Temporal Reasoning for Planning and Scheduling in Complex Domains: Lessons Learned

Mark S. Boddy*
boddy@src.honeywell.com
Honeywell Technology Center, MN65-2200
3660 Technology Drive
Minneapolis, MN 55418

Appears in "Advanced Planning Technology," AAAI Press, 1996.

Abstract

Over the past five years, we have implemented and applied efficient, general-purpose temporal reasoning as a substrate for building planning and scheduling systems. We have also investigated the kinds of temporal reasoning that will be most useful, and for what problems. Our results confirm that temporal reasoning *is* a sufficiently self-contained activity to be implemented entirely independently of the overlying application, modulo some assumptions about how problem-solving is to proceed. We have also shown that constraint-based temporal reasoning supports a "least-commitment" style of planning and scheduling that is efficacious in a wide variety of complex problem domains. There have been some surprises, as well, for example in the fact that causal reasoning in general, and *projection* in particular, have been less useful than we anticipated.

In this paper, we sketch the design, implementation, and semantics of our current temporal reasoning engine, loosely based upon Dean's Time Map Manager (TMM), discuss how that engine has been applied to a range of planning and scheduling problems, and draw some conclusions. The primary lesson to be drawn is that constraint-based temporal reasoning provides an effective basis for building planning and scheduling systems, particularly in applications where problem-solving is only weakly directed by domain-specific solution methods.

Introduction

The research and implementations described in this paper could plausibly be described as an extended experiment testing a hypothesis first advanced more than ten years ago by Tom Dean, to the effect that "temporal reasoning" was a commonly-needed capability that could usefully abstracted and implemented in a separate module, which could then be used over and over again in different applications (Dean 1984). Our preliminary result is a confirmation of Dean's hypothesis, though just what functions are required of or useful in a temporal reasoner has undergone some modification. For example, causal reasoning in general, and *projection* in particular, have been less useful than we anticipated.

In this paper, we sketch the design, implementation, and semantics of our current temporal reasoning engine, loosely based upon Dean's Time Map Manager (TMM), discuss how that engine has been applied to a range of planning and scheduling problems, and draw some conclusions. The primary lesson to be drawn is that constraint-based temporal reasoning provides an effective basis for building planning and scheduling systems, particularly in applications where problem-solving is only weakly directed by domain-specific solution methods. The version of the TMM described here, developed under a contract as part of Phase I of the Planning Initiative, is presented more fully in (Schrag, Carciofini, & Boddy 1992).

Time map managers

In this section, we briefly discuss the history, architecture, and implementation of the temporal reasoning system implemented at Honeywell as part of Phase I of the Planning initiative. This system was a reimplementation of Dean's *Time Map Manager* (TMM) (Dean & McDermott 1987). The TMM, conceived as a foundation for building planning and scheduling systems, includes capabilities for reasoning about both qualitative and metric temporal constraints, as well as two forms of causal reasoning: projection and persistence, all in the presence of incremental change in a dynamic environment. The version of the TMM described here, developed under a contract as part of Phase I of the Planning Initiative, is presented more fully in (Schrag, Carciofini, & Boddy 1992). The semantics of the system are described in (Schrag, Boddy, & Carciofini 1992).

The TMM provides users and application programs (e.g., planners and schedulers) with the following functionality:

- Qualitative and metric constraints between time points.
- Causal reasoning.

*The work described here was supported in part by ARPA and the Air Force Rome Laboratory under contract F30602-90-C-0102.

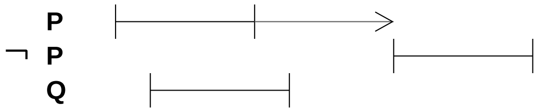

Figure 1: A simple temporal database

- Database monitors for temporal conditions and protections.
- Optimizations for large temporal databases.

The structure and capabilities of the TMM are described in more detail below.

Temporal Relations

The TMM lets users assert *constraints* between pairs of *time points*, resulting in a partial ordering among the points. TMM supports queries regarding necessary and possible temporal relations among the time points. The truth of facts over intervals of time is represented by *tokens,* which may include properties of *persistence* beyond their observed endpoints. In the current implementation, tokens may persist both forward and backward in time. The truth of a proposition over an interval is determined based on the ordering of token endpoints and the token's persistence properties. For example, Figure 1 is a simple temporal database, involving three tokens of three different types. In this example, P is true over the interval bounded by the vertical lines, and persists into the future. (not P) becomes true at a later time, and *clips* the forward persistence of P. The statement "P and Q" is true for an interval defined by the overlap of the tokens labelled P and Q.

Causal Reasoning

The TMM currently supports reasoning about the changing state of the world as activities occur using two forms of inference:

- The persistence assumption. As described above, users of the TMM specify that certain facts are believed to be true over specific intervals of time. In addition, they can specify that those facts can be assumed to remain true until something occurs to make them false.
- Projection. This is inference of the form: given an event E and a set of preconditions $\mathsf{P}_1, \mathsf{P}_2, \ldots \mathsf{P}_k$, and a result R, whenever the preconditions are believed to be true for the entire event E, R is believed to become true immediately following E.

These forms of inference are handled completely automatically: the user specifies which facts are persistent and asserts a set of projection rules, and the requisite inference is performed by the system.

As Dean and Boddy show in (Dean & Boddy 1987), reasoning about projection and persistence with partially-ordered points is an NP-complete problem. The current implementation employs a decision procedure which approximates corrects inference in a polynomial algorithm. The approximation is sound but incomplete: the system never concludes incorrectly that a proposition P necessarily holds at some point t, but may fail to conclude that relation when it does in fact hold. [1]

Nonmonotonic Reasoning and Database Monitors

TMM supports two basic kinds of nonmonotonic reasoning:

- Possibly true temporal relations between time points (which may be invalidated by additional constraints), and
- Assumed truth of a temporal proposition over an interval based on a time token's persistence (which may be invalidated by the addition of a contradictory token, which *clips* the proposition during that interval).

In addition, the database itself is "nonmonotonic", in the sense that information can be deleted, and the inference performed by the system thus far will be checked to ensure that it continues to be supported by the current state of the database. [2]

The existence of specified database properties as changes are made over time can be tracked through the use of *monitors.* The existing types of TMM database monitors are *temporal conditions* and *protections.* Temporal conditions monitor whether specified relations among points can be derived from the current state of the database, maintaining this information as the database changes. Protections do the same thing for the truth of some fact over an interval. Between them, these two mechanisms provide support for monitoring the continued validity of previous inference, or triggering demons based on complex properties of the temporal database.

Efficiency

Current and planned TMM optimizations for handling large databases include the use of a global reference point where appropriate (rather than forcing its use as some systems do), limiting search to that necessary to prove or disprove a query, caching search results for later use, graph decomposition, temporal indexing, lazy monitor evaluation, and algorithms that are designed to search only those parts of the database that may result in useful answers.

Temporal reasoning for planning and scheduling

Our efforts in building or extending planning and scheduling systems using the temporal reasoning capa-

[1] See (Schrag, Boddy, & Carciofini 1992) for a more complete discussion.

[2] This capability (*temporal reason maintenance*) is described in (Dean & McDermott 1987).

bilities of the TMM and its descendants started with a fairly complete picture of what was desired and what was required to achieve those desires. These desiderata were in some measure untested, and part of what we have done is to validate them.

We are interested in providing solutions to large, complex planning and scheduling problems. A "solution" as we use the term is not simply an implementation of an algorithm for solving a particular constraint satisfaction or constrained optimization problem. For many domains, constructing plans or schedules is an extended, iterated process that may involve negotiation among competing agents or organizations, choices made for reasons not easily implementable in an automated system, and last-minute changes when events do not go as expected. In such an environment, the process by which a plan or schedule is constructed must be considered in any attempt to provide a useful system.

Scheduling

In addition to the difficulties attendant upon providing support for scheduling as a *process*, the more circumscribed problem of generating a single schedule is becoming increasingly complex. Application domains may involve large problem instances, context-dependent activity effects, complex resource structures, and user preferences that may or may not be susceptible to automation. The operational customers for scheduling systems are increasingly sophisticated about the nature of their applications and the scheduling problems involved, and thus less and less willing to settle for gross simplifications in how the problem is solved. To provide an effective solution, a scheduling system must be expressive enough to represent or reflect these domain complexities as well as supporting the process by which a schedule is constructed.

The assumptions underlying our scheduling work are as follows:

1. Explicitly modelling the constraints resulting from specific scheduling decisions makes the schedule easier to construct and modify.
2. Representing only those relationships required by the current set of constraints (the decisions made so far) provides a more useful picture of the current state of the scheduling effort.

The main consequence of this approach is that the scheduler does not manipulate totally-ordered timelines of activities and resource utilization. Instead, the evolving schedule consists of a partially ordered set of activities, becoming increasing ordered as additional constraints are added (or less so, as those decisions are rescinded).

In our approach, which we call *Constraint Envelope Scheduling* (CES), schedules are constructed by a process of "iterative refinement," in which scheduling decisions correspond to constraining an activity either with respect to another activity or with respect to some

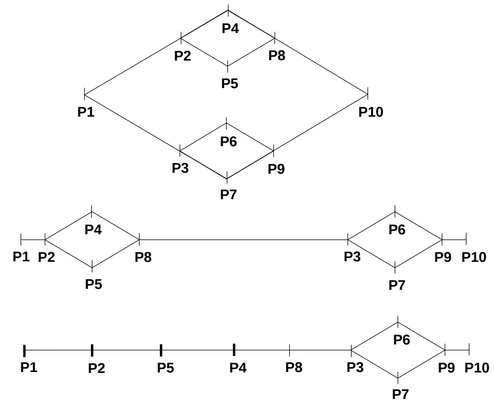

Figure 2: Gradual hardening of a partial order

timeline. The schedule becomes more detailed as activities and constraints are added. Undoing a scheduling decision means removing a constraint, not removing an activity from a specified place on the timeline. Figure 2 depicts the process by which a partially ordered schedule is gradually refined into an executable, totally ordered schedule.

This increased flexibility comes at a cost. The explicit representation of partially-ordered activities in the time map makes reasoning about resource usage and other state changes more complicated. Causal reasoning and resource profiles both depend on precise orderings of facts and activities in time, that is, on what propositions are true and what activities occur when. For a given partial order, we can determine what facts might *possibly* or *necessarily* hold at a point, in some or all of the total orders consistent with the given partial order. With even a very simple causal model, this is an NP-complete problem (Dean & Boddy 1988).

We finesse this problem in several ways. First is by using the approximate reasoning described in the previous section, implementing *strong* and *weak* reasoning as approximations for what is possibly or necessarily true, given the current partial order. For certain classes of inference, in particular problems involving resource capacity or the aggregate duration of mutually exclusive activities, strong and weak (even exact "necessary" and "possible") reasoning occasionally provides insufficient information. For these cases, there are two possible approaches: simulation (sampling) of totally-ordered sequences, or some kind of static graph analysis to determine better bounds on the system's behavior. The end result in either case is an estimate of the likelihood that further constraints on the partial order will uncover inconsistencies requiring the scheduler to backtrack to earlier choices.

Despite the approximate nature of this reasoning, we are still ahead of the game: where the least-commitment approach to scheduling can at least provide approximate answers in support of scheduling decisions (e.g. what order activities should occur in), timeline schedulers make the same decisions arbitrarily—putting an activity on the timeline is a stronger commitment than constraining it to occur (say) between two other activities, or within a given time window.

Planning

TMM support for efficient reasoning about partial orders was initially intended to support nonlinear planning. The original intuition behind nonlinear planning was the happy thought that we could plan by a process of adding orderings only as required to ensure that our plan has the right effects. This "least commitment" approach was extended to other forms of constraints in constraint-posting planning (e.g., (Stefik 1981)). As an incremental process of eliminating possible plans from a set defined by the current set of constraints, constraint-posting planning exactly parallels CES.

The fact that actions take time was abstracted out in the earliest domain models, but metric time and durations are slowly being incorporated. This kind of reasoning tends to be computationally expensive. Forbin, Deviser, and Sipe all suffer from performance problems limiting the size of the problems to which they can be applied. In addition, various people have implemented "temporal reasoning systems" in an attempt to isolate and optimize reasoning about the relations among actions, or between actions and other processes (Allen 1983; Dean & McDermott 1987; Arthur & Stillman 1992). OPlan-2 contains a separate reasoning module (the *time point network manager*) with similar functionality (Tate, Drabble, & Kirby 1992).

The correct level of integration between a planner and a temporal reasoning system such as the TMM is an open question. For example, in doing projection, the TMM builds data dependencies that look much like causal links in an SNLP-style planner, but without the bookkeeping required to explore the set of possible links in a systematic manner (McAllester & Rosenblitt 1991). An SNLP-style planner could make use of the lower-level temporal reasoning capabilities of the TMM without employment of the causal reasoning machinery. Backstrom and Nebel provide additional arguments that projection may not be a useful form of inference to support current generative planning techniques (Nebel & Backstrom 1992).

Applications

Planning and scheduling tools using the TMM have been developed for a wide variety of domains, including operations planning for a Space Shuttle science module, satellite data analysis and retrieval for NASA's Mission to Planet Earth, and processor and communication scheduling for the Boeing 777 Flight Management System, and batch scheduling. These systems run the gamut from predominantly manual to completely automatic.

Spacehab: The *Kronos* scheduler was developed for the purpose of assisting humans in scheduling experiments for a science module that has flown on several Space Shuttle missions. The scheduling process is largely manual. Activities are manually selected to be added to the schedule. Constraints such as execution windows and predecessor/successor relationships are enforced automatically. The availability of needed resources is also checked automatically—a real benefit, since resource requirements can be complex—but the choice of resource assignment is left up to the human user, unless there is only a single consistent assignment. Possible and necessary resource conflicts are detected and presented to the user for resolution, either through the addition of ordering links between activities, or by changing resource assignments. These "resource bounds" are the visible manifestation of the envelope of schedules consistent with the current set of constraints.

EOSDIS NASA's Earth Observing System (EOS) is a multi-year, multi-billion-dollar project aimed at gathering scientific information about the Earth's environment through satellite-based remote sensing. The EOS Data and Information System (EOSDIS) will be responsible for archiving and analyzing the resulting data. EOSDIS functions include managing mission information, archiving and distributing data, and generating and disseminating scientific data products (Dozier & Ramapriyan 1990). We have provided two separate applications in support of this process. The first is the *Data Archiving and Distribution Scheduler* (DADS), which schedules the retrieval of data from terabyte tape archives, its staging on disk during processing, and eventual distribution to the interested parties (Short *et al.* 1995). DADS includes an interface to an execution and dispatch system, through which the schedule is updated automatically as the course of events diverges from that predicted in the current schedule. This updating capability highlights one of the strengths of the least-commitment CES approach. As long as the course of events does not diverge too far from that predicted, the system simply updates the projected start and end times of activities as appropriate. When a deadline or some other commitment is threatened, that fact is detected and the system will reschedule, either automatically or by prompting a human user to make a decision.

The other system we have provided is an extension of an existing NASA planning tool for image analysis, based on Nonlin. Our extension of this tool in a system called PlaSTiC (Boddy *et al.* 1995), adds the ability to backtrack in planning based on estimates on the duration of planning (sub)tasks at any level in the planning hierarchy. These duration estimates are ultimately derived from worst-case bounds on execution times for the primitive subtasks involved, but can also be modified directly based on previous experience. The estimates are updated automatically as the system runs, permitting a smooth response to a changing computational environment.

Avionics scheduling Another of the applications to which we have applied constraint envelope scheduling is the generation of static schedules for processing time and bus communications, involving safety-critical applications running on flight hardware on a commercial airplane. The schedule is static for reasons having to do with verifiability and repeatability of behavior, and ultimately with FAA certification for flight safety. This problem is both large and complex. In a typical problem instance, there are approximately 30000 activities. There are six processors, all between 80% and 90% loaded, with processes periodic at rates between 5 Hz and 80 Hz. Data communication is specified between processes, not between process instances. To make matters worse, we are constructing a schedule for a 200 mS "frame" which itself runs at 5 Hz. Communication from one instance of this frame to the next is entirely legal, and so there is in some sense a circular model of time, in which constraints on activities late in the frame may affect activities early in the frame.

The system constructed for this problem generates an initial schedule entirely automatically—the problem is too big for any but the most abstract level of manual intervention. Rescheduling is a more interactive process, in which the user may specify preferences regarding what in the existing schedule will change. These preferences are structured according to how changes in the schedule affect the certification level of various system functions. Further information on implementation details is available in (Boddy & Goldman 1994).

Batch manufacturing The term *batch manufacturing* is used for a range of manufacturing processes best characterized in terms of what they are not. Batch manufacturing processes are neither *continuous* nor *discrete*. Batch processes usually involve the manufacture of "stuff" measurable and divisible by volume or weight, but in discrete lots of limited size. Products commonly manufactured using batch processes include cosmetics, plastics, and beer, among many others. Batch manufacturing presents special challenges for scheduling systems, including the unpredictability of batch manufacturing processes, the dynamic nature of the environment, and the nature of batch manufacturing plants, which typically involve a wide variety of shared resources. The *Honeywell Batch Scheduler* (Goldman & Boddy 1996) is a prototype batch scheduling system for batch applications, designed to accommodate the uncertain, dynamic nature of batch manufacturing processes, as well as the complex resource and process models that are required. The system has met with considerable success, and has served as a valuable source of data regarding the strengths and weaknesses of our scheduling technology.

Lessons learned

The implementations described in the last section have served to test, and largely confirm, our initial assumptions regarding the useful attributes of a temporal reasoning system to support planning and scheduling. A constraint-based, least-commitment approach is in fact useful across a wide range of domains, and for a wide variety of interaction styles, involving both human users and other automated components. There are a few additional lessons to be drawn, as well.

For example, any effective scheduling system for a moderately complex domain needs to support the user in constructing an initial schedule, modifying an existing schedule, and detecting and characterizing infeasibility. The last of these functions is not unanimously recognized as crucial, or necessarily achievable (consider the popularity of unsystematic heuristic-repair techniques for scheduling), but we have found it to be of great importance. The initial stages of solving a scheduling problem can involve an extended period in which those responsible for generating the initial schedule requirements must be persuaded to modify those requirements to achieve consistency. In at least one case, the opening moves in this process involved decreasing *required* processor loading from something over a hundred percent. So, we do systematic search.

To support both culprit identification in inconsistent problems and incremental rescheduling, we need to be able to refer to previous scheduling decisions, and to associate them with specific effects on the current (partial) schedule. Our current approach to providing the necessary functionality is to restrict search operations ("scheduling decisions") to those implementable as added variables for which the assignment can be reflected in the underlying CSP as explicitly added constraints (this is a specialized form of propagation). There are some potential difficulties with this approach, including first and foremost the choice of what search variables to add, so as to structure the problem in the appropriate way. This approach presents a significant problem for our use of Constraint Logic Programming tools such as CHiP or ILog Solver.

Another observation we can make is that there is a very significant qualitative distinction between scheduling domains that are strongly "temporal" and those that are not. In particular, a lot of fairly elaborate machinery that we have developed for use on the strongly temporal problems is much less useful when the real difficulty is in terms of, say, resource assignment.

Dynamic backtracking

Our scheduling tools use an extended variant of Ginsberg's Dynamic Backtracking algorithm (Ginsberg 1993). One of these extensions was to enable the search engine to report the set of inconsistent variables involved, should it fail to find a solution. This provides the culprit identification necessary, while the systematic search ensures that inconsistencies will be detected.

The second extension that we made was necessitated by the nature of scheduling problems, or at least of how we represent them. Ginsberg's algorithm involves generating *eliminations*: explanations of why a given value for some variable is ruled out given the current partial assignment. The assumption that eliminations are available by inspection does not work for complex temporal constraints: frequently we discover that a given ordering is infeasible only through some form of propagation. Accordingly, we have extended the algorithm to handle unsuccessful attempts to assign a given value to a variable. In this case, the search engine undoes the assignment (including removing any added constraints), records an elimination explanation for that value, and reports failure back to the scheduler.

Empirically, this extended implementation of Ginsberg's algorithm has been invaluable. Some of our scheduling problems involve some tens of thousands of variables representing choices on ordering, resource assignments, and other relationships. Given the difficulty of localizing variable interaction, sorting related variables to be close to each other is impractical or impossible. Despite considerable effort, we have not managed to find variable or value ordering heuristics that result in backtrack-free solutions (though we do use a variant of Smith's "slack" heuristic for value ordering (Smith & Cheng 1993)).

For these reasons, having a search method that leaves intact that part of a partial assignment not involved in a given inconsistency is crucial. One of the ways in which we might have run into trouble using dynamic backtracking has not materialized, either: inconsistencies typically involve less than 30 variables. This means that the elimination bookkeeping is kept within bounds, as well.

Related work

The idea that schedules should be constructed "from the side," looking at part or all of the schedule history rather that just sweeping forward or backward in time, has been implemented in several scheduling systems, e.g. (Fox 1990; Biefeld & Cooper 1989; Smith *et al.* 1986). Typically, these systems also support an iterative process of schedule refinement or repair. Recent work on COMPASS provides a protocol for allowing different agents to modify the same schedule, wherein commitments made by one agent cannot be affected by the actions of any other. Research in constraint-based scheduling (Fox & Smith 1984; Smith *et al.* 1990; Sadeh & Fox 1990) has demonstrated the advantages of considering the structure of problem constraints over time and using this structure to dynamically focus decision-making on the most critical decisions. However, these systems have historically had a weak model of the interaction of activities and the evolving state of the domain.

Research in generative planning has focused on the construction of activity networks that bring about desired goal states, given basic representations of the effects of actions in the world. Classical domain modeling assumptions (Fikes, Hart, & Nilsson 1972) make it difficult to reason about the duration of activities, continuously varying quantities, and resource consumption. The consequence of these limitations is that automatic planners have not had much success in applications to significant planning and scheduling problems (Dean, Firby, & Miller 1988; Vere 1983). Recent work addressing the integration of constraints and classical planning may result in a resolution of some of these issues (Muscettola 1993; Penberthy & Weld 1993; Tate, Drabble, & Kirby 1992).

Summary and discussion

We have made successful application of our implemented temporal reasoning capabilities to planning and scheduling in a wide variety of domains, supporting a wide range of interaction styles. One of the primary strengths we see in this approach is in that range: being able to move smoothly across a continuum from manual to automatic problem-solving is a valuable characteristic for an operational system. In our experience, users desire for automation increases with increasing familiarity with and trust in the tool they have been given, as well as with an increasing understanding of how to solve the problem at hand using that tool.

Our experiences confirm that temporal reasoning *is* a sufficiently self-contained activity to be implemented entirely independently of the overlying application, modulo some assumptions about how problem-solving is to proceed. We have identified the following functionality as useful in any substrate for providing effective planning and scheduling solutions in complex domains:

- Systematic search, or at least feasibility checks.
- Culprit identification on the detection of inconsistencies (*why* is the problem as specified infeasible?).
- Incremental rescheduling.
- Support for a mix of manual and automatic scheduling.
- A clear mapping from some feature of the current state of the schedule to the decisions responsible for that feature.

We have constructed a variety of flexible and powerful scheduling tools, where the necessary extensions to the TMM were small in comparison to the size of the TMM itself, consisting primarily of interface support and activity and resource models. We have also shown that constraint-based temporal reasoning supports a "least-commitment" style of planning and scheduling that is efficacious in a wide variety of complex problem domains.

There have been some surprises, as well, for example in the fact that causal reasoning in general, and *projection* in particular, have been less useful than we anticipated. Reasoning about scheduling with state is one place that projection might come in handy, but, as in planning, scheduling is frequently more of a synthetic task than an analytic one. So, for example, instead of reasoning using projection about what states hold when, we might specify the state desired at a given point in time, then constrain further development of the schedule so that that specification is satisfied. Projection could be an important capability for planning in the presence of exogenous events, for simulation, or for diagnosis.

Our current work in planning and scheduling addresses several areas, including research on scheduling with state information and the integration of optimization methods with our current system, new system capabilities, such as a negotiation-based distributed scheduling architecture, and further extensions to the efficiency and expressive power of the basic software.

References

Allen, J. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26:832–843.

Arthur, R., and Stillman, J. 1992. Tachyon: A model and environment for temporal reasoning. Notes of AAAI Workshop on Implementing Temporal Reasoning, 1–13.

Biefeld, E., and Cooper, L. 1989. Scheduling with chronology-directed search. In *Proc. AIAA Computers in Aerospace VII*, 1078–1087.

Boddy, M., and Goldman, R. 1994. Empirical results on scheduling and dynamic backtracking. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation for Space.*

Boddy, M.; White, J.; Goldman, R.; and Short, N. 1995. Integrated planning and scheduling for earth science data processing. In Hostetter, C. F., ed., *Proceedings of the 1995 Goddard Conference on Space Applications of Artificial Intelligence and Emerging Information Technologies*, 91–101. Available as NASA Conference Publication 3296.

Dean, T., and Boddy, M. 1987. Incremental causal reasoning. In *Proceedings AAAI-87 Sixth National Conference on Artificial Intelligence*, 196–201.

Dean, T., and Boddy, M. 1988. Reasoning about partially ordered events. *Artificial Intelligence* 36:375–400.

Dean, T., and McDermott, D. V. 1987. Temporal data base management. *Artificial Intelligence* 32:1–55.

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

Dean, T. 1984. Planning and temporal reasoning under uncertainty. In *Proceedings IEEE Workshop on Principles of Knowledge-Based Systems.* IEEE.

Dozier, J., and Ramapriyan, H. 1990. Planning for the eos data and information system (eosdis). In *The Science of Global Environmental Change.* NATO ASI.

Fikes, R.; Hart, P.; and Nilsson, N. 1972. Learning and executing generalized robot plans. *Artificial Intelligence* 3:251–288.

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

Fox, B. R. 1990. Non-chronological scheduling. In *Proc. AI, Simulation, and Planning in High Autonomy Systems.* IEEE Computer Society Press.

Ginsberg, M. 1993. Dynamic backtracking. *Journal of Artificial Intelligence Research* 1:25–46.

Goldman, R. P., and Boddy, M. S. 1996. Constraint-based scheduling for batch manufacturing. *IEEE Expert, to appear.*

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 634–639.

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

Nebel, B., and Backstrom, C. 1992. On the complexity of temporal projection and plan validation. In *Proceedings AAAI-92 Tenth National Conference on Artificial Intelligence*, 748–753.

Penberthy, J. S., and Weld, D. S. 1993. Temporal planning with constraints (preliminary report). In *Working Notes, AAAI Spring Symposium on Classical Planning*, 112–116. Also available as a AAAI tech rpt, forthcoming.

Sadeh, N., and Fox, M. 1990. Variable and value ordering heuristics for activity-based job-shop scheduling. In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, Hilton Head Island, S.C.*

Schrag, R.; Boddy, M.; and Carciofini, J. 1992. Managing disjunction for practical temporal reasoning. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning.*

Schrag, R.; Carciofini, J.; and Boddy, M. 1992. Tmm manual. Technical Report CS-R92-012, Honeywell SRC.

Short, N.; Bedet, J.-J.; Bodden, L.; Boddy, M.; White, J.; and Beane, J. 1995. Constraint based scheduling for the goddard space flight center distributed active archive center's data archive and distribution system. In *Proceedings of the Fourth NASA/Goddard Conference on Mass Storage Systems and Technologies.*

Smith, S. F., and Cheng, C.-C. 1993. Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings AAAI-93*, 139–144. AAAI.

Smith, S.; Ow, P.; LePape, C.; McLaren, B. S.; and Muscettola, N. 1986. Integrating multiple scheduling perspectives to generate detailed production plans. In *Proceedings of the SME Conference on AI in Manufacturing.*

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

Stefik, M. J. 1981. Planning with constraints. *Artificial Intelligence* 16:111–140.

Tate, A.; Drabble, B.; and Kirby, R. 1992. Spacecraft command and control using ai planning techniques. Technical Report AIAI-TR-109, University of Edinburgh.

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