Accelerating Partial Order Planners by Improving Plan and Goal Choices *

Lenhart Schubert[1] and Alfonso Gerevini[2]

[1] Dept. of Computer Science, University of Rochester
Rochester, NY 14627-0226, USA email: schubert@cs.rochester.edu
tel.: 716-275-8845 fax: 716-461-2018

[2] IRST, 38050 Povo, Trento, Italy
email: gerevini@irst.it tel.: +39-461-314333 fax: +39-461-314591

Abstract

We describe some simple domain-independent improvements to plan-refinement strategies for well-founded partial order planning that promise to bring this style of planning closer to practicality. One suggestion concerns the strategy for selecting plans for refinement among the current (incomplete) candidate plans. We propose an A* heuristic that counts only steps and open conditions, while ignoring "unsafe conditions" (threats). A second suggestion concerns the strategy for selecting open conditions (goals) to be established next in a selected incomplete plan. Here we propose a variant of a strategy suggested by Peot & Smith and studied by Joslin & Pollack; the variant gives top priority to unmatchable open conditions (enabling the elimination of the plan), second-highest priority to goals that can only be achieved uniquely, and otherwise uses LIFO prioritization. The preference for uniquely achievable goals is a "zero-commitment" strategy in the sense that the corresponding plan refinements are a matter of deductive certainty, involving no guesswork. In experiments based on modifications of UCPOP, we have obtained improvements by factors ranging from 5 to several hundred for a variety of problems that are nontrivial for the unmodified version. Crucially, the hardest problems give the greatest improvements.

Keywords: Improving Planning Efficiency, Planning Strategies, Partial Order Planning, Least Commitment Planning

*The work of the first author was supported by Rome Lab contract F30602-91-C-0010.

1 Introduction

The history of planning research shows at least two major strands, whose respective goals are to achieve *practical* planning and *well-founded* planning. Practical planning research seeks to provide planning frameworks and tools that are sufficiently expressive, flexible and efficient to be effectively usable in applications such as planning robot actions, transportation planning, factory scheduling, genetic engineering and conversation planning. Some of the earliest practically motivated planning "formalisms" were the MICROPLANNER implementation of Hewitt's PLANNER language [13, 24], STRIPS [8], and NOAH [23], and some familiar later examples are NONLIN [25], DEVISER [26], SIPE [27], PRS [9], FORBIN [7], and O-Plan [6].

The emphasis in well-founded planning is on constructing planners that can be *proved* to have certain desirable properties, such as soundness and completeness for their intended class of problems, or the ability to find optimal or near-optimal solutions. The first well-founded planner was probably C. Green's QA3 [12], offering sound and complete planning within the expressively quite rich situation calculus. However, its performance was impractical, and this provided some of the impetus behind the development of STRIPS and its descendants. The subsequent quest for more practical planners contributed many valuable ideas to planning theory and practice, but there remained a lingering dissatisfaction in the planning community with the lack of formal foundations and guarantees for the resultant planners (highlighted by troublesome problems such as the Sussman anomaly and the register exchange problem). This led to a renewal of efforts in the 80's to find viable approaches to well-founded planning, exemplified by novel algorithms such as BIGRESS [22, 16] (based on dynamic logic), Bibel's linear connection method for plan generation [3], TWEAK [4] (Chapman's partial-order planner based on his "modal truth criterion") and SNLP [18] (another systematic partial-order planner using propositional STRIPS operators). These efforts have gained considerable momentum in recent years, leading both to extensions of earlier approaches, such as UCPOP [20] and BURIDAN [17], and systematic comparative performance evaluations (e.g., [2, 15]).

Despite these efforts, it seems fair to say that well-founded planners still perform dismally in practical terms. For example, when we tried to apply the programs evaluated in [15] to the standard UCPOP suite of test problems, we found that none achieve reasonable performance on the 3-disk Towers of Hanoi (T of H) puzzle (requiring 7 moves for its solution), or on some other simple problems. UCPOP did best on T of H but still took over 3 minutes of CPU time on a SUN 10, generating tens of thousands of partial plans. (This was with the "delay separation" switch on [21];[1] with this switch off, performance

[1] i.e., delaying the use of "promotion" and "demotion" to avert threats until all variables appearing in the conflict conditions are bound; and disabling altogether the use of inequations to block unification of threatening effects with threatened causal links

was typically several times worse.) This is disappointing, since puzzles like T of H are easily solved by inexperienced people, with very little trial and error search; moreover, the very first well-founded planner, C. Green's QA3, reportedly solved some (carefully formulated) versions of this problem rather easily [12]. It should be noted that such toy problems are not particularly outlandish from a practical perspective; for instance T of H and blocks world problems resemble problems that arise in such areas as connecting railroad cars into trains (with use of sidetracks) and pallet management in automated warehouses. (Some of the other problems in the test suite, such as the "ferry domain", are more directly evocative of real-world applications.)

Some recent studies of partial-order planning strategies (e.g., [15]) could be interpreted as implying that the level of planning performance achieved so far is about the best that is possible for *domain-independent* planners; any real improvements from this point on will have to come from exploiting domain-specific information. Our outlook on well-founded, domain-independent planning is more optimistic. In the following, we suggest improved planning strategies based on the one hand on more carefully formulated heuristics for selecting plans for refinement, and on the other on "deductively oriented" (or "zero commitment") strategies for choosing subgoals. We describe these two classes of techniques in Sections 2 below, and in Section 3 we report our preliminary experimental results based on slightly modified versions of UCPOP. These results suggest that order-of-magnitude improvements in the performance of well-founded planners are possible, bringing them closer to practical usability.

2 Plan Selection and Goal Selection

2.1 UCPOP

We will be basing our discussion and experiments on UCPOP, an algorithm exemplifying the state of the art in well-founded partial-order planning. Thus we begin with a sketch of this algorithm, referring the reader to [1, 20] for details.

In essence, UCPOP explores a space of partially specified plans, each paired with an agenda of goals still to be satisfied and threats still to be averted. The initial plan contains a dummy *start* action whose effects are the given initial conditions, and a dummy *end* action whose preconditions are the given goals. Thus goals are uniformly viewed as action preconditions, and are uniformly achieved through the effects of actions, including the *start* action.

The plans themselves consist of a collection of *steps* (i.e., actions obtained by instantiating the available operators), along with a set of *causal links*, a set of *binding constraints*, and a set of *ordering constraints*. When an open goal (precondition) is selected from the agenda, it is established (if possible) either by adding a step with an effect that unifies with the goal, or by using an existing step with an effect that unifies with the goal. (In the latter case, it must be consistent with current ordering constraints to place the existing step before the goal, i.e., before the step whose preconditions generated the goal.) When a new or existing step is used to establish a goal in this way, there are several "side effects":

- A causal link (S_p, Q, S_c) is also added, where S_p indicates the step "producing" the goal condition Q and S_c indicates the step "consuming" Q. This causal link serves to protect the intended effect of the added (or reused) step from interference by other steps.
- Binding constraints are added, corresponding to the unifier for the action effect in question and the goal (precondition) it achieves.
- An ordering constraint is added, placing the step in question before the step whose precondition it achieves.
- If the action in question is new, its preconditions are added to the agenda as new goals (except that eq/neq conditions are integrated into the binding constraints – see below).
- New threats (called "unsafe conditions") are determined. For a new step and its causal link, other steps threaten the causal link if they have effects unifiable with the condition protected by the causal link (and these effects can occur temporally during the causal link); and the effects of the new step may similarly threaten other causal links. In either case, new threats are placed on the agenda.

Binding constraints assert the identity (eq) or nonidentity (neq) of two variables or a variable and a constant. Eq-constraints arise from unifying open goals with action effects, and neq-constraints arise from neq-preconditions of newly instantiated actions and from matching negative goals containing variables to the initial state. (We set aside "separation" as a means of averting threats, which also leads to neq-constraints.) Neq-constraints may be disjunctive, but are handled simply by generating separate plans for each disjunct.

The overall control loop of UCPOP consists of selecting a plan from the current list of plans (initially the single plan based on *start* and *end*), selecting a goal or threat from its agenda, and replacing the plan by the corresponding refined plans. If the agenda item is a goal, the refined plans are those corresponding to all ways of establishing the goal using a new or existing step. If the agenda item is a threat to a causal link (S_p, Q, S_c), then with the "delay separation" switch on there are two refined plans, respectively constraining the threatening step to be before step S_p (demotion) or after step S_c (promotion), thus averting the threat.

Inconsistencies in binding constraints and ordering constraints are detected when they first occur (as a result of adding a new constraint) and the corresponding plans are eliminated. Planning fails if no plans remain. The success condition is the creation of a plan with consistent binding and ordering constraints and an empty agenda.

What we have described so far is actually POP. The "U" and "C" in UCPOP correspond to a liberalized form of STRIPS-like operator specifications, allowing universally quantified preconditions (and goals) and conditional effects. For instance, it is permissible to have a precondition for a PICKUP(x) action that says that for all y, (not (on y x)) holds. Also, it is permissible to have conditional effects for a PUTON(x,y,z) action ("put x on y from z"), stating that when y is not the table, it will not be clear at the end of the action, and when z is not the table, it will be clear at the end of the action. We need not be concerned here with the details of how such conditions are handled. They cause only minor perturbations in the operation of UCPOP; for instance, conditional effects can lead to multiple matches against operators for a given goal, each match generating different preconditions. (Of course, there can be multiple matches even without conditional effects, if some predicates occur more than once in the effects.)

The key issues for us are the strategic ones: how plans are selected from the current set of plans (discussed in section 2.2), and how goals are selected for a given plan (discussed in section 2.3).

2.2 The trouble with counting unsafe conditions

The choice of the next plan to refine in the UCPOP system is based on an A* best-first search. Recall that A* uses a heuristic estimate $f(p)$ of overall solution cost consisting of a part $g(p)$ = cost of the current partial solution (plan) p and a part $h(p)$ = estimate of the additional cost of the best complete solution that extends p. In the current context it is helpful to think of $f(p)$ as a measure of plan *complexity*, i.e., "good" plans are simple (low-complexity) plans.

There are two points of which the reader should be reminded. First, in order for A* to guarantee discovery of an *optimal* plan (i.e., the "admissibility" condition), $h(p)$ should not *over*estimate the remaining solution cost [19]. Second, if the aim is not necessarily to find an optimal solution but to find a satisfactory solution *quickly*, then $f(p)$ can be augmented to include a term that estimates the remaining cost of *finding* a solution. One common way of doing that is to use a term proportional to h(p) for this as well, i.e., we "emphasize" the h-component of f relative to the g-component. This is reasonable to the extent that the plans that are most nearly complete (indicated by a low h-value) are likely to take the least effort to complete. Thus we will prefer to pursue a plan p' that seems closer to being complete to a plan p further from completion, even though the *overall* complexity estimate for p' may be greater than for p [19] (pages 87–88). Alternatively, we could add a heuristic estimate of the remaining cost of finding a solution to $f(p)$ that is more or less independent of the estimate $h(p)$.

With these considerations in mind, we now evaluate the advisability of including the various terms in UCPOP's function for guiding its A* search, namely

S, OC, CL, and UC,

where S is the number of steps in the partial plan, OC is the number of open conditions (unsatisfied goals and preconditions), CL is the number of causal links, and UC is the number of unsafe conditions (the number of pairs of steps and causal links where the step threatens the causal link). The default combination used by UCPOP is S+OC+UC.

(a) Concerning S, the number of steps currently in the plan, this can naturally be viewed as comprising $g(p)$, the plan complexity so far. Intuitively, a plan is complex to the extent that it contains many steps. While in some domains we might want to make distinctions among the costs of different kinds of steps, a simple step count seems like a reasonable generic complexity measure.

(b) Concerning OC, the number of open conditions, this can be viewed as playing the role of $h(p)$, since each remaining open condition must be established by some step. The catch is that it may be possible to use existing steps in the plan (including *start*, i.e., the initial conditions) to establish remaining open conditions. Thus OC can overestimate the number of steps still to be added, forfeiting admissibility.

On the other hand, even open conditions that do not require new steps do require some work on the part of the planning algorithm. So these open conditions can be viewed as contributing to the remaining cost of *finding* a solution, biasing UCPOP slightly toward trading off solution cost against solution-finding cost. As such, OC appears to be a reasonable generic component of the A* heuristic function.

(c) Concerning CL, the number of causal links, one might motivate the inclusion of this term by arguing that numerous causal links are indicative of a complex plan. As such, CL appears to be an alternative to step-counting. In fact, as long as a partial plan does not yet link any preconditions to the initial state, and as long as each step establishes just one precondition or goal, CL is essentially the same as S. Once preconditions are linked to the initial state, or if steps are used to establish multiple conditions, CL will differ from S by preferring plans in which steps have few preconditions to plans in which steps have more preconditions, even when the total number of steps are the same.

This seems like a reasonable alternative to S. However, if we simply add CL to S, we will again tend to emphasize plan cost relative to plan-completion cost, and thus decrease the chances of finding a solution quickly. So it appears that if CL is used as a g-measure, then the S term should be dropped from the overall heuristic.

(d) Concerning UC, the number of unsafe conditions, we note first of all that this is clearly not an h-measure. The number of unsafe conditions bears no definite relation to the number of steps that must still be added, and in fact arbitrarily many "unsafe" conditions may cease to be unsafe upon addition of ordering constraints or binding constraints. When such expired threats are selected from the agenda, they are recognized as such and discarded without further action.

Can we then view UC as a g-measure? Or as a measure of the remaining cost of finding a plan? The former possibility seems plausible at first glance for plans in which we have added no constraints to avert unsafe conditions. For such plans, UC should generally increase with the number of steps in those plans, since adding steps typically adds unsafe conditions.[2] However, once we have refined *some* plans to remove unsafe conditions, the UC count need no longer vary systematically with the number of steps. Besides, even if it did, augmenting g in this way would work *against* finding a solution quickly, since it would emphasize g rather than h.

That leaves us with the question whether UC is indicative of the remaining cost of *finding* a solution. One could argue that unsafe conditions are "flaws" that will have to be remedied by refinement steps. The more refinements a plan requires, the longer it will take to complete.

However, this argument is dubious at best. As already noted, unsafe conditions include many *possible* conflicts which may eventually vanish as a result of subsequent partial ordering choices and variable binding choices not specifically aimed at removing these conflicts. Thus counting unsafe conditions can arbitrarily overestimate the number of genuine refinements still needed to complete the plan. In fact if we consider a plan that already contains all n steps that will be needed, we can see that in the worst case there may be $O(n^2)$ unsafe conditions, yet there must exist $O(n)$ refinements that fully linearize these steps, completing the plan. This observation also suggests that UC could easily swamp the S+OC terms, suppressing their role in guiding the A* search.

The conclusions we can draw are thus that S+OC and CL+OC are the most promising general heuristic measures for plan selection, while the UC term

[2]UC will also tend to increase when existing steps are used to establish open conditions, since this adds causal links. Since CL could also serve as a plan complexity measure – see (c) – this tendency is still consistent with the supposition under consideration.

should probably not be included. Note that with both S+OC and CL+OC there will be a preference for those offspring of a plan that reuse actions already in the plan rather than adding new actions. With CL+OC, such offspring have the same cost as the parent, while with S+OC they actually have a *lower* cost, emphasizing the preference for action reuse. This emphasis appears to give the S+OC measure considerable advantages in some domains. Because of its nearly uniform experimental superiority to the CL+OC measure, we will not further consider the latter here. The S+OC heuristic was in fact previously considerd by Peot and Smith [21], but because their focus was on threat-removal strategies neither they nor other researchers appear to have fully recognized the advantages of this measure.

2.3 The goal selection strategy

An important opportunity for improving planning performance independently of the domain lies in identifying "forced refinements", i.e., refinements that can be made *deterministically*. Specifically, it makes sense to give top priority to open conditions that cannot be achieved; and then preferring open conditions that can only be achieved in one unique way – either through addition of an action not yet in the plan, or through a unique match against the initial conditions.

The argument for giving top priority to unachievable goals is just that plans containing such goals can be eliminated. Thus we prevent allocation of effort to the refinement of doomed plans, and to the generation and refinement of their doomed successor plans.

The argument for preferring open conditions that can only be achieved uniquely is equally apparent. Since every open condition must eventually be established by *some* action, it follows that if this action is unique, it must be part of every possible completion of the partial plan under consideration. So, adding the action is a "zero-commitment" refinement, involving no choices or guesswork. At the same time, adding *any* refinement in general narrows down the search space by adding binding constraints and adding a causal link and further effects that can temporally constrain other threatening or threatened actions. For unique refinements this narrowing-down is monotonic, never needing revocation. In short, the strategy cuts down the search space without loss of access to viable solutions.

Peot and Smith [21] studied the strategy of preferring forced threats to unforced threats, and also suggested possible use of a "least commitment" strategy for handling open conditions. "Least commitment" always selects an open condition which generates the fewest refined plans. Thus it *entails* the priorities for unachievable and uniquely achievable goals above (while also entailing a certain prioritization of nonuniquely achievable goals). Joslin and Pollack [14] studied the uniform application of such a strategy to both threats and open conditions in UCPOP, terming this strategy "least cost flaw repair" (LCFR).[3] Combining this with UCPOP's default plan selection strategy, they obtained significant search reductions (though less significant running time reductions, for implementation reasons) for a majority of the problems in the UCPOP test suite.

In UCPOP, goals are selected from the agenda according to a LIFO (last in first out, i.e., stack) discipline. Based on experience with search processes in AI in general, such a strategy has much to recommend it, as a simple default. It will tend to maintain focus on the achievement of a particular higher-level goal by regression – very much as in prolog goal chaining – rather than attempting to achieve multiple goals in breadth-first fashion. We have therefore chosen to stay with UCPOP's LIFO strategy whenever there are no unachievable or forced open conditions. This has led to very substantial improvements over LCFR in our experiments.

3 Experiments Using UCPOP

In order to test our ideas we have modified version 2.0 of UCPOP [1], replacing its default plan-selection strategy (S+OC+UC) and goal-selection strategy (LIFO) to incorporate strategies discussed in the previous sections. We use "ZLIFO" ("zero-commitment last in first out") to denote the goal-selection strategy that assigns highest priority to open conditions that can be achieved with zero-commitment plan refinements, and second-highest priority to open conditions most recently added to the agenda.

We have tested the modified planner on several problems in the UCPOP suite, emphasizing those that had proved most challenging for previous strategies. We have also included one of the two artificial domains (ART-$\#_{est}$-$\#_{clob}$) that served as a testbed for Kambhampati *et al.*'s extensive study of the behavior of various planning strategies as a function of problem parameters [15].[4] The experiments were conducted on a SUN 10.

Figures 1 and 2 give the formalizations of the two versions of the T of H domain in terms of UCPOP's language, while the formalizations of the other problems from the UCPOP suite are not repeated here.[5] We thought it important to test more than one version of T of H, since this was the hardest problem for UCPOP (as well as other algorithms we tried before focusing on UCPOP), and its difficulty has long been known to be sensitive to the formalization (e.g., [12]). Figure 3 supplies a UCPOP formalization of ART-$\#_{est}$-$\#_{clob}$.

[3]We would find "least *commitment* flaw repair" more accurate.

[4]This domain was chosen since absolute performance data are provided for it in [15].

[5]The formalizations of these domains except the 3-operator version of the T oh H and the artificial domain from [15] are available along with UCPOP via anonymous FTP from cs.washington.edu

```
(define (operator move-disk)
   :parameters ((disk ?disk) ?below-disk ?new-below-disk)
   :precondition (:and (smaller ?disk ?new-below-disk)  ;handles pegs
                       (:neq ?new-below-disk ?below-disk)
                       (:neq ?new-below-disk ?disk)
                       (:neq ?below-disk ?disk)
                       (on ?disk ?below-disk)
                       (clear ?disk)
                       (clear ?new-below-disk))
   :effect (:and (clear ?below-disk)
                 (on ?disk ?new-below-disk)
                 (:not (on ?disk ?below-disk))
                 (:not (clear ?new-below-disk))))

Initial state: ((smaller D1 P1) (smaller D2 P1) (smaller D3 P1) (smaller D1 P2)
                (smaller D2 P2) (smaller D3 P2) (smaller D1 P3) (smaller D2 P3)
                (smaller D3 P3) (smaller D1 D2) (smaller D1 D3) (smaller D2 D3)
                (clear P2) (clear P3) (clear D1) (disk D1) (disk D2) (disk D3)
                (on D1 D2) (on D2 D3) (on D3 P1))
   Goal state: (and (on D1 D2) (on D2 D3) (on D3 P3))
```

Figure 1: Formalization of T-of-H1

plan-selection	**goal-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	204.51	160911/107649
LIFO	S+OC	0.97	751/511
ZLIFO	S+OC+UC	6.90	1816/1291
ZLIFO	S+OC	0.54	253/184

Table I: Performance of plan/goal selection strategies on T-of-H1

Tables I–IX show the CPU time (seconds) and the number of plans created/explored by UCPOP on nine problems in the following domains: Towers of Hanoi with three disks and either one operator (T-of-H1) or three operators (T-of-H3), the blocks world (tower-invert4 and sussman-anomaly), Russell's tire changing domain (fix3), the ferry domain (ferry-test), "Dan's fridge" domain (fixa), and the artificial domain ART-$\#_{est}$-$\#_{clob}$ (specifically, ART-3-6 and ART-6-3). Note that the number of plans is probably more meaningful than the CPU time for evaluating the performance of the strategies examined. In fact our implementation of these strategies was committed to not altering UCPOP's data structures; they could have been implemented more efficiently with modified data structures.

Tables I and II show that for the T of H the plan selection strategy S+OC gives dramatic improvements over the default S+OC+UC strategy. (In these tests

plan-selection	**goal-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	> 600	> 500000
LIFO	S+OC	8.54	5506/3415
ZLIFO	S+OC+UC	> 600	> 500000
ZLIFO	S+OC	1.24	641/420

Table II: Performance of plan/goal selection strategies on T-of-H3

plan-selection	**goal-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	2.45	2131/1903
LIFO	S+OC	2.48	2131/1903
ZLIFO	S+OC+UC	0.33	96/74
ZLIFO	S+OC	0.33	96/74

Table III: Performance of plan/goal selection strategies on fixa

plan-selection	**goal-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	6.50	3396/2071
LIFO	S+OC	0.43	351/215
ZLIFO	S+OC+UC	1.12	357/221
ZLIFO	S+OC	1.53	574/373

Table IV: Performance of plan/goal selection strategies on fix3

plan-selection	**goal-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	1.35	808/540
LIFO	S+OC	0.19	148/105
ZLIFO	S+OC+UC	2.81	571/378
ZLIFO	S+OC	0.36	142/96

Table V: Performance of plan/goal selection strategies on invert-tower4

plan-selection	**goal-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	0.63	718/457
LIFO	S+OC	0.32	441/301
ZLIFO	S+OC+UC	0.24	136/91
ZLIFO	S+OC	0.22	140/93

Table VI: Performance of plan/goal selection strategies on test-ferry

goal-selection	**plan-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	.67	558/392
LIFO	S+OC	1.36	1299/840
ZLIFO	S+OC+UC	0.16	72/49
ZLIFO	S+OC	0.18	79/54

Table VII: Performance of plan/goal selection strategies on ART-$\#_{est}$-$\#_{clob}$ with $\#_{est} = 3$ and $\#_{clob} = 6$ (averaged over 100 problems)

goal-selection	**plan-selection**	**CPU-time**	**plans created/explored**
LIFO	S+OC+UC	1.32	985/653
LIFO	S+OC	2.08	1743/1043
ZLIFO	S+OC+UC	0.14	57/37
ZLIFO	S+OC	0.14	57/37

Table VIII: Performance of plan/goal selection strategies on ART-$\#_{est}$-$\#_{clob}$ with $\#_{est} = 6$ and $\#_{clob} = 3$ (averaged over 100 problems)

```
(define (operator MOVE-D1)
  :parameters ((thing ?from) (thing ?to))
  :precondition (:and (on D1 ?from)
                      (clear ?to)
                      (:not (on D1 ?to))
                      (:neq ?to D1))
  :effect
  (:and (on D1 ?to)
        (:not (clear ?to))
        (clear ?from)
        (:not (on D1 ?from)))
```

```
(define (operator MOVE-D2)
  :parameters ((thing ?from) (thing ?to))
  :precondition (:and (on D2 ?from)
                      (clear ?to)
                      (:not (on D2 ?to))
                      (:not (on D1 D2))
                      (:not (on D1 ?to))
                      (:neq ?to D1)
                      (:neq ?to D2))
  :effect (:and (on D2 ?to)
                (:not (clear ?to))
                (clear ?from)
                (:not (on D2 ?from))))
```

```
(define (operator MOVE-D3)
  :parameters ((thing ?from) (thing ?to))
  :precondition (:and (on D3 ?from)
                      (clear ?to)
                      (:not (on D3 ?to))
                      (:not (on D1 D3))
                      (:not (on D2 D3))
                      (:not (on D1 ?to))
                      (:not (on D2 ?to))
                      (:neq ?to D1)
                      (:neq ?to D2)
                      (:neq ?to D3))
  :effect (:and (on D3 ?to)
                (:not (clear ?to))
                (clear ?from)
                (:not (on D3 ?from))))
```

```
Initial state: ((on D1 D2) (on D2 D3)
                 (on D3 P1) (clear D1)
                 (thing D1) (thing D2)
                 (thing D3) (thing P1)
                 (thing P2) (thing P3)
                 (clear D1) (clear P2)
                 (clear P3))

   Goal state: (and (on D1 D2) (on D2 D3)
                    (on D3 P3))
```

Figure 2: Formalization of T-of-H3

plan-selection	goal-selection	CPU-time	plans created/explored
LIFO	S+OC+UC	0.06	44/26
LIFO	S+OC	0.04	36/21
ZLIFO	S+OC+UC	0.12	67/43
ZLIFO	S+OC	0.07	41/25

Table IX: Performance of plan/goal selection strategies on sussman-anomaly

the default LIFO goal selection strategy was used.) In fact, UCPOP solved T-of-H1 in 0.97 seconds using S+OC versus 204.5 seconds using S+OC+UC. T-of-H3 proved harder to solve than T-of-H1, requiring 8.5 seconds using S+OC and an unknown time in excess of the 600 CPU-second limit using S+OC+UC.

Our ZLIFO goal-selection strategy can significantly accelerate planning compared with the simple LIFO strategy. In particular, when ZLIFO was combined with the S+OC plan-selection strategy in solving T of H, it further reduced the number of plans generated by a factor of 3 in T-of-H1 (obtaining an overall reduction by a factor of 636, and decreased the required CPU time from 204.5 to 0.54 seconds!), and by a factor of 8 in T-of-H3.

Tables III–VIII provide data for problems that are easier than T of H, but still challenging to UCPOP operating with its default strategy, namely fixa, tower-invert4, fix3, test-ferry and the artificial domain ART-$\#_{est}$-$\#_{clob}$ (with $\#_{est} = 3$ and $\#_{clob} = 6$ and with $\#_{est} = 6$ and $\#_{clob} = 3$). The results show

```
;Replace i by 0, ..., 9 in the following two operators:
;
(define (operator Ai1)              (define (operator Ai2)
   :parameters ()                      :parameters ()
   :precondition ((Ii))                :precondition ((Pi))
   :effect (:and (Pi) [(Ii+1)]         :effect (:and (Gi) [(Pi+1)]
              {(:not (Ii-1))}))                   {(:not (Pi-1))}))

Initial state: ((I0) (I1) (I2) (I3) (I4) (I5) (I6) (I7) (I8) (I9))
Goal state: (and (G0) (G1) (G2) (G3) (G4) (G5) (G6) (G7) (G8) (G9))
```

Figure 3: Formalization of ART-$\#_{est}$-$\#_{clob}$. The square brackets (not part of the syntax) indicate parts to be included only for $i < n_+$ ($\#_{est}$); the braces (not part of the syntax) indicate parts to be included only for $0 < i < n_-$ ($\#_{clob}$).

that the combination of S+OC and ZLIFO substantially accelerates UCPOP in comparison with its performance using OC+S+UC and LIFO. The number of plans generated dropped by a factor of 22 for fixa, by a factor of 5.9 for fix3, by a factor of 5.7 for tower-invert4, by a factor of 5.1 for test-ferry, by a factor of 7 for ART-3-6, and by a factor of 17 for ART-6-3.

Concerning ART-$\#_{est}$-$\#_{clob}$, note that the performance we obtained with unenhanced UCPOP (624 plans generated for ART-3-6 and 985 for ART-6-3) was much the same as (just marginally better than) reported in [15] for the best planners considered there (700 - 1500 plans generated for ART-3-6, and 1000-2000 for ART-6-3). This is to be expected, since UCPOP is a generalization of the earlier partial-order planners. Relative to standard UCPOP and its predecessors, our "accelerated" planner is thus an order of magnitude faster. Interestingly, the entire improvement here can be ascribed to ZLIFO (rather than S+OC plan selection, which is actually a little worse than S+OC+UC). This is probably due to the unusual arrangement of operators in ART-$\#_{est}$-$\#_{clob}$ into a "clobbering chain" ($A_{n_-,1}$ clobbers $A_{n_--1,1}$'s preconditions, ..., $A_{1,1}$ clobbers $A_{0,1}$'s preconditions; similarly for $A_{i,2}$), which makes immediate attention to new unsafe conditions an unusually good strategy.

In experimenting with various combinatorially trivial problems that unmodified UCPOP handles with ease, we found that the S+OC and ZLIFO strategy is neither beneficial nor harmful in general; there may be a slight improvement or a slight degradation in performance. Results for the Sussman anomaly in table IX provide an illustrative example.

For direct comparison with Joslin and Pollack's LCFR strategy, we implemented their strategy and applied it to a few problems. It did very well (sometimes better than ZLIFO) for problems on the lower end of the difficulty spectrum, but poorly for harder problems. For T-of-H3 (the hardest problem), LCFR in combination with the default S+OC+UC plan selection strategy ran in 96.3 cpu seconds, creating/ exploring 9942/6402 plans (cf., 641/420 for ZLIFO). With S+OC plan selection, results were marginally better (87.7 cpu

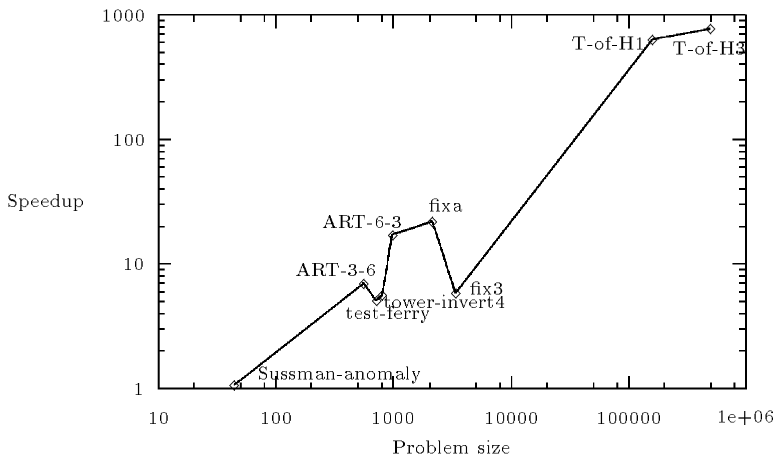

Figure 4: Increased speedup of ZLIFO and S+OC relative to the number of plans generated by LIFO and S+OC+UC (log-log scale).

seconds, 9387/6998 plans).

We summarize our results in Figure 4, showing the speedup obtained with the combined ZLIFO goal selection strategy and S+OC plan selection strategy *as a function of problem difficulty* (as indicated by the number of plans generated by the default LIFO plus S+OC+UC strategy). The trend toward greater speedups for more complex problems (though somewhat dependent on problem type) is quite apparent from the log-log plot.

4 Conclusions and Further Work

We have argued in favor of some simple, domain-independent improvements to partial order planning strategies, based on the one hand on a carefully considered choice of terms in the A* heuristic for plan selection, and on the other on a preference for choosing open conditions that cannot be achieved at all or can only be achieved uniquely (with a default LIFO prioritization of other open conditions). Since the plan refinements corresponding to uniquely achievable goals are logically necessary, we have termed this strategy a "zero-commitment" strategy.

Our experiments based on modifications of UCPOP indicate that our strategies can give large improvements in planning performance, especially for problems that are hard for UCPOP (and its "relatives") to begin with. The best performance was achieved when our strategies for plan selection and goal selection were used in combination. Further, our results indicate that zero-commitment is best supplemented with a LIFO strategy for open conditions achievable in multiple ways, rather than a generalization of zero-commitment favoring goals with the fewest children. A sufficient variety of problems were tried to indicate that our techniques are of broad potential utility.

One promising direction for further work is to make the zero-commitment strategy apply more often by developing ways of identifying "false options" as early as possible. That is, if a possible action instance (obtained by matching an open condition against available operators as well as against existing actions) is easily recognizable as inconsistent with the current plan, then its elimination may leave us with a single remaining match and hence an opportunity to apply the zero-commitment strategy.

One way of implementing this strategy would be to check at once, before accepting a matched action as a possible way to attain an open condition, whether the temporal constraints on that action force it to violate a causal link, or alternatively, force its causal link to be violated. In that case the action could immediately be eliminated, perhaps leaving only one (or even no) alternative. This could perhaps be made even more effective by broadening the definition of threats so that preconditions as well as effects of actions can threaten causal links, and hence bring to light inconsistencies sooner. Note that if a precondition of an action is inconsistent with a causal link, it will have to be established with another action whose *effects* violate the causal link; so the precondition really poses a threat from the outset.

Another direction for further work is to apply efficient temporal reasoning methods to the problem of eliminating inconsistent promotion/demotion alternatives for threat elimination, given the set of all (definite) threats and ordering relations in the plan under development. Though this problem is in principle NP-hard, algorithms that are very efficient on average are described in [10, 11]. This could be far more efficient than trying each possible promotion and demotion, checking in isolation for consistency with ordering constraints. A similar idea was previously explored in [28] using arc consistency techniques, but we think further gains are possible with the algorithms mentioned above, which are more general than arc-consistency testing and employ intelligent backtracking for efficient search.

Finally, another direction that seems very promising to us (based on some hand simulations) is to precompute certain constraints that must hold throughout the search space of a given problem, based on the structure of the operators, initial conditions and goal conditions. This often permits some matching actions for open conditions to be immediately eliminated, as they would violate the precomputed constraints.

Our conclusion, both from the results we have presented and from the possibilities for further speedups we have mentioned, is that ample opportunities still exist for major improvements in the performance of well-founded, domain-independent planners. These may be sufficient to make such planners competitive with current more pragmatically designed planners.

References

[1] A. Barrett, K. Golden, S. Penberthy, and D. Weld. UCPOP user's manual. Technical Report 93-09-06, Dept. of Computer Science and Engineering, University of Washington, Seattle, WA 98105, 1994.

[2] A. Barrett and D. S. Weld. Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, 67:71–112, 1994.

[3] W.A. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4(2):115–132, 1986.

[4] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.

[5] K. Currie and A. Tate. O-Plan: The open planning architecture. *Artificial Intelligence*, 51(1), 1991.

[6] J. Dalton, B. Drabble, and A. Tate. The O-Plan constraint associator. In *Thirtheenth workshop of the UK Planning Special Interest Group*, Glasgow, UK, 1994.

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

[8] R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[9] M.P. Georgeff and A.L Lansky. Reactive reasoning and planning. In *Proc. of the 6th National Conference of the American Association for Artificial Intelligence*, pages 677–682, Seattle, WA, 1987. Morgan Kaufmann.

[10] A. Gerevini and L.K. Schubert. An efficient method for managing disjunctions in qualitative temporal reasoning. In *Principles of Knowledge Representation and Reasoning: Proc. of the 4th Int. Conf. (KR-94)*, pages 215–225, San Francisco, CA, 1994. Morgan-Kaufmann.

[11] A. Gerevini and L.K. Schubert. Efficient algorithms for qualitative reasoning about time. *Artificial Intelligence*, 1995. To appear. Also available as: IRST Tech. Rep. 9307-44, Istituto per la Ricerca Scientifica e Tecnologica, 38050 Povo, Trento Italy; Tech. Rep. 496, Computer Science Dept., University of Rochester, Rochester, NY 14627, USA.

[12] C. Green. Application of theorem proving to problem solving. In *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-69)*, pages 219–239, 1969.

[13] C. Hewitt. Planner: A language for proving theorems in robots. In *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-69)*, pages 295–301, Bedford, MA, 1969. Morgan Kaufmann.

[14] D. Joslin and M.E. Pollack. Least-cost flaw repair: a plan refinement strategy for partial-order planning. In *Proc.of the 12th Nat. Conf. of the American Association for Artificial Intelligence (AAAI-94)*, pages 1004–1009, Seattle WA, 1994.

[15] S. Kambhampati, C. A. Knoblock, and Q. Yang. Planning as refinement search: A unified framework for evaluating design tradeoff in partial-order planning. *Artificial Intelligence. Special Issue on Planning and Scheduling*, 1995. To appear; also available as Tech. Rep. ASU-CSE-TR 94-002, Dept. of Computer Science and Engineering, Arizona State Univ., Temple, AZ.

[16] H.A. Kautz. Planning within first-order dynamic logic. In *4th Bienn. Conf. of the Can. Soc. for Computational Stud. of Intelligence (CSCSI-82)*, pages 19–26, niv. of Saskatchewan, Saskatoon, Sask., 1982.

[17] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic least-commitment planning. In *Proc.of the 12th Nat. Conf. of the American Association for Artificial Intelligence (AAAI-94)*, pages 1073–1078, Seattle WA, 1994.

[18] D. McAllester and Rosenblitt D. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 634–639, July 1991.

[19] N.J. Nilsson. *Principles of Artificial Intelligence.* Tioga Pub. Co., Palo Alto, CA, 1980.

[20] J.S. Penberthy and D.S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In B. Nebel, C. Rich, and W. Swartout, editors, *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR-92)*, pages 103–114, Boston, MA, 1992. Morgan Kaufmann.

[21] M. A. Peot and D. E. Smith. Threat-removal strategies for partial-order planning. In *Proc. of the 11th Nat. Conf. of the American Association for Artificial Intelligence (AAAI-93)*, pages 492–499, Washington, D.C.,1993.

[22] S. Rosenschein. Plan synthesis: a logical perspective. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 331–337, 1981.

[23] E. D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, pages 206–214, Tbilisi, Georgia, USSR, September 1975.

[24] G. Sussman, T. Winograd, and E. Charniak. Micro-planner reference manual. AI Memo 203, AI Lab, MIT, 1970.

[25] A. Tate. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 888–889, Cambridge, MA, 1977. MIT.

[26] S.A. Vere. Planning in time: Windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):246–267, 1983.

[27] D.E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, San Mateo, CA, 1988.

[28] Q. Yang. A theory a conflict resolution in planning. *Artificial Intelligence 58*, pages 361–392, 1992.