Deadlines, Travel Time, and Robot Problem Solving

David Miller, R. James Firby, and Thomas Dean
Yale University
Department of Computer Science
P.O. Box 2158 Yale Station
New Haven, Connecticut 06520

1 Abstract

This paper describes some extensions to the reductionist planning paradigm typified by Sacerdoti's *NOAH* program. Certain inadequacies of the partial ordering scheme used in *NOAH* are pointed out and a new architecture is detailed which circumvents these problems. An example from the semi-automated factory domain is used to illustrate features of the new planner. Techniques for eliminating unnecessary travel time by the robot and avoiding backtracking due to deadline failures are discussed and their incorporation in the planner is described.

2 Introduction

Most planners since *NOAH* [8] have represented a plan for a set of tasks as a partial (ie., non-linear) ordering of the steps required for carrying out those tasks. *NOAH* and its successors (eg., *NONLIN* [9] and *DEVISER* [10]) employ a partially ordered network of tasks to avoid early and unnecessary commitment to task orderings. The motivation for this is to eliminate backtracking. However, maintaining a consistent partial order is difficult in domains where the fact that tasks actually take time plays an important role: domains in which deadlines or robot travel time are serious considerations.

That a partial order leads unavoidably to a deadline failure usually cannot be discovered until an attempt is made to linearize the ordering. However, failure to notice a deadline violation early will require backtracking later.

In addition, planning with a partial order is not well suited to efficiently managing factors like travel or machine running time. It's not difficult to represent that moving from one workstation to another takes time proportional to the distance separating the two workstations. However, generating a plan that eliminates unnecessary travel between workstations requires exploring some of the linearized task orderings. It is not until tasks are completely ordered that the source and destination workstations of each movement can be known and the travel time computed with any accuracy.

In this paper we describe an approach to planning that combines the use of a partial order with a method for exploring the possible repercussions of that partial order. This approach has been implemented in the *FORBIN* planner (First Order RoBot INtender). *FORBIN* is a planner capable of solving a significantly wider class of problems than any of its predecessors.

2.1 The Factory Domain

One problem domain that has been used for exploring our approach to planning is what we refer to as the *semi-automated factory.* In this domain a mobile robot operator wanders about the factory floor (see Figure 1) performing basic maintenance and supply operations to the factory machinery. The purpose of the factory is the creation of *widgets* and *gizmos*. Whether a *widget* or *gizmo* is produced depends on which bit from the *tool area* has been inserted into the factory's *lathe*. Once a *widget* or *gizmo* has been created it is placed onto the appropriate shelf. The amount of time needed to create an object depends on the type of object being cut.

This factory differs from a typical job-shop factory in several important respects. Two of these are that a complete *job* is not run at one workstation at a time (ie., not all of the widgets in an order are turned on the lathe before being moved to the storage shelves) and the travel time of the robot from one place in the factory to another can be a significant part of the overall factory production time. Like any factory, this one has production deadlines to meet. A typical problem is to construct a number of widgets and gizmos where some of each must be done within a given time limit.

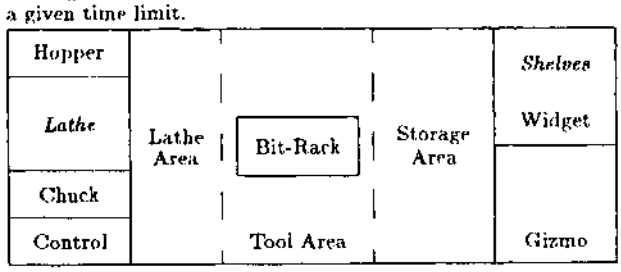

Figure 1: Layout of the Factory

3 An Overview of the FORBIN Project

The most important issue *FORBIN* is designed to explore is the use of spatial and temporal reasoning in planning. To do this the system supplements the usual hierarchical planner with two new modules: The *Time Map Manager* and the *Time Optimizing Scheduler.*

3.1 The Time Map Manager [TMM]

The TMM [3] is a set of routines for reasoning about the occurrence of tasks and the spans of time over which facts can be said to hold. The data structure that the TMM uses to store and manipulate the temporal relationships between these facts is called a *time map.*

Plans generally have assumptions or facts which must be true in order that the plan be applicable in a given situation. The TMM is queried during plan selection to locate intervals over which these facts are true. It returns a number of interval-constraint set pairs each of which specifies a set of constraints on the partial order that must be imposed to keep the assumptions valid over its interval. The TMM monitors the continued validity of plans by setting up nonmonotonic data dependency justifications composed of assertions called *protections.* If a protection fails then the TMM notifies the planner of any plans threatened

by the failed protection. The failure is annotated in such a way as to facilitate corrective action. The TMM also anticipates possible protection failures and suggests ordering constraints to avoid undesirable interactions. The *FORBIN* planner uses this same machinery to handle simple resource management chores (eg., reserve the lathe for a 30 minute stretch beginning after 8:00 but ending before noon). The TMM subsumes and extends the functionality of the TOME (Table Of Multiple Effects) mechanism used in *NOAH.*

3.2 The Time Optimizing Scheduler [TOS]

The nature of tasks in the *FORBIN* domain necessitate that the robot spend considerable amounts of time in transit between workstations. The *FORBIN* system uses the TOS to make plan selection decisions that reduce this travel time and make the overall plan as short as possible.

In the semi-automated factory, where travel distances are sizable with respect to production times, travel time considerations are very important in deciding which of the possible task expansions will eventually yield the most efficient final plan. The TOS chooses the plan that fits best in the overall scheme of things by producing the most efficient schedule of execution using each possible expansion, and choosing the best. The chosen expansion is used in further planning and the schedule is used to guide the order of expansion of further subtasks. When planning is complete the schedule provides the order in which the robot should execute the final set of primitive actions.

The TOS offers an inexpensive method of exploring the schedules that can be formed from the partial order of a set of tasks. The search space for scheduling a set of tasks is factorial in the number of tasks, but [7] and [11] contain heuristics for trimming it significantly. These heuristics mainly use temporal constraints to eliminate impossible schedules before they have been fully elaborated. Ordering constraints and deadlines often eliminate all but a few possible schedules.

A further computational saving is derived by rating the quality of the schedules being constructed, and pursuing only the most promising ones. For the *FORBIN* factory domain the overall execution time is the chief determiner of a schedule's quality and is the discriminating feature among legal, consistent schedules.

No rating system is perfect and it is possible that the schedule first picked by the TOS may not be the optimal one. Since the TOS uses a heuristic search to produce schedules, a longer search time will increase the probability of the program finding the optimal schedule. The TOS can be set to search at any level of detail and can therefore find the best schedule that balances planning time against execution time.

3.3 Flow of Control

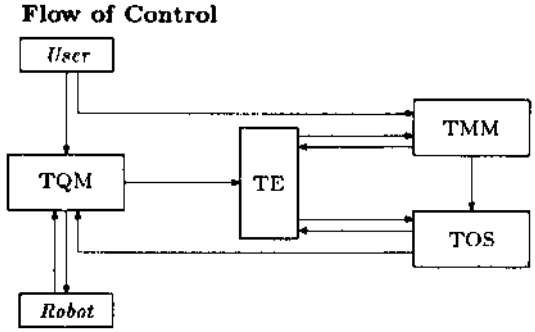

Figure 2: Flow of Control in the FORBIN System

Along with the TMM and TOS, the *FORBIN* system has two other major modules: the *Task Expander* [TE], and the *Task Queue Manager* [TQM]. The TE is responsible for finding all possible plans that can be used to expand a task, and the TQM maintains two queues: the unexpanded tasks in the order they should be expanded and the primitive actions in the order they should be executed. The communication paths between these modules are shown in Figure 2. In addition to the basic modules, interfaces are provided for a *User* which gives the system new tasks to perform, and a *Robot* which performs the primitive actions directed by the planner.

While there are non-primitive tasks in the TQM, *FORBIN:*

1. Pops the first task in the TQM and passes it to the TE.
2. The TE finds all the plan descriptions in the plan library which match the task. For each plan it finds, it asks the TMM for a time or times when the plan could be used.
3. The TMM elerives all of the constraints necessary to make each plan suggested by the TE feasible.
4. The TOS takes all the plan-constraint sets and finds the one that produces the best schedule when combined with the contents of the rest of the time map.
5. The TQM gets the schedule and the TE is passed the selected plan.
6. The TQM takes the schedule, extracts the new subtasks from it and adds them to its queues. The ordering in the schedule is used to help order the items in the queues.
7. The cycle then repeats until the TQM has no more unexpanded tasks.

3.4 The Plan Formalism

To facilitate this flow of control, the plan formalism used by the *FORBIN* system specifies not only the action and ordering information found in other formalisms, but also the temporal and spatial features required by the TMM and TOS. Each plan specification includes:

- Conditions that must hold true before and during plan execution.
- How to expand the plan into lower level actions.
- Effects of some actions and the protections on those effects.
- Aproximately how long the plan will take to execute.
- Where the robot must be to carry out the plan.
- The utility of the plan compared to others for that task.

Plans come in two parts: the *property* descriptor and the *plan* descriptor. For any given task, such as (make ?thing), there will be only a single property descriptor no matter how many plans there are. The property descriptor gives the approximate duration and position of a task before a plan is chosen for it: a combination of the durations and positions of all the plans known for the task. These estimates are required so that the TMM and TOS can deal effectively with tasks that are not yet expanded.

The plan descriptor is used when it is time to expand a task. The plan descriptor contains a list of the plan's subtasks and the mandatory ordering and time constraints that exist on those subtasks. The plan descriptor may also contain *assumptions* which must be predicted to hold true over the intervals specified in or-

dor for that particular plan descriptor to be able to be chosen as the actual plan for the task. The plan formalism is discussed in more detail in [4].

4 FORBIN and What Has Gone Before

The *FORBIN* planning system shares many of the characteristics of earlier planners like *NOAH* and *NONL1N* since it is a hierarchical planner that attempts to leave subtasks only partially ordered *as* long as possible. However, the TOS and TMM give *FORBIN* important new capabilities:

1. to deal with tasks that require specific amounts of time and must be performed at specific locations.
2. to represent deadlines so that plan steps can be synchronized with events outside the planner's control.
3. to produce near-optimal schedules that eliminate unnecessary robot travel and idle time.

The necessity of representing time in planning has been recognized by many researchers: [6], [1], [2]. The *FORBIN* system uses the time map to reason about the temporal intervals associated with tasks. All real tasks take time and hence it is critical that a planner be able to represent and deal with information concerning the duration and separation of tasks. The *FORBIN* treatment of time allows the system to deal with deadlines on tasks, to recognize explicit overlap of tasks where that is possible or necessary, and to compare the predicted execution time of different, planning choices.

Some previous systems that have made extensive use of time are *ISIS,* [5), and *DEVISER,* [10]. *ISIS* uses a heuristic scheduling module to solve job-shop scheduling problems. However job-shop scheduling is too restrictive to handle many common aspects of typical problem solving domains. Thus *ISIS* docs not incorporate travel time between workstations into its scheduling representation. *DEVISER,* though it produces a schedule, does not contain a scheduler. Instead it relies on the general hierarchical planning mechanism combined with backtracking to eventually produce a schedule. By not having a scheduler guide the plan expansion and ordering the results of *DEVISER's* work can be very inefficient and may have involved very large amounts of backtracking.

Earlier planners often made no effort to produce a linear schedule of primitive actions from the partial order of the final plan expansion. In the *FORBIN* factory domain, such a linear schedule is required because the robot can do only one thing at a time. The TOS is used to help keep the best linear schedule implied by the partial order at each planning step as near to optimal (ie., short) as possible. To allow this, travel between tasks is not represented in the plan formalism, instead the location of the task is. Thus, as the TOS is examining possible linear schedules, it calculates the travel time between the ordered tasks as it fits them together. In this way, unneeded travel tasks are not generated and the TOS is free to order travel any way it chooses as long as it does not violate any other constraints on the tasks. The TOS can also overlap the execution of several tasks provided that the overlap is consistent with the constraints in the time map and it does not demand that the robot be more than one place at a. time. These abilities of the TOS give *FORBIN* the opportunity to produce plans that make better use of time than previous planners.

5 Summary

Solutions to planning problems that involve realworld actions must take time and travel into account. The *FORBIN* planning system does this by using two special purpose modules. The TMM constructs and maintains a temporal database in which to reason about tasks and their consequences over time. It then uses this time map to anticipate and suggest methods of avoiding undesirable interactions. The TOS manipulates the partial order of subtasks to find the arrangement that takes best advantage of executing tasks in parallel and eliminates unnecessary travel in order to minimize overall plan execution costs.

A plan formalism has been given that allows all the necessary constraints needed to interface with these modules, to be expressed clearly and cleanly.

The overall system can plan solutions for tasks that have a wide variety of temporal and spatial constraints. The solutions produced by the system are not only consistent with the constraints placed on the problem, but arc also near optimal with regards to their cost in time.

Acknowledgments

The authors wish to thank Steven Hanks, James Spohrer, and Drew McDerniott for their comments and suggestions on this research. This work was supported in part by the Advanced Research Projects Agency of the Department of Defence and monitored under the Office of Naval Research under contract N000l4-83-K-0281.

Bibliography

[I] Allen, James, *Maintaining knowledge about temporal intervals,* Comm. ACM, 26/11 (1983), pp. 832-843.

[2] Cheeseman, Peter, A Representation of Time for Automatic Planning, *Proc. IEEE Int. Conf. on Robotics,* 198-1.

[3] Dean, T., Temporal Reasoning Involving Counterfactuals and Disjunctions, *Proc. of the. Ninth Int. Joint Conf. on Artificial Intelligence,* IJCAI, AAAI, Los Angeles, CA, August 1985.

[4] Firby, R.J., Dean, T., Miller, D., Efficient Robot Planning with Deadlines and Travel Time, *I'roc. of the 6th Int. Symp. on Robotics and Automation,* IASTED, Santa Barbara, CA, May 1985.

[5] Fox, Mark S., *Constraint-Directed Search: A Case Study of Job-Shop Scheduling,* Technical Report CMU-R1-TR-83-22, CMU Robotics Institute, December 1983.

[6] McDerniott, Drew V., *A temporal logic for reasoning about processes and plans,* Cognitive Science, 6 (1982), pp. 101-155.

[7] Miller, David, *Scheduling Heuristics for Problem Solvers,* Technical Report 264, Yale University Dept. of Comp. Sci., 1983.

[8] Sacerdoti, Earl, *A Structure, for Plans and Behavior,* American Elsevier Publishing Company, Inc., 1977.

[9] Tate, Austin, Generating Project Networks, *Proc. IJCAI 5,* IJCAI, 1977, pp. 888-893.

[10] Vere, Steven, *Planning in Time: Windows and Durations for Activities and Goals,* IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-5/3 (1983), pp. 240-267.

[11) Vere, Steven, *Temporal Scope of Assertions and Window Cutoff,* 1984. JPL, AI Research Group Memo.