State-Based Scheduling: An Architecture for Telescope Observation Scheduling

Nicola Muscettola and Stephen F. Smith[1]

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper, we extend the applicability of constraint-based scheduling, a methodology previously developed and validated in the domain of factory scheduling, to problem domains that require attendance to a wider range of state-dependent constraints. We focus specifically on the problem of constructing and maintaining a short-term observation schedule for the Hubble Space Telescope (HST), which typifies this type of domain. We examine the nature of the constraints encountered in the HST domain, discuss system requirements with respect to utilization of a constraint-based scheduling methodology in such domains, and present a general framework for state-based scheduling.

1. Introduction

Many planning problems of practical importance involve the allocation of resources over time subject to a large and complex set of constraints. The construction of short-term observation schedules for the Hubble Space Telescope (HST), which is the domain of focus in the research reported in this paper, provides a representative example. HST observation scheduling involves the placement of several thousand exposures on a time line so as to satisfy a wide range of constraints relating to orbit characteristics, power and thermal balance requirements, instrument capabilities, viewing conditions, guidance requirements, and proposer specific restrictions and preferences. Scheduling under constraints, which is characteristic of most real-world scheduling problems, is extremely complex. Complexity derives not only from the diversity and number of constraints that must be attended to, but also from the fact that it is generally not possible to satisfy all constraints. In situations of conflicting objectives, appropriate compromises must be determined.

One approach to scheduling that has demonstrated an ability to effectively cope with a large and conflicting set of constraints is *constraint-based scheduling* [7, 17, 15]. This approach has been investigated and validated in the context of complex factory scheduling problems [15]. Constraint-based scheduling is an incremental problem solving methodology based on repeated analysis of the characteristics of problem constraints induced by the current partial solution (e.g. intervals of likely resource contention, relative flexibility of different activity time constraints, conflicts in the current partial

[1]This research has been sponsored in part by the National Aeronautics and Space Administration under contract # NCC 2-531 and The Robotics Institute.

schedule, etc.) as a means for structuring and exploring the underlying search space. It presumes an ability on the part of the problem solver to selectively reason from local perspectives, and analysis is concerned with subproblem formulation (i.e. which decisions to consider next and which scheduling criteria to emphasize). Commitments are thus made in an *opportunistic* manner (i.e. there is no a priori constraint on the order in which decisions are made). As specific commitments are made, current solution constraints are updated to reflect their consequences. This is a radical departure from traditional dispatch-based approaches to reasoning about efficient resource utilization over time (e.g. [16, 11]), wherein commitments are generated in a strict forward time order. It enables the scheduler to focus immediately on those decisions most critical to overall optimization of scheduling objectives as opposed to encountering them only after other restricting decisions have necessarily been committed to.

In this paper, we introduce a framework for *state-based scheduling*, which extends the applicability of this opportunistic scheduling methodology to domains, like the HST domain, where scheduling decisions must satisfy a wide variety of state-dependent constraints in addition to resource availability. Section 2 considers the nature of the short-term HST observation scheduling problem and the constraints that must be attended to. In Section 3, we consider HST problem constraints in light of the modeling assumptions that were made within the factory scheduling domain, and discuss implications with respect to use of an opportunistic scheduling methodology. In Section 4, we describe a representation and system architecture for state-based scheduling.

2. The HST Scheduling Problem

The HST is a sophisticated observatory due to be placed into low earth orbit in late 1989 and expected to have a lifetime of around 15 years. HST will offer new opportunities to the astronomical community, and contention for viewing time can be expected to be high. Generally speaking, the HST scheduling problem involves determination of execution times for observations specified in a set of previously accepted observation programs subject to a complex and conflicting set of constraints. As in [13], we presume a hierarchical decomposition of the overall problem over different time horizons, and we focus specifically on the short term scheduling problem (one week to one month) where all orbital constraints are known with certainty. Our intent in this section is to provide an indication of the diversity and nature of the problem constraints. The reader is referred to [12] for a more complete description.

An observation program accepted for execution consists of a set of observations designed to meet specific scientific objectives. The number of programs accepted over a given time horizon is expected to exceed the actual capabilities of the telescope. Some programs are thus designated as supplemental, and their inclusion in the schedule is not guaranteed. Within a given observation program, a variety of relationships among specific observations may be specified, including partial orderings on observations, separation constraints, temporal grouping constraints, coordinated parallel observations, same telescope orientation constraints, and conditional execution The observations in a given program may themselves be prioritized, and, in some cases preferences as to completion levels are specified (e.g. 25% minimally, 50% would be adequate, no more than 75%). Some programs specify observations intended to be executed in parallel with those of other programs if their specific constraints can be mutually satisfied (e.g. same telescope pointing position, sufficient power to operate required instruments).

The execution of a specific observation implies the simultaneous satisfaction of many constraints. Various proposer specified requirements (e.g. target, dark time requirements, time critical exposure), orbital characteristics (e.g. passage through the South Atlantic Anomaly, wherein observing is severely restricted), resource availability constraints (e.g. power) and operational constraints (e.g. pointing restrictions relative to the sun, bright and earth; spacecraft roll constraints) combine to delineate possible execution intervals for a given exposure activity. Some of these constraints can be selectively relaxed (e.g. roll constraints can be sometimes compromised - implying less energy from the solar arrays - provided sufficient time is subsequently spent on-roll to recharge the batteries). Execution of an exposure also requires an ability to establish the "state" specified by observing requirements. This typically requires the execution of sequences of auxiliary "setup" activities. The telescope may require repositioning to point at the target (called slewing), guide stars must be acquired and locked onto by the telescope's fine guidance sensors, the designated instrument and detectors must be brought to the appropriate configuration state, etc. Furthermore, specified communication requirements dictate the execution of additional communication activities. The execution of each of these supporting activities is subject to constraints that typically differ from those of the actual exposing activity (e.g. visibility of communication satellites, tape recorder capacity). Observations may be designated as interruptible (e.g. if the target is occulted for some portion of the telescope's orbit), necessitating the execution of additional setup activities (e.g. guide star reacquisition). Setup activities can often be performed in parallel (e.g. slewing while warming up the required instrument), and it is advantageous to do so as long as relevant constraints (e.g. required power) can be mutually satisfied.

Thus, the HST scheduling problem is one of maximizing the amount of science viewing time while attending as much as possible to the diverse preferences of specific observation programs and insuring feasibility with respect to the complex set of constraints surrounding operation of the telescope and execution of observations.

3. Implications for Constraint-Based Problem Solving

Characteristics of the HST scheduling problem, call into question some of the modeling assumptions that were possible in the factory scheduling domain. This, in turn, has implications with respect to providing an ability to generate and revise scheduling decisions in an opportunistic manner. This issue is considered below.

One broad distinction that can be drawn relative to the characteristics of factory scheduling and HST scheduling, is that factory scheduling problems are typically much less dominated by absolute temporal constraints than is the HST scheduling problem. There are of course deadlines in factory scheduling (and meeting them is very important), but these do not place rigid constraints on the execution of particular production activities. The point is that there is a certain degree of robustness in any factory schedule that is generated. Minor deviations from the schedule during its execution do not have a drastic effect on the overall performance of the factory (e.g. whether a given activity is performed 5 minutes ahead or behind schedule typically has little global impact).

A second distinction, owing more to the specific manufacturing domains we have addressed, concerns the level of interaction between the setup activities that must necessarily be performed to satisfy state-dependent constraints on production activities and HST observations respectively, and the presence or absence of constraints on the execution of these setup activities. In the manufacturing scheduling problems we have considered, such interactions have been minimal and setup activities themselves have been relatively unconstrained, allowing the prespecification and use of setup duration constraints[2].

These constraint characteristics have been exploited within our factory scheduling work to facilitate use of an opportunistic, constraint-based scheduling methodology. The OPIS factory scheduling system [17], which exhibits this capability, operates with respect to simplified assumptions regarding the state of resources over time, explicitly modeling only their available capacity, and assuming all other aspects of their state to be a function of the last activity performed. Resource setup activities are implicitly modeled as adjustments to the durations of activities that require them. These assumptions enable advance instantiation of the possible sequences of activities required to produce the set of production units that must be manufactured, and thus enable the scheduler to maintain an accurate characterization of current solution constraints [10].

In the HST domain, in contrast, it is simply not possible to operate under such modeling assumptions. The dominating presence of state-dependent constraints requires reasoning relative to an explicit model of the actual world state and the on-line expansion of sequences of activities to satisfy observation setup constraints. At the same time, given the overall size of the problem, such detailed reasoning can only be feasibly approached once some commitment has been made relative to where on the time line specific observations are to be placed. Thus, it is evident that analysis and opportunistic commitment with regard to specific observations must take place relative to approximate models of current solution constraints, and as such, these commitments can, at best, provide constraints on the actual decisions that must ultimately be taken. Such commitments must be subsequently refined so as to both insure their feasibility (i.e. that requisite activities can be accomplished in a manner consistent with the decision and the current partial schedule) and attend, as much as is possible, to their optimality (i.e. that the final placement of all constituent activities on the time line reflects relevant scheduling objectives). In the following section, we define a general scheduling framework that supports such decision-making.

4. A Generalized Scheduling Framework

In this section we introduce the general purpose framework that we are developing to solve the HST scheduling problem. We begin by introducing the main assumptions and conceptual primitives on which the architecture rests. First we discuss how the physical system over which the scheduler has to reason is represented. Then we discuss how we specify to the scheduler what it should accomplish on the physical system, both in terms of what to do and under what conditions. Finally, we describe the architecture, outlining three modules that constitute it.

4.1. Modeling the Physical System

Every scheduling problem is defined with respect to a physical system. Classical formulations of scheduling problems [8, 2] describe the physical system only in terms of two entities: **actions** and **resources**. For each resource the amount of *processing capacity* available over time is defined. The fundamental assumption is that when an action is executed, it consumes a fixed amount of capacity from a single resource for the course of its duration. The evolution of such a physical system is consistent if there is never an instant of time t at which the sum of the requests of processing capacity of each action in-process at time t exceeds the capacity available on the resource at that time.

[2]Note, however, that this is certainly not true of all manufacturing environments (e.g. an automated manufacturing cell)

The system description implied by this classical formulation is insufficient for scheduling problems, like the HST scheduling problem, where the execution of actions depends not only on resource availability. If we want to observe a target, for example, we need to insure that the target is visible throughout the observing action. It is conceptually incorrect to interpret a target as a resource and visibility as a processing capacity since it is never the case that the target loses any fraction of its visibility during an observation. The assumption of renewability (i.e. that capacity is required only for the duration of the action) is also problematic in many cases. For example, capacity on the on-board tape recorder is consumed by "write" actions and is not again available until the data is read out.

Our approach to the representation of the physical system is philosophically in line with that of [4, 9, 6]. In the following we will highlight the main characteristics of the corresponding description language; its complete description can be found in [14].

At any point in time we can describe the state of the system with a finite number of predicates. Each predicate represents one of the following:

- **actions**: these are transformations of the state of the system that have a known duration and are explicitly initiated by the executor of the schedule;
- **events**: these are transformations of the state of the system with fixed duration that are outside of the direct control of the executor of the schedule;
- **stable states**: these are reached after an action or an event has terminated. Their duration can depend on the occurrence of other actions or events occurring after their start time.

Let's consider some examples drawn from the HST domain:

The predicate:
LOCKED (Target_X)

represents a stable state. It will appear in the description of the state of the system whenever the telescope is pointing toward *Target_X* and is locked on it.

An example of an action is the predicate:
INSTRUMENT-STATE-TRANSITION (Wf/Pc, StandBy, Operational)

which represents the warmup transition on the *Wf/Pc* instrument that brings it from state *StandBy* to state *Operational*.

An example of event would be:
UNLOCKING (Target_X)

that express the fact that the telescope is losing its lock on *Target_X*. This event will start when the state of the system contains a predicate indicating that HST is locked on *Target_X* and a predicate indicating that *Target_X* has become not visible.

The basic temporal representation used to describe predicates associated with a temporal duration is the *time map* (TM), described in [3]. To each action, event and stable state is associated a **time token**, consisting of a triple $<P, t_{start}, t_{end}>$, where P is the corresponding predicate and t_{start} and t_{end} are nodes in the time map. All the nodes in a time map (except one) designate either the start time or the end time of a token; the exception is the node **ref** that represents the origin of the time axis. For each node in a TM we maintain two numbers $<d_{MAX}, D_{min}>$ representing respectively the maximum of the minimum distances from **ref** and the minimum of the maximum distances of the node from **ref**. The two couples associated with the nodes of a time token represent a generalization of a time bound as described in [10].

Time tokens are also organized along another dimension. Specific sets of predicates are associated with specific **state variables**. For example, the following formulas:

LOCKED (*?target*)
UNLOCKED (*?target*)
LOCKING (*?target*)
SLEWING (*?target_1*, *?target_2*)
UNLOCKING(*?target*)

constitute the descriptors of all possible values of the state variable **HST-pointing-status**. The basic constraint implied by a given state variable is that only one of its possible predicates can hold at any point in time. A state variable has a function similar to the *clipping constraints* described in [3].

The last thing we have to express in order to completely describe the physical system is its dynamics. This includes explicit representations of which predicates across state variables can hold simultaneously at any point in time, the preconditions of a predicate, etc. For example we have to express the fact that the telescope can be locking on a target only while the target is visible. This is expressed by saying that while the predicate *LOCKED* (*?target*) holds, the predicate *VISIBLE* (*?target*) must hold too, where the variable *?target* unifies with the same individual in both predicates. Another way to say this is that in any description of the behavior of the system over time, the presence of a time token $\mathbf{tk}_2$ of type *LOCKED* (*Target_X*) implies the presence of a time token $\mathbf{tk}_2$ of type *VISIBLE* (*Target_X*) that temporally contains it; contains has the same semantics as in [1]. A complete presentation of the description language of the system's dynamics can be found in [14].

4.2. Specification of a Scheduling Problem

The framework for describing the physics of the system presented in Section 4.1 gives us the possibility to express more general scheduling problems than those classically considered. In general, we can say that in order to specify a scheduling problem one has to formulate a set of **scheduling goals** and a set of **scheduling constraints**. In the following we specify what we intend with this terminology.

A **scheduling goal** is a specification of *what* we want the system to do. It takes the form of a predicate that has to hold in any solution. Generally speaking, scheduling goals include both actions to be executed and states to be achieved. In classical formulations of the scheduling problem [2], the latter type of goal has not been accorded full status (e.g. allowing expression of only required resource capacity). By contrast, a solution of to the HST scheduling problem (as well as many others) requires full treatment of both types of scheduling goals. Specification of an observation implies both the definition of actions to be executed (e.g. taking an exposure and communicating to Earth) and the definition of sets of stable states to be achieved (e.g. the viewing instrument and detector in operate mode, the telescope in the Earth's shadow, etc.). The description language presented in Section 4.1 provides a general framework for expression of both types of goals.

The second component of a scheduling problem is a specification of a set of **scheduling constraints**. An important class concerns *when* we want the system to reach the scheduling goals. This generally includes specification of both relative and absolute temporal restrictions on scheduling goals. With respect to this issue as well, classical formulations have typically imposed limiting assumptions (e.g. total orderings over the set of operations to be performed). In the HST domain, a variety of temporal restrictions on observations is possible, including parallel observations, partial orderings of sets of observations, and specific observation time windows. Representation of such temporal constraints is straightforward in the time map formalism. Figure 4-1 represents an observation of duration d that is constrained to start after time t_1 and to end before time t_2. Figure 4-2 represents an expose and communicate action sequence with no intervening temporal gap.

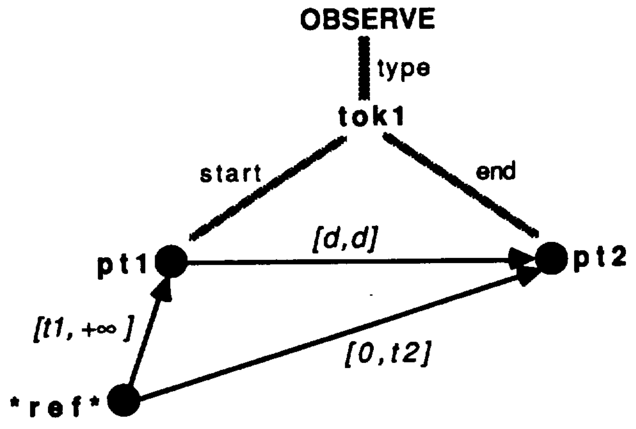

Figure 4-1: Representation of absolute temporal constraints

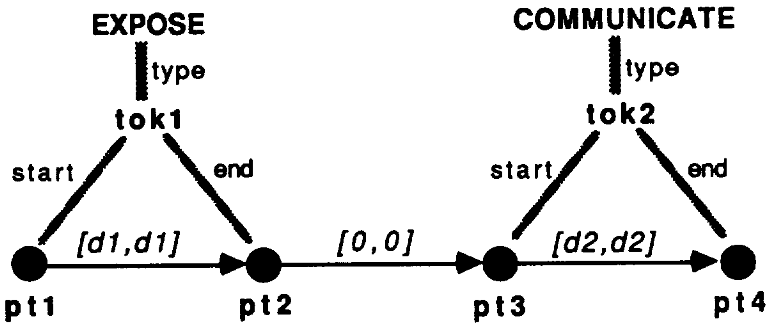

Figure 4-2: Representation of relative temporal constraints

Another class of scheduling constraints relates to objectives and preferences that we would like the system to satisfy to the extent possible. In some cases, such constraints may be defined relative to specific temporal restrictions, for example *"Execute action x as soon as possible after action y"*. In other cases, they define priority relationships among sets of scheduling goals. In the HST domain, for example, an observation program may designate preferences with respect to the number of observations that must be executed. The representation of such constraints has not been discussed in this paper, but we assume use of a utility-based formulation as in [7].

4.3. The Architecture

As in OPIS [17], the scheduler builds its schedule incrementally according to evolving characteristics of solution constraints. This is accomplished through the iterative application of three modules: a **Sub-Problem-Selector**, a **Planner**, and a **Reserver**. This process is outlined below.

The scheduler starts with a description of the expected evolution of the state of the physical system over time and an initial scheduling problem. This information is represented by two separate TMs. Namely:

- **Scheduling Problem TM (SPTM)**: This is a representation of the **scheduling goals** and **temporal scheduling constraints** that constitute the current scheduling problem.
- **System's Simulation TM (SSTM)**: This represents a complete deterministic simulation of the state of the system over time.

The first step of the iterative scheduling process is accomplished by the **Sub-Problem Selector** module. The role of this module is to opportunistically focus the attention of the system. Minimally, this involves selection of a sub-problem TM from the SPTM for solution relative to the full model of the underlying physical system. The introduction of additional scheduling constraints into the selected sub-problem TM is also possible (e.g. restricting attention to the interval between two previously achieved goals), if further constraining of the detailed problem solving effort is deemed appropriate. Decision-making at this level is based on analysis of the scheduling constraints associated with as yet unachieved scheduling goals in the SPTM. To this end, we assume that the consequences of scheduling commitments recorded in the SSTM (see below) are reflected back into the SPTM (similar to the manner in which operation time bounds are modified by resource unavailabilities in OPIS [10]). Since the focus of this module encompasses the entire scheduling problem, any consideration of tradeoffs relative to achievement of scheduling objectives necessary to support subproblem formulation must necessarily make use of approximate models of actual setup constraints (e.g. proximity of targets as a means for approximating slewing time in the HST domain).

The sub-problem TM determined by the sub-problem selector module forms the nucleus of a third type of TM:

- **Plan TM (PTM)**: It represents all the possible evolutions of the system deriving from the execution of a given set of actions that reaches the scheduling goals of a sub-problem TM under the specified scheduling constraints

The process of augmenting the sub-problem TM to form a complete PTM is performed by a **Planner** module: the complete description of the planning algorithm can be found in [14]. In synthesis, the planner keeps a set of planning goals (PGs); a planning goal is a specification of a token that has to be present in the PTM. Initially the set contains all the tokens that form the sub-problem TM. After selecting a PG from the set, the planner will expand it both backwards and forward. The backward expansion is analogous to the one performed in classical linear planning systems [5]. The forward expansion is equivalent to a forward simulation and it is performed in order to detect possible inconsistencies with the **reservations** (defined below) in the current SSTM. While processing the current PG, a new PG is generated and introduced into the set of current sub-goals if:

1. the new PG is pre-condition of the current PG;
2. the new PG is an effect of the current PG;
3. the new PG is has to hold in parallel with the current PG;

4. the new PG corresponds to a reservation that clashes with the current PG.

The new PGs in 1, 2 and 3 are directly obtainable from the System's dynamic description mentioned in Section 4.1. A plan is found when the set of pending PGs is empty.

The last step in the scheduling cycle is performed by the **Reserver** module. This selects a single start time for each of the actions in the PTM. The corresponding evolution of the state of the system is merged with the current SSTM, forming a new SSTM that solves the current scheduling sub-problem. The reserver has also to mark some tokens in the new SSTM as **reservations**, indicating that they need to be preserved in any further extension of the SSTM.

The scheduling cycle is repeated until either all scheduling goals in the SPTM have been achieved or it has been determined that it is not possible to achieve those that remain.

5. Summary

In this paper, we have described a scheduling framework that extends the applicability of an opportunistic scheduling methodology previously developed and validated in factory scheduling domains to problem domains where solutions must satisfy complex state-dependent constraints. We examined the characteristics of the HST observation scheduling problem, which is representative of this type of problem domain, pointing out the dominating presence of state-dependent constraints, the inadequacy of modeling assumptions that were possible in previous work, and the implications with respect to opportunistic scheduling. This led to the presentation of a representation and architecture for state-based scheduling, which we are currently developing to solve the HST observation scheduling problem. This framework enables complete treatment of state-dependent constraints while retaining the flexibility to incrementally construct schedules in an opportunistic manner.

Acknowledgements

The authors wish to thank Katia Sycara for helpful comments on an earlier draft of this paper.

References

[1] Allen, J.
Maintaining Knowledge about Temporal Intervals.
Communications ACM 26:832-843, 1983.

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

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

[4] Dean, T.L., R.J. Firby, and D. Miller.
The FORBIN Paper.
Technical Report YALE/CSD/RR#550, Yale University, Dept. of Computer Science, July, 1987.

[5] Fikes, R.E.,P.E. Hart, and N.J. Nilsson.
Learning and Executing Generalized Robot Plans.
Artificial Intelligence 3:251-288, 1972.

[6] Forbus, K.D.
Introducing Actions into Qualitative Simulation.
Technical Report UIUCDCS-R-88-1452, Univeristy of Illinois at Urbana-Champaign, Dept. of Computer Science, August, 1988.

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

[8] Garey, M.R. and D.S. Johnson.
Computers and Intractability.
W.H. Freeman and Company, San Francisco, 1979.

[9] Hogge, J.C.
TPLAN: A Temporal Interval-Based Planner with Novel Extensions.
Technical Report UIUCDCS-R-87-1367, Univeristy of Illinois at Urbana-Champaign, Dept. of Computer Science, September, 1987.

[10] LePape, C. and S.F. Smith.
Management of Temporal Constraints for Factory Scheduling.
In C. Rolland, M. Leonard, and F. Bodart (editors), *Proceedings IFIP TC 8/WG 8.1 Working Conference on Temporal Aspects in Information Systems (TAIS 87).* Elsevier Science Publishers, held in Sophia Antipolis, France, May, 1987.

[11] Miller, D.P.
Planning by Search Through Simulations.
Technical Report YALEU/CSD/RR#423, Yale University, Computer Science Dept., October, 1985.

[12] Miller, G., D. Rosenthal, W. Cohen, and M. Johnston.
Expert Systems Tools for Hubble Space Telescope Observation Scheduling.
In *Proceedings of the 1987 Conference on Artificial Intelligence Applications.* NASA Goddard Space Flight Center, 1987.

[13] Miller,G., M. Johnston, S. Vick, J. Sponsler, and K. Lindenmayer.
Knowledge-Based Tools for Hubble Space Telescope Planning and Scheduling: Constraints and Strategies.
In *Proceedings Goddard Conference on Space Applications of Artificial Intelligence.* NASA, May, 1988.

[14] Muscettola, N.
Incremental Temporal Planning.
Technical Report, Carnegie Mellon University, The Robotics Institute, forthcoming.

[15] Ow, P.S. 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.

[16] Panwalker S.S. & Iskander W.
A Survey of Scheduling Rules.
Operations Research 25:45-61, 1977.

[17] Smith, S.F.
A Constraint-Based Framework for Reactive Management of Factory Schedules.
In M.D. Oliff (editor), *Intelligent Manufacturing.* Benjamin Cummings Publishers, 1987.