE calculations), and a higher quality schedule will result. Figure 1 illustrates the graphical interface to the SPIKE system.

Neural network schedule optimizing system. The scheduling constraint satisfaction problem (CSP) can be represented as a Hopfield discrete neural network (Johnston, 1989), which can be thought of as a matrix where the rows represent scheduling clusters and the columns represent discrete time segments. The output state of each neuron in the matrix can be either 0 or 1, where 1 indicates a commitment of the activity to the time segment. A column of guard cells is used to bias the network in such a way as to maximize the number of clusters that are scheduled (neurons that are on).

A congruous connection matrix stores the *connections* between neurons representing relative constraints. Those connections are derived by analyzing the effect of committing an activity A to a time segment S on all other possible activity/segment neurons in the matrix. Another matrix stores the *biases* associated with each neuron. These biases are assigned by analyzing the absolute constraints on activities that comprise clusters, in such a way that higher biases indicate greater suitability.

The term *summed suitability* is a numeric value that is the sum of all inputs to each neuron that is *on* in the network. This is one way to measure how good the overall schedule is.


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 -0.9 10836546111
385 -0 4 124809 16284072)>
magnitude: 1 560580 1085697075e-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 1. Example screen from SPIKE showing the scheduling of a few HST observations. The upper left window represents a six-month scheduling interval and includes only scheduling clusters. The bottom window shows one of those clusters, its component activity and constraints. The upper right window displays textual information about several of the constraints.

Interactions between SPIKE and SPSS

SPIKE TO SPSS communication

SPIKE sends information to SPSS concerning what scheduling units are to be placed in a specific week. The granularity is very coarse in that SPIKE assures SPSS that the SU is schedulable at some unspecified point in that week. The operators of SPSS then must attempt to place each SU onto a detailed timeline.

SPSS TO SPIKE communication

It might occur that SPSS will be unable to place certain SUs on its timeline. The reasons for this might include the following:

1) The philosophy of scheduling at STScI includes the goal of oversubscribing exposures. The SPIKE system parameters associated with various resource constraints (exposure time, data volume) are set so that 30% oversubscription is the scheduling goal.
2) Since SPIKE is a long-term scheduler, some constraints, such as the South Atlantic Anomaly (SAA) of the Van Allen Belts, and Tracking and Data Relay Satellite System (TDRSS) availability, must be calculated only on a statistical basis. This is due to the fact that the ST in-track position is not accurately predictable on long timescales.
3) The logical context may have changed from the time that SPIKE calculated its best schedule to the time when SPSS attempts to place SUs on its calendar.
   - Minor changes in the orbit model may invalidate certain SPIKE decisions. For example, solar activity may have changed unpredictably such that constraints based on such activity (e.g., SAA) become more severe at SPSS scheduling time.
   - On-orbit experience with the spacecraft may change the manner in which activities are scheduled.
4) The greedy algorithm employed in the auto-scheduling SPSS subsystem may select from the search space a set of SU/time assignments such that certain mutually conflicting constraints make a complete scheduling of all SUs assigned to the week impossible.

It is a possibility that the execution by the spacecraft of certain exposures may fail (or that the data resulting from an exposure may be lost). In such a case, the SPIKE system will have to be alerted to this partial failure. This may require the creation of a new SU consisting of only the affected activities that will have to be considered for rescheduling.

In the event that SPSS is unable to schedule a subset of the SUs for a week, information concerning that subset will be sent to the SPIKE system along with some rudimentary explanation information (e.g., "constraint C violated," "instrument X unavailable"). This new information will, of course, make invalid some portions of the SPIKE schedule. The corresponding scheduling clusters will have to be removed from the schedule and reprocessed. Those steps are the focus of this discussion.

Constraint cascading over planning sessions (CCOPS)

A SPIKE subsystem, CCOPS has been developed to do the following:

CCOPS facilitates the decomposition of the full scheduling problem into more manageable portions. If the SPIKE system is called upon to build a schedule and all proposals are loaded at once, about 15,000 activities may be memory resident. As the number of activities and constraints increases, the time required to load files and instantiate the database and the time required for computation will increase. If it is the case that all activities and constraints are memory-resident at one time and a complete schedule has been computed, the problems associated with a hardware or software crash are exacerbated because a major loss is sustained. The strategy, then, has been to break the pool of proposals into groups.

The CCOPS system processes *session monitors*, which retain an abstract memory about what proposals are grouped, what scheduling decisions have been made, and what resources have been consumed. The CCOPS system interface is menu-oriented and provides the user with the tools to group proposals. Each group can be loaded into and processed by the SPIKE scheduling tools. Scheduling decisions made are stored by CCOPS in a symbolic format in a database that can be saved to disk. The important feature provided by CCOPS is a protocol for communicating the consumption of resource from one group of proposals to another.

The CCOPS system helps to solve the rescheduling problem by providing a mechanism for dealing with first the high priority items followed by the supplemental ones since, at the very least, the pool will be divided by *director's priority*. The director of the STScI assigns a priority to each proposal based on the recommendations of the Time Allocation Committee (TAC).

Let SM_a and SM_b be session monitors that are ordered (i.e., $a < b$). The CCOPS system supports *constraint cascading*, where information about the resources consumed by SM_a is communicated to SM_b. The cascade is unidirectional and so no information may be passed from SM_b to SM_a. Thus, it is important that SM_a be fully scheduled before any scheduling is done in SM_b. Otherwise incorrect scheduling decisions would be made. Similarly, clusters assigned to SM_a should be rescheduled before those assigned to SM_b.

FUNCTIONS USED TO QUANTIFY THE PROBLEM

An important component of the SPIKE scheduling methodology is based on the notion that constraint information (e.g., "schedule A before B") can be represented numerically as suitability functions. In that spirit, two new measures, *rescheduling priority* and *disruptivity*, are proposed. These functions map preferences related to rescheduling into numeric values so that they can be considered along with other constraints and preferences.

Rescheduling priority

Given a specific scheduling cluster, it is desirable to deduce a numeric preference that can be used via comparison to select such objects for rescheduling.

The term *rescheduling priority* is defined to be the relative measure of how important it is to reschedule a cluster. This priority is a single numeric value determined in the following manner. Each element in a set of criteria is considered. With respect to a specific criterion, the cluster is analyzed, yielding a numeric value, the *subpriority*. All such values are multiplied producing the rescheduling priority. The behavior of the multiply function is such that if any value (determined for a specific criterion) is zero then the rescheduling priority is zero. Each analysis is therefore done with that fact in mind. The criteria that are proposed for consideration are described below.

Partial scheduling of a dependency cluster. In certain cases, it may be that the activities in a proper subset of the scheduling clusters in a dependency cluster are not scheduled. The numeric value associated with such a case is calculated in the following manner: Let C be the number of clusters in a dependency cluster, S_u be the number of unscheduled clusters, and S_s be the number of scheduled clusters. The priority is the ratio S_s/C. This is a subjective measure based on the notion that a dependency cluster that has a higher percentage of scheduled clusters (and thus is closer to being completely scheduled) ought to be processed before a cluster with a lower percentage. The activities in a dependency cluster are linked via constraints and therefore represent a scientific experiment.

Partial scheduling of a proposal. Similar logic utilized in the case of the dependency cluster can be applied to the set of activities in a proposal. Here the scientific value of completion may be even stronger. If C is the number of activities in a proposal, then the same ratio S_s/C is used to find the numeric value for this preference.

Director's priority. Each proposal has an assigned *director's priority* which is either **high** or **supplemental.** Using this information, the subpriority of a cluster is 1 if the director's priority (of the source proposal) is high and 0 otherwise. The current philosophy states that supplemental proposals are not guaranteed scheduling. Since the pool of supplementals is large, this criterion (director's priority) will give supplementals originally not scheduled by SPIKE a chance.

Priority based on repeated SU failures. It is possible that a specific SU will repeatedly fail to be scheduled by SPSS. One reason for this is the oversubscription philosophy mentioned above. It is proposed that a priority value be calculated to capture those iterations for use by the SPIKE rescheduling machinery. If N is the number of times that SPSS has rejected a specific SU, then the *repeat failure priority* is $1/N$ (unless $N \geq$ **threshold**, in which case it is 0). The threshold is currently assigned the value three. For example, if the SU has been rejected by SPSS 2 times, then its priority is 1/2.

Some clusters cannot be rescheduled. If C_i is a cluster to be rescheduled, other components of the dependency cluster (to which C_i belongs) have been either executed by ST or scheduled in the very near term (and hence may not be unscheduled), and if there is no suitable time segment for C_i due to constraints, then it is impossible to reschedule C_i and its priority must be zero.

Disruptivity

The function *SD*(*scheduling cluster, time segment*), the suitability based on disruptivity, is defined to be a relative measure of the effects of scheduling a specific *scheduling cluster* in a specific *time segment.* Such changes would include other clusters, being uprooted and rescheduled. Unlike the rescheduling priority, SD takes the form of the classic suitability function. An SD of numeric value one means that little if any disruption is expected. An SD of zero means that an unacceptably high disruption is predicted. Disruptivity can be calculated by taking the following factors into account:

Disruptivity and estimated propagated effects. If one reschedules an activity, what are the effects on the other activities in the dependency cluster of the activity? The best case scenario would be if no other activities must be moved from their positions on the preexisting schedule. Assuming that one is able to get all other activities back on the schedule, the worst case exists when all other activities in the cluster must be shuffled within the schedule in order to accommodate the activity.

The other important criterion related to determining SD is what happens to the overall suitability of the schedule. The suitability of the old (and now invalid) schedule is the baseline. If the suitability of the new schedule increases or remains constant, then disruption is low and SD is close to one. If the suitability decreases, the SD is less than one.

Resource consumption and disruptivity. Determining the overall summed suitability of a schedule can be used to determine how placing one activity will affect resource consumption. The neural network system maintains a network that encodes how a specific scheduling decision affects other activities based on resources consumed. If a decision is made that causes many activities to become unschedulable based on available resources, then this will be included in the calculation to produce a relatively higher disruptivity.

TWO ALGORITHMS FOR RESCHEDULING AN ACTIVITY

In the following paragraphs, two algorithms that can be used to solve the *single activity rescheduling problem* are proposed. In both algorithms, schedule time is divided, by the concepts of *now* (some time segment), the *past* (all segments lower in ordinality than now), and the *future* (all segments higher than but including now). The selection of *now* should represent not real clock time but instead the point in the real time future that is where one can reasonably make changes to the schedule. For instance, one probably would not want to routinely make changes to a schedule for the next week in real time. However, making changes two months in the future might be acceptable. Therefore, assigning *now* to be a month into the real time future is reasonable.

The first step in either algorithm is to order the list of scheduling clusters to reschedule based on their relative rescheduling priority (see above). The following discussions relate only to rescheduling a single cluster that is selected from such a list.

In both cases disruptivity is calculated in the following manner. If any unschedulables are noted then disruptivity is 1. Otherwise the disruptivity is percent of activities that moved. The suitability based on disruptivity is (1 − disruptivity).

A stochastic NN algorithm

The stochastic NN rescheduling algorithm is based primarily on the notion that enough intrinsic knowledge of the clusters and constraints is stored in the biases and weights of the neural network system to quickly (and optimally) replace an activity on a preexisting schedule. Let A be the cluster that is to be rescheduled. The steps are described below.

1) Freeze the past.
   a) Turn on the neurons in the past that represent accepted commitments (of clusters to segments). Clamp those neurons (with a high bias) so that their state cannot change.
   b) For rows that are in the past that have no neurons on, clamp all neurons in those rows with a negative bias to prevent any neurons there from being turned on.
2) Eliminate the original commitment (of the cluster to be rescheduled) from the

range of commitment possibilities by turning the corresponding neuron off and clamping it with a large negative bias.

3) Turn on all neurons in the future that correspond to legal commitments. The assumption here is that these preexisting commitments are valid and represent a baseline schedule. These neurons are **not** clamped using bias and so during a network run may change state. The underlying logic is this: It is desirable to preserve as much of the preexisting schedule as possible. However, any scheduling decision that lies in the future could be revoked in order to reschedule A. Figure 2 illustrates an example neural network representation at this point.
4) Run the neural network scheduler. Since the clusters in the dependency cluster of A except for A are scheduled, their weighted effects on the network will tend to place A in a legal place (which should be very close in time to its original placement). If such a legal place does not exist (because it is in the past), then some portion (perhaps all) of the dependency cluster must be moved. The more activities that are moved to accommodate the rescheduling of A the more the solution violates our goal of minimized disruptivity.
5) Given a solution determined in step 4, calculate the measure of disruption that has occurred.

An exhaustive algorithm to minimize disruptivity

In this algorithm, the neural network environment and external functions are also used to determine disruptivity. However, in this algorithm an exhaustive search is effected to determine the best possible place(s) where cluster A can be rescheduled. The method used to estimate disruptivity will operate in the following manner:

1) Freeze the past, and turn on legal commitments in the future (same as above).
2) For each time segment in the future (except for the disallowed segment), find the disruptivity that results from scheduling A there.
   a) First turn on the corresponding neuron and set its bias high.
   b) Run the network. The network may cause clusters in the future to move because the immutable commitment of A may cause certain commitments to be inconsistent (based on the weights and biases).
   c) Analyze the disruption.
3) The segment that produces the best (lowest) disruptivity is selected as the place for rescheduling A. In the case of ties, the earliest segment is selected.

Informing SPIKE about disruptivity. Once the suitability of disruptivity has been calculated for a given activity, it may be useful to communicate that inferred knowledge to the core SPIKE scheduling system. The suitability can be integrated into the planning session data structures as an *absolute time constraint*, represented graphically for users, and may guide automatic or manual rescheduling of activities.

PROTOTYPE SYSTEMS AND EXPERIMENTAL RESULTS

We have implemented a prototype based on the discussion above to test whether the behavior of the exhaustive rescheduling algorithm justifies the computational expense relative to the stochastic neural network rescheduling approach.

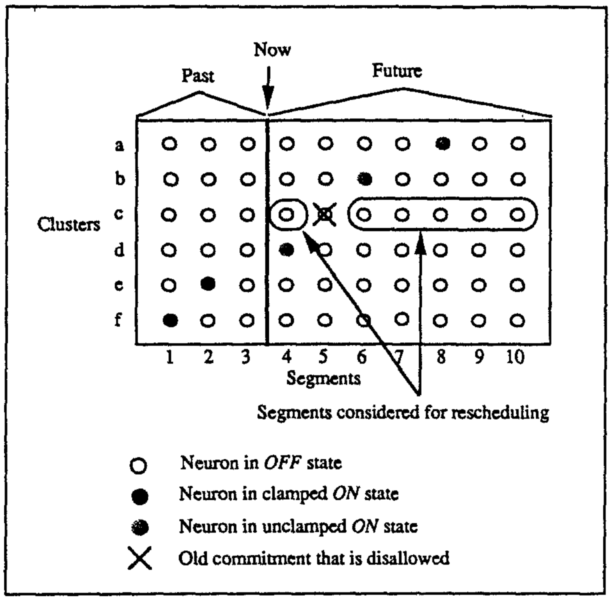

Now
Past
Future
Clusters
a
b
c
d
e
f
1 2 3 4 5 6 7 8 9 10
Segments
Segments considered for rescheduling
Neuron in OFF state
Neuron in clamped ON state
Neuron in unclamped ON state
Old commitment that is disallowed

Figure 2. Neural network representation of a schedule. Each circle in the figure depicts a neuron (representing a possible commitment). Cluster C, in this scenario, had previously been committed to segment 5. That commitment is now illegal. Black circles respresent commitments that are considered unchangeable (because they are in the past). Gray circles represent commitments in the schedule that can be uncommitted (in order to reschedule cluster C). Both algorithms begin by creating a network that has this general organization.

Hypotheses

Let $\mu 1$ be the mean disruptivity of the neural network approach and $\mu 2$ be the mean disruptivity of the exhaustive algorithm:

$$H_0: \mu 1 = \mu 2$$
$$H_1: \mu 1 \neq \mu 2.$$

In a setup that was composed of 60 segments, 30 scheduling clusters, and dependency clusters of size 3, 50 trials were run for both the network rescheduling algorithm and the exhaustive rescheduling algorithm to determine which would find the best place to reschedule, such that disruptivity was minimal. For the network rescheduling algorithm the mean was 0.139 and the standard deviation was 0.318. For the exhaustive rescheduling algorithm, the mean was 0.038 and the standard deviation was 0.141. A 95% confidence interval (0.005, 0.197) for the difference in population mean scores was determined using the *Z statistic* (Bhattacharyya, 1977). Since the interval does not include zero, the null hypothesis is rejected in favor of H_1.

DISCUSSION

The exhaustive approach to rescheduling is to produce better results. Statistics reveal that the differences however are not great and so one might argue that the computation involved in the exhaustive approach is too costly given the marginal benefit. Although a large number (50) of trials were executed, the algorithms were only tested on a single problem. More testing on a varied set of problems is required to more accurately assess the comparative usefulness of these approaches. It is also possible that the selected parameters (e.g., number of links, position in time of the rejected cluster) may have biased the results. Again, only more tests will tell.

It is considered odd that the exhaustive system was only a little better statistically. First, one may argue that statistical measures are designed to be conservative with respect to supporting differences that result from varying treatments. Another important point is that, in general, the best place to reschedule a cluster is another point in time that is close to the original segment. If such a place exists that is legal (based on constraints), then the stochastic algorithm should find this solution. It is when that nearby place is not legal that the exhaustive algorithm should prevail because the stochastic algorithm will then find any legal configuration without regard for disruptivity.

It is believed that the basic approach described in this report is sound and when fully implemented will provide an effective mechanism to repair broken schedules when that need arises.

REFERENCES

Bhattacharyya, G., & Johnson, R. (1977). *Statistical concepts and methods.* New York: John Wiley & Sons.

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

Hall, D., ed. (1982). *The space telescope observatory,* (NASA CP2244).

Johnston, M. (1990, March). SPIKE: AI scheduling for NASA's Hubble Space Telescope. To appear in *Proc. of the Sixth IEEE Conference on Artificial Intelligence Applications.*

Johnston, M., & Adorf, H.-M. (1989). Learning in stochastic neural networks for constraint satisfaction problems. *Proc. NASA Conf. on Space Telerobotics.*

Miller, G., Johnston, M., Vick, S., Sponsler, J., & Lindenmayer, K. (1988). Knowledge based tools for Hubble Space Telescope planning and scheduling: Constraints and strategies. *Proc. 1988 Goddard Conference on Space Applications of Artificial Intelligence*; reprinted in *Telematics and Informatics 5,* 197 (1988).

Acknowledgments—The authors would like to thank the following persons for their ideas concerning this report: Dr. Glenn Miller and Shon Vick.