Bottleneck Identification Using Process Chronologies

Eric Biefeld and Lynne Cooper
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
U. S. A.

Abstract

We present a heuristics-based approach to deep space mission scheduling which is modeled on the approach used by expert human schedulers in producing schedules for planetary encounters. New chronological evaluation techniques are used to focus the search by using information gained during the scheduling process to locate, classify, and resolve regions of conflict. Our approach is based on the assumption that during the construction of a schedule there exist several disjunct temporal regions where the demand for one resource type or a single temporal constraint dominates (bottleneck regions). If the scheduler can identify these regions and classify them based on their dominant constraint, then the scheduler can select the scheduling heuristic.

1 Introduction

Tn a heuristics-based automated scheduling system, it is possible to construct a large set of scheduling heuristics such that each scheduling heuristic optimizes a different aspect of the schedule [Smith and Ow, 1985]. For example, one scheduling heuristic could temporally colocate steps that use the same resources, reducing set-up times. While a particular scheduling heuristic optimizes one aspect of the schedule, it can cause inefficiencies in the other aspects of the schedule. Thus, the scheduler must carefully choose which scheduling heuristic to run and limit the scope of its application.

This paper presents the results of research on developing a heuristics-based approach to automated scheduling. The approach is based on the assumption that during the construction of a schedule there exist several disjunct temporal regions of the schedule where the demand for one resource type or a single temporal constrain dominates (referred to as bottlenecks). If the scheduler can identify these regions, and choose the appropriate heuristics to optimize for the dominant constraint within the regions, then the system can construct an efficient solution.

In this paper, we will provide the background and motivation for this approach, discuss the object representations necessary to support it, describe the mechanisms used for locating, classifying, and resolving bottlenecks, and describe the results achieved toward implementing this approach. Discussion of related work is also included.

2 Background

Mission scheduling is characterized by large numbers of science experiments which are considered mutually independent except for their competition for limited spacecraft resources [Biefeld and Cooper, 1989]. For missions such as the Voyager Grand-Tour of the planets, the requested tasking far exceeds the capability of the spacecraft resources. Most of the science requests will never be scheduled, so the problem is one of configuring a subset of the requested tasks in such a way as to optimize the total science return. While there is considerable flexibility in scheduling an individual task, producing schedules with the necessary science return density requires the interleaving of tasks in a manner which increases the total number of experiments that can be performed.

Due to the complexity and size [Dean, 1986] of the domain, mission scheduling is currently performed by teams of expert human schedulers. When constructing a mission schedule, the expert human schedulers work incrementally. They first lay out a rough draft of the schedule complete with large numbers of conflicts and high levels of over-subscription [Biefeld and Cooper, 1988]. They proceed by identifying problem areas, classifying these areas, and using the appropriate techniques to resolve any conflicts. As they cycle through the problem regions, they build up knowledge about this particular scheduling problem, e.g. where the major resource

*This research was done at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the U.S. Department of Defense through an agreement with the National Aeronautics and Space Administration. In addition to the authors, individuals participating in this project include David Atkinson, Len Charest, Richard Doyle, Loretta Falcone, Kirk Kandt, Raymond Lam, Gaius Martin, Elmain Martinez, and Harry Porta.

[1] Optimize is not used in the mathematical sense. Because of the intractability of the problem domain, a working definition is: to develop a schedule which an expert human scheduler cannot easily improve.

contentions are located, and which tasks have a high level of interaction or are difficult to place into the schedule. This information is then used to guide their future scheduling actions.

The approaches used in classical planning systems such as NONLIN [Tate, 1976], FORBIN [Dean *et* a/., 1987], and DEVISER [Vere, 1983], are fundamentally different from the approaches used by the human mission schedulers. The classical planning systems build a schedule sequentially, with each scheduling action resulting in either a conflict-free schedule ready to have additional tasks added to it, or an illegal schedule which requires backtracking to correct.

An alternative approach, presented in this paper, is modeled after expert human schedulers. The Operations Mission Planner (OMP) models the inherently cyclic nature of the human scheduling process and the types of information that the human schedulers develop as they are generating and refining a schedule.

3 Representations

OMP makes use of two primary object types: tasks and resources. A task is a request to the scheduler for the use of resources (e.g. a science experiment on Voyager). It includes all of the internal constraints such as time windows (e.g. when the red spot of Jupiter is visible to the spacecraft). A resource is a commodity on the spacecraft (e.g. cameras, downlink, power). It is modeled as a resource timeline representing the constraints on its use (e.g. capacity, direction, bandwidth). Detailed descriptions of how OMP implements tasks and resources are given in the following sections.

3.1 Tasks

Tasks are represented in OMP as data objects with parametrized slots and demons which ensure internal consistency for the task representation. A task is composed of a set of steps and the temporal constraints that exist between the steps. A task can have several different sets of steps which satisfy it. The template for a task request is shown below.

Name: A symbol representing the name of the task.

Description: A string providing background information on the task.

Windows: The list of temporal intervals in which the task can be scheduled.

Repetitions: The number of times to repeat the task during the schedule.

Priority: The priority of this task (relative to other tasks) for this schedule.

Scenarios: The order(s) in which to perform the task steps and their associated preference(s), this can include alternative ways in which to implement a given task breakdown.

Steps: A list of steps.

Step#: The step's identification number.

Duration: Minimum and maximum (desired) duration for the step.

Delay: The minimum and maximum time delay between this step and the previous step.

Resources: The resources requested by the step.

3.2 Steps and Activities

In OMP, a step is a temporal interval which consumes resources. Each step has specified duration constraints, temporal windowing constraints, and a list of the resources requested. A step object, S, specifies the start and end times of the step (Interval, /), and pairs representing the resource assigned, *R,* and the amount of resource assigned to the step (Usage, *SU).*

$S=[I,(R_i,SU_i(I))]$ where $(R_i \in (R_1...R_N))$

An activity is a set of steps and the associated constraints [Le Pape and Smith, 1987] which satisfy a request. There may be more than one activity that satisfies a task request (see SCENARIO slot above). While OMP actually schedules at the step level, demons attached to the activity ensure that the task is always consistent and that all task constraints are met. Therefore, the only constraint satisfaction that the scheduling engine (discussed below) must perform is to satisfy resource constraints.

3.3 Resources

There are four fundamental types of resources: capacity, consumable/renewable, state-continuous, and state-discrete [Starbird, 1987]. A capacity resource is basically a pooled resource. Tasks request and use the resource, then free it up for other tasks to use (e.g. spacecraft downlink). A consumable resource is one for which there is a limited supply, and once it is used by a task, it, is no longer available (e.g. spacecraft fuel). A renewable is a special case of consumable, where the resource can be replenished (e.g. storage tape; it is used up during recording, and "replenished" during playback). A state resource represents a resource whose state (configuration, position, etc.) must be a certain value in order to support a task. A State-continuous resource is one in which the state of the resource can best be described by a continuous variable (e.g. the direction that an antenna is pointing). State-discrete resources, on the other hand, are represented by discrete values (e.g. on/off, low-gain/medium-gain/high-gain).

The OMP system has represented two of these types of resources: capacity and state-continuous. However, they were represented independently in two separate implementations of OMP. The following discussion assumes a capacity type resource. The Conflict Level function must be redefined for the other types of resources.

OMP resources are represented as timelines. Each timeline is a sequence of contiguous, discrete temporal intervals represented by the following parameters:

Interval *(t_s,te)'-* The start and end times of the interval.

Resource Capacity *(CA):* The maximum capacity of the resource, known *a priori.* Can be a function of time.

Step Assignments *(SA):* A list of steps requesting the resource during a given temporal region.

Resource Usage *(RU):* The average amount of the resource requested by the assigned steps during a given temporal region where:

$$RU(I) = \sum_{(S_i|S_i \in SA)} SU_i/\mathrm{dur}(I)$$

Conflict Level *(CL):* The average level of conflict (contention for resources) during a given temporal region where:

$$CL(I) = F(RU(I), CA(I))$$

Each resource, *RS,* is represented by a temporal ordered list of tuples, where for each time tick, t,

$$RS(t) = [CA(t), SA\{t), RU(t), CL(t))$$

3.4 Regions

OMP adds a descriptive layer over the timelines called regions. A region is a variable-length, continuous time segment on a single resource. It is designated by a start time and end time and is defined dynamically by the scheduler. In addition to the general parameters associated with the resource timeline *(RS),* each region, *REi,* has special-purpose parameters.

$$RE = [I, F, CL, RU]$$

Interval (I): The temporal area bounded by the start and end times of the region.

Focus Level *(F):* The number of times a given time slot has been the FOCUS of the scheduler.

OMP defines several types of specialized regions:

Conflict Region *(CR):* A variable length, continuous time interval where the requested usage is greater than the theoretical capacity of the resource during that interval.

$$CLRS(I) > a$$

where *a* corresponds to the percent of over subscription allowed by the current scheduling phase.

Focus Region *(FR):* The conflict region which the scheduler has chosen to work on during the upcoming scheduling pass.

Effected Region *(ER):* Any region which is changed as a result of a scheduling action in a focus region.

Bottleneck Region *(BNR):* A *set* of regions (not necessarily continuous) which comprise a bottleneck.

4 Scheduling

The OMP approach to scheduling is based on the following assumptions:

1. The schedule can be decomposed into several disjunct temporal regions where a small set of constraints dominates (bottlenecks).
2. A parametrized scheduling heuristic instituting a specialized search can be used to optimize for the dominant constraint within the bottleneck.
3. The scheduler can merge the resulting partial schedules and complete any unspecified portions while maintaining the highly optimized sections.

OMP decomposes a scheduling problem by using chronologies to locate, classify, and resolve bottlenecks. An intuitive, working definition of a bottleneck is: A set of disjunctive temporal regions where:

1. A small set of resource constraints dominate;
2. The potential resource demand based on possible requests exceeds the available resource capacity.

4.1 Locate Bottlenecks

OMP's approach to locating bottlenecks is not based on a static evaluation of requests versus resource limits [see CORTES; Section 6]. Instead it is based on the observation that repeatedly applying a conflict-repair strategy to a schedule will cause the conflict level(s) within a region(s) to change. By tracking changes in the conflict levels in different regions, OMP is able to locate interdependent regions and merge them into a bottleneck. The chronology subsystem is responsible for capturing the information needed to locate the bottlenecks.

4.1.1 Chronograms and Chronology

After performing the initial expansion of the tasks into activities and making preliminary resource assignments, the scheduler focuses on the area of the schedule with the most conflicts. The scheduler performs a shallow search which lowers the number of conflicts in this area. Only the activities that are involved in the conflict are modified. The impact of these modifications on the resources is recorded in an object called a chronogram. The procedure is:

1. Select a *FR* from the various *CR* on the schedule using the appropriate selection criteria (e.g. select the *CR* with the maximum *CL);*
2. Move tasks out of the *FR* until the selection parameter (e.g. *CL)* is less than the threshold value set by scheduler;
3. Create a chronogram and add it to the chronology;
4. Repeat untill no conflicts exist or a bottleneck is identified.

The shallow search used by OMP is a simple hill climber whose utility function is composed of the conflict level within the focus region, the total conflict level of the schedule and the number of times an activity has been modified. While the search tries to minimize the total conflict level, it will at times increase the conflict level over the total schedule in order to lower the conflict level within the focus region.

A newly created conflict region will eventually become the focus of the scheduler. Solving this region may, in turn, cause other conflicts and so on, until the original conflict region is once again in conflict. As the search progresses through the oversubscribed resources, the level of conflict in these and other areas oscillate. The conflict areas that oscillate in this manner are classified as potential bottlenecks.

Each time the system focuses on a new region it creates an entry into the chronology called a chronogram. A chronogram includes the focus region, the change in conflict level of the focus region, and a list of any other resource regions whose conflict level changed (the effected regions), and the actual change in conflict $\Delta CLER$ for the effected regions.

Chronogram(C_i), C_i = *[FR,F,CL,* Δ *CL,(ERS,CL$_{ER}$)1*

and

Chronology(CY), *CY* = (C_i, *i* = 1, *M)*

for *M* scheduling passes

Eventually, the number of times the system has focused on the same region will reach a heuristic threshold. This signifies that the search is wandering around in circles and it is time to analyze the chronology to identify the bottlenecks.

4.1.2 Chronograph

The first step in processing the chronology is to build a graph for each chronogram. Each region in the chronogram is a node and an arc is placed between the focus region node and the effected region nodes in the chronogram. A pulse count of one is assigned to each of the newly created nodes indicating the number of times this region has changed its conflict level. The strength of the arc is the ratio of the change in conflict level of the region and the focus region.

Node *(Ni):* For *($R_i \backslash R_i \in ERs$),* a node, N_i,is created with a

Pulse Count *(PC):* $PC(N_i)$ = 1.

Arc (A): For every node, A^7,, there exists an arc, .4,, which connects i?R_i and the *FR,* with an

Arc Strength *(Si):* S_i — $|\Delta CL_{R,}/\Delta CL_{FR}|$.

The next step in processing the chronology is to merge the individual chronographs. OMP combines any nodes that represent the same region. If any two regions on the same resource overlap temporally, the ratio of overlap duration to total temporal duration is used to determine if the regions should be combined. When two nodes are combined, the temporal extent of the resulting node is the union of the original two nodes[1] temporal extents and the pulse count is the sum of the original two nodes-pulse count. Any two arcs which link the same nodes are combined and the resulting strength is the sum of the strengths of the original two arcs. After all the graphs have been merged, any node whose pulse count is below a heuristic threshold is deleted from the graph and any arc whose strength is below a threshold is also deleted. The graph is then split into a set of connected graphs and a bottleneck is created for each of the connected graphs.

4.2 Classify and Resolve Bottlenecks

Once OMP has identified the bottlenecks, the next goal is to classify them.

OMP calculates a set of metrics for each bottleneck. These metrics evaluate the conflict level, temporal scope (number, size, and continuity of temporal intervals involved), and resource scope (number of different resources involved) of the individual bottlenecks. Once a bottleneck has been evaluated based on these metrics, appropriate strategies can be employed to resolve the bottleneck. The goal of the strategies is to either produce a conflict-free schedule within the bottleneck region or break-down the bottleneck into more manageable pieces.

If the metrics indicate that a bottleneck is massively oversubscribed (i.e. the requested usage within the bottlenecks is greater then 150% of the theoretical capacity of the bottleneck), OMP will employ strategies which delete the lower priority activities from the bottleneck until the requested demand is reduced to about 120% over the theoretical capacity. After the demand has been lowered OMP will once again search for bottlenecks.

The variance in the resource usage is a metric which checks for uniform usage throughout the bottleneck. If, for example the variance is high then a simple hill climber, which minimizes the square of the resource demand, is applied to the bottleneck to level out the resource usage. After this leveling strategy has completed then the system will apply a more specific strategy to reduce the total conflict within the bottleneck.

Another metric used to classify a bottleneck is the total temporal extent and the number of resources the bottleneck employs. If the bottleneck is temporally large in scope then a general purpose strategy is used which adds constraints to the activities in the bottleneck, causing the large bottleneck to split into several smaller bottlenecks. If the bottleneck is small in scope, then a strategy that optimizes for a specific type of resource is applied to the bottleneck. For example, if the resource tracks the pointing direction of an antenna and the direction changes frequently during the temporal extent of the bottleneck, a lot-sizing[2] strategy is applied which collects requests with similar direction requirements together. This strategy reduces the panning (setup time) of the antenna and thus frees up the antenna for additional requests.

5 Status

The current implementation of OMP has been tested on a Space Station Freedom (SSF) scenario consisting of 400 tasks (—1000 steps), using 17 different capacity resources. OMP generates a 7 day schedule. Using the approach presented in this paper, OMP is able to develop a schedule in approximately 15 minutes. Randomly reordering the input requests has a less than two percent impact on the number of tasks in the schedule. (Although the actual schedule that's produced is highly input dependent, the general characteristics of the schedule such as total number of tasks in schedule and profile with respect to task priority, vary only slightly).

OMP currently performs no preprocessing on the input tasking. It uses an extremely simple loading algorithm which randomly assigns the steps to the resources.

[2]Lot-sizing is a concept from manufacturing scheduling where several tasks of the same type are scheduled together as a lot in order to reduce equipment setup time.

Additional analysis is needed to identify the potential benefits which could be realized by incorporating a more intelligent loading scheme.

The chronology analysis techniques discussed in this paper center around the merging and separating of individual chronographs. We are currently investigating a more efficient methodology for building the chronographs which takes advantage of the fact that by careful selection of the next focus region, you can build up a chronograph incrementally with a high degree of confidence that all of the nodes of the graph are part of the same bottleneck region. We believe that this technique, chronological coherence, could eliminate the need for an in-depth analysis and reduction of the chronograph into a series of directed graphs and improve the overall efficiency of the scheduler.

The heuristic threshold values referred to throughout Section 4 are currently set manually and cannot be changed during the course of a scheduling session. A future goal for OMP is to incorporate learning capabilities which would enable OMP to dynamically set the values of those parameters during the production of a schedule.

OMP has proven itself capable of handling a complex, real-world domain. It produced realistic schedules in a reasonable amount of time for a large, detailed problem domain. The crux of OMP research up to this point has been to prove that an automated approach modeled after expert human schedulers could be developed. The possibility of producing such an automated scheduling system has been established. Future work will concentrate on the feasibility of this approach.

6 Related Work

Other work in mission scheduling done at JPL includes DEVISER, which was tested on Voyager's Uranus near-encounter sequence. While DEVISER was able to produce Voyager schedules for cruise mode operations, it was impractical to use for the highly packed encounter sequences [McLaughlin and Wolff, 1989]. In order to constrain the search, the expert human scheduler who built the knowledge base had to 1) divide the encounter tasks into several smaller groups; 2) modify the knowledge base for each set of tasks to limit the search; and 3) manually integrate the partial schedules into a complete schedule.

After the near-encounter test was finished, various techniques for heuristically limiting the search were investigated. These techniques included: 1) dynamic rule subsetting; 2) dynamic rule ordering; 3) pruning search paths; and 4) backtracking to check points instead of the previous state. While these mechanisms were technically possible to implement, the domain experts could not formulate domain-specific heuristics which used these mechanisms.

When attempting to limit the search of DEVISER, the expert would construct a schedule by hand and analyze the bottlenecks within this schedule. The expert was unable to use DEVISER to extract this type of information. This (and other observations of the human expert) led to the use of chronologies for identifying bottlenecks.

The FORBIN planner uses the Time Map Manager (TMM) [Dean and McDermontt, 1987] as a temporal truth maintenance database and incorporates many classical planning techniques. The TMM uses the constraints knowledge base to make temporal predictions on plan viability. However, since the TMM is not capable of detecting all of the steps' interactions, FORBIN has a scheduling module, Heuristic Task Scheduler (HST) [Miller, 1988], which searches the partially constructed plan for a totally specified schedule. If a viable schedule is found, FORBIN goes on to the next goal to be expanded while the particular parameters chosen by the scheduler will not be enforced, in keeping with a policy of least commitment.

One of the major heuristics used by HST for limiting the search space is *"generate only the feasible schedules"* [Miller, 1988]. Like HST, the TMM can only work with feasible schedules. OMP, on the other hand, does most of its processing on infeasible schedules. As Dean noted "*It is often convenient in debugging plans to determine what the repercussions are of modifying certain constraints .. . What you want is a blow-by-blow account of what things might go wrong, so you can assess what might be done*"[1] [Dean, 1985]. A basic premise of OMP's architecture is that a careful analysis of the implications of the infeasibility of a schedule allows the scheduler to tightly control the search and that the increase in search control more than compensates for the increase in the search space.

The Opportunistic Intelligent Scheduler (OPIS) [Smith *et* al., 1986] is a job shop scheduler that grew out of ISIS [Fox and Smith, 1984]. OPIS implements multiple scheduling perspectives. It incorporates the order-based perspective used in ISIS and adds a resource-based perspective. The resource-based perspective is used to focus the search on the resource bottlenecks.

Like OMP, OPIS uses different strategies to resolve a bottleneck depending on the metrics of the bottleneck. However, OPIS uses a simple infinite capacity scheduler to find bottlenecks that is easily fallible under conditions other than its test domain [Ow and Smith, 1988]. The technique described in this paper for finding bottlenecks is more robust and allows the system to distinguish the structure of a bottleneck.

CORTES [Fox and Sycara, 1990] and RALPH Johnson an d Werntz, 1987] use a look ahead technique Johnson and Roadifer, 1986; Muscettola and Smith, 1987; Sadeh and Fox, 1990] that is conceptually similar to the technique presented in this paper. This technique is based on first constructing a probabilistic model of the activities. If an activity can be scheduled in a large temporal window, then it has a non-zero probability distribution within this temporal window which depends inversely on the duration of the window and is proportional to the duration of the activity. The activity probability distribution is then weighted by its resource usage and the activity resource distribution is summed for all the activities. The resulting resource usage distribution is compared to the resource capacity to locate potential bottlenecks.

OMP uses a statistical technique to generate the same

information found in the resource usage distributions. Instead of calculating the probability of a resource usage level, OMP generates resource usage samples during its initial search. These samples are used to statistically calculate a resource usage distribution.

A significant advantage of the OMP approach is that not only are the bottleneck regions identified but the regions are collected into independent bottlenecks. With the probability approach mentioned above all the bottleneck regions are identified but there is no direct information on how these regions are connected.

While OMP is not a learning program there has been recent work [Eskey and Zweben, 1990] in extending Explanation-Based Learning of search control rules [Minton, 1988] to learn the general conditions under which chronic contention occurs. A chronic contention is similar to a bottleneck. However, instead of having to analyze the schedule to find the bottlenecks, the general conditions under which they will occur are learned.

The advantage of an EBL approach is that the system can quickly identify passible chronic conditions. However, the connection between the bottleneck regions depends on the particular set of tasking and the exact timing of the orbit of the spacecraft. Since this can radically vary from day to day it makes exact prediction of the bottleneck regions very hard without actually roughing out the schedule. While a good *a prion* guess of the chronic resources could speed up the initial processing of the schedule, OMP can produce its first rough set of bottlenecks in three to five minutes, so the potential time savings is small. Extending this concept to the learning of how to classify a bottleneck once it has been found does, however, offer substantial performance gains.

7 Summary

In this paper, we have presented a heuristics-based automated scheduling prototype, the Operations Mission Planner (OMP), modeled after the expert human schedulers who produce schedules for planetary encounters. OMP uses new chronological evaluation techniques to focus the search by using information gained incrementally during the scheduling process to locate, classify, and resolve bottlenecks. Our approach is based on the assumption that during the construction of a schedule there exist several disjunct temporal regions where the demand for one resource type or a single temporal constraint dominates (bottleneck regions). Using chronologies, the scheduling system builds a limited history of the scheduling process and then analyzes the information inherent in that history using a set of metrics. A rule-based system then classifies the bottlenecks based on the metrics and determines the appropriate scheduling actions to resolve the bottleneck. In order to implement the chronology system, OMP incorporates special data items in its representations of both tasking and resources. A special data type, the chronogram, is also used to record the effects of a scheduling pass. The chronograms are transformed into a graph representation and, using graph analysis techniques, the chronographs are merged into a set of connected graphs which represent the bottleneck regions.

References

[Biefeld and Cooper, 1988] Eric Biefeld and Lynne Cooper. Replanning and Iterative Refinement in Mission Scheduling. *Sixth Annual Intelligence Community AI Symposium,,* Washington, DC, October 1988.

[Biefeld and Cooper, 1989] Erie Biefeld and Lynne Cooper. Comparison of Mission and Job Shop Scheduling. *Proceedings of the Third International Conference on Expert Systems and the Leading Edge tn Production Planning and Control,* pages 483-494, Hilton Head Island, South Carolina, May 1989.

[Dean, 1985] Thomas Dean,. An Approach to Reason about Time for Planning and Problem Solving. Technical Report 433, Yale University, 1985.

[Dean, 1986] Thomas Dean,. Intractability and Time Dependent Planning. *Proceedings of Workshop on Planning and Reasoning About Action,* pages 143-164, June 1986.

[Dean and McDermontt, 1987] Thomas Dean and Drew McDermott. Temporal Data Base Management. *Artificial Intelligence Journal,* 32(1): 1-55, 1987.

[Dean *et al,* 1987] Thomas Dean, R. James Firby, and David Miller. Hierarchical Planning involving Deadlines, Travel Time, and Resources. *Computational Intelligence,* 4(4), November 1988.

[Eskey and Zweben, 1990] Megen Eskey and Monte Zweben. Learning Search Control for Constraint-Based Scheduling. Internal NASA Ames Research Document, 1990.

[Fox and Sycara, 1990] Mark Fox and Katia Sycara. Overview of CORTES: A Constraint Based Approach to Production Planning, Scheduling and Control. *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management,* pages 1-15, Hilton Head Island, South Carolina, May 1990.

[Fox and Smith, 1984] Mark Fox and Stephen Smith. ISIS: A Know ledge-Based System for Factory Scheduling. *Expert Systems,* 1(1):25 49, July 1984.

[Johnson and Roadifer, 1986] Craig Johnson and James Roadifer. A Look-Ahead Strategy for Heuristic Activity Scheduling. *Joint Conference of the Operations Research Society of America and the Institute of Management Sciences,* October 1986.

[Johnson and Werntz, 1987] Craig Johnson and David Werntz. Automation of the Resource Allocation and Planning System at NASA's Jet Propulsion Laboratory. *SPACE: Technology, Commerce and Communications,* Houston, Texas, November, 1987.

[Le Pape and Smith, 1987] Claude Le Pape and Stephen Smith. Management of Temporal Constraints for Factory Scheduling. *Proceedings of the Working Conference on Temporal Aspects in Information Systems,* May 1987.

[McLaughlin and Wolff, 1989] W. I. McLaughlin and D. M. Wolff. Automating the Uplink Process for Plan-

etary Missions. *AIAA 21th Aerospace Science Meeting,* Reno, Nevada, January 1989.

[Miller, 1988] David Miller. A Task and Resource Scheduling System for Automated Planning. *Annals of Operations Research,* 12(1—4):69—198, February 1988.

[Minton, 1988] Steven Minton. Quantitative results Concerning the Utility of Explanation-Based Learning. *Proceedings of the Seventh National Conference on Artificial Intelligence,* pages 564-569, Saint Paul, Minnesota, August 1988.

[Muscettola and Smith, 1987] Nicola Muscettola and Stephen Smith. A Probabilistic Framework for Resource-Constrained Multi- Agent Planning. *Proceedings of the Tenth International Joint Conference On Artificial Intelligence,* page 1063-1066, Milan, August 1987.

[Ow and Smith, 1988] Peng Si Ow and Stephen Smith. Viewing Scheduling as an Opportunistic Problem-Solving Process. *Annals of Operations Re search,12(1-*4):85-108, February 1988.

[Sadeh and Fox, 1990] N orman Sadeh and Mark Fox. Variable and Value Ordering Heuristics for Activity-Based Job-Shop Scheduling. *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management,* pages 134-144, Hilton Head Island, South Carolina, May 1990.

[Smith *et ai,* 1986] Stephen Smith, Mark Fox, and Peng Si Ow. Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems. *AI Magazine,* 7(4):45-61, 1986.

[Smith and Ow, 1985] Stephen Smith and Peng Si Ow. The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks. *Proceedings of the Ninth International Joint Conference On Artificial Intelligence,* pages 10130-1015, Los Angeles, California, August 1985.

[Starbird, 1987] Tom Starbird. Space Flight Operations Center Sequence Subsystem (SEQ) Functional Requirements Document for Planning. JPL Internal Document D-4697; NASA, Jet Propulsion Laboratory, California Institute of Technology, Pasadena Califonia, August 1987.

[Tate, 1976] Austin Tate, Project Planning Using a Hierarchical Non-linear Planner. Department of Artificial Intelligence Report No. 25, Edinburgh University, 1976.

[Vere, 1983] Steven Vere. Planning in Time: Windows and Durations for Activities and Goals. *IEEE Transactions on Machine Intelligence PAMI-5,* No. 3, pages 246-267, May 1983.