Proceedings of the 1991 IEEE
International Conference on Robotics and Automation
Sacramento, California - April 1991

Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture

Nicola Muscettola, Stephen F. Smith, Amedeo Cesta[1], Daniela D'Aloisi[2]

Center for Integrated Manufacturing Decision Systems
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper we describe HSTS, an integrated planning and scheduling architecture that has been applied to the problem of generating observation schedules for the Hubble Space Telescope. The solution of problems like this is complicated by the interaction of resource allocation and auxiliary task expansion during schedule development. HSTS deals with this interaction by viewing planning and scheduling as two complementary aspects in the construction of a behavior of a system. First we describe how HSTS specifies the dynamics of a system, how it represents schedules at multiple levels of abstraction, and the specific problem solving machinery it provides. Then we give an example of the use of the architecture in the Hubble Space Telescope domain. Finally, we give performance results that indicate the practicality of the HSTS approach.

1. Introduction

Automated robotic systems offer new opportunities in many diverse areas, from flexible manufacturing to space exploration. However, the extent to which their potentiality can be actually exploited depends on effective management of their complex operations. The specific domain of focus in our research - coordinating the use of the Hubble Space Telescope (HST) - is a good example of such complex management problems.

The efficient operation of automatic systems like HST requires attendance to several classes of interacting constraints. On one hand, it is necessary to reconcile a large and conflicting set of user requests with the overall objective of optimizing resource utilization. At the same time, allocation decisions must satisfy a series of physical constraints, insuring that the various components of the system and its environment will behave in mutually compatible ways.

Research in scheduling and planning has variably addressed different aspects of this problem. Recent scheduling research [3] [9] [8] deals with efficiently allocating resources to competing activities under conflicting objectives; however this has been done under simplified representational assumptions regarding the behavior of the underlying physical system. Research in planning [11] [12] [2] alternatively, has focused on generation of courses of action from more general representations of physical constraints, but has ignored issues of global resource optimization and, with few exceptions [5], has not attempted to exploit the inherent structure of the underlying physical system.

In this paper, we describe HSTS, an integrated planning and scheduling architecture. HSTS is based on a uniform view of planning and scheduling as a process of constructing a behavior of a dynamical system [6]. HSTS provides three principal capabilities:

- a *domain description language* for modeling the structure and dynamics of the physical system at multiple levels of abstraction;
- a *temporal data base* for representing possible evolutions of the state of the system over time;
- a *scheduling/planning methodology* for integrating decision-making at multiple levels of abstraction.

Within the HSTS architecture, we have addressed the problem of constructing short-term executable observation schedules for HST. In the rest of the paper, we first describe the HST operating environment (Section 2. Then we present the description language (Section 3) and illustrate its use by detailing the current model of the HST operating environment (Section 4). Next, we give an overview of the HSTS scheduling and planning framework (Sections 5 and 6), followed by a description of the heuristics employed by the current HST scheduler (Section 7). We then present performance results obtained on a realistically-sized scheduling problem (Section 8). Finally, we briefly discuss the implications of our solution (Section 9).

---

[1]Current Address: Dipartimento di Informatica e Sistemistica, Universita' di Roma "La Sapienza",Via Salaria 113, 00198 Roma (Italy).

[2]Current address: Fondazione Ugo Bordoni, Via B. Castiglione 59, 00142 Roma (Italy).

2. The HST Domain

Before we introduce the HSTS approach to modeling complex systems, let us give a brief overview of the HST operating environment.

Considered as a whole, HST is a very sophisticated "sensor"; it gathers light from celestial objects, named targets, and communicates scientific data back to Earth through one of two TDRSS communication satellites (Figure 2-1). Since the telescope is in low earth orbit, both the targets and the satellites are periodically occulted by the Earth.

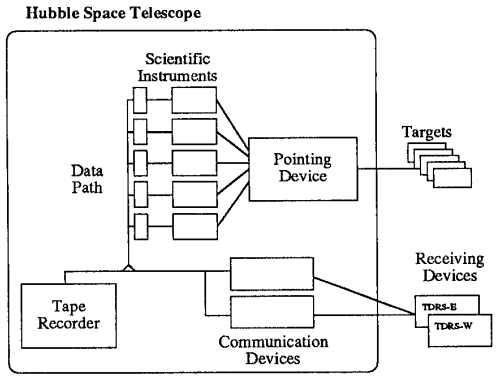

Figure 2-1: The Hubble Space Telescope Domain

The telescope itself consists of several components; their behavior must be coordinated over time due to the limitation of available electric power, the need to maintain acceptable temperature profiles on the telescope structure, etc. The pointing device represents the HST subsystem responsible for orienting it in the direction of a target and locking the target at the center of the field of view of a designated scientific instrument. HST has 6 different scientific instruments. Limitations on available electric power prevent all of them from being switched on simultaneously. Data can be read from the instruments and directly communicated to Earth through one of two links operating at different communication rates (1Mb/sec and 4kb/sec); data can also be temporarily stored on an on-board tape recorder and communicated to Earth at a later time.

Astronomers formulate observation programs according to a fairly sophisticated specification language [10]. The basic structure of each program is a partial ordering of observations, each specifying the collection of light from a celestial object with one of the telescope's six scientific instruments. A diverse set of temporal constraints can be imposed on the observations in a program, including precedences, windows of opportunity for groups of observations, minimum and maximum temporal separations, and coordinated parallel observations with different viewing instruments.

Solving the HST observation scheduling problem requires the generation of the sequence of commands needed both to carry out observations (e.g., "take an exposure", "communicate to Earth data from instrument X") and to reconfigure the telescope to enable observation execution (e.g., "point telescope to target Y", "switch on instrument Z").

3. Modeling Complex System Dynamics in HSTS

In this section we introduce the primitives of the HSTS Domain Description Language (HSTS-DDL). The HSTS-DDL supports modelling of a system at multiple levels of abstraction as a collection of interacting components.

An HSTS model is subdivided into a set of **system components**, each of which has an associated set of **properties**. At any instant of time, each property of the system can have one and only one associated **value**. Some properties are **static**, i.e., their value does not change over time. Others are **dynamic**, i.e., usually their values change over time; in the following we will also refer to these as **state variables**.

The HSTS Domain Description Language requires explicit declaration of the set of possible values that can be assumed by each dynamic property. In general, a value has the form $R(x_1, x_2, ..., x_n)$, where $<x_1, x_2, ..., x_n>$ represents a tuple in the relation R.

A **behavior** of the system is an evolution over time of the values of its state variables: it is completely specified once each state variable has an associated value for each instant of time. In a behavior, each state variable changes its value a discrete number of times and a value persists for a continuous interval of time.

To fully describe a value, HSTS-DDL requires the specification of its **duration**, a constraint on its temporal length due to the value's intrinsic characteristics. A duration is a pair of temporal distances $[d, D]$, $D \geq d \geq 0$ where d and D are respectively the lower bound and the upper bound of the duration: in general both may be functions of some arguments of the value.

The complexity of the dynamics of a system stems from the interactions between different state variables. In fact, a value can be present in a behavior of the system only if well-defined patterns of values occur over time on the state variables. In HSTS-DDL these patterns are specified by associating a **compatibility specification** with each value. A compatibility specification consists of one or more sets of compatibilities organized as an AND/OR graph. A **compatibility** associated to a (constrained) value specifies how this value must be temporally related to another (constraining) value or sequence of values. More precisely, a compatibility is a 4-tuple:

<comp–class, st–var, type, temp–rel>

where *comp–class* is either the symbol VALUE or the symbol SEQUENCE, indicating whether the compatibility involves a single constraining value or a constraining

sequence of contiguous values; *st–var* is the state variable of the constraining value or sequence; *type* indicates the subset of values from which the constraining value or sequence is extracted; and *temp–rel* is a temporal relation that specifies a pattern of distance constraints. For example, the temporal relation *before*($[d, D]$) means that the end of the constraining value/sequence must precede the start of the constrained value by an interval of time δ, such that $d \leq \delta \leq D$. The relation *contains*($[d_1, D_1], [d_2, D_2]$), indicates that the constrained value must be contained within the constraining value/sequence; $[d_1, D_1]$ defines the distance between the two start times and $[d_2, D_2]$ defines the distance between the two ends. In Section 4 we will give several examples of compatibility specifications.

A model represented in HSTS-DDL can be subdivided into a number of **layers of abstraction**. An abstract model consists of system components and state variables that aggregate several components and state variables at more detailed levels. The relationship among the layers is established by **refinement descriptors** that map some of the values associated with an abstract layer into a network of values associated with the immediately more detailed layer. The mapping also specifies the correspondence between the start and end times of each abstract value and those of the corresponding detailed values.

4. The HST Domain Model

Within HSTS, we have developed a model of the HST operating environment. At present, it includes 2 of the 6 HST scientific instruments, telescope pointing, data communication to Earth and the on-board tape recorder. The components of this model are summarized below.

The environment external to the telescope consists of targets and TDRSS satellites. For each of them, a distinct state variable characterizes its visibility with respect to the space telescope.

The possible values of the pointing apparatus (state variable HST-POINTING) are: *LOCKED*($?T$), *UNLOCKED*($?T$), *LOCKING*($?T$), and *SLEWING*($?T1, ?T2$). Here $?T$, $?T1$, and $?T2$ are variables denoting possible targets. For example, *SLEWING*($?T1, ?T2$) represents the movement of the telescope from the direction pointing to $?T1$ to that pointing to $?T2$. Figure 4-1 shows the possible transitions among values. An edge connects two nodes if the corresponding values are related by a *before*([0,0]) compatibility. The highlighted nodes represent "stable values", i.e., values with intrinsic duration $[0, +\infty]$; all the other nodes represent states with finite durations.

During telescope repointing, some of the values can occur only while a target is visible. More precisely, a window of visibility for target $?T$ must necessarily contain the occurrence of a *LOCKING*($?T$) operation and of a *LOCKED*($?T$) state. Figure 4-2 lists the complete compatibility specification for *LOCKED*($?T$). It indicates that in every correct behavior of HST, *LOCKED* can

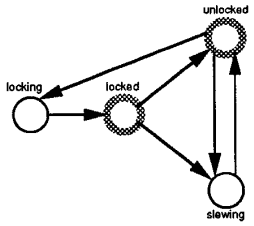

Figure 4-1: Value transition graph for HST-POINTING

appear on HST-POINTING either in a sequence *LOCKING*, *LOCKED*, *UNLOCKED* or in a sequence *LOCKING*, *LOCKED*, *SLEWING*, together with the constraint of co-occurrence between *LOCKED* and a target visibility window.

```
AND [ < VALUE, HST-POINTING, LOCKING (?T), before([0,0])>,
      < VALUE, VISIBILITY (?T), VISIBLE, contains([0, +∞], [0, +∞])>,
      OR [ < VALUE, HST-POINTING, UNLOCKED (?T), after ([0, 0])>,
           < VALUE, HST-POINTING, SLEWING (?T), after ([0, 0])> ] ]
```

Figure 4-2: Compatibilities for *LOCKED*($?T$)

Of the two instruments currently modeled, the Wide Field Planetary Camera is the most complex. It consists of two different sets of CCD sensors, or detectors, the Wide Field Camera (WF) and the Planetary Camera (PC). Both share the same support module that provides them with temperature control, optical filters reconfiguration operations, etc.; we will denote the support module with WFPC. Each of the three components of the Wide Field Planetary Camera is modeled as a separate state variable. Figure 4-3 shows the value transition graph for the WFPC-STATE state variable; similar graphs describe the value transitions of WF-STATE and PC-STATE.

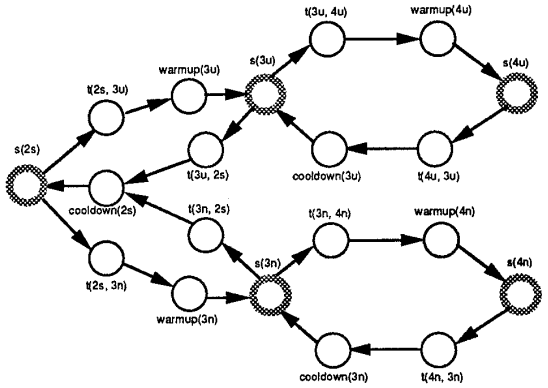

Figure 4-3: Value transition graph for WFPC-STATE

The value *s(2s)* represents the state where the Wide Field Planetary Camera is switched off, and the values *s(4u)* and *s(4n)* represent fully operational states. Both WF-STATE and PC-STATE can assume, as a possible value, *EXPOSE*(*?config*, *?target*, *?duration*), which represents a picture taking operation; here the variable *?config* can be

either $4u$ or $4n$, depending on the operational configuration required on the WFPC. Figure 4-4 expresses the compatibilities that have to be satisfied for the occurrence of *EXPOSE*(4*n*, ?*target*, ?*duration*) on the WF-STATE.

```
AND [ < VALUE, HST-POINTING, LOCKED (?target), contains ([0, +∞], [0, +∞])>,
      < VALUE, PC-STATE, S (2s), contains ([0, +∞], [0, +∞])>,
      < VALUE, WFPC-STATE, S (4n), contains ([0, +∞], [0, +∞])> ]
```

Figure 4-4: Compatibilities for *EXPOSE*(4*n*, ?*T*, ?*D*)

Further physical constraints require synchronization of the warmup and cooldown processes of each of the three instrument components. For example while the WF-STATE has value *s(3n)*, WFPC-STATE can undergo any warm-up or cool-down sequences between *s(3n)* and *s(4n)*; this is represented as a compatibility associated with the *s(3n)* of the WF-STATE (Figure 4-5).

```
< SEQUENCE,
  WFPC-STATE,
  { S (3n), T (3n, 4n), WARMUP (4n), S (4n), T (4n, 3n), COOLDOWN (3n) },
  contains ([0, +∞], [0, +∞])>
```

Figure 4-5: A sequence compatibility

Data are read out of a detector either by the tape recorder or by one of the two communication links. The values of TAPE-RECORDER-STATE indicate that the quantity of stored data increases at each read-out operation and that the content of the tape recorder has to be "dumped" to Earth when a given capacity threshold is exceeded. The read-out operations on the communication links and the dump operation on the tape recorder must occur during a window of visibility of one of the two TDRSS satellites.

To provide a global perspective on HST operation, the detailed model just described is augmented with an abstract model. Here the whole telescope is described as a single state variable, HST-STATE. The possible values of the abstract state variable are *OBSERVE*(?*P*, ?*I*, ?*S*, ?*T*, ...) (where ?*P* designates an observation program, ?*I* designates a viewing instrument, ?*S* designates the required operating state of ?*I*, and ?*T* designates a target), *RECONFIGURE*(?*FROM–I*, ?*TO–I*, ...), which represent the reconfiguration process between two observations, and *IDLE*, which represent a stationary state of the telescope where all detectors are off and the tape recorder is empty. The value *RECONFIGURE*(?*FROM–I*, ?*TO–I*, ...) provides an abstract description of an entire segment of detailed behavior; its duration can be determined by analyzing the network of values that implements it at the detailed level.

Correspondence between abstract and detailed layers is insured by a refinement descriptor that maps an abstract *OBSERVE*(?*P*, ?*I*, ?*S*, ?*T*, ...) value into the detailed *EXPOSE* and *READ–OUT* values that actually implement it.

5. The Temporal Data Base

In HSTS, the construction of an executable schedule is viewed as an incremental process of constraint posting and propagation on a central data base. The HSTS Temporal Data Base extends the philosophy of the time map formalism developed in [1] by tightly connecting the state of the data base and the model of a system. This association provides a strong basis to support planning and enforce data base consistency.

The unit of description of temporal behavior is the **token**, a quadruple <*state–variable*, *type*, *st*, *et*,>, where *state–variable* is the identifier of one of the state variables in the system model, *type* is a subset of the state variable values, and *st* and *et* are two variables representing the token's start and end times respectively. The meaning of a token is that *state–variable* assumes some values included in *type* within *st* and *et*.

We distinguish two kinds of tokens depending on the cardinality of their *type*:

1. *value token*: *type* is a singleton set. A value token indicates that, within *st* and *et*, *state–variable* assumes the constant value in *type*.
2. *constraint token*: *type* is a multiple value set. A constraint token specifies that, within *st* and *et*, *state–variable* must assume a sequence of values of indefinite length (possibly empty) each belonging to *type*. Since a constraint token can be substituted with any sequence of value tokens satisfying the type constraint, it enables the description of partially specified evolutions of a state variable over time.

As in [1], the *st* and *et* of each token are represented as nodes in a network of constraints. A directed edge labelled with the pair $[d, D]$ from time node t_i to t_j indicates the existence of a relative temporal constraint between t_i and t_j that restricts their distance to vary within the interval $[d, D]$. Absolute temporal constraints on a time node are represented as relative distances from a *reference* time constant, a time node whose value is fixed to the origin of the temporal axis.

Since a value can appear in the evolution of a state variable only if it satisfies the constraints imposed by the physics of the system, the HSTS-TDB associates with each value token an instance of the compatibility specification graph associated with the value.

The HSTS temporal data base is subdivided into a number of communicating layers, each corresponding to an abstraction layer of the system model. In each layer, we can distinguish two networks of tokens: a *goal network* and a *behavior network*. The *goal network* specifies the requirements imposed by the problem on the evolution of the system's state variables over time; for example, when an HST observation scheduling problem is formulated, unscheduled observation programs are represented as goal networks. Each elementary goal is a

value token. Goals can be related by the same kind of temporal relations used to specify compatibilities. The interpretation of refinement specifications generates corresponding goal networks in adjacent layers of the data base.

Planning at a given level of abstraction consists of repeatedly selecting goal tokens and building system behaviors that achieve them. The construction of system behaviors proceeds in a separate token network, the *behavior network*. A behavior network is a set of linear sequences of tokens, one for each state variable. Each sequence covers the entire scheduling horizon; in fact, the *st* of the first token and the *et* of the last token occur respectively at the beginning and at the end of the scheduling horizon, and the *et* of a token is identical to the *st* of the following token in the sequence. If a segment of state variable behavior is only partially specified in a given state of the data base, it will be covered by one or more constraint tokens.

During the planning process, each layer of the data base is repeatedly refined through the application of three basic data base modification operations:

1. *commitment on the achievement of a goal.* This results in the insertion of a value token belonging to the goal network into the behavior network. This can be achieved in two ways: a) merging the goal token with a matching value token in the behavior network, or b) inserting the goal token into a compatible constraint token, i.e., one whose type contains the type of the goal token. After the operation, the two networks share the value token; therefore any constraint subsequently imposed on one network will propagate to the other (e.g., the expansion of an auxiliary task in the behavior network will affect how to achieve the remaining goals).
2. *value compatibility implementation.* Given a value compatibility belonging to a value token in the behavior network, first a value token is either selected from the existing ones or created by refining a constraint token. This designated token is then connected to the token to be constrained according to the specified temporal relation. A compatibility can be consistently implemented if the behavior contains tokens that satisfy both the compatibility's type and temporal relation. Type consistency requires the type of a token to have a non empty intersection with the type of the compatibility. Temporal consistency requires the network of temporal constraints not to contain cycles of distance links with total length necessarily different from 0. Figure 5-1 summarizes the process of implementing a *contains* compatibility.
3. *sequence compatibility implementation.* This operation is analogous to the previous one. In this case the temporal relation connects the constrained value token to a sequence of value and constraint tokens whose type matches the compatibility type constraint. The type of each constraint token in the constraining sequence must now also satisfy the compatibility type constraint.

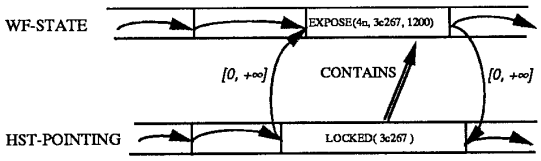

Figure 5-1: Implementation of a *contains* compatibility

6. The Planning Process

To manage the combinatorics of planning and scheduling for complex systems, planning at multiple levels of abstraction can provide significant leverage. Abstract views provide a basis for globally focussing the detailed planning effort. Similarly, detailed views provide a basis for sharpening abstract predictions. The HSTS planning and scheduling methodology supports flexible integration of planning in different layers of the temporal data base.

The objective of planning/scheduling at a given layer of abstraction is to transfer goal tokens from the goal network into the behavior network and generate a consistent system behavior that achieves these goals. This process of commitment and behavior generation is carried out incrementally by repeated applications of the HSTS planning procedure depicted in Figure 6-1.

```
PlanningProcedure (GoalNet, BehaviorNet)

  (1) select some goals from GoalNet;
  (2) insert selected goals in BehaviorNet;

  {repeat

     (3) select an open compatibility
         in BehaviorNet;
     (4) implement selected compatibility

   until no more open compatibilities}
```

Figure 6-1: The planning procedure

In order to instantiate the planning procedure for a particular domain, we must determine several parameters. All 4 steps in Figure 6-1 typically require choices among alternatives. For example, a compatibility can be implemented in several ways. In fact, in general we can select the constraining token or sequence of tokens in different positions with respect to the planned evolution of

a state variable. The selection criterion, however, depends on the characteristics of both the application domain and the problem to be solved. The HSTS architecture provides just a skeletal heuristic search methodology from which we can obtain actual algorithms by introducing selection and pruning heuristics. In Section 7 we will discuss the heuristics presently used by the HST observation scheduler.

Each representational layer has an associated planning process. A planning process exchanges information with its two adjacent layers (or to the external world if a layer does not exist). Given two adjacent layers i and i+1, where i is less abstract than i+1, i+1 can communicate to i a refined goal network and preferences on how the goals should be achieved (e.g., "achieve all the goals as soon as possible"). Process i communicates back to level i+1 temporal information on the possible occurrence of the abstract goals corresponding to the refined goal network.

At its core, a planning process repeatedly calls the planning procedure described before, using heuristics that are most suitable to the characteristics of the model it operates on. When a procedure returns, the process has to decide whether to continue planning at this level or to communicate information to one of the adjacent layers and pass the control to the corresponding process. Processing stops when either all the goals communicated by the user have been achieved or it has been determined that no more goals can be achieved.

The development of a scheduler for an application domain also requires the definition of the pattern of coordination among its planning processes.

7. Observation scheduling in the HST domain

In this section we describe the scheduler for the HST domain currently implemented in the HSTS architecture. Section 8 reports some performance results obtained with the program on realistically-sized observation scheduling problems.

Given the two-level model described in Section 2, the current HSTS observation scheduler has two planning processes acting on each layer of a two-layer temporal data base. Planning at the abstract level has responsibility for determining the sequence of observations to be executed; planning at the detailed level is responsible for developing a detailed system behavior that implements this observation sequence.

To make use of more precise information about actual telescope reconfiguration durations as scheduling proceeds, decision-making at the abstract level and at the detailed level are tightly coupled. Each time an observation is selected and inserted into the abstract behavior network, on the basis of the abstract estimates of telescope reconfiguration times, control is passed to detailed planning that expands the detailed setup tasks to achieve the new expose and communicate goals. Abstract planning currently utilizes a dispatch-based approach (see Section 7.1 below); on each iteration, an additional observation is appended to the abstract behavior extending the schedule strictly forward in time.

The heuristics employed by each planning process reflects an overall objective of maximizing the amount of time during which the telescope actually collects scientific data. These heuristics are described in the rest of the section.

7.1. Sequencing heuristics

At the abstract level, a single state variable characterizes the behavior of the telescope and there is only one possible path through its associated value transition graph. The complexity of decision-making at this level lies entirely in goal selection.

Selection of the next goal to achieve is accomplished by a local greedy heuristic designed to minimize dead time between observations. Specifically, a one-step look-ahead search is performed where each of the eligible candidates in the goal network are hypothesized as the next observation to be executed. A goal may be ineligible due to ordering constraints with other unachieved goals or due to absolute time constraints. Among the eligible goals, the one that yields the estimated earliest start time is identified. However, selection of this goal might prevent the subsequent achievement of one or more other goals due to some temporal constraint in the network. If this is not the case, the earliest start goal becomes the final selection; otherwise, the temporally constrained goal with the highest priority is selected, and the remaining unachievable goals are removed from the goal network.

7.2. Detailed planning heuristics

At the detailed level, the selection of goals results from the refinement of the choice made at the abstract level. However, heuristics guide the compatibility selection and implementation decisions.

7.2.1. Selection of a compatibility to implement

The compatibility specification graphs of the detailed model contain OR nodes. Once these graphs are instantiated in the temporal data base, the planner must select only one of the alternative OR branches for implementation. Heuristics perform this selection. All compatibilities that remain after this selection process must be achieved to obtain a complete plan. Therefore, we need an additional mechanism to select the order in which they will be implemented. The order determines how the topology of the behavior network evolves, a factor that directly influences the effort required to implement successive compatibilities.

Let us give an example of the OR compatibility selection heuristic. The graph describing value adjacencies can present branching (as in the case of value $s(3n)$ for the WFPC-STATE (Figure 4-3). The current HST model presents this situation for all the state variables

associated with the two scientific instruments, and for the telescope pointing state. The topology of each of these graphs, however, grants the existence of a single acyclical path for each ordered pair of values. The existing interactions between the durations of sequences of values and the compatibility temporal relations in the HST model assure that the minimal length path correctly coordinates on time with other state variables. The heuristic therefore consists of a table, indexed by a pair *< branching–value, destination–value >*, where each entry represents the first value encountered in the acyclical path starting from the branching value and reaching the destination value.

For another example, let us consider the communication of data to Earth through the fast communication link (1Mb-link); this process requires locking the telescope onto one of the two TDRSS satellites. To minimize idle time, an heuristic chooses the satellite that allows scheduling of the communication operation as soon as possible after the corresponding exposure. The selection considers how the visibility windows of the target required by the exposure overlap with those of the two alternative satellites, together with the kind of synchronization between expose and communicate (e.g., *before*, *contains*).

The mechanism that selects the order of compatibility implementations gives priority to compatibilities that relate values on the same state variable. Whenever such a compatibility is open, a gap (i.e., a constraint token) exists among two values on a state variable. The planner continues to extend the sequence of values on the state variable until the gap is closed.

If no compatibility that relates values on the same state variable is open, then the planner selects a compatibility that involves different state variables. The open "cross" compatibilities among each pair of state variables are organized in a separate queue. The selection mechanism chooses a queue and tries to achieve all the contained compatibilities one by one. However, the satisfaction of a cross compatibility might create a gap on a state variable. In this case, the dequeuing of cross compatibilities is suspended until the new gap is closed.

7.2.2. Implementation of a compatibility

The implementation of a compatibility requires the determination of the position of the requested value or sequence in the behavior network.

A fetch operation linearly scans the sequence of tokens associated to the required state variable in the behavior network; as a result it returns a list of candidate tokens or sequences of tokens. HSTS uses look-ahead temporal constraint propagation to detect inconsistencies. Mechanisms are provided to limit such propagation to a subnetwork of temporal constraints. By relying on the decomposition of the model into interacting state variables, it is possible to associate each cross compatibility with a set of tightly coupled state variables. The look-ahead constraint propagation will limit the search for cycles to these state variables.

Among the alternatives returned by the fetch operation, a heuristic gives priority to the alternative that is temporally closest to the constrained value, given the goal of minimizing the time spent in reconfiguring the telescope.

8. Performance Results

We conducted experiments with three models of the HST operating environment of increasing complexity and realism, respectively denoted as SMALL, MEDIUM and LARGE model. All models share a representation of the telescope at the abstract level as a single state variable; they differ with respect to the number of components modeled at the detailed level. The SMALL model consists only of the telescope pointing device and the Wide Field Planetary Camera. The MEDIUM model adds the two state variables for the Faint Object Spectrograph to the previous model, while the BIG model includes also data communication and tape recorder management. The test problem consists of a set of 50 observation programs, each containing a single observation with no user-imposed time constraints. The experiments were run on a TI Explorer II+ with 16 Mbytes of RAM memory.

The data in Table 8-1 give some measures relative to the final executable schedule that was produced with each model. The number of tokens indicates the total number of distinct state variable values that constitute the schedule. The temporal separation constraints are distance constraints that relate two time points on different state variables; their number gives and indication of the amount of synchronization needed to coordinate the evolution of the state variables in the schedule.

With respect to the processing times, notice that since the heuristics that guide the planning search exploit the modularity of the model and the locality of interactions, the average CPU time (excluding garbage collection) spent implementing each required compatibility in the three models remains relatively stable. The total elapsed time (including garbage collection) spent generating an executable schedule for the 50 observations is an acceptable fraction of the time horizon covered by the schedules; this indicates the practicality of the framework in the actual HST operating environment.

	SMALL	MEDIUM	LARGE
State Variables	4	6	13
Tokens	587	604	843
Temporal Constraints	1296	1328	1474
Shedule Horizon	41h37' 20"	54h 25' 46"	52h 44' 41"
CPU Time/Observation	11.62"	12.25"	21.74"
CPU Time/Compatibility	0.29"	0.29"	0.33"
Total CPU time	9' 68"	10' 52"	18' 12"
Elapsed Time	1h 08'36"	1h 13' 16"	2h 34' 07"

Table 8-1: Performance results

With regard to the quality of the schedules, the percentage of time spent taking exposures with respect to the time horizon covered varies between 22% and 25%. This result is comparable with general expectations of telescope use. However further research is needed to represent the full extent of the actual operating constraints (e.g. treatment of availability of electric power) and to devise strategies to generate efficient schedules for problems involving more complex program constraints. To this end, we are currently investigating the integration of global problem space analysis and focusing techniques [7] [8] [4].

9. Conclusions

To efficiently operate complex robotics systems, it is necessary to allocate system resources to competing user requests while coordinating system reconfiguration activities. HSTS addresses both concerns within an integrated planning and scheduling architecture. HSTS has been applied to the Hubble Space Telescope observation scheduling domain. Its modeling capabilities allow explicit representation of the various system components and their interactions over time; this leads naturally to a framework for incrementally addressing complex problems, through development of solutions to a series of increasingly more realistic scenarios. The structural characteristics of the model can be exploited by heuristics that guide the coordinated planning processes; this allows the design of efficient planning and scheduling algorithms operating at multiple levels of abstraction. Finally, the evolving schedule is represented as an explicit network of constraints; the flexibility remaining in the final solution (e.g., on the start and duration of various activities) eases the adjustment of a schedule when reacting to unexpected external events.

Acknowledgements

We thank Gilad Amiri and Dhiraj Pathak for their contributions to the HSTS project. This work was sponsored in part by the National Aeronautics and Space Administration under contract # NCC 2-531 and the Robotics Institute. Amedeo Cesta was supported by a scholarship of the Italian National Research Council. Daniela D'Aloisi carried out her work in the framework of the agreement between the Italian P.T. Administration and the Fondazione Ugo Bordoni.

References

[1] Dean, T.L., McDermott, D.V., Temporal Data Base Management, *Artificial Intelligence* 32:1-55, 1987.

[2] Dean, T., Firby, R.J., Miller, D., Hierarchical Planning Involving Deadlines, Travel Time, and Resources , *Computational Intelligence*, 4:381-398, 1988.

[3] Fox, M.S., Smith, S.F., ISIS: A Knowledge-Based System for Factory Scheduling, *Expert Systems*, 1(1):25-49, 1984.

[4] Johnston, M.D., SPIKE: AI Scheduling for NASA's Hubble Space Telescope, in *Proceedings of the 6th Conference on Artificial Intelligence Applications*, pages 184-190, IEEE Computer Society Press, 1990.

[5] Lansky, A., Localized Event-based Reasoning for Multiagent Domains, *Computational Intelligence*, 4:319-340, 1988.

[6] Muscettola, N., *Planning the Behavior of Dynamical Systems*, Technical Report CMU-RI-TR-90-10, The Robotics Institute, Carnegie Mellon University, 1990.

[7] Muscettola, N., S.F. Smith, A Probabilistic Framework for Resource-Constrained Multi-Agent Planning, in *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1063-1066, Morgan Kaufmann, 1987.

[8] Sadeh, N, Fox, M.S., Variable and Value Ordering Heuristics for Activity-based Job-shop Scheduling, in *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Island, S.C., 1990.

[9] Smith, S.F., Ow, P.S., Muscettola, N., Potvin, J.Y., Matthys, D., An Integrated Framework for Generating and Revising Factory Schedules, *Journal of the Operational Research Society*, 41(6):539-552, 1990.

[10] STScI, *Proposal Instructions for the Hubble Space Telescope*, Technical Report, Space Telescope Science Institute, 1986.

[11] Vere, S., Planning in Time: Windows and Durations for Activities and Goals, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5, 1983.

[12] Wilkins, D.E., *Practical Planning*, Morgan Kaufmann, 1988.