N94-34046

Constraint-Based Integration of Planning and Scheduling for Space-Based Observatory Management*

Nicola Muscettola and Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
phone: (412) 268-8811
net: sfs@isl1.ri.cmu.edu

Introduction

The generation of executable schedules for space-based observatories is a challenging class of problems for the planning and scheduling community. Existing and planned space-based observatories vary in structure and nature, from very complex and general purpose, like the Hubble Space Telescope (HST), to small and targeted to a specific scientific program, like the Sub-millimeter Wave Astronomy Satellite (SWAS). However, they all share several classes of operating constraints including periodic loss of target visibility, and limited on-board resources like battery charge and data storage.

The complexity of these problems stems from two sources. First, the execution of astronomy observation programs requires the solution of a classical scheduling problem: objectives relating to overall system performance must be optimized (e.g., maximization of return of science data) while satisfying a diverse set of constraints. These constraints relate to both the observation programs to be executed (e.g., precedence and temporal separation among observations) and observatory capacity limitations (e.g., observations requiring different targets cannot be executed simultaneously). Second, a safe mission requires the detailed description of all transitions and intermediate states that support the achievement of observing goals. Such description must guarantee consistency with respect to the detailed dynamics of the observatory; this constitutes a classical planning problem.

Another characteristic of the problem is its large scale. The size of the pool of observations to be performed on a yearly horizon can range from thousands to even tens of thousands. Large observatories can consist of several tens of interacting system components with complex interacting dynamics. To effectively deal with problems of this size, it is essential to employ problem and model decomposition techniques. In certain cases, this requires an ability to represent and exploit the available static structure of the problem (e.g., interacting system components). In other cases an explicit structure is not immediately evident (e.g., interaction among large numbers of temporal and capacity constraints); therefore the problem solver should be able to dynamically focus on different parts of the problem, exploiting the structure that emerges during the problem solving process itself.

In this paper, we report on our progress toward the development of effective, practical solutions to space-based observatory scheduling problems within the HSTS scheduling framework. HSTS was developed and originally applied in the context of the HST short-term observation scheduling problem. Our work was motivated by the limitations of the current solution and, more generally, by the insufficiency of classical planning and scheduling approaches in this problem context. HSTS has subsequently been used to develop improved heuristic solution techniques in related scheduling domains, and is currently being applied to develop a scheduling tool for the upcoming SWAS mission. We first summarize the salient architectural characteristics of HSTS and their relationship to previous scheduling and AI planning research. Then, we describe some key problem decomposition techniques underlying our integrated planning and scheduling approach to the HST problem; research results indicate that that these technique provide leverage in solving space-based observatory scheduling problems. Finally, we summarize more recently developed constraint-posting scheduling procedures and our current SWAS application focus.

Planning and Scheduling for Space-Based Observatories

The management of the scientific operations of the Hubble Space Telescope is a formidable task. Its solution is the unique concern of an entire organization, the Space Telescope Science Institute (STScI). The work of several hundred people is supported by several software tools, organized in the Science Operations Ground System (SOGS). At the heart of SOGS is a FORTRAN-based software scheduling system, SPSS. The original task of SPSS was to take astronomer viewing programs for a yearly period as input and produce executable spacecraft instructions as output, with minimal intervention from human operators. SPSS has had a somewhat checkered history [Wal89], due in part to the complexity of the scheduling problem and in part to the difficulty of developing a solution via traditional software engineering practices and conventional

*The work reported in this paper has be supported in part by the National Aeronautics and Space Administration, under contract NCC 2-531, and by the Advanced Research Projects Agency under contract F30602-90-C-0119.

programming languages. To confront SPSS's computational problems, STScI has developed a separate, knowledge-based tool for long term scheduling called SPIKE [Joh90]. SPIKE accepts programs approved for execution in the current year and partitions observations into weekly time buckets. Each bucket constitutes a smaller, more tractable, short-term scheduling problem. Detailed weekly schedules are then generated through the efforts of a sizable group of operations astronomers, who interactively utilize SPSS to place observations on the time line.

In the HSTS project we have addressed the short term problem in the HST domain, i.e., the problem of efficiently generating detailed schedules that account for the major operational constraints of the telescope and for the domain's optimization objectives. The basic assumption is to treat resource allocation (scheduling) and auxiliary task expansion (planning) as complementary aspects of a more general process of constructing behaviors of a dynamical system. [Mus90].

Two basic mechanisms provide the basis of the HSTS approach:

1. a *domain description language* for modeling the structure and dynamics of the physical system at multiple levels of abstraction.
2. a *temporal data base* for representing possible evolutions of the state of the system over time (i.e. schedules).

In HSTS the natural approach to problem solving is by iterative posting of constraints, extracted either from the external goals or from the description of the system dynamics. Consistency is tested through constraint propagation. For more details, see [MSCD92, Mus93b].

Three key aspects distinguish HSTS from other approaches:

1. the state of the modeled system is explicitly decomposed into a finite set of "state variables" evolving over continuous time. This enables the specification of algorithms exploiting problem decomposability, and provides the necessary structure for optimizing resource utilization.
2. the temporal data base permits flexibility along both temporal and state value dimensions. The time of occurrence of each event does not need to be fixed but can float according to the temporal constraints imposed on the event by the process of goal expansion. This flexibility contributes directly to scheduling efficiency. Since overcommitment can be avoided, there is a lower possibility of backtracking.
3. the constraint posting paradigm accommodates a range of problem solving strategies (e.g. forward simulation, back chaining, etc.). This allows the development of algorithms that opportunistically exploit problem structure to consistently direct problem solving toward the most critical tradeoffs.

The importance of integrating these three features within a single framework can be appreciated by considering the limitations of other approaches that address them separately or partially.

In planning, most Artificial Intelligence research adopts the classical representational assumption proposed by the STRIPS planning system [FHN72]. In this view action is essentially an instantaneous transition between two world states of indeterminate durations. The structural complexity of a state description is not limited, but the devices provided for its description are completely unstructured, such as complete first order theories or lists of predicates. Some frameworks [Wil88, CT91] have demonstrated the ability to address practical planning problems. However, the classical assumption lacks balance between generality and structure; this is a major obstacle in extending classical planning into integrated planning and scheduling. Past research has attempted partial extensions in several important directions: processes evolving over continuous [AK83] and metric time [Ver83, DFM88], parallelism [Lan88], and external events [For89]. However, no comprehensive view has yet been proposed to address the integration problem.

Classical scheduling research has always exploited much stronger structuring assumptions [Bak74]. Domains are decomposed into a set of resources whose states evolve over continuous time. This facilitates the explicit representation of resource utilization over extended periods of time. Several current scheduling systems exploit reasoning over such representations [FS84, SOM+90, Sad91, MJPL92, ZG90, BC91]. Empirical studies have demonstrated the superiority of this approach [OS88, Sad91] with respect to dispatching scheduling [PI77], where decision making focuses only on the immediate future. However, the scheduling view of the world also has very strong limitations. No information is kept about a resource state beyond its availability. Additional state information (e.g., in which direction the observatory is pointing at a given time) is crucial to maintain causal justifications and to dynamically expand support activities during problem solving.

Issues in Integrating Planning and Scheduling

Use of Abstraction

The use of abstract models has long been exploited as a device for managing the combinatorics of planning and scheduling. In HSTS models are expressed in terms of interacting state variables representing different components of the system (in our case, the space-based observatory) and of its operating environment (e.g., celestial objects to be observed). An abstract model can summarize system dynamics with state variables that aggregate several structural components; alternatively abstract models can selectively simplify system dynamics by omitting one or more component state variables. Given the structure of space-based observatory scheduling problems, an abstract model provides a natural basis for isolating overall optimization concerns. This provides global guidance in the development of detailed, executable schedules.

In the case of HST, a two-level model has proved sufficient. At the abstract level, telescope dynamics is summarized in terms of a single state variable, indicat-

ing, at any point in time, whether the telescope (as a whole) is taking a picture, undergoing reconfiguration, or sitting idle. At this level reconfiguration transitions have duration constraints that estimate the time required by the actual reconfiguration activities implied by the detailed model (e.g., instrument warmup and cool-down, data communication, telescope repointing). Execution of an observation at the abstract level requires only satisfaction of this abstract reconfiguration constraint, target visibility constraints, and any user specified temporal constraints. Thus, the description at the abstract level looks much like a classic scheduling problem: a set of user requests that must be sequenced on a single resource subject to specified constraints and allocation objectives.

Planning on a fully detailed model ensures the viability of any sequencing decisions made at the abstract level. This corresponds to generating and coordinating required supporting system activities. The degree of coupling between reasoning at different levels depends in large part on the accuracy of the abstraction. In the case of HST, decision-making at abstract levels is tightly coupled; each time a new observation is inserted into the sequence at the abstract level, control passes to the detailed level to develop detailed segments of system behavior necessary to achieve the new goal. Given the imprecision in the abstract model, goals posted for detailed planning cannot be rigidly constrained; instead only preferences are specified (e.g., "execute as soon as possible after obs1"). The results of each detailed planning stage are propagated at the abstract level to provide more precise constraints for subsequent abstract level decision-making.

Model Decomposability and Incremental Scaling

To approach large problems it is typically necessary to decompose them into smaller sub-problems, solve each sub-problem separately, and then assemble the sub-solutions. We can judge how the problem solving framework supports modularity and scalability if it displays the following two features:

- the search procedure for the entire problem can be assembled by combining heuristics independently developed for each sub-problem, with little or no modification of the heuristics;
- the computational effort needed to solve the complete problem does not increase with respect to the sum of the efforts needed to solve each component sub-problem.

We conducted an experiment with three increasingly complex and realistic models of the HST, in order to evaluate the support that HSTS provides towards the realization of these features (issues relating to optimization of the overall mission performance criteria will be discussed in the following sections).

We identify the three models as SMALL, MEDIUM, and LARGE. All share the same abstract level representation. At the detail level the three models include state variables for different telescope functionalities. The SMALL model has a state variable for the visibility

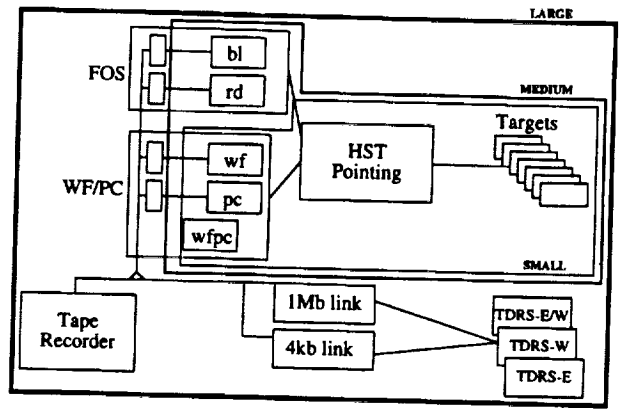

Figure 1: The SMALL, MEDIUM and LARGE HST models.

of each target of interest, a state variable for the pointing state of the telescope, and three state variables to describe a single instrument, the Wide Field/Planetary Camera (WFPC). The MEDIUM model includes SMALL and two state variables for an additional instrument, the Faint Object Spectrograph (FOS). Finally, the LARGE model extends MEDIUM with eight state variables accounting for data communication. The LARGE model is representative of the major operating constraints of the domain. Figure 1 shows the relations among the various models.

For each model we use the same pattern of interaction between problem solving at the abstract and at the detail level. At the abstract level observations are selected and dispatched using a greedy heuristic to minimize expected reconfiguration time. The last dispatched observation is refined into the corresponding detail level problem; then control is passed to planning/scheduling at the detail level. This cycle is repeated until the abstract level sequence is complete.

The detail planner/scheduler for SMALL is driven by heuristics which deal with the interactions among its system components. A first group ensures the correct synchronization of the WFPC components; one of them, for example, states that, when planning to turn on the WF detector, preference should be given to synchronization with a PC behavior segment already constrained to be off. A second group deals with the pointing of HST; for example, one of them selects an appropriate target visibility window to execute the locking operation. A final group manages the interaction between the state of WFPC and target pointing; an example from this group states a preference to observe while the telescope is already scheduled to point at the required target. To solve problems in the context of MEDIUM, additional heuristics must deal with the interactions within FOS components, between FOS and HST pointing state, and between FOS and WFPC. However, the nature of these additional interactions is very similar to those found in SMALL. Consequently, it is sufficient to extend the domain of applicability of SMALL's heuristics to obtain a complete set of heuristics for MEDIUM. For example, the heuristic excluding

Model	SMALL	MEDIUM	LARGE
State Variables	4	6	13
Tokens	587	604	843
Time Points	588	605	716
Temporal Constraints	1296	1328	1474
CPU Time / Observation	11.62	12.25	21.74
CPU Time / Compatibility	0.29	0.29	0.33
Total CPU time	9:41.00	10:11.50	18:07.00
Total Elapsed Time	1:08:36.00	1:13:16.00	2:34:07.00
Schedule Horizon	41:37:20.00	54:25:46.00	52:44:41.00

Table 1: Performance results. The times are reported in hours, minutes, seconds, and fractions of second

WF and PC from being in operation simultaneously can be easily modified to ensure the same condition among the two FOS detectors. Finally, for LARGE we include the heuristics used in MEDIUM with no change, plus heuristics that address data communication and interaction among instruments and data communication; an example of these prevents scheduling an observation on an instrument if data from the previous observation has not yet been read out of its data buffer. By making evident the decomposition in modules and the structural similarities among different sub-models, HSTS made possible the reuse of heuristics and their extension from one model to another. We therefore claim that HSTS displays the first feature of a modular and scalable planning/scheduling framework.

To determine the relationship between model size and computational effort, we ran a test problem in each of the SMALL, MEDIUM, and LARGE models. Each test problem consisted of a set of 50 observation programs; each program consisted of a single observation with no user-imposed time constraints. The experiments were run on a TI Explorer II+ with 16 Mbytes of RAM memory.

As required by the second feature of a scalable framework, the results in table 1 indicate that the computational effort is indeed additive. In the table, the measure of model size (number of state variables) excludes visibilities for targets and communication satellites, since these can be considered as given data. The temporal constraints are links between two time points that lie on different state variables; the number of these links gives an indication of the amount of synchronization needed to coordinate the evolution of the state variables in the schedule.

Since the detail level heuristics exploit the modularity of the model and the locality of interactions, the average CPU time (excluding garbage collection) spent posting an elementary temporal relation constraint (compatibility) remains relatively stable. In particular, given that the nature of the constraints included in SMALL and MEDIUM is very similar, the time is identical in the two cases. The total elapsed time to generate a schedule in the case of LARGE is an acceptable fraction of the time horizon covered by the schedule during execution. Even if this implementation is far from optimal, it nonetheless shows the practicality of the framework for the actual HST operating environment.

Exploiting Opportunism to Generate Good Solutions

In space-based observatory scheduling a critical tradeoff is between: (1) maximizing the time spent collecting science data; (2) satisfying absolute temporal constraints associated with specific user requests. The scheduling problem is typically over-subscribed, i.e., it will generally not be possible to accommodate all user requests in the current short term horizon, and some must necessarily be rejected. A lost opportunity corresponds to the rejection of a request whose user-imposed time windows fall inside the current scheduling horizon. Observation requests without such execution constraints are not lost because the scheduler may reattempt to honor them in subsequent scheduling episodes.

The experiment described in the previous section uses a dispatch-based strategy for sequence development. Simulating forward in time at the abstract level, the strategy repeatedly added to the end of the current sequence the candidate observation estimated to incur the minimum amount of wait time (due to HST reconfiguration and target visibility constraints). This heuristic strategy, termed "nearest neighbor with look-ahead" (NNLA), attends directly to the first global objective of maximizing the time spent collecting science data.

However, forward simulation does not sufficiently address the second global objective: the minimization of rejections of absolutely constrained requests. A request's window of opportunity may be gone by the time the scheduler rates the request as the choice with minimum dead time. Coupling forward simulation with look-ahead search (i.e. evaluation of possible "next sequences" and potential rejections) can provide protection against unnecessary request rejection. However this approach has limited effectiveness because of combinatorics. A second sequencing strategy directly attends to the minimization of the number of rejected requests with absolute constraints: "most temporally constrained first" (MCF). MCF's computational complexity is comparable to that of NNLA. Under the MCF scheme, the sequence is built by repeatedly selecting the pending request with the tightest execution bounds and inserting it in the schedule. Unlike NNLA this strategy does not build a sequence with a simulation-based approach. Honoring the temporal constraints of the selected requests will create availability "holes" over the scheduling horizon. Incidentally, the MCF strategy is quite close to the algorithm currently employed in the operational system at STScI.

As is the case with the NNLA strategy, one objective is also emphasized at the expense of the other within the MCF strategy. The availability holes opened by MCF can result in considerable telescope idle time and therefore a sequence insertion heuristic should seek to minimize such dead time. However, its effectiveness is largely determined by the specific characteristics and distribution over the horizon of the initially placed requests.

Both NNLA and MCF manage combinatorics by

Sequencing Strategy	Pctg. Constrained Goals Scheduled	Pctg. Telescope Utilization
NNLA	72	21.59
MCF	93	17.20
MCF/NNLA	93	20.54

Table 2: Comparative Performance of NNLA, MCF and MCF/NNLA

making specific problem decomposition assumptions and localizing search according to them. NNLA assumes an event based decomposition (considering only the immediate future) while MCF assumes that the problem is decomposable by degree of temporal constrainedness. Previous research in constraint-based scheduling [SOM+90] has indicated the leverage of dynamic problem decomposition and selective use of local scheduling perspectives. In our case, we can also evaluate problem structure to select the appropriate strategy between NNLA and MCF at any point during sequence development. In particular we estimate the current variance of the number of feasible start times remaining for individual unscheduled requests. If the variance is high, there is an indication that some remaining requests might be much more constrained than others; this prompts the use of MCF to emphasize placement of tightly constrained goals. If the variance is low, there is an indication that all pending requests have similar temporal flexibility; the emphasis can switch to minimizing dead time within current availability "holes" using NNLA.

To test this multi-perspective approach, we solved a set of short-term (i.e., daily) scheduling problems using three separate strategies: NNLA, MCF and the composite strategy just described (referred to as MCF/NNLA). The results are given in Table 2. They confirm our expectations as to the limitations of both NNLA and MCF. We can also see that MCF/NNLA produces schedules that more effectively balance the two competing objectives. Further details on the sequencing strategies and the experimental may be found in [SP92].

These results should be viewed as demonstrative and we are not advocating MCF/NNLA as a final solution. We can profitably exploit other aspects of the current problem structure and employ other decomposition perspectives. For example, the distribution of goals over the horizon implied by imposed temporal constraints has proved to be a crucial guideline in other scheduling contexts [SOM+90, Sad91], and we are currently investigating the use of analogous look-ahead analysis techniques within the problem solving framework provided by HSTS (see below). There are also additional scheduling criteria and preferences (e.g., priorities) in space-based observatory domains that are currently not accounted for.

Constraint Posting Scheduling

Most constraint-based scheduling research addresses the problem of finding a single, consistent assignment of start times for each activity [BC91, Joh90, KY89, MJPL92, SOM+90, Sad91, ZG90]. HSTS, in contrast, advocates a problem formulation more akin to least-commitment planning frameworks. The problem is most naturally treated as one of posting sufficient additional precedence constraints between pairs of activities so as to ensure feasibility with respect to time and capacity constraints. Solutions generated in this way typically represent a set of feasible schedules (i.e., the sets of activity start times that remain consistent with posted sequencing constraints), as opposed to a single assignment of start times.

While HSTS does not prohibit the use of "fixed time" scheduling techniques, there are several potential advantages to a solution approach that retains solution flexibility as problem constraints permit. A flexible schedule provides a measure of robustness against uncertainty during schedule execution. The determination of actual start times can be delayed until execution and solution revision can be minimized. A constraint posting approach can also provide a more convenient search space in which to operate during schedule development. Alternatives are not unnecessarily pruned by (over) committing on specific start times. When the need for schedule revision becomes apparent, modifications can often be made much more directly and efficiently through simple adjustment of posted constraints. Our recent research has developed and evaluated two novel algorithms for generating schedules via constraint posting that demonstrate the previous potential advantages: Conflict Partition Scheduling (CPS)[Mus92] and Precedence Constraint Posting Scheduling (PCP)[SC93].

The CPS procedure builds on previous research into techniques for estimating resource contention[MS87]. Great emphasis is put in the recognition of resource capacity bottlenecks - time periods with high predicted contention among activities for the same resource capacity. In CPS capacity bottleneck are detected through use of a stochastic simulation technique. After identifying resource capacity bottlenecks, CPS acts to lessen the level of contention by posting ordering constraints among the activities competing for capacity at the most severe bottleneck. The iterative process continues until no capacity conflict remains; at this point a final schedule has been determined. If the process reaches an infeasible solution state, the search is simply restarted. CPS has been experimentally tested on a set of benchmark constraint satisfaction scheduling problems. The performance analysis demonstrated superior problem solving performance to two currently dominant "fixed-times" scheduling approaches - micro-opportunistic scheduling [Sad91] and min-conflict iterative repair [MJPL92]. The reader is referred to [Mus92] for details. More recent work has aimed at the evaluation of different alternative CPS configurations (e.g., micro vs macro decision making, focused on capacity conflicts vs randomly focused) to establish the relative importance of different steps of the procedure and the corresponding performance trade-offs [Mus93a].

The PCP procedure combines two techniques: dom-

inance conditions for incremental pruning of the set of feasible sequencing alternatives [EV76] and a simple look-ahead analysis of the temporal flexibility associated with different sequencing decisions. At each step of the search, a measure of "residual temporal slack" is computed for each sequencing decision that remains to be made. PCP choses the decision with the smallest residual slack as the most critical, and posts a precedence constraint in the direction that retains the most flexibility. Posting a constraint might leave other sequencing decisions with only a single feasible ordering; these unconditional decisions are taken by posting the implied precedence before recomputing estimates of residual slack. Unlike CPS, which posts constraints only until all resource contention has been resolved, PCP terminates when either all pairs of activities contending for the same resource have been sequenced, or an infeasible state has been reached. PCP has also been tested on the same suite of constraint satisfaction used in the performance analysis of CPS. PCP has shown comparable problem solving performance to other contention-based scheduling approaches at a small fraction of the computational cost[SC93].

Our current research pursues the following goals: (1) extension of both the CPS and PCP approaches to address optimization criteria; (2) investigation of complementary techniques for exploiting solution flexibility in reactive contexts; (3) evaluation of the effectiveness of these techniques in the context of space-based observatory scheduling problems.

Conclusions

In this paper, we have considered the solution of space-based observatory scheduling problems. These problems require a synthesis of the processes of resource allocation and expansion of auxiliary activities. We briefly outlined the HSTS framework and contrasted it with other scheduling and AI planning approaches. To illustrate the adequacy of the framework, we then examined its use in solving the HST short-term scheduling problem. We identified three key ingredients to the development of an effective, practical solution: flexible integration of decision-making at different levels of abstraction, use of domain structure to decompose the planning problem and facilitate incremental solution development/scaling, and opportunistic use of emergent problem structure to effectively balance conflicting scheduling objectives. The HSTS representation, temporal data base, and constraint-posting framework provide direct support for these mechanisms. Finally, we summarized more recent research aimed at further demonstration of the efficacy of constraint posting scheduling in HSTS.

We are currently utilizing HSTS to develop a scheduling tool for support of the upcoming Submillimeter Wave Astronomy Satellite (SWAS) mission. This scheduling domain requires attendance to many types of scheduling constraints similar to the the HST domain (e.g., target visibility, power). However, the SWAS scheduling problem also differs in character from the HST problem; whereas the HST problem involves synchronization of sets of observations with pre-specified targets and durations, the SWAS scheduling requirement is to distribute viewing (or data integration) time among a prioritized set of desired targets. We have developed and are currently experimenting with an initial solution to the SWAS problem. This consists of a heuristic algorithm that utilizes target priority and dead time minimization criteria to create and interleave individual target observations of various durations. These results indicate the flexibility of HSTS to accommodate different types of scheduling problems. Current plans call for refinement and subsequent transfer of this solution to the SWAS mission team by the end of the year.

References

[AK83] J. Allen and J.A. Koomen. Planning using a temporal world model. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 741–747, 1983.

[Bak74] K.R. Baker. *Introduction to Sequencing and Scheduling.* John Wiley and Sons, New York, 1974.

[BC91] E. Biefeld and L. Cooper. Bottleneck identification through chronology-directed search. In *Proceedings 12th International Conference on Artificial Intelligence*, pages 218–224, Sydney, Australia, August 1991.

[CT91] K. Currie and A. Tate. O-plan: the open planning architecture. *Artificial Intelligence*, 52(1):49–86, 1991.

[DFM88] T. Dean, R.J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence*, 4:381–398, 1988.

[EV76] F. Roubellat Erschler, J. and J.P. Vernhes. Finding some essential characteristics of the feasible solutions for a scheduling problem. *Operations Research*, 24:772–782, 1976.

[FHN72] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[For89] K.D. Forbus. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1273–8. Morgan Kaufmann, 1989.

[FS84] M.S. Fox and S.F. Smith. Isis: A knowledge-based system for factory scheduling. *Expert Systems*, 1(1):25–49, 1984.

[Joh90] M.D. Johnston. Spike: Ai scheduling for nasa's hubble space telescope. In *Proceedings of the 6th IEEE Conference on Artificial Intelligence Applications*, pages 184–190, 1990.

[KY89] N. Keng and D.Y. Yun. A planning/scheduling methodology for the constrained resource problem. In *Proceedings IJCAI-89*, Detroit, MI., Aug 1989.

[Lan88] A. Lansky. Localized event-based reasoning for multiagent domains. *Computational Intelligence*, 4:319–340, 1988.

[MJPL92] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:361–205, 1992.

[MS87] N. Muscettola and 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.

[MSCD92] N. Muscettola, S.F. Smith, A. Cesta, and D. D'Aloisi. Coordinating space telescope operations in an integrated planning and scheduling architecture. *IEEE Control Systems Magazine*, 12(1), February 1992.

[Mus90] N. Muscettola. Planning the behavior of dynamical systems. Technical Report CMU-RI-TR-90-10, The Robotics Institute, Carnegie Mellon University, 1990.

[Mus92] N. Muscettola. Scheduling by iterative partition of bottleneck conflicts. Technical report, The Robotics Institute, Carnegie Mellon University, 1992.

[Mus93a] N. Muscettola. An experimental analysis of bottleneck-centered opportunistic scheduling. Technical Report CMU-RI-TR-93-06, The Robotics Institute, Carnegie Mellon University, March 1993.

[Mus93b] N. Muscettola. Hsts: Integrating planning and scheduling. Technical Report CMU-RI-TR-93-05, The Robotics Institute, Carnegie Mellon University, March 1993.

[OS88] P.S. Ow and S.F. Smith. Viewing scheduling as an opportunistic problem solving process. In R.G. Jeroslow, editor, *Annals of Operations Research 12*. Baltzer Scientific Publishing Co., 1988.

[PI77] S.S. Panwalker and W. Iskander. A survey of scheduling rules. *Operations Research*, 25:45–61, 1977.

[Sad91] N. Sadeh. *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling*. PhD thesis, School of Computer Science, Carnegie Mellon University, March 1991.

[SC93] S.F. Smith and C. Cheng. Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings AAAI-93*, Washington DC, July 1993.

[SOM+90] S.F. Smith, J.Y. Ow, P.S. Potvin, N. Muscettola, , and D. Matthys. An integrated framework for generating and revising factory schedules. *Journal of the Operational Research Society*, 41(6):539–552, 1990.

[SP92] S.F Smith and D.K. Pathak. Balancing antagonistic time and resource utilization constraints in over-subscribed scheduling problems. In *Proceedings 8th IEEE Conference on AI Applications*, March 1992.

[Ver83] S. Vere. Planning in time: Windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5, 1983.

[Wal89] M. Waldrop. Will the hubble space telescope compute ? *Science*, 243:1437–1439, March 1989.

[Wil88] D.E. Wilkins. *Practical Planning*, volume 4. Morgan Kaufmann, 1988.

[ZG90] M. Deale Zweben, M. and R. Gargan. Anytime rescheduling. In *Proc. DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*. Morgan Kaufmann Pub., Nov. 1990.

Session A2: ARTIFICIAL INTELLIGENCE II

Session Chair: Dr. Abe Waksman