In *Proceedings of the Eleventh National Conference on Artificial Intelligence* (AAAI-93), 1993

Postponing Threats in Partial-Order Planning

David E. Smith

Rockwell International
444 High St.
Palo Alto, California 94301
de2smith@rpal.rockwell.com

Mark A. Peot

Department of Engineering Economic Systems
Stanford University
Stanford, California 94305
peot@rpal.rockwell.com

Abstract

An important aspect of partial-order planning is the resolution of threats between actions and causal links in a plan. We present a technique for automatically deciding which threats should be resolved during planning, and which should be delayed until planning is otherwise complete. In particular we show that many potential threats can be provably delayed until the end; that is, if the planner can find a plan for the goal while ignoring these threats, there is a guarantee that the partial ordering in the resulting plan can be extended to eliminate the threats.

Our technique involves: 1) construction of an *operator graph* that captures the interaction between operators relevant to a given goal, 2) decomposition of this graph into groups of related threats, and 3) postponement of threats with certain properties.

1 Introduction

In (McAllester & Rosenblitt 1991), the authors present a simple elegant algorithm for systematic partial-order planning (SNLP). Much recent planning work (Barrett & Weld 1993, Collins & Prior 1992, Kambhampati 1992, Penberthy & Weld 1992, Peot & Smith 1992) has been based upon this algorithm.

In the SNLP algorithm, when threats arise between steps and causal links in a partial plan, those threats are resolved before attempting to satisfy any remaining open conditions in the partial plan. In (Peot & Smith 1993) we investigate several other strategies for resolving threats. Although some of these strategies work much better than the SNLP strategy, they are all fixed, dumb strategies. In practice, we know that some threats that occur during planning are easy to resolve, while others are difficult to resolve. What we would like is a smarter threat-selection strategy that can recognize and delay resolution of the easy threats in order to concentrate effort on the difficult ones.

In this paper, we present techniques for automatically deciding whether threats should be resolved during partial-order planning, or delayed until planning is otherwise complete. In particular, we show that certain threats can be provably delayed until the end; that is, if the planner can find a plan for the goal while ignoring these threats, there is a guarantee that the partial ordering in the resulting plan can be extended to eliminate the threats.

In Section 2, we construct *operator graph*s that capture the interaction between operators relevant to a goal and set of initial conditions. In Section 3, we develop theorems and decomposition rules that use the operator graph to decide when threats can be postponed. In Section 4 we discuss our experience with these techniques and related work.

For purposes of this paper, we adopt a simple STRIPS model of action, and assume the SNLP model of planning. Many of the results and ideas can be applied to other causal-link planners such as (Kambhampati 1993, Tate 1977). Full proofs of the theorems appear in (Smith & Peot 1993).

2 Operator graphs

Following (McAllester & Rosenblitt 1991), we define special `Start` and `Finish` operators for a problem:

Definition 1: The `Start` operator for a problem is defined as the operator having no preconditions, and having all of the initial conditions as effects. The `Finish` operator for a problem is defined as the operator having no effects, and having all of the goal clauses as preconditions.

Given these operators we construct an operator graph for a problem recursively, according to the following rules:

Definition 2: An *operator graph* for a problem is a directed bipartite graph consisting of *precondition nodes* and *operator nodes* such that:

1. There is an operator node for the `Finish` operator.
2. If an operator node is in the graph, there is a precondition node in the graph for each precondition of the operator and a directed edge from the precondition node to the operator node.
3. If a precondition node is in the graph, there is an operator node in the graph for every operator with an effect that unifies with the precondition and there is a directed edge from the operator node to the precondition node.

To illustrate, consider the simple set of operators below (relations, operator names, and constants are capitalized, variables are lower case):

Shape(x)		
	Prec's:	Object(x), ¬ Fastened(x, z)
	Effects:	Shaped(x), ¬ Drilled(x)
Drill(x)		
	Prec's:	Object(x), ¬ Fastened(x, z)
	Effects:	Drilled(x)
Bolt(x, y)		
	Prec's:	Drilled(x), Drilled(y)
	Effects:	Fastened(x, y)
Glue(x, y)		
	Prec's:	Object(x), ¬ Fastened(x, z), Object(y), ¬ Fastened(y, z)
	Effects:	Fastened(x, y)

Suppose that the goal is

$$\text{Shaped}(x) \wedge \text{Shaped}(y) \wedge \text{Fastened}(x, y),$$

and the initial conditions are:

$$\begin{array}{ll} \text{Object}(A) & \neg\text{Fastened}(A, z) \\ \text{Object}(B) & \neg\text{Fastened}(B, z) \\ \ldots & \ldots \end{array}$$

The operator graph for this problem is shown in Figure 1. Note that each operator appears at most once in the graph; but a clause such as Object(x) may appear more than once, if it appears more than once as a precondition. Note that the graph can also contain cycles. If the Bolt operator had an effect Drilled(z) there would be directed edges from Bolt(x, y) to both Drilled(x) and Drilled(y), forming loops with the directed edges from Drilled(x) and Drilled(y) to Bolt(x,y). In this paper we will only consider acyclic operator graphs. The basic results and techniques also apply to cyclic graphs, but the definitions

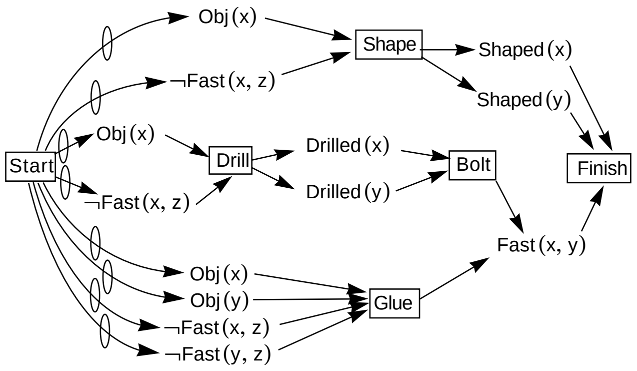

Figure 1: Operator graph for simple machine shop problem. Fastened and Object have been abbreviated for clarity. Circled arcs represent a bundle of arcs.

and theorems are more complicated. The full theory is given in (Smith & Peot 1993).

The operator graph tells us what operators are relevant to the goal, and also tells us something about the topology of partial plans that will be generated for the problem. In this example, it tells us that if the Bolt operation appears in the plan, at least one Drill operation will precede it in the plan. The operator graph has an implicit *and/or* structure to it. Predecessors of an operator node are *ands*, since all preconditions of the operator must be achieved. Predecessors of a precondition node are *ors*, since they correspond to different possible operators that could be used to achieve the precondition.

2.1 Use Count

In the example above, the Glue and Bolt operators are used for only one purpose, while the Drill and Shape operators are used for more than one purpose. This information is important for our analysis of operator threats. We therefore introduce the following notion:

Definition 3: The *use count* of a node in the graph is defined as the number of directed paths from the node to Finish.

The use count of an operator is an upper bound on the number of times the operator could appear in a partial plan. It can be infinite for graphs with cycles.

2.2 Threats

So far, operator graphs only tell us which subgoals and operators may be useful for a problem.

Definition 4: Let O be an operator node, and P be a precondition node in an operator graph. If some effect of O unifies with the negation of P we say that O threatens P and denote this by $O \otimes P$.

The threats for our example are shown in Figure 2.

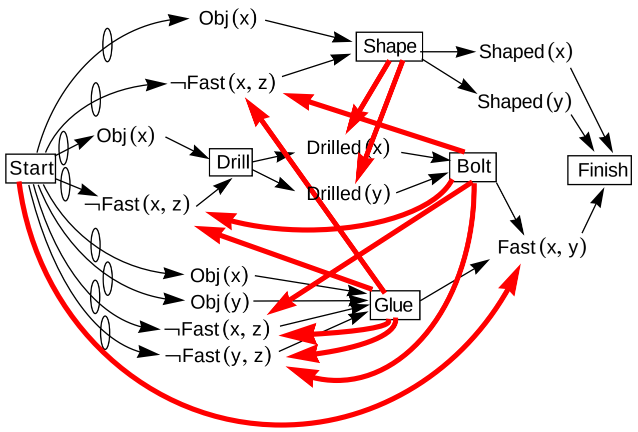

Figure 2: Operator graph with threats (heavy lines).

2.3 Eliminating Threats

Not all threats in the operator graph are important. Some of them will never actually occur during planning. In particular, consider the threat from Start to Fastened(x,y). The initial conditions operator always precedes all other operators in a plan. As a result, there is no possibility that Start can ever clobber an effect produced by another operator. Therefore we can eliminate these threats from the graph.

Theorem 1: Threats emanating from Start can be eliminated.

A related class of threats are those between operators and preconditions that are successors of each other. Consider the threat from Glue(x,y) to its precondition ¬Fastened(x,z). This says that gluing clobbers its precondition. This is not a problem since the clobbering follows the consumption of the precondition. As a result, this threat can be eliminated. Similar arguments can be made for the threat from Bolt(x,y) to the ¬Fastened(x,z) precondition of Drill(x) .

Note that our arguments rely on the fact that Glue and Bolt will only appear once in the final plan. If Glue(x,y) appeared more than once, there is a distinct possibility that one gluing operation might clobber the precondition of another gluing operation. As a result, we can only eliminate such threats when the use count of the operator is 1.

Theorem 2: Threats from an operator to any predecessor or successor in the graph can be eliminated if the threatening operator has use count 1.

A third source of superfluous threats are *disjunctive* branches in the operator graph. In our example, there are two different ways of achieving the subgoal Fastened(x,y): bolting and gluing. Only one of these two alternatives will appear in any given plan. As a result, we can ignore threats that go from one branch to the other. This means that the edges from Bolt(x,y) to the ¬Fastened preconditions of Glue(x,y), and from Glue(x,y) to the ¬Fastened(x,z) precondition of Drill(x) can be eliminated.

As with Theorem 2, we need to consider use count. Suppose that there was a second subgoal of the form Fastened(x,y). The planner might choose Bolt(x,y) for one of these subgoals, and Glue(x,y) for the other. In this case, a threat between Bolt(x,y) and the ¬Fastened(x,y) precondition of Glue(x,y) could occur.

Theorem 3: If a threat is between two disjunctive branches in the operator graph, and the threatening operator has use count 1, the threat can be eliminated.

To decide if a threat is between two disjunctive branches we need to look at the nearest common ancestor of the threatening operator and precondition. For the threat between Bolt(x,y) and the ¬Fastened(x,y) precondition of Glue(x,y), the nearest common ancestor is Fastened(x,y). Since this is a precondition node, the threat is between disjunctive branches.

After applying Theorems 1, 2, and 3, there are only four remaining threats, as shown in Figure 3. These are the only possible threats that can actually arise during planning.

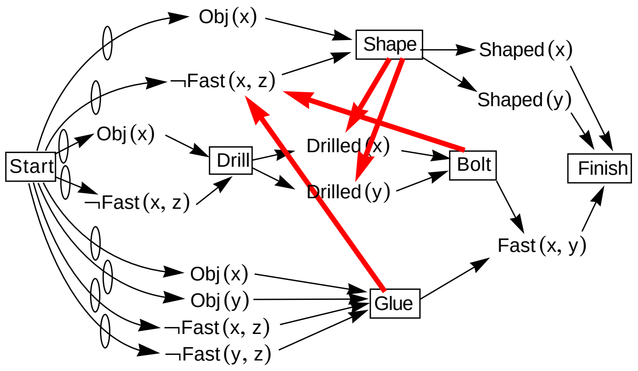

Figure 3: Threats remaining after the elimination theorems have been applied.

3 Postponing Threats

In Figure 3, consider the threats Shape(x) ⊗ Drilled(x) and Shape(x) ⊗ Drilled(y). These threats tell us that allowing Shape operations to occur between Drill and Bolt operations may cause problems. However, in considering the graph, we can see that there is an easy solution. If we add the ordering constraint that Shape operations must occur before Drill operations, both threats are eliminated.

We could have the planner automatically add these constraints every time Shape or Drill operations were added to a partial plan. Although this strategy would work, it is more restrictive than necessary. In our example, if two different objects, A and B are used for x and y, there would be two different Shape operations, and two different Drill operations in the final plan. To get rid of the threats it would only be necessary that Shape(A) precede Drill(A) and Shape(B) precede Drill(B). The other potential threats go away by virtue of the different variable bindings for x and y.

To avoid this over-commitment, it is better to *postpone* the threats between Shape(x) and Drilled(x). No matter what plan is generated, we can always eliminate this threat *later* by imposing the necessary ordering constraints between Shape and Drill operations.

The argument made above relies on two things:

1. There are ordering constraints that will resolve the threats,
2. The other potential threats do not interfere with these ordering constraints.

The first part of this argument is straightforward; in our case, demoting Shape before Drill did the trick, since it prevents Shape from occurring between Drill and Bolt. The second part of the argument is tougher. It requires showing that none of the possible resolutions of the remaining threats will prevent the ordering of Shape operations before Drill operations. To show this, we need to consider all possible ways that the planner might choose to resolve the remaining set of threats.

First consider the threat $\text{Bolt}(x, y) \otimes \neg\text{Fastened}(x, y)$. Since Bolt cannot come before Start, demotion is not possible for this threat. However, promotion is possible, since $\text{Shape} < \text{Bolt}$ is consistent with the operator graph. We therefore need to consider the possibility that this constraint might be added to the operator graph. (Separation is not possible in this case. Even if it were, separation adds no ordering constraints to the graph, and therefore does not concern us.)

Next consider the threat $\text{Glue}(x, y) \otimes \neg\text{Fastened}(x, y)$. As before, demotion is not possible since Glue cannot come before Start. However, promotion is possible, since $\text{Shape} < \text{Glue}$ is consistent with both the operator graph, and with the constraint $\text{Shape} < \text{Bolt}$.

Since the addition of $\text{Shape} < \text{Glue}$ and $\text{Shape} < \text{Bolt}$ do not interfere with $\text{Shape} < \text{Drill}$, condition (2) is also satisfied.

The general form of this argument is summarized in the following theorem:

Theorem 4: Let T be the set of threats in an operator graph, and let P be a subset of those threats. The threats in P can be postponed if there is a set of ordering constraints that resolves the threats in P for every possible resolution of the remaining threats in $T - P$.

Proof Sketch: Suppose that SNLP ignores all threats corresponding to the threats P in the operator graph. Consider a final plan F produced by SNLP. Let R be the set of threats that were resolved in the construction of F. The threats in R are instances of the threats T-P in the operator graph. Thus there is some resolution of threats in T-P that corresponds to the resolution of R in the plan. By our hypothesis, there is some set of ordering constraints in the operator graph that resolves the threats in P, for each possible resolution of T-P. These ordering constraints will therefore resolve any instances of P ignored during the construction of F. As a result, there is an extension of the partial ordering of F that will resolve all of the postponed threats. Since F was an arbitrary plan, the theorem follows.

Corollary 5: Let T be the set of threats in an operator graph. The (entire) set of threats T can be postponed if there is a set of ordering constraints that resolves the threats in T.

In the machine shop example, we could use Corollary 5 to postpone the entire set of threats at once, since the three ordering constraints $\text{Shape} < \text{Drill}$, $\text{Shape} < \text{Glue}$, and $\text{Shape} < \text{Bolt}$ resolve all four of the threats. In general, however, this corollary cannot be applied as frequently as Theorem 4. The reason is that this corollary requires the resolution of all threats by ordering constraints. There may be some threats that can only be resolved by separation during planning or that cannot be resolved. In these cases Corollary 5 cannot be applied, but Theorem 4 may still allow us to postpone some subset of the threats. As an example, consider the operator graph shown in Figure 4.

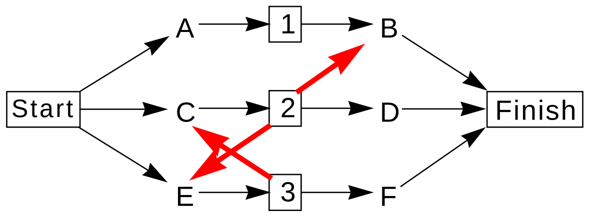

Figure 4: Operator graph where only two of the three threats can be postponed.

In this example, the two threats $2 \otimes E$ and $3 \otimes C$ can only be resolved by separation. However, the threat $2 \otimes B$ can always be resolved by imposing the ordering constraint $2 < 1$ (demotion). Since demotion is consistent with the only possible resolution of the remaining threats, the threat $2 \otimes B$ can be postponed according to Theorem 4.

3.1 Over-constraining

The primary difficulty with applying Theorem 4 and Corollary 5 is that they both take time that is exponential in the number of threats being considered. In fact it can be shown that:

Theorem 6: Given a partial ordering and a set of threats, it is NP-complete to determine whether there exists an extension to the partial ordering that will resolve the threats.

The proof of this theorem (Kautz 1992) involves a reduction of 3-SAT clauses to a partial ordering and set of threats. The complete proof can be found in (Smith & Peot 1993).

Although the general problem of postponing threats is computationally hard, there are some special cases that are more tractable. The first technique that we consider involves over-constraining the operator graph. In particular, we simultaneously impose both promotion and demotion ordering constraints on the operator graph for all threats in the graph but one. We then check to see if there is an ordering constraint on the remaining threat (demotion or promotion) that is still possible in the over-constrained graph. If so, we know that the ordering constraint will work for all possible resolutions of the remaining threats, and we can therefore postpone the threat. More precisely:

Theorem 7: Let t be the threat $O_t \otimes \left(O_p \xrightarrow{g} O_c\right)$. For each remaining threat $r_t \otimes \left(r_p \xrightarrow{g} r_c\right)$ in the operator graph augment the operator graph with the edges $r_t \rightarrow r_p$ and $r_c \rightarrow r_t$. The threat t can be postponed if either:

1. $O_t < O_p$ is consistent with the operator graph, and $O_p \notin \mathrm{Predecessors}(O_t)$ in the augmented operator graph.
2. $O_c < O_t$ is consistent with the operator graph, and $O_c \notin \mathrm{Successors}(O_t)$ in the augmented operator graph.

Proof Sketch: Let A be the set of augmentation edges added to the graph. Every consistent way of resolving the set of remaining threats corresponds to some subset of these constraints. Suppose that case 1 holds for the above theorem. Since $O_p \notin \mathrm{Predecessors}(O_t)$ in the augmented graph, we know that it will also hold for every subset of the augmentation edges. As a result, we know that this condition holds for every possible way of resolving the remaining threats in the graph. Furthermore, in a consistent graph, $O_p \notin \mathrm{Predecessors}(O_t)$ implies that t can be resolved by demotion. As a result, Theorem 4 says that t can be postponed. The argument for case 2 is analogous.

To see how this theorem applies, consider the threat $\mathrm{Glue}(x, y) \otimes \neg\mathrm{Fastened}(x, z)$ in Figure 3. To see if this threat can be postponed, we need to augment the operator graph with all ordering constraints that resolve the remaining three threats. For the threat $\mathrm{Bolt}(x, y) \otimes \neg\mathrm{Fastened}(x, z)$ we need to add the edge $\mathrm{Shape}(x) \rightarrow \mathrm{Bolt}(x, y)$ to the graph, since it is the only way of resolving the threat. For the other two threats, $\mathrm{Shape}(x) \otimes \mathrm{Drilled}(x)$ and $\mathrm{Shape}(x) \otimes \mathrm{Drilled}(y)$, we need to add the two edges $\mathrm{Shape}(x) \rightarrow \mathrm{Drill}(x)$ and $\mathrm{Bolt}(x, y) \rightarrow \mathrm{Shape}(x)$. The resulting graph is shown in Figure 5.

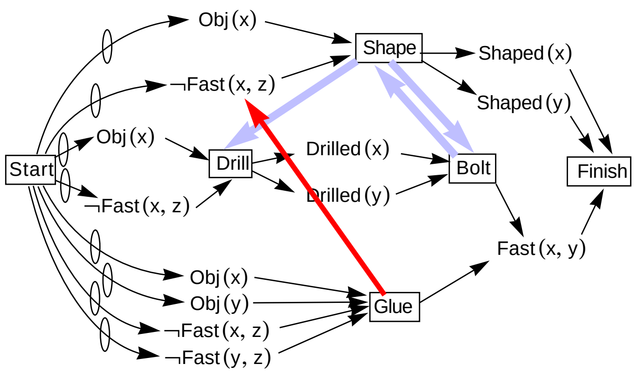

Figure 5: Machine shop example with over-constrained threats. Partial-ordering constraints are shown as grey arrows.

Now consider the two possibilities for resolving $\mathrm{Glue}(x, y) \otimes \neg\mathrm{Fastened}(x, z)$. Glue cannot be ordered before Start in the operator graph, so case 1 is out. Glue can be ordered after Shape, however, so we need to consider case 2. In the augmented graph, the only successor of Glue is Finish. Since Shape is not in this set, the second condition is satisfied. Therefore we can postpone the threat $\mathrm{Glue}(x, y) \otimes \neg\mathrm{Fastened}(x, z)$.

Theorem 7 can also be used to show that each of the remaining threats in the machine shop problem can be postponed. More generally, Theorem 7 can be applied in a serial fashion: after one threat is postponed, it does not need to be considered in the analysis of subsequent threats.

3.2 Threat Blocks

Although Theorem 7 is considerably weaker than Theorem 4, it can be applied in time that is linear in the size of the operator graph. As a result, it can often be used to quickly eliminate many of the easiest threats from consideration. Unfortunately, there are some sets of threats where the full power of Theorem 4 is still needed. Consider the graph shown in Figure 6. The four threats shown in the top half of

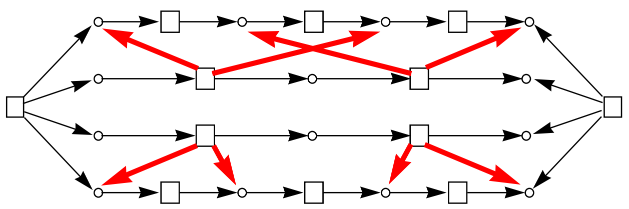

Figure 6: Threat graph with difficult threats.

the graph can be resolved and postponed using Theorem 4 but not Theorem 7. The set of threats in the bottom half of the graph cannot be resolved using only ordering constraints, and therefore cannot be postponed. In a case like this, the top and bottom halves of the graph are independent, and we should be able to examine the threats in the two halves separately.

To do this we first need some definitions. We define a block as a subset of an operator graph having a common beginning and ending. More precisely:

Definition 5: Let Begin and End be two operators such that Begin is a predecessor of End in the operator graph. A *block* is a subset of the operator graph (ignoring threats), such that, for each node N in the block:

1. Begin occurs on all paths from Start to N.
2. End occurs on all paths from N to Finish.
3. Every node and edge on a path from Begin to N is in the block.
4. Every node and edge on a path from N to End is in the block.

In the graph above, each of the four branches constitutes a block. Any two or three of these branches also constitute a block.

Definition 6: A *threat block* is a block where all threats that touch any node in the block are contained completely within the block. A threat block is *minimal* if no subset of the block is a threat block.

According to this definition, there are two minimal threat blocks in the above graph, one containing the top two and one containing the bottom two branches.

Theorems 4 and 7 can now be extended to threat blocks. We restate Theorem 4 for threat blocks.

Theorem 8: Let T be the set of threats in a minimal threat block and let P be a subset of those threats. The threats in P can be postponed if there is a set of ordering constraints that resolves the threats in P for every possible resolution of the remaining threats in $T - P$.

Proof Sketch: Consider the set of threats not in the threat block. If we consider every possible way of resolving these outside threats it is easy to see that the resulting ordering constraints can have no impact on any ordering decisions within the block. Thus, if the conditions of Theorem 8 hold, we can expand the set T to include the threats outside the block and Theorem 4 will apply.

Using this theorem, we could examine and postpone the threats in the top half of the graph of Figure 6.

It is relatively easy to find minimal threat blocks. We start with one threat, and find the common descendents and ancestors of both ends of the threat. If other threats are encountered in the process, we include the endpoints of these new threats in our search for a common ancestor and descendent. With pre-numbering of the graph, this process can be done in time linear in the size of the graph.

4 Discussion

4.1 Implementation

In our current implementation, we first attempt to eliminate as many individual threats as possible using Theorem 7. After this, we construct minimal threat blocks for the remaining threats. We then use Corollary 5 on each individual threat block. We have not yet implemented the more powerful Theorems 4 or 8, but expect to apply them only after other more tractable alternatives have failed.

Our preliminary testing indicates several things:

1. The number of threats that can be postponed varies widely across problems and domains. As we would expect, many more threats can be postponed in domains with loosely-coupled operators. The techniques do little to help highly recursive sets of operators.
2. The time taken to build an operator graph and analyze threats is computationally insignificant in comparison to the time required to do planning. For non-trivial planning problems, this time is less than 10% of planning time, and is often much smaller than that.

Our experience suggests that the speed of these procedures is not a concern and that even the use of Theorem 8 on threat blocks will probably not cause serious computational problems. We speculate that if the threats in a block are so numerous and tangled that the speed of these techniques is a problem, the planner is in deep trouble anyway.

4.2 Related Work

Both (Etzioni 1993) and (Knoblock 1990, 1991) have proposed goal ordering mechanisms to reduce the number of threats that arise during planning. In particular, Etzioni and Knoblock construct and analyze graphs similar to the operator graphs developed here. Etzioni derives goal-ordering rules from this graph, while Knoblock constructs predicate hierarchies to guide a hierarchical planner. Unfortunately, both of these systems were developed for a total-order planner. In a total-order planner the order in which goals are processed affects the ordering of actions in the plan. This, in turn, determines the presence or absence of threats in the plan.

In contrast, for partial-order planning, the order in which goals are processed does not determine the ordering of actions within the plan. As a consequence, goal ordering does not affect the presence or absence of threats in the plan, and cannot be used to help reduce threats. Although goal ordering can be used to reduce search in partial-order planning (Smith 1988, Smith & Peot 1992), it cannot be used to reduce the number of threats. A more detailed critique of Knoblock's technique can be found in (Smith & Peot 1992).

4.3 Extensions

Originally, we thought it was possible to use local analysis techniques to postpone many threats. However, all of our conjectures in this area have proven false. The one area that we think still holds promise is division into threat blocks. We think that there may be criteria that will allow threats to be broken up into smaller blocks.

Another approach that we think holds promise is variable analysis in the operator graph (Etzioni 1993). By a careful analysis of variable bindings in the operator graph, it is often possible to eliminate many phantom threats from the graph. This, in turn, makes it more likely that other threats can be postponed.

There are other possibilities for analysis of the operator graph, including analysis of potential loops. Here, the recognition and elimination of unnecessary loops among the operators can allow the postponement of additional threats. Some of these possibilities are discussed in (Smith & Peot 1993).

4.4 Final Remarks

The techniques developed in this paper have a direct impact on the efficiency of the planning process. Whenever possible, they separate the tasks of selecting actions from the task of ordering or scheduling those actions. This is a natural extension of the least-commitment strategy inherent to partial-order planning.

But perhaps as important as threat postponement is the ability to recognize threats that are difficult to resolve. If a block of threats cannot be postponed, the planner should pay attention to those threats early. This information could be used to help the planner avoid partial plans with difficult threat blocks. It could also be used to help determine the order in which to work on open conditions. In particular, if the planner is faced with a difficult threat block it should probably generate and resolve that portion of the plan early. In our experience, both the choice of partial plan and the choice of open condition can dramatically influence the performance of a planner. For this reason, information about difficult threat blocks could make a significant difference.

Acknowledgments

The idea of analyzing threats in the operator graph was motivated by the work of Craig Knoblock (Knoblock 1990, 1991). Thanks to Mark Drummond, Steve Minton, Craig Knoblock, and Oren Etzioni for comments and discussion. Thanks to Henry Kautz and David McAllester for help with the NP-completeness result. This work is supported by DARPA contract F30602-91-C-0031.

References

Barrett, A. and Weld, D. 1993. Partial-Order Planning: Evaluating Possible Efficiency Gains, Technical Report 92-05-01, Dept. of Computer Science, University of Washington.

Collins, G., and Pryor, L. 1992. Representation and Performance in a Partial Order Planner, Technical Report 35, The Institute for the Learning Sciences, Northwestern University.

Etzioni, O. 1993. Acquiring Search-Control Knowledge via Static Analysis, *Artificial Intelligence*, to appear.

Harvey, W. 1993. Deferring Threat Resolution Retains Systematicity, Technical Note, Department of Computer Science, Stanford University.

Kambhampati, S. 1992. Characterizing Multi-Contributor Causal Structures for Planning, *In Proceedings of the First International Conference on AI Planning Systems*, College Park, Maryland, 116-125.

Kambhampati, S. 1993. On the Utility of Systematicity: Understanding Tradeoffs between Redundancy and Commitment in Partial-Ordering Planning, *In Proceedings of the Thirteenth International Conference on AI*, Chambéry, France.

Kautz, H. 1992, personal communication.

Knoblock, C. 1990, Learning abstraction hierarchies for problem solving. *In Proceedings of the Eight National Conference on AI*, Boston, MA, 923–928.

Knoblock, C. 1991. Automatically Generating Abstractions for Problem Solving, Technical Report CMU-CS-91-120, Dept. of Computer Science, Carnegie Mellon University.

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

Penberthy, J. S., and Weld, D. 1992. UCPOP: A Sound, Complete, Partial Order Planner for ADL, *In Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, Cambridge, MA.

Peot, M., and Smith, D. 1992. Conditional Nonlinear Planning, *In Proceedings of the First International Conference on AI Planning Systems*, College Park, MD, 189-197.

Peot, M., and Smith, D. 1993. Threat-removal Strategies for Partial-Order Planning, *In Proceedings of the Eleventh National Conference on AI*, Washington, D.C.

Smith, D. 1988. A Decision Theoretic Approach to the Control of Planning Search., Technical Report 87-11, Stanford Logic Group, Department of Computer Science, Stanford University.

Smith, D., and Peot, M. 1992. A Critical Look at Knoblock's Hierarchy Mechanism, *In Proceedings of the First International Conference on AI Planning Systems*, College Park, Maryland, 307-308.

Smith, D., and Peot, M. 1993. Threat Analysis in Partial-Order Planning. Forthcoming.

Tate, A. 1977. Generating Project Networks, *In Proceedings of the Fifth International Joint Conference on AI*, Boston, MA, 888-893.

Yang, Q., A Theory of Conflict Resolution in Planning, *Artificial Intelligence*, 58:361–392, 1992.