UNCLASSIFIED DTIC FILE COPY

(2)

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

AD-A196 075 D	1b. RESTRICTIVE MARKINGS
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 88-0526
6a. NAME OF PERFORMING ORGANIZATION: University of Iowa; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR
6c. ADDRESS (City, State and ZIP Code): Iowa City, Iowa 52242	7b. ADDRESS (City, State and ZIP Code): BLDG #410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR- 87-0118
8c. ADDRESS (City, State and ZIP Code): BLDG #410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A7	

11. TITLE (Include Security Classification): Temporal Knowledge Representation and Reasoning For Project Planning

12. PERSONAL AUTHOR(S): Colin E. Bell

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 2/1/87 TO 1/31/88	88/04/25	5

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR	Temporal reasoning, knowledge-based systems, planning

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes research done in temporal reasoning for project planning. The principal investigator produced a computer program to specify a set of temporal constraints, including disjunctive constraints. The program identifies a feasible solution to the constraints, if one exists, and a contradiction toherwise. Dependency-directed bascktracking is employed.

DTIC ELECTED MAY 20 1988 S D CoD

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Abraham Waksman	202-767- 5025	AFOSR/NM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

TEMPORAL KNOWLEDGE REPRESENTATION AND REASONING FOR PROJECT PLANNING

GRANT AFOSR-87-0118, FINAL REPORT

Colin E. Bell
Department of Management Sciences
University of Iowa
Iowa City, IA 52242

ABSTRACT.

I have written a working computer program which allows a user (or a different module of an AI planning program) to specify a set of temporal constraints including disjunctive constraints. In the context of a point-based temporal model, my program has required several innovative design choices. The program will find a feasible solution to the constraints if one exists, otherwise it will identify a contradiction. It is especially appropriate when adding new constraints to currently satisfiable existing constraints.

The philosophical approach of dependency-directed backtracking is employed. This is implemented in a way that takes account of my problem's special structure. The program is particularly successful in certain examples for which chronological backtracking would be hopelessly inefficient. Other notorious examples require time-consuming search. These examples reveal important tradeoffs between explicitly storing temporal knowledge and deriving appropriate temporal knowledge on demand. Particular instances have been identified where explicit storage is critical for avoiding expensive search.

Somewhat independently, I have begun to investigate a novel approach to solving a resource-based project scheduling problem. Although this problem has been investigated by others, I am hopeful that a different approach to it will prove competitive.

1. DISJUNCTIVE TEMPORAL CONSTRAINTS.

Enclosed with this report are copies of the research paper [Bell87] submitted to Artificial Intelligence and my previous progress report [Bell87] submitted to AFOSR on 25 September 1987. The main purpose of my work has been to incorporate explicit disjunctive constraints into a point-based temporal constraint model and to reason with such constraints to prove whether or not they are satisfiable. A point-based model has a network representation whose nodes are instants in time and whose arcs are precedence constraints. An arc I->J of length L_{IJ} is a constraint that at least L_{IJ} time units must elapse between the firing dates of nodes I and J.

Disjunctive constraints arise in many contexts, but one frequent situation involves the use of a resource by competing activities in a plan. For example, if you have only one forklift truck and activities A and B both require it for the entire duration of their execution then you must impose a disjunctive constraint: "either the finish of A (F_A) must precede the start of B (S_B) or the finish of B (F_B) must precede the start of A (S_A)". In other words, the network must contain either an arc
F_A->S_B of length 0 or an arc F_B->S_A of length 0.

Scheduling problems involving such constraints are notoriously intractable. I have developed two approaches which I hope will develop into valuable constraint manager modules of automated AI planning programs. One approach seems appropriate for situations where the number of disjunctive constraints is small relative to the number of non-disjunctive (easier to handle) constraints. The other approach is used in situations dominated by disjunctive constraints. I expect that many AI planning programs would fit the former category except where they involved a heavy sequencing and scheduling component. Thus applications to job-shop scheduling might well require the latter approach.

Let me simply refer to the two approaches as "interaction avoidance" and "resource-based scheduling" respectively. In interaction avoidance relatively few resource usage conflicts are expected but appropriate disjunctive constraints must be added when such conflicts are discovered. In resource-based scheduling it is known in advance that there are a multitude of resource usage constraints to be satisfied. These two approaches have been addressed somewhat independently. The next two sections describe progress so far along each of these two fronts.

2. INTERACTION AVOIDANCE.

The form of disjunctive constraint assumed by my model is motivated by the least-commitment response to the discovery of protection interval violations in planning. If an action in a plan is discovered to interfere with the purpose of other actions, then a (typically disjunctive) additional scheduling constraint must be imposed. At the University of Edinburgh in 1985, I wrote a module for the planner O-PLAN [Currie85] to discover a least-commitment response to protection interval violations. My analysis of this situation led me to believe that an appropriate form for a disjunctive temporal constraint is as a disjunction of conjunctions of precedence arcs.

Figure 1 of [Bell87] gives a typical situation where an additional disjunctive constraint is required to respond to a protection interval violation. The appropriate constraint is given below the network in Figure 1. I call a conjunction of arcs a "cluster"; a constraint is thus a disjunction of clusters. As far as I can tell, mine is the first attempt to explicitly represent disjunctive temporal constraints in a point-based temporal constraint model. The interval-based approach of [Allen83] expresses disjunctive constraints but does not allow for "*metric time*" (*using numerical time units in constraints*).

The implementation uses dependency-directed backtracking specialized to take advantage of the mathematical structure of my constraint model. With this approach to search, one records additional constraints as "nogood" sets of assumptions. A nogood in my context is a set of arcs which cannot be simultaneously present in the temporal constraint network without causing an illegal cycle of non-negative overall length. One contribution of my work is to develop an efficient methodology for discovering appropriate nogood sets. When my program signals that there are one or more nogoods to be found, a procedure is called upon to find those nogoods which are likely to be most powerful in constraining the search. My procedure finds (a subset of) all nogoods of minimal cardinality. Nogoods of low cardinality tend to be the most helpful in further constraining the search space.

Additional contributions come through my choice of strategy for organizing search. Given a current set of constraints, I seek a network representation with (1) enough arcs so that every constraint has at least one cluster all of whose arcs are present in the network (and thus the constraint is satisfied) but also (2) few enough arcs that the network contains no

A-1

illegal cycles of non-negative overall length. The search is organized so that (2) is always maintained while I try to achieve (1).

The user is free to add new (possibly disjunctive) constraints or to retract old ones. Thus reasoning always takes place in the context of some current set of constraints all of which must be satisfied. In this context, each arc can be in any of 4 states: fixedin, in, out, fixedout corresponding to "the constraints are provably unsatisfiable unless the arc is in the network", "the arc is currently in the network but we haven't proved that the constraints are unsatisfiable without it", "the arc is currently not in the network but we haven't proved that the constraints are unsatisfiable with the arc in the network", and "the constraints are provably unsatisfiable unless the arc is not in the network".

When additional constraints are imposed, I am often in a position to reason that certain arcs which were "in" can be made "fixedin" or that certain arcs which were "out" can be made "fixedout". When an existing constraint is retracted, support for some of this reasoning may be removed. Thus retracting a constraint could cause an arc to be reclassified from "fixedin" ("fixedout") to "in" ("out"). My program manages these reasoning processes. [Bell87] describes this work.

3. RESOURCE-BASED SCHEDULING.

More recently I turned attention to a class of resource-based scheduling problems previously attacked by Stinson, Davis, and Khumawala (hereafter "SDK") in [Stinson78]. Although my programming effort is incomplete and I do not yet have computational results to compare with SDK, my approach appears to be novel and complementary to theirs. The class of problem attacked is simply described as: find a minimal "makespan" (i.e. overall project duration) schedule for a project consisting of a fixed number of inter-related activities. The activities have known durations and there is a partial order expressing precedence constraints between them of the form "activityI must be finished before activityJ can start". In addition, there are R shared resources with n_r units of resource r available. Each activity uses a known number of units of each resource (for its entire duration). Thus, additional resource constraints forbid the simultaneous execution of any set of activities which collectively would require more than n_r units of resource r for any r, $1 \leq r \leq R$. We call a set of activities for which parallel execution is consistent with the precedence constraints and which together over-consume at least one of the R resources a "resource-violating set".

Finding a minimal makespan schedule without resource constraints is an easy task. With resource constraints added, computational complexity is known to increase dramatically. The SDK approach is a tree search in which any node is a partially-built schedule starting from time 0. Some subset of the activities have already been assigned a start time consistent with precedence and resource constraints. The children of any node in this tree represent extensions of this partial schedule by assigning start times to any feasible non-empty subset of all not-yet-scheduled activities which are eligible to be executed next. Leaf nodes of this tree are complete schedules in full detail. Of course, pruning techniques are employed wherever possible so that the resulting tree does not grow to include all feasible schedules.

My approach is to avoid the generation of completely detailed schedules by organizing the tree search differently. Any node of the search tree is a network representing a partial order on the set of all tasks. The initial node is simply the original given partial order (ignoring resource constraints). Any node in the search tree can be viewed as representing a convex set of detailed schedules each of which satisfy all constraints specified by the partial order and achieve the minimal makespan allowed by the partial order. To generate the children of

a given node, I identify a resource-violating set. The children are then new networks each with one additional arc inserted to "break up" such a resource-violating set. A leaf node of the resulting search tree is a network which admits no resource violations.

Without having made computational comparisons between my approach and that of SDK, it is still safe to make some general statements which lead me to believe that further investigation is merited. First, as the number of effective resource constraints decrease (other properties of the problem being held equal) the resulting SDK tree grows because any given node is guaranteed to have no fewer children and may have more. On the contrary, with fewer resource constraints to worry about, the size of tree in my approach will typically shrink. Thus our two approaches are complementary in a sense. At least along one important dimension of describing a resource-constrained problem, the SDK approach gets worse as ours gets better (and vice versa).

An encouraging feature of my approach is the fact that a node of my search tree represents a large number of possible schedules. Hopefully my final search tree would have fewer leaf nodes than SDK's since each node in my tree could represent a multitude of individual schedules which each appeared as individual leaf nodes in the SDK search tree.

4. TANGIBLE RESULTS AND FUTURE WORK.

There is no question that the approach taken to interaction avoidance is very worthwhile in some problems. It solves certain potentially combinatorially explosive problems quickly and elegantly (e.g. the example in Figure 2 of [Bell87] for any number of jobs). This approach has been summarized in [Bell87] submitted to Artificial Intelligence and in a companion paper [Bell87b] submitted to Management Science. The latter paper contains essentially the same research results but offers more explanation of AI planning to a management science audience. Both papers are currently under review by the respective journals; no feedback has been received other than acknowledgement of submission.

I have been asked to organize an invited paper session for the Spring National Joint Meeting of The Institute of Management Sciences and the Operations Research Society of America in Washington, DC, 25-27 April. Participants in my session include Drew McDermott of Yale University and Thomas Dean of Brown University. Both of them will report on other aspects of knowledge representation and reasoning in AI planning. I will report on the interaction avoidance phase of my research under AFOSR 87-0118. This talk and accompanying paper are now being prepared. In addition to [Bell87] and [Bell87b] these are the mechanisms for disseminating knowledge on progress so far in interaction avoidance. In the next year of the grant, my interaction avoidance research will receive further attention particularly in improving the control structure of its algorithms. Additional areas of investigation are spelled out in sections 3 and 4 of the accompanying progress report submitted in September 1987.

AI planners generally incorporate little ability to reason about resource constraints. This is often one of their main weaknesses. It is well-known that many resource-based scheduling problems are NP-complete, even those which are as easily described as the set of problems attacked by SDK. I am hopeful that I will be able to test my approach to the SDK problems on the same numerical examples which they used. These examples represented a variety of job-shop, flow-shop, and other production scheduling contexts. Although much effort remains in my research in this area, I have clear immediate plans to experiment with different control structures for search and to test my algorithms on SDK problems. (I am using just one of their sample problems as a test example.)

I hope that I can contribute in two ways:

- by concluding that there are important classes of problems where my algorithms perform better than those of SDK and thereby providing results of interest to production scheduling researchers (perhaps through a paper in a journal such as the International Journal of Production Research),

- by evaluating the contribution of my algorithm within the context of AI planning and thereby clarifying resource balancing issues for researchers in that field.

I thus look forward to continuing my work on two fronts: interaction avoidance and resource-based scheduling.

5. REFERENCES.

[Allen83] Allen, J.F., "Maintaining Knowledge about Temporal Intervals", Communications of ACM, Vol. 26, pp. 832-843 (1983).

[Bell87] Bell, C.E., "Representing and Reasoning with Disjunctive Temporal Constraints in a Point-Based Model", submitted to Artificial Intelligence (1987).

[Bell87a] Bell, C.E., "A Least Commitment Approach to Avoiding Protection Violations in Nonlinear Planning", accepted for publication in special Artificial Intelligence volume of Annals of Operations Research (1987).

[Bell87b] Bell, C.E., "Maintaining Project Networks in Automated Artificial Intelligence Planning", submitted to Management Science (1987).

[Currie85] Currie, K. and A. Tate, "O-Plan: Control in the Open Planning Architecture", Expert Systems 85, Cambridge University Press, (1985).

[Stinson78] Stinson, J.P., E.W. Davis, and B.M. Khumawala, "Multiple Resource-Constrained Project Scheduling Using Branch and Bound," AIIE Transactions, Vol. 10, No. 3, pp. 252-259 (1978).