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

Threat-Removal Strategies for Partial-Order Planning

Mark A. Peot

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

David E. Smith

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

Abstract

McAllester and Rosenblitts' (1991) systematic nonlinear planner (SNLP) removes threats as they are discovered. In other planners such as SIPE (Wilkins, 1988), and NOAH (Sacerdoti, 1977), threat resolution is partially or completely delayed. In this paper, we demonstrate that planner efficiency may be vastly improved by the use of alternatives to these threat removal strategies. We discuss five threat removal strategies and prove that two of these strategies dominate the other three--resulting in a provably smaller search space. Furthermore, the systematicity of the planning algorithm is preserved for each of the threat removal strategies. Finally, we confirm our results experimentally using a large number of planning examples including examples from the literature.

1 Introduction

McAllester and Rosenblitt (1991) present a simple elegant algorithm for systematic nonlinear planning (SNLP). Much recent planning work (Barrett & Weld, 1993; Collins & Pryor, 1992; Harvey, 1993; Kambhampati, 1993a; Penberthy & Weld, 1992; Peot & Smith, 1992) has been based upon this algorithm (or the Barrett & Weld (1993) implementation of it).

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. From a practical standpoint, we know that this is not always the most efficient course. When there are only a few loosely coupled threats in a problem, it is generally more efficient to delay resolving those threats until the end of the planning process.[1] However, if there are many tightly-coupled threats (causing most partial plans to fail), those threats should be resolved early in the planning process to avoid extensive backtracking. These two options, resolve threats immediately, and resolve threats at the end, represent two extreme positions. There are several other options, such as waiting to resolve a threat until it is no longer separable, or waiting until there is only one way of resolving the threat. Given a reasonable mix of problems, what is the best strategy or strategies?

In this paper, we introduce four alternative threat removal strategies and show that some are strictly better than others. In particular, we show that delaying separable threats generates a smaller search space of possible plans than the SNLP algorithm.

In Section 2 we give preliminary definitions and a version of the SNLP algorithm. In Section 3 we introduce four different threat removal strategies and investigate the theoretical relationships between them. In Section 4 we give empirical results that confirm the analysis of Section 3. In Section 5, we discuss work related to the work in this paper.

2 Preliminaries

Following (Kambhampati, 1992), (Barrett & Weld, 1993), (Collins & Pryor, 1992) and (McAllester & Rosenblitt, 1991), we define causal links, threats and plans as follows:

Definition 1: An open condition, g, is a precondition of an operator in the plan that has no corresponding causal link.

Definition 2: A causal link, $L\colon s_e \xrightarrow{g} s_c$, protects effect g of the establishing plan step, s_e, so that it can be used to satisfy a precondition of the consuming plan step, s_c.

Definition 3: An ordering constraint $O\colon\ s_1 > s_2$ restricts step s_1 to occur after step s_2.

Definition 4: A plan is a tuple $\langle S, L, O, G, B\rangle$ where S denotes the set of steps in the plan, L denotes the set of causal links, O denotes the set of ordering constraints, G denotes the outstanding open conditions, and B denotes the set of equality and inequality constraints on variables contained in the plan.

Definition 5: A threat $T\colon\ s_t \overset{b}{\otimes} (s_e \xrightarrow{g} s_c)$ represents a potential conflict between an effect of a step in the plan, s_t,

[1] Hacker (Sussman, 1973), Noah (Sacerdoti, 1977), and (to a certain extent) Sipe (Wilkins, 1988), delay the resolution of threats until the end of the planning process. Yang (1993) has also explored this strategy.

and a causal link $s_e \xrightarrow{g} s_c$. s_t threatens causal link $s_e \xrightarrow{g} s_c$, if s_t can occur between s_e and s_c and an effect, e, of s_t possibly unifies with either g or $\neg g$ when the additional binding constraints b are added to the plan bindings. Unification of two literals, e and g, under bindings b is denoted by $e \overset{b}{\equiv} g$. If $e \overset{b}{\equiv} g$, we refer to T as a *positive threat*. If $e \overset{b}{\equiv} \neg g$, we refer to T as a *negative threat*.

To make our analysis easier, we will work with the following modified version of the SNLP algorithm given below. The primary difference between this algorithm and the algorithms given in (Barrett, Soderland & Weld, 1991; Collins & Pryor, 1992; and Kambhampati 1993a) is that threat resolution takes place immediately after Add-Link and Add-Step. As a result, the set of partial plans being considered never contains any plans with unresolved threats.

Plan (initial-conditions, goal):

1. **Initialization**: Let Finish be a plan step having preconditions equal to the goal conditions and let Start be a plan step having effects equal to the initial conditions. Let Q be the set consisting of the single partial plan $\langle \{\text{Start}, \text{Finish}\}, \varnothing, \{\text{Start} < \text{Finish}\}, G, \varnothing \rangle$, where G is the set of open conditions corresponding to the goals.
2. **Expansion:** While Q is not empty, select a partial plan $p = \langle S, L, O, G, B \rangle$ and remove it from Q.
   A. **Termination:** If the open conditions G are empty, return a topological sort of S instantiated with the bindings B.
   B. **Establish Open Condition:** Select some $g \in G$ and do the following:
      i. **Add Link:** For each $s \in S$ with an effect e such that $e \overset{b}{\equiv} g$ and s is possibly prior to s_c call
      $$\text{Resolve-Threats}\left(S, L + \left(s \xrightarrow{g} s_c\right), O + (s < s_c), G - g + G_s, B + b\right)$$
      ii. **Add Step:** For each $a \in A$ with an effect e such that $e \overset{b}{\equiv} c$ call
      $$\text{Resolve-Threats}\left(S + s, L + \left(s \xrightarrow{g} s_c\right), O + (s < s_c), G - g + G_s, B + b\right)$$
      where A denotes the set of all action descriptions, s denotes the new plan step constructed by copying a with a fresh set of variables and G_S is the preconditions of S.

Resolve-Threats (S, L, O, G, B):

1. Let T be the set of threats between steps in S and causal links in L.
2. If T is empty add $\langle S, L, O, G, B \rangle$ to Q.
3. If T is not empty select some threat $s_t \overset{b}{\otimes} \left(s_e \xrightarrow{g} s_c\right) \in T$ and do the following:
   A. **Demotion**: If $s_t < s_e$ is consistent with O $\text{Resolve-Threats}(S, L, O + s_t < s_e, G, B + b)$.[2]
   B. **Promotion**: If $s_c < s_t$ is consistent with O $\text{Resolve-Threats}(S, L, O + s_c < s_t, G, B + b)$.
   C. **Separation**: For each $(x_i = y_i) \in b$:
   $$\text{Resolve-Threats}\left(S, L, O, G, B + \{x_k = y_k\}_{k=1}^{i-1} + (x_i \neq y_i)\right)$$
   where $\{x_k = y_k\}_{k=1}^{i-1}$ denotes the set of the first $i-1$ equality constraints in b.[3]

3 Threat Delay Strategies

In the SNLP algorithm, when threats arise in a partial plan, they are immediately resolved before attempting to satisfy any remaining open conditions. There are three ways that a threat can be resolved: separation, promotion, and

[2] The criterion for separation, promotion, and demotion, used in (Barrett & Weld, 1993) and (Collins & Pryor, 1992) are not mutually exclusive. In order to preserve systematicity, one must either restrict separation so that the threatening step occurs between the producer and consumer steps, or restrict the variable bindings for promotion and demotion so that separation is not possible. For our purposes, the latter restriction is simpler.

[3] The addition of equality constraints during separation is required to maintain systematicity [3].

demotion. Separation forces the variable bindings in the clobbering step to be different than those in the threatened causal link. Promotion forces the clobbering step to come before the producing step in the causal link, and demotion forces the clobbering step to come after the consumer of the causal link. If all three of these are possible, there will be at least a three way branch in the search space of partial plans. In fact, it can be worse than this, because there may be many alternative ways of doing separation, and all of them must be considered.

In practice, however, threat removal is often deferred in order to improve planning performance. Harvey (1993) has shown that any threat removal order may be used in the SNLP algorithm without compromising the algorithm's systematicity. In this section, we describe four alternative threat deferral strategies and the effect of these strategies on the size of the planner search space.

3.1 Separable Delay

Many of the threats that occur during planning are ephemeral. As planning continues, variables in both the clobbering step and the causal link may get bound, causing the threat to go away. This causes the promotion and demotion branches for that partial plan to go away, and causes all but one of the separation branches for the plan to go away. Thus, it would seem to make heuristic sense to postpone resolving a threat until the threat becomes definite; that is, until the bindings of the clobbering step and the causal link are such that the threat is guaranteed to occur. Thus we could modify the SNLP algorithm in the following way:

Resolve-Threats (S, L, O, G, B):

1. Let $T = \{s \overset{\varnothing}{\otimes} l\}$ be the set of unseparable threats between steps $s \in S$ and causal links $l \in L$. These threats are those that are guaranteed to occur regardless of the addition of additional binding constraints.
2. If T is empty add $\langle S, L, O, G, B\rangle$ to Q.
3. If T is not empty select some threat, $s_t \overset{\varnothing}{\otimes} \left(s_e \overset{g}{\to} s_c\right) \in T$, and do the following:
   A. **Demotion**: If $s_t < s_e$ is consistent with O $\text{Resolve-Threats}(S, L, O + s_t < s_e, G, B)$
   B. **Promotion**: If $s_c < s_t$ is consistent with O $\text{Resolve-Threats}(S, L, O + s_c < s_t, G, B)$.

We refer to this threat resolution strategy as DSep. Note that DSep is a complete, systematic planning algorithm that does not require the use of separation.

Theorem 1: The space of partial plans generated by DSep is no larger than (and often smaller than) the space of partial plans generated by SNLP. (We are assuming that the two algorithms use the same strategy to decide which open conditions to work on.)

Sketch of Proof: The essence of the proof is to show that each partial plan generated by DSep has a unique corresponding partial plan generated by SNLP. Let $p = \langle S, L, O, G, B\rangle$ be a partial plan generated by DSep and let $Z = z_1, \ldots, z_n$ be the sequence of planner operations (add-link, add-step, demote, and promote) used by DSep to construct p. For each threat t introduced by an operation z_t in Z, there are three possibilities:

1. t became unseparable and was resolved using a later Promote or Demote operation z_r.
2. t is still separable and hence is unresolved in p
3. t was separable, but eventually disappeared because of binding or ordering constraints introduced by a later operation z_r.

For each threat t, we perform the corresponding modification to the sequence Z indicated below:

1. move z_r to immediately after z_t.
2. add a Separate operation for t immediately after z_t
3. add the appropriate Promote, Demote, or Separate operation immediately after z_t that mimics the way in which the threat is eventually resolved.

In this new sequence Z', all threats are resolved immediately after they are introduced. As a result, Z' is now the sequence of planning operations that would have been generated by SNLP. The plan $p' = \langle S, L, O', G, B'\rangle$ generated by this sequence differs from the original plan only in that 1) B' is augmented with separation constraints for each unresolved threat in p, and 2) O and O' may differ in redundant ordering constraints, but $\text{Closure}(O) = \text{Closure}(O')$. As a result, the mapping from DSep plans to SNLP plans is one to one, and the theorem follows. n

Although the space of partial plans generated by DSep is smaller than that generated by SNLP, we cannot guarantee that DSep will always be faster than SNLP. There are two reasons for this:

1. There is overhead associated with delaying threats because the planner must continue to check separability.
2. When a threat becomes unseparable, DSep must check to see if demotion or promotion are possible. Because the space of partial plans may have grown considerably since the threat was introduced, there might be more of these checks than if resolution had taken place at the time the threat was introduced. (Of course, the reverse can also happen.)

3.2 Delay Unforced Threats

A natural extension of the DSep idea would be to delay resolving a threat until there is only one (or no) threat resolution option remaining. (This is the ultimate least-commitment strategy with regard to threats.) Thus, if demotion were the only possibility for resolving a threat, the appropriate ordering constraint would be added. Alternatively, if separation were the only option, and there was only one way of separating the variables, the appropriate not-equals constraint would be added to the plan. We refer to this threat resolution strategy as DUnf. The threat resolution procedure for DUnf is shown below:

Resolve-Threats (S, L, O, G, B):

1. Let $T = s_t \overset{b}{\otimes} \left(s_e \xrightarrow{g} s_c\right)$ be the set of threats between steps in S and causal links in L such that either:
   - A. $s_t < s_e$ is consistent with O, $s_t \le s_c$, and $b = \varnothing$
   - B. $s_c < s_t$ is consistent with O, $s_t \ge s_e$, and $b = \varnothing$.
   - C. $s_e \le s_t \le s_c$ and b contains exactly one constraint.
2. If T is empty add $\langle S, L, O, G, B\rangle$ to Q.
3. If T is not empty select some threat $s_t \overset{b}{\otimes} \left(s_e \xrightarrow{g} s_c\right) \in T$ and do the following:
   - A. **Demotion**: If $s_t < s_e$ is consistent with O $\text{Resolve-Threats}(S, L, O + s_t < s_e, G, B + b)$
   - B. **Promotion**: If $s_c < s_t$ is consistent with O $\text{Resolve-Threats}(S, L, O + s_c < s_t, G, B + b)$.
   - C. **Separation**: For the single binding constraint, $(x = y)$, $\text{Resolve-Threats}(S, L, O, G, B + (x \ne y))$.

It might seem that this strategy would always expand fewer partial plans than DSep. Unfortunately this is not the case for at least two reasons. First of all, it is possible to construct examples where both promotion and demotion are possible for each individual threat, but the entire set of threats is unsatisfiable (a direct conclusion from the fact that the problem of determining whether a set of threats might be resolved by the addition of ordering constraints is NP-Complete (Kautz, 1993)). For such a case, the DUnf planner would choose not to work on the threats and therefore wouldn't recognize that the plan was impossible. In contrast, SNLP and DSep would commit to either promotion or demotion for each threat, and would therefore discover that the plan was impossible earlier than would the DUnf strategy. In addition, when the DUnf strategy postpones the addition of ordering constraints to a plan, it allows plan branches to be developed that contain Add-links that might have been illegal if those ordering constraints were added earlier in the planning process.

On the other hand, it is also easy to construct domains where DSep is clearly inferior to DUnf. Therefore we can conclude:

Theorem 2: Neither DUnf nor DSep are guaranteed to generate a smaller search space than the other for all planning problems.

Another possible drawback to DUnf is that checking to see if there is only one option for resolving a threat may be costly, whereas the separability criterion used in DSep is relatively easy to check.

3.3 Delay Resolvable Threats

An alternative to DUnf would be to ignore a threat until it becomes impossible to resolve, and then simply discard the partial plan. We refer to this alternative as DRes:

Resolve-Threats (S, L, O, G, B):

1. Let T be the set of threats between steps in S and causal links in L.
2. For all $s_t \overset{b}{\otimes} \left(s_e \xrightarrow{g} s_c\right) \in T$ if either b is nonempty, $s_t < s_e$ is consistent with O, or $s_c < s_t$ is consistent with O, then add $\langle S, L, O, G, B\rangle$ to Q.

To see the difference between DUnf and DRes, consider a partial plan with a threat that can only be resolved by demotion. Using DUnf, we would generate a new partial plan with the appropriate ordering constraint. This ordering constraint could, in turn, prevent any number of possible add-link operations, and could reduce the possible ways of resolving other threats. If DRes were used, this additional ordering constraint would not be present. As a result, DUnf will consider fewer partial plans than DRes.

It is relatively easy to show that:

Theorem 3: The space of partial plans generated by DRes is at least as large as the space generated by DUnf.

Since the cost of checking to see if a threat is unresolvable is just as expensive as checking to see if a threat has only one resolution, there should be no advantage to DRes over DUnf.

3.4 Delay Threats to the End

The final extreme approach is to delay resolving threats until all open conditions have been satisfied. We refer to this algorithm as DEnd. Threat resolution strategies similar to DEnd are used in (Sussman, 1973; Sacerdoti, 1977; Wilkins, 1988; and Yang 1993).

The primary advantage to this approach is that there is no cost associated with checking or even generating threats until the plan is otherwise complete. In problems where there are few threats, or the threats are easy to resolve by ordering constraints, this approach is a win. However, if most partial plans fail because of unresolvable threats, this technique will generate many partial plans that are effectively dead.

It is relatively easy to show that:

Theorem 4: The space of partial plans generated by DRes is no larger than (and is sometimes smaller than) the space generated by DEnd.

The search space relationships between the five threat removal strategies is summarized in Figure 1.

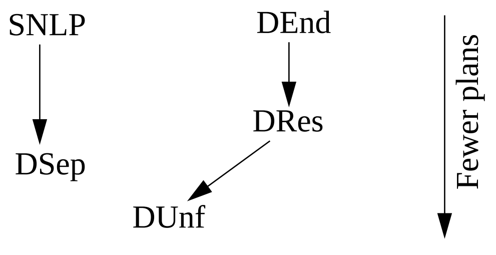

Figure 1: Search space relationships for five threat removal strategies.

4 Empirical Results

We tested the five threat resolution strategies on several problems in each of several different domains; a discrete time version of Minton's machine shop scheduling domain (1988), a route planning domain, Russell's tire changing domain, and Barrett & Weld's (1993) artificial domains D^mS^1 and D^1S^1 (also called the ART-MD and ART-1D domains), and ART-1D-RD and ART-MD-RD, Kambhampati's (1993a) variations on these domains.

Ordinarily, the performance of a planner depends heavily on first, the order in which the partial plans are selected, and second, the order in which open conditions are selected. In order to try to filter out these effects, we tested each domain using several different strategies.

We used the A* search algorithm in our testing. The 'g' function is the length of the partial plan, and 'h' is the number of open conditions. We did not include the number of threats in 'h' because this number varies across different threat resolution strategies. The search algorithm is engineered so that it always searches partial plans with equivalent causal structures (generated by the same chain of add-link and add-step operations) in the same order regardless of the threat resolution strategy selected. These tests demonstrate the relative search space theorems by showing that an inefficient threat resolution strategy (for example, SNLP) generates at least one (and often more than one) partial plan for each equivalent partial plan generated by one of the more efficient threat resolution strategies (for example, DSep).

For selecting open conditions, we used a LIFO strategy, a FIFO strategy, and a more sophisticated least-commitment strategy. The LIFO and FIFO strategies refer to the order that open conditions are attacked: oldest or youngest first. The least-commitment strategy selects the open condition to expand that would result in the fewest immediate children. For example, assume that two open conditions A and B are under consideration. Open condition A can be satisfied by adding either of two different operators to the plan. Condition B, on the other hand, can only be satisfied by linking to a unique initial condition. In this situation, the least-commitment strategy would favor working on B first. Note that the least-commitment strategy violates the assumption of Theorem 1; the action of the least-commitment strategy can depend on previous threat resolution actions.

The planner used for these demonstrations differs from the algorithm described in this paper in that it can ignore positive threats.[4] For most of our testing, we turned off detection of positive threats in order to reduce the amount of time spent in planning.

In the following plots, we have made no attempt to distinguish between individual planning problems. Instead, we have plotted the relative size of the search space of each problem in the sample domain. Each line corresponds to a single problem/conjunct-ordering-strategy pair. The relative size plotted in these figures is the quotient of the number of nodes explored by the planner when using a particular threat resolution strategy and the number of nodes explored when using a reference strategy. For example, in Figure 2, all of the search spaces sizes are normalized relative to the size of the most efficient threat resolution strategy for this domain, DSep. The shapes of these plots demonstrate the superiority of the DSep and DUnf threat resolution strategies over all of the other threat resolution strategies. The pseudoconvex shape of each of these curves illustrates the dominance relationships illustrated in Figure 1.

Figures 2 through 4 illustrate the relative search space size of several problems drawn from the tire changing, machine shop and route planning domains, respectively. Figures 5 and 6 illustrate the relative search space sizes for a variety of problems drawn from the ART-MD-RD and ART-1D-RD domains. The choice of threat delay strategy has no effect on the D^mS^1 and D^1S^1 domains (that is, the relative search space sizes are identical).

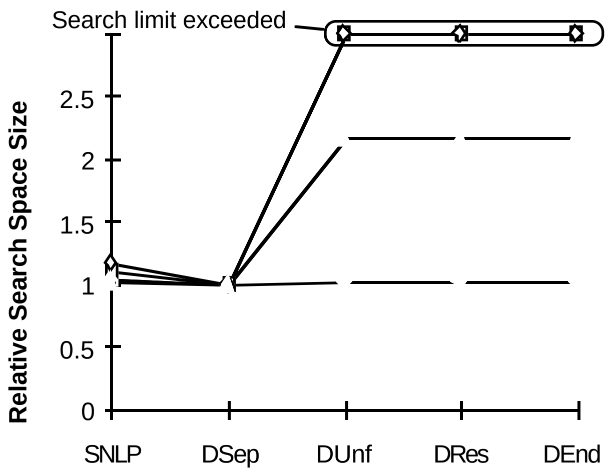

Figure 2: Russell’s Tire Changing Domain (3 Problems)

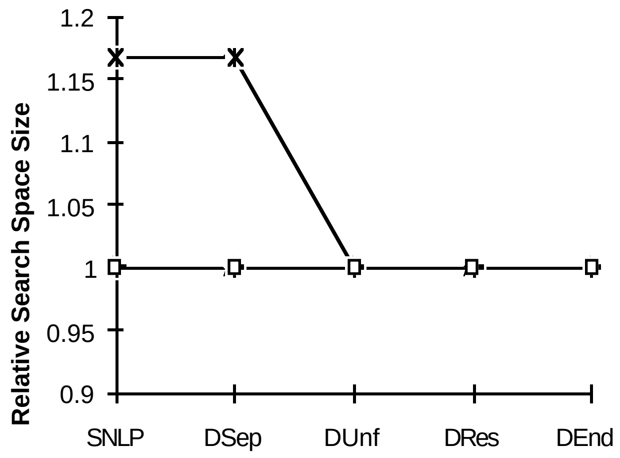

Figure 3: Machine Shop (3 Problems)

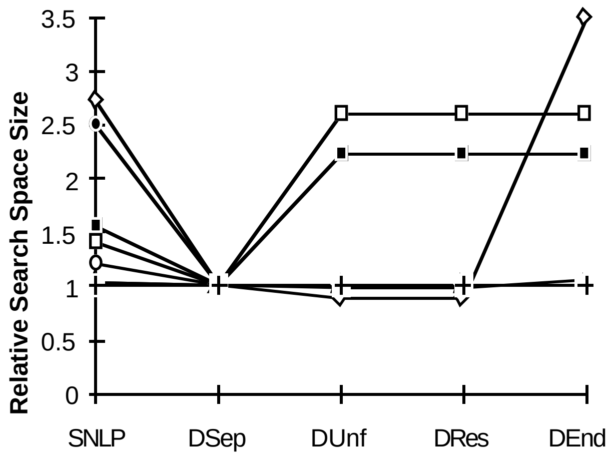

Figure 4: Route Planning Domain (5 Problems)

[4] and is, therefore, nonsystematic.

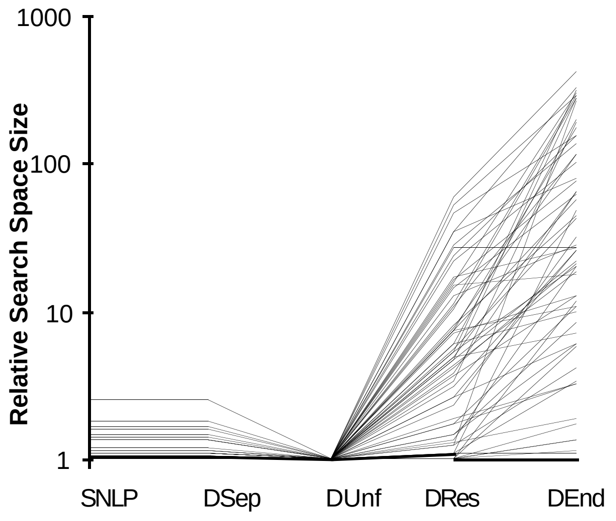

Figure 5: ART-MD-RD Planning Domain (29 Problems)

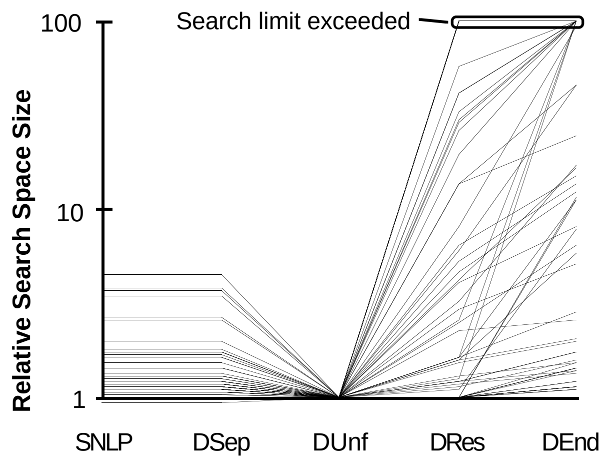

Figure 6: ART-1D-RD Domain (29 Problems)

Note that the artificial domains are propositional. Thus DSep has exactly the same effect as the default SNLP strategy because threats are never separable.

In Figure 7, we plot the CPU time required for planning using three of the threat resolution strategies on an assortment of ART-MD-RD problems. On small problems, the additional computation required for the more complicated threat resolution strategies dominates. For larger problems, however, we expect that these factors will become unimportant and that we will realize a savings that is roughly exponential in the number of threats.

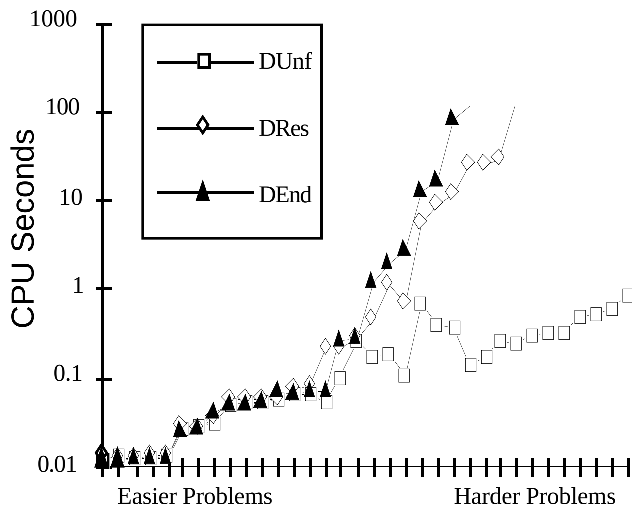

Figure 7: CPU time required for ART-1D-RD Problems

5 Related Work

Recently, Kambhampati (1993a) performed tests with several different planning algorithms, including SNLP, and argues that systematicity reduces the redundancy of the search space at the expense of increased commitment by the planner. He shows data indicating that for some classes of problems SNLP performs more poorly than planners that are nonsystematic.

Although we find these results interesting (and find Kambhampati's multi-contributor planners intriguing) we are left wondering to what extent his results would be affected by a more judicious selection of threat resolution strategies.

In his tests, Kambhampati also considers a variant of SNLP where positive threats are ignored (NONLIN). It is possible to construct examples where ignoring a positive threat in SNLP results in an arbitrarily large increase in search. Conversely, we believe that the consideration of positive threats does not cost a great deal. In particular, if t is the potential number of positive threats in a partial plan, we conjecture that considering positive threats in the delayed separability algorithm will never result in more than a factor of 3^t increase in the size of the search space of partial plans.

Kambhampati (1993b) also observes that the DSep threat resolution strategy is identical to SNLP if the definition for threats is modified. In particular, we would only recognize a threat when the post condition of a potentially threatening operator unifies with a protected precondition regardless of any bindings that might be added to the plan. Thus, with the appropriate threat definition, separation is not required for a complete, systematic variation of SNLP.

In addition, Kambhampati claims that these threat resolution strategies can be applied to planners that use multi-contributor causal structures (Kambhampati, 1992).

Yang (1993) has investigated the use of constraint satisfaction methods for resolving sets of threats. In our experience, the time at which the planner resolves threats has far more impact on performance than the method used for resolving sets of threats.

In (Smith & Peot, 1993), we have been investigating a more involved method of deciding when to work on threats. In particular, we show that for certain kinds of threats it is possible to prove that the threats can always be resolved at the end, and can therefore be postponed until planning is otherwise complete. The work reported in this paper is complementary since it suggests strategies for resolving threats that cannot be provably postponed.

Acknowledgments

This work was supported by DARPA contract F30602-91-C-0031 and a NSF Graduate Fellowship. Thanks to Will Harvey, Rao Kambhampati, and Dan Weld for their contributions to this paper.

References

Barrett, A., and Weld, D., Partial Order Planning: Evaluating Possible Efficiency Gains, to appear in *Artificial Intelligence*, 1993.

Collins, G., and Pryor, L., *Representation and performance in a partial order planner*, technical report 35, The Institute for the Learning Sciences, Northwestern University, 1992.

Harvey, W., Deferring Conflict-Resolution Retains Systematicity, Submitted to AAAI, 1993.

Kambhampati, S., Characterizing Multi-Contributor Causal Structures for Planning, Proceedings of the First International Conference on Artificial Intelligence Planning Systems, College Park, Maryland, 1992.

Kambhampati, S., On the utility of systematicity: understanding trade-offs between redundancy and commitment in partial-ordering planning, submitted to IJCAI, 1993a.

Kambhampati, S., A comparative analysis of search space size, systematicity and performance of partial-order planners. CSE Technical Report, Arizona State University, 1993b.

Kautz, Henry. Personal communication, April 6, 1992.

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

Minton, S., *Learning Effective Search Control Knowledge: An Explanation-Based Approach.* Ph.D. Thesis, Computer Science Department, Carnegie Mellon University, 1988.

Penberthy, J., S., Weld, D., UCPOP: A sound, complete, partial order planner for ADL, In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, Cambridge, MA, 1992.

Peot, M., and Smith, D., Conditional nonlinear planning, In *Proc. First International Conference on AI Planning Systems*, College Park, Maryland, 1992.

Sacerdoti, E., *A Structure for Plans and Behavior*, Elsevier, North Holland, New York, 1977.

Smith, D. and Peot, M., Postponing conflicts in nonlinear planning, AAAI Spring Symposium on Foundations of Planning, Stanford, CA, 1993, to appear.

Sussman, G., *A Computational Model of Skill Acquisition*, Report AI-TR-297, MIT AI Laboratory., 1973.

Wilkins, D., *Practical Planning: Extending the Classical AI Planning Paradigm*, Morgan Kauffman, San Mateo, 1988.

Yang, Q., A Theory of Conflict Resolution in Planning, *Artificial Intelligence*, 58 (1992) pg. 361-392.