Management of Temporal Constraints for Factory Scheduling

Claude Le Pape[1] and Stephen F. Smith

CMU-RI-TR-87-13

Intelligent Systems Laboratory
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

June 1987

This work was sponsored in part by the Air Force Office of Scientific Research under contract F49620-82-K-0017 and The Robotics Institute.

This paper appeared in *Proceedings IFIP TC 8/WG 8.1 Working Conference on Temporal Aspects in Information Systems (TAIS 87)*, eds. C. Rolland, M. Leonard, and F. Bodart, Elsevier Science Publishers, held in Sophia Antipolis, France, May 1987.

---

[1]Current address: Ecole Normale Superieure, 45 rue d'Ulm, 75230 PARIS CEDEX 05, FRANCE

Table of Contents

List of Figures

Abstract

In this paper, we present constraint propagation techniques used in the OPIS scheduling system to update schedule descriptions and detect introduced inconsistencies. Our approach is summarized as follows:

- A hierarchical model is used to represent resources and operations to be performed. Schedules are developed and maintained at different levels of precision which are explicitly associated with resources and operations; constraint propagation is correspondingly performed at different levels.
- Various scheduling constraints are attached to resources and operations, and combined to derive time bound constraints. A description of the original constraints that collectively impose a bound is explicitly recorded.
- Time bound constraints are maintained by an object-oriented propagation process: through messages, resources and operations communicate constraints and cooperate to compute time bounds.
- When time bounds are inconsistent, their origins provide the information required to construct an appropriate description of the conflicting situation. This description provides OPIS with information needed to make reactive decisions.

1. Introduction

For the past several years, we have been investigating knowledge-based approaches to the difficult problem of job shop scheduling. In addressing this problem, there have been two broad objectives. The first concerns an ability to effectively *predict* shop behavior through the generation of production plans that reflect both the full complexity of the factory environment and the stated objectives of the organization. The second concerns an ability to intelligently *react* to changing circumstances, as the shop floor is a dynamic environment where unexpected events continually occur and quickly force changes to planned activities.

Previous work in knowledge-based factory scheduling has characterized the generation of production schedules as a constraint-directed activity. This work has recognized the importance of exploiting temporal knowledge as a basis for constraint propagation, and various propagation techniques aimed at maintaining the consistency of scheduling decisions were implemented in both the ISIS [Fox 84] and SOJA [LePape 85a] scheduling systems. More recent work with the OPIS scheduling system [Smith 86] has advocated a more opportunistic approach to scheduling. Scheduling decisions are not made according to a fixed problem decomposition strategy (e.g., in an order-by-order fashion) but instead characteristics of the problem at hand and the current state of the schedule are used to dynamically decompose the problem and focus the scheduler in the most appropriate fashion (e.g., schedule the critical resources first). This leads to the use of multiple scheduling perspectives [Smith 85], each differing in the focal point of the scheduler (e.g., order or resource) and the specific types of scheduling concerns that are emphasized. At the same time, our increased emphasis on reactive schedule revision has underscored the fact that, while it is important to provide a detailed schedule for the immediate future, the unpredictability of factory operation makes it counterproductive to do so over an extended time horizon. Not only are the chances of actually executing such a schedule remote, but efforts to reactively revise the schedule become increasingly more complex. Thus, OPIS develops and maintains schedules at different levels of abstraction over different time horizons. Implementation of these ideas has demanded more sophisticated support in the areas of constraint propagation and consistency checking. In this paper, we describe the schedule management system developed to fill this need.

In the context of scheduling, two objectives for temporal constraint propagation can be distinguished. First, constraint propagation provides a distributed characterization, at any point during schedule development, of the set of alternatives that exist with respect to the scheduling decisions that remain to be made. This enables the scheduler to adopt different local views of the schedule with some assurance that the scheduling decisions made will be globally consistent. Just as important, it enables the scheduler to determine which parts of the schedule are more constrained, and structure the overall scheduling task accordingly.

The second objective of propagation relates to consistency checking and the detection of constraint violations. The need for consistency checking arises in two contexts. The first concerns the occurrence of unanticipated events during the schedule execution; it is necessary to determine when and how such events invalidate the predictive schedule so that appropriate reactive decisions can be taken. A second context where consistency checking can play an important role is in the synthesis of partial solutions proposed during schedule generation. Given a decomposition of the scheduling problem, guaranteeing the consistency of solutions adopted for distinct subproblems is in general a very complex task. Indeed, abandoning the assumption that subproblem results must be compatible, and instead implementing techniques to detect and correct situations in which subproblem results are incompatible is often a much more viable approach.

Section 2 discusses the models of temporal propagation and consistency checking previously or currently used within planners and scheduling systems. Section 3 briefly describes our multiple level representation of factory schedules. Section 4 and 5 are respectively concerned with the temporal constraint propagation and the consistency checking techniques we have implemented in OPIS. Finally,

in Section 6, we remark on the contributions and limitations of our approach.

2. Related Work

In representing temporal constraints, two kinds of representations can be distinguished: symbolic representations, which allow the expression of various temporal relationships between events (e.g., *X before Y*), and numeric representations, which provide a basis for describing how important dates associated with the occurrence of events are numerically related along the time line (e.g., *end-time(X) < start-time(Y)*). These two kinds of representation are often integrated in the same planning or scheduling system, in which case the numeric representation is used to refine the relations expressed by the symbolic representation.

As pointed out by Rit [Rit 86], we can similarly distinguish two kinds of time propagation systems: the symbolic and the numeric ones. Symbolic systems combine relationships with a temporal logic a la Allen [Allen 81] in order to infer additional relationships and detect contradictions. A complete set of axioms such as *(X before Y) and (Y before Z)* $\rightarrow$ *(X before Z)* is used for that purpose. Such systems are naturally oriented towards reasoning tasks requiring determination of the relative occurrence of events in time (e.g., natural language comprehension). Numeric systems, alternatively, allow deduction of new temporal equalities and inequalities from a set of existing ones. For example, $start(Z) \geq (today+2)$ can be deduced from the following inequalities: $today \leq end(X)$, $end(X) \leq start(Y)$, $start(Y)+2 \leq end(Y)$ and $end(Y) \leq start(Z)$. These systems are appropriate for tasks involving the absolute placement of activities along the time line, and generally exploit a pre-existing set of symbolic relationships among activities. Variants have appeared in the context of several planning and scheduling systems: NONLIN [Tate 76] and ISIS [Fox 84, Smith 83] use a critical path method to compute earliest and latest start and end times of activities, DEVISER [Vere 83] maintains consistency by narrowing time windows associated with activities as decisions are made, and SOJA [LePape 85b] guarantees a schedule's feasibility by checking for the introduction of positive circuits in a "scheduling graph". More recent work [Bell 84, Rit 86] has proposed techniques that permit uncertainty with respect to activity durations and delays between successive activities.

The work of McDermott and Dean [McDermott 82] [Dean 86] is an exception to this classification since they provide a temporal logic able to make both symbolic and numeric deductions. Furthermore, the recording of data dependencies provides a basis for understanding any inconsistencies that arise. The resulting temporal constraint propagation and consistency checking systems are the most powerful. However it would appear that their computational requirements place practical limits on the types of problems that can be addressed. Nonetheless, the Time Map Maintenance system [Dean 86] has been used with success by the FORBIN [Miller 85] single robot planning system.

In providing support for the OPIS scheduler, we have been concerned with numerical representation and propagation of temporal constraints. Descriptions of part production processes define the possible temporal relationships among the operations that must be scheduled, and provide the framework for time bound propagation. Our approach diverges from existing numerical propagation systems for the following reasons:

- In order to perform propagation at several levels of abstraction, it is necessary to reason about the properties of the specific operations and resources concerned with the propagated temporal information.
- In order to accurately characterize an inconsistency introduced into the schedule, it is necessary to determine which constraints and decisions collectively led to the inconsistency. For example, adding $start(Z) < (today+2)$ to the set of inequalities mentioned above leads to an inconsistency. To properly understand and react to it, we need to know not only which inequalities led to $start(Z) \geq (today+2)$, but also what kinds of constraints are represented by these inequalities. Consequently, we advocate the explicit recording of a symbolic description

of the constraints that enable deduction of a given numeric inequality.
The resulting system for temporal propagation and consistency checking is described in the next sections.

3. Representing a Schedule and the Relevant Constraints

As alluded to above, propagation of the temporal consequences of scheduling decisions within OPIS is based on a rich underlying model of the production environment. Production processes are characterized at different levels of abstraction, with various time and capacity constraints associated with the operations and resources defined at each level. Three broad classes of constraints are distinguished within the OPIS factory model. *Restrictions* delineate the space of admissible solutions to the considered problem while *preferences* are constraints whose satisfaction may be compromised if necessary. The third class, referred to as *conflict-avoidance preferences*, is composed of the constraints introduced as a result of specific allocation decisions. In this section, we consider the nature of these constraints and the time bound information that is derived from them.

3.1. Scheduling Constraints

A shop schedule is characterized within OPIS by instantiating the appropriate production plan for each pending order, and associating time bound information with the instantiated plans. A production plan is represented as a hierarchy of operations: the root operation represents the entire production process to fulfill an order, and any given operation can be refined into either a sequence of more detailed sub-operations, a set of operations to be performed in parallel, or a set of exclusive alternatives. Resource requirements are associated with each operation in the plan, as are a variety of temporal constraints:

- Release and due date constraints specify that the production plan can not start before a given date and should ideally be finished at another given date.
- Expected durations and set-up times are designated for each operation.
- Precedence relations state that some operations must be finished for others to start. They can be refined by the specification of minimum and/or maximum delays between successive operations.

Finally, a large group of temporal constraints are related to the status of the current schedule and to the events that happen on the shop floor. They state that it is impossible to modify the actual start and end times of an operation that is completed or in-process on the shop floor, and undesirable to do so with the scheduled start and end times that have been assigned to a scheduled operation.

To parallel the operations defined in a given production plan, resources are also described at various levels of abstraction. Work-cells (i.e., resources that can perform one operation at a time) are successively grouped, by function, into work-areas. Two major kinds of work-areas can be distinguished: coupled work-areas are composed of resources that are configured to operate in a serial fashion while decoupled work-areas are composed of resources that can be used alternatively to perform the same operations. Capacity constraints are defined at each level. For example, lists of structures of the form *(interval-of-time available-capacity list-of-operations)* are associated with decoupled work-areas. These structures specify, for each interval of time, the number of available work-cells (capacity) and the list of operations currently scheduled (some may actually have started) in the work-area during that interval. An operation can be assigned to the work-area during a given interval of time only if the capacity it requires (e.g., a single work cell) is available throughout the desired interval. At any level of description, a capacity limitation can appear either as a restriction or as a conflict-avoidance preference.

The instantiated production plans provide a framework for generating and maintaining schedules at different levels of precision. Specific levels are defined relative to these plans by attaching precision level information to operations and resources. As a system default (which can easily be changed or extended by the user), characteristics of the resources to be allocated are used to define five distinct levels:

. *shop-level:* The only resource defined at this level is the shop:itself. S.cheduling at this level merely enables estimation of whether due-dates are l.kely to be sat.sf.ed.

.*interchanaeable-cell-group-tevel:* A natural partition of the shop is obtained by grouping t o S T t h t f e r e ^ u L s that can be used to perform similar operations. Scheduling at th.s K performed without differentiating among the interchangeable resources of a group.

.*ideniical-cell^roup-levet* A more detailed level of precision is obtained by considering S S T t h a t represent functionally equivalent sets of subresources (i.e. hav.ng the same operating characteristics). Equivalent subresources are not distinguished.

. *setup-levet* This refines the previous level by individually considering those work-cells where operation setup time is dependent on the operation just performed.

. *work-cell-level:* This is the most detailed level of scheduling. All work-cells are individually allocated to operations and a time-table is generated for each of them.

This delineation of levels at which scheduling might be performed is certainly not exhaustive, and we expect that the specific levels of interest in particular application domains will vary. However, the ability to schedule at different levels of precision is an important tool in dealing with the uncertainty inherent in manufacturing activities. Indeed, the precision at which production schedules are generated should reflect the amount of uncertainty that is present. One rough indicator of uncertainty, the time to execution, suggests the use of distinct time horizons; a detailed schedule is maintained only over an immediate short term horizon (e.g., the next day) and less precise schedules are maintained over longer term horizons.

&2. Time Bound Constraints and Their Origins

The time bound constraints posted by the propagation system in response to a change in the current state of the schedule are derived through composition of the constraints identified above. These new constraints state that a given operation can not start or end before or after a given date without altering the satfefacifen of the original constraints. The specific constraints that have collectively imposed a given time bound dictate both the importance of the bound and the type of problem that will arise in the event that the bound is violated (e.g., poor satisfaction of a due date constraint, an operation precedence violation, aBocatfon of a resource that is not available during the designated time interval). A time bound constraint can itself be viewed as a *preference,* if its violation results in an inability to totally satisfy a particular preference constraint (e.g., a due date win be missed), a *conflict avoidance preference,* if its violation impfies some amount of rescheduling, or a *restriction,* if it was derived only from scheduling restrictions (In which case the violation of the time bound constraint is an illegal move). To provide a basis for making such distinctions, the propagation system explicitly records the origins of each time bound constraint that fe posted.

Different sets of constraints may be considered in the computation of a particular bound. For example, the earfiesi start time of the first operation of an order might be defined with respect to either a preferential release date or the scheduling restriction that states that nothing can be scheduled before today. By default, al preferences are taken into account until they are explicitly relaxed. In other cases, specific *commkts* may "shadow" others. For example, the *due* date of an order will play no role in the definition of the latest and ime of an operation that precedes a scheduled operation. The due date preference is, in th» case, hidden by the scheduled start time of the downstream operation. Generally speaking, 'If different seta of constraints lead to the same time bound constraint, we designate the *origins* of the bound to be the mott riQtt of these sets, La. the one that contains the most important constraints. Given a partial orderinc, *, of scheduling constraints (Cj» Cj means: Ci is more important than Cj), we say that a set S_i of constraints is more rigidthan&jifandorUyifthe following property is verified: for all c, in S,, there exuts a c, **fci** S_t such that Cj is more important than c^ By default, a restriction is considered more **m p « I M** than a cortta-avoidance preference which is in turn more important than a preference. The deraul rtiat«n can be reined by the scheduling system's policies concerning the importance of various

Time bounds of an aggregate operation are obtained either by computing the time bounds of its sub-operations and propagating the results upwards, or by making a direct estimation at the level of abstraction of the operation. A direct estimation is performed only when the operation is at the required level of precision.

4. Propagating Time Bounds

4.1. General Behavior of the System

The computation and maintenance of operation time bounds over time is accomplished by an object-oriented propagation process. The required level of precision and a temporal constraint with its origins are passed with every propagation message. The required level of precision can be either a constant (a designated level of precision), or a function of two parameters, the receiving operation and the time bound passed, which returns a specific level of precision. The use of such a function enables the specification of propagation strategies that vary the level of precision over distinct time horizons.

Upon revision of an operation time bound or its origins, propagation messages may be communicated in any of five directions:

- forward propagation: messages are sent forward through the instantiated production plan whenever an operation has its actual, scheduled or earliest end time updated, or if the corresponding order's requested start date is set or modified. The constraint passed in this case specifies how the new status of the preceding operations restrain the choice of a start time for the operation that receives the message.
- backward propagation: messages are sent backward through an instantiated production plan whenever an operation has its scheduled or latest start time updated, or if the corresponding order's due date is set or modified. The constraint specifies how the status of the following operations restrain the choice of an end time for the operation that receives a message.
- upward propagation: a message is passed upward to an operation by one of its sub-operations in the event that either of the sub-operation's bounds have been updated. The constraint specifies the new value of this changed bound.
- downward propagation: messages are sent downward by an aggregate operation in situations where it does not reside at the required level of precision. The types of constraints communicated in this case are the same as those passed during forward and backward propagation.
- inter-order propagation: Whenever the available capacity of a resource changes, time bounds of the unscheduled operations that require this resource are also likely to change, and messages are sent to these operations for that purpose. The constraint describes the change made in the resource availability.

When an operation receives a message, it interprets the constraint and determines which of its bounds or bound-origins may be affected by the introduction of the constraint. If the operation resides at the required level of precision, the constraint is combined with other temporal constraints that concern the operation (e.g., duration, set-up time) and the capacity limitations of its required resource. The new values of the time bounds and their origins are then deduced. If the operation does not reside at the required level of precision, the time bounds of its sub-operations are computed (via downward propagation) and the bounds of the operation are deduced from these more precise bounds. Three kinds of aggregation can be handled by the propagation system:

- aggregation over exclusive alternatives: in this case, the earliest (resp. latest) start and end times of the operation are the minimal (resp. maximal) values of the earliest (resp. latest) start and end times of its sub-operations.

- aggregation of an ordered sequence of operations: in this case, the earliest and latest start (resp. end) times of the operation are equal to the earliest and latest start (resp. end) times of the first (resp. last) sub-operation of the sequence.
- aggregation of a set of operations that can be performed in parallel: in this case, the earliest and latest start (resp. end) times of the operation are the minimal (resp. maximal) earliest and latest start (resp. end) times of its sub-operations.

Another interesting aggregation for scheduling would be the aggregation of an unordered list of operations that can not be performed in parallel. However, such an aggregation can be obtained by combining the first two, so there was no need to introduce it explicitly in the system.

When an operation has its bounds or the origins of its bounds updated, it determines how this can affect the time bounds of other operations and sends them messages when necessary.

4.2. Initiation of a Propagation Process

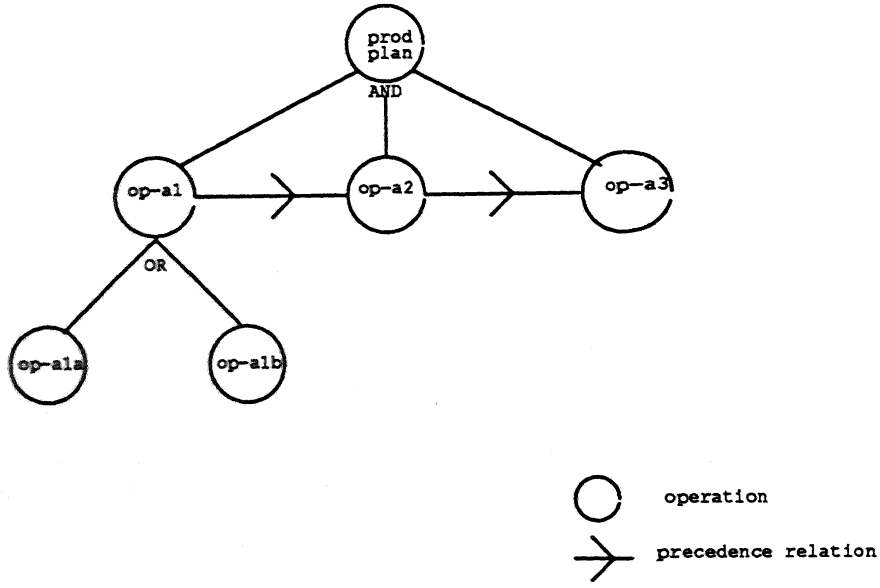

Figure 4-1: An operation hierarchy

To understand when and how constraint propagation is initiated, we will consider the operation hierarchy of figure 4-1 and the two following levels of precision: *level-1* is defined by the operations *op-a1*, *op-a2* and *op-a3* and *level-2* by the operations *op-a1a*, *op-a1b*, *op-a2* and *op-a3*. We will assume that maintenance of time bounds is originally required at *level-1*. When the hierarchy is first created, all earliest start and end times are set to $-\infty$ and latest start and end times are set to $+\infty$.

First, a requested release time for the order is specified. The root operation *prod-plan* receives the release time constraint, checks the required level of precision and sends the constraint to *op-a1*. *op-a1* belongs to *level-1*, so its earliest start and end times are updated. Then a message is sent to *op-a2* and so on. Once the propagation reaches *op-a3*, upward propagation is initiated to establish the earliest start and end times of the root operation. A similar process is initiated upon specification of the requested due-date for the order, except that it concerns manipulation of latest start and end times, and starts by propagating the constraint to *op-a3*.

Several types of changes can then occur:

- The level of precision can be changed from *level-1* to *level-2*. Two propagation processes are then started. The first one concerns the earliest start and end times. *prod-plan* receives a message and responds to it by sending a message to *op-a1*. Since *op-a1* does not belong to *level-2*, messages are sent to *op-a1a* and *op-a1b*, and their earliest start and end times are computed. Earliest start and end times of *op-a1* are then deduced and a message is sent to *op-a2*, etc. The second propagation process concerns the latest start and end times. *prod-plan* receives a message, sends one to *op-a3*. *op-a3* notices that its bounds do not have to change, but propagates the request for precision to *op-a2*. This operation, in turn, sends a message to *op-a1* which sends messages to *op-a1a* and *op-a1b*.
- One of the operations in the hierarchy may be scheduled. In this case, the operation sends messages to the neighboring operations and the propagation starts.
- One of the operations *op-a1a* or *op-a1b* may start on the shop floor. An actual start time and an estimation of the actual end time are provided, a message is sent (upwards) to *op-a1* and the propagation starts. The same thing happens when the operation actually ends.
- Inter-order propagation steps are performed whenever the capacity of a resource is modified and may lead to a modification of time bounds throughout all the operation hierarchy. This happens for example when an operation (belonging to another production plan) is scheduled or starts on the shop-floor, or when a machine breaks down. Capacity changes are propagated throughout the resource hierarchy and their consistency with the current schedule is checked prior to the initiation of any time bound propagation processes.

5. Temporal Consistency Checking

5.1. Detecting Constraint Violations

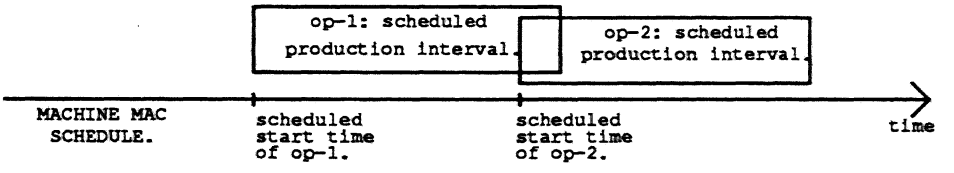

Figure 5-1: Machine *mac* is allocated to operations *op-1* and *op-2* during overlapping intervals of time.

The association of time bound constraints and their origins with individual operations provides a straightforward basis for detecting situations in which either a constraint is violated or several constraints are in conflict with one another. The aim of consistency checking is to detect such situations and provide the OPIS scheduler with the information necessary to take appropriate reactive decisions.

Figure 5-1 gives an example of a simple capacity constraint violation in which machine *mac* has been allocated to operations *op-1* and *op-2* during overlapping intervals of time. A similar situation arises when a machine fails, and projected down time overlaps with previously made allocation decisions. Capacity constraint violations are detected as soon as they occur. Indeed, whenever a scheduling decision or a change in the shop state occurs, the concerned resources are informed of the change and check the satisfaction of their capacity constraints.

Figure 5-2 shows an operation *op-1* starting later than predicted, making the scheduled start time of

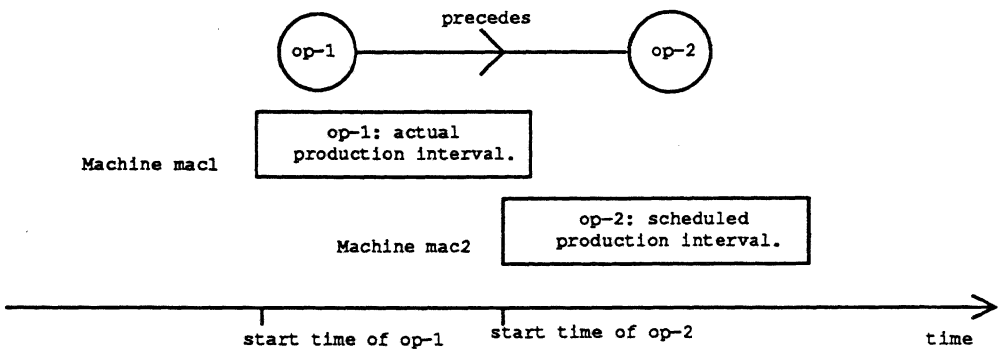

Figure 5-2: Operation *op-1* is late and meeting the scheduled start time of *op-2* is now impossible.

op-2 inconsistent with the current state of the shop. Actually, the earliest start time of *op-2* happens to be greater than its scheduled start time and the anomaly is detected as a consequence of the propagation of the expected actual end time of *op-1*.

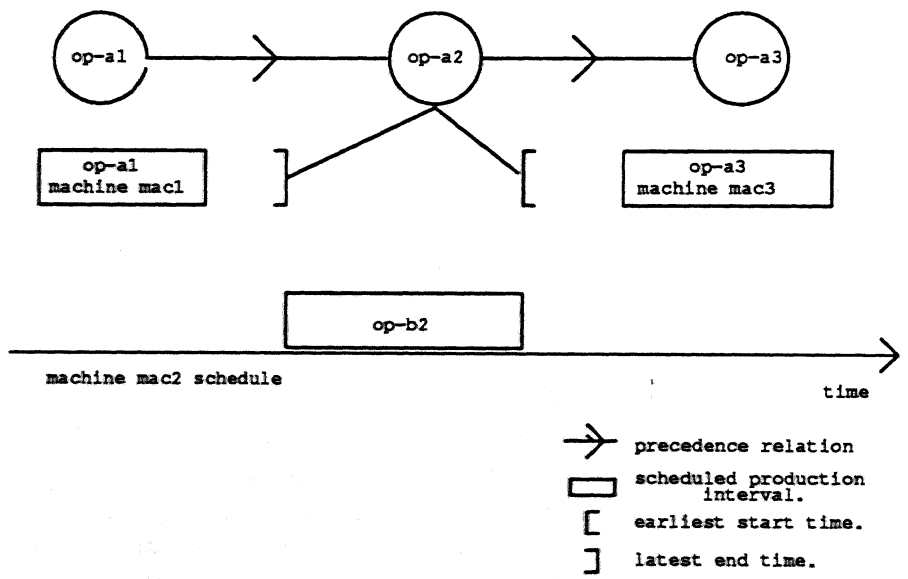

Figure 5-3: A conflicting situation involving both resource availability and temporal constraints.

Performing an operation during its associated time bound interval is necessary to satisfy all of its originating constraints. Therefore, a conflicting situation is detected whenever the width of the interval (i.e., the difference between the latest end time and the earliest start time) becomes smaller than the operation duration. Such a situation is illustrated in figure 5-3: *op-a2* must be performed between *op-a1* and *op-a3*, but the presence of *op-b2* in the schedule of *mac-2* obliges *op-a2* to start after the end of

op-b2 and to end before the beginning of *op-b2.* The origins of the conflicting time bounds provide the information needed to characterize the conflict.

5.2. Characterizing Conflicts

Communication of a detected conflicting situation to the OPIS scheduler involves the construction of an appropriate event description. Different types of events are distinguished according to the kind of constraints that are conflicting with each other. When some of these constraints are scheduling preferences, an event is created to state that the current solution requires some degree of compromise with respect to the satisfaction of these preferences. Otherwise, the conflicting situation is described as an inconsistency. A distinction between time constraints and capacity limitations provides a finer level of discrimination amongst inconsistencies. In fact, the three examples of the previous section (figures 5-1, 5-2 and 5-3) correspond precisely to the three elementary classes of inconsistencies that are distinguished. They show respectively a pure *capacity conflict,* a pure *time conflict,* and a conflicting situation involving both time and capacity concerns (referred to as a *time-vs-capadty conflict).*

A set of *conflicting commitments,* i.e. the set of scheduling decisions made that can not coexist, is associated with each conflict description. Resolution of the conflict implies the revision of at least one of these commitments. Commitments related to time are distinguished from those related to capacity. For example, in figure 5-2, there is a unique time-related commitment: the start and end times assigned to *op-2* must change in order to satisfy a precedence constraint between *op-1* and *op-2.* The inconsistency of figure 5-1 involves two capacity-related commitments: either the scheduled interval of *op-1* or *op-2* (or both) must be revised. Figure 5-3 shows an example in which there are two time-related commitments *(op-a1* and *op-a3)* and a commitment related to the capacity of *mac-2 (op-b2).*

In a conflicting situation like the one of figure 5-3, the set of operations that cannot be scheduled without violating one of the conflicting constraints constitutes another important piece of information provided about the conflict. Indeed, since a reaction will aim at introducing possibilities for scheduling these operations, it will depend on their characteristics. This information is also obtained from the recorded origins of the conflicting situation.

6. Conclusion

The approach to maintaining a factory schedule described in this paper has been integrated with an event-based control mechanism to provide the current framework for reactive control in the OPIS scheduling system [Smith 87]. The framework provides a basis for opportunistic development of the schedule from both resource and order based perspectives, with the propagation system comrminicating relevant constraints between individual subtasks, and detecting problems that arise during the synthesis of subtask results. Control heuristics which operate on the event descriptions produced by the consistency checker provide the necessary guidance in resolving any constraint conflicts that arise. We are currently investigating the usefulness of various reactive rescheduling strategies in responding to inconsistencies imposed by unanticipated external events.

Several interesting issues relating to our approach to temporal constraint propagation remain to be addressed. One concerns the level of effort that should be expended in detecting inconsistencies. Our consistency checking currently does the minimum amount of analysis possible in detecting temporal inconsistencies in the schedule (i.e. inspection of the relationships between start and end time bounds). While this is sufficient to detect all conflicts that arise, some amount of additional analysis can often enable earlier detection of conflicts, which suggests the possibility of less wasted scheduling effort. For example, assume two unscheduled operations must compete for the same resource, and according to current time bound information there is sufficient capacity and time to schedule each operation. It may in fact be the case that there will be a problem once one of the operations is scheduled {i.e., there is time and capacity to perform either one of them but not both). Had a more extensive analysis of the bounds

been made prior to scheduling one of the operations, the future conflict could have been anticipated. Of course, there is a tradeoff with respect to the amount of computational effort one would like to spend in consistency checking. The nature of this tradeoff should be explored.

A second issue concerns the management of schedules at different levels of precision over different time horizons. As the propagation system is currently implemented, any change made to the schedule at the detailed level will result in unrestrained propagation of consequences to all higher levels. Ideally, one would like upward propagation to be more selective; only detailed changes that are significant at higher levels (i.e., require more global reaction) should in fact reach the higher levels. For example, minor adjustments in the detailed, short-term horizon as factory operation continues would not generally be expected to drastically alter the aggregate long-term schedule. The solution to this problem lies in the development of appropriate temporal abstractions that "filter out" minor deviations as they are propagated upward. Varying the granularity of the time line employed at different levels of aggregation is one such abstraction that we are currently exploring.

Acknowledgements

The work described in this paper was performed while Claude LePape was a visiting scholar in the Intelligent Systems Laboratory of the Robotics Institute funded by a grant from INRIA. The authors would like to thank Nicola Muscettola, Peng Si Ow, and Bruce McLaren for many helpful discussions during the design of the schedule management system and Kevin Neel for his implementation efforts.

References

[Allen 81] James F. Allen.
An Interval-Based Representation of Temporal Knowledge.
In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence.* Vancouver, Canada, August, 1981.

[Bell 84] Colin E. Bell and Austin Tate.
Using Temporal Constraints to Restrict Search in a Planner.
Technical Report, Artificial Intelligence Applications Institute, University of Edinburgh, December, 1984.

[Dean 86] Thomas Dean.
Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving.
PhD thesis, Yale University, May, 1986.

[Fox 84] Mark S. Fox and Stephen F. Smith.
ISIS: a Knowledge-Based System for Factory Scheduling.
Expert Systems 1(1):25-49, July, 1984.

[LePape 85a] Claude Le Pape and Bernard Sauve.
SOJA: un Systeme d'Ordonnancement Journalier d'Atelier.
In *Les systemes experts et leurs applications : cinquiemes journees internationales.* Avignon, France, May, 1985.

[LePape 85b] Claude Le Pape.
SOJA: a Daily Workshop Scheduling System.
In *Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems.* Warwick, Great Britain, December, 1985.

[McDermott 82] Drew McDermott.
A Temporal Logic for Reasoning About Processes and Plans.
Cognitive Science 6:101-155, 1982.

[Miller 85] David Miller, James Firby and Thomas Dean.
Deadlines, Travel Time, and Robot Problem Solving.
In *Proceedings of the Ninth Internationa/Joint Conference on Artificial Intelligence.* Los Angeles, California, August, 1985.

[Rit 86] Jean-Francois Rit.
Propagating Temporal Constraints for Scheduling.
In *Proceedings of the 5th National Conference on Atificial Intelligence.* Philadelphia, Pennsylvania, August, 1986.

[Smith 83] Stephen F. Smith.
Exploiting Temporal Knowledge to Organize Constraints.
Technical Report, The Robotics Institute, Carnegie-Mellon University, July, 1983.

[Smith 85] Stephen F. Smith and Peng Si Ow.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.* Los Angeles, California, August, 1985.

[Smith 86] Stephen F. Smith, Peng Si Ow, Claude Le Pape, Bruce McLaren and Nicola Muscettola.
Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans.
In *Proceedings of the SME Conference on AI in Manufacturing.* Long Beach, California, September, 1986.

[Smith 87] Stephen F. Smith.
A Constraint-Based Framework for Reactive Management of Factory Schedules.
In M. Oliff (editor), *Proceedings International Conference on Expert Systems and the Leading Edge in Production Planning and Control,* pages 349-366. Charleston, South Carolina, May, 1987.

[Tate 76] Austin Tate.
Project Planning Using a Hierarchical Non-Linear Planner.
Technical Report, Department of Artificial Intelligence, University of Edinburgh, August, 1976.

[Vere831 Vere,S.A.
Planning in Time: Windows and Durations for Activities and Goals.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5(3):246-267, May, 1983.