Published in in *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambery, France, August 1993, 1436–1441

Plan Modification versus Plan Generation: A Complexity-Theoretic Perspective*

Bernhard Nebel **Jana Koehler**
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
e-mail: {nebel|koehler}@dfki.uni-sb.de

Abstract

The ability of a planner to modify a plan is considered as a valuable tool for improving efficiency of planning by avoiding the repetition of the same planning effort. From a computational complexity point of view, however, it is by no means obvious that modifying a plan is computationally as easy as planning from scratch if the modification has to follow the principle of "conservatism," i.e., to reuse as much of the old plan as possible. Indeed, considering propositional STRIPS planning, it turns out that conservative plan modification is as hard as planning and can sometimes be harder than plan generation. Furthermore, this holds even if we consider modification problems where the old and the new goal specification are similar. We put these results into perspective and discuss the relationship to existing plan modification systems.

1 Introduction

Plan generation in complex domains is normally a resource and time consuming process. One way to improve the efficiency of planning systems is to avoid the repetition of planning effort whenever possible. For instance, in situations when the goal specification is changed during plan execution or when execution time failures happen, it seems more reasonable to *modify* the existing plan than to plan from scratch again. In the extreme, one might go as far as basing the entire planning process on plan modification, a method that could be called *planning from second principles.*

Instead of generating a plan from scratch, that method tries to exploit knowledge stored in previously generated plans. The current problem instance is used to find a plan in a plan library that—perhaps after some modifications—can be used to solve the problem instance at hand. Current approaches try to integrate methods from analogical or case-based reasoning to achieve a higher efficiency [Hammond, 1990; Veloso, 1992], integrate domain-dependent heuristics or investigate reuse in the general context of deductive planning [Koehler, 1992; Bauer *et al.*, 1993].

Some experiments give evidence that planning based on second principles might indeed be more efficient than planning from scratch [Kambhampati and Hendler, 1992; Veloso, 1992; Hanks and Weld, 1992]. However, it is by no means clear to what extent these results generalize. In fact, it is not obvious that *modifying* an existing plan is computationally as easy as *generating* one from scratch, in particular, if we adopt the principle of *conservatism* [Kambhampati and Hendler, 1992], that is to try to recycle "as much of the old solution as possible" [Veloso, 1992, p. 133] or to "produce a plan ... by minimally modifying [the original plan]" [Kambhampati and Hendler, 1992, p. 196].

Considering, for instance, the revision of logical theories, most revision schemata turn out to be computationally harder than deduction [Nebel, 1991; Eiter and Gottlob, 1992].A similar result holds for abduction [Eiter and Gottlob, 1993], which may be viewed as "modifying the assumptions in a proof." Hence, it seems worthwhile to have a closer look at the computational nature of the process of modifying a plan in order to find out why and under which circumstances plan modification and reuse promises to be more efficient than planning from scratch.

The computational complexity of different forms of planning has been recently analyzed by a number of authors [Chapman, 1987; Bäckström and Klein, 1991; Bylander, 1991; Chenoweth, 1991; Gupta and Nau, 1991; Bylander, 1992; Erol *et al.*, 1992]. However, the computational complexity of plan modification has not been investigated yet. We will analyze this problem in the formal framework of *propositional STRIPS planning* as defined by Bylander [1991; 1992]. As Bylander [1991] notes, this model of planning is "impoverished compared to working planners" and is only intended to be a "tool for theoretical analysis." However, since we are mainly interested in *comparing* plan generation with plan modification from a *complexity-theoretic perspective*, this framework is appropriate for our purposes.

As it turns out, modifying a plan is not easier than planning from scratch. On the positive side, we show that modification does not add any complexity to planning if we consider the *general case*. However, there ex-

*This work was supported by the German Ministry for Research and Technology (BMFT) under contracts ITW 8901 8 and ITW 9000 8 as part of the WIP project and the PHI project.

ist special cases when modifying a plan *conservatively*, i.e., *by using as much of the old plan as possible*, can be harder than creating one from scratch, as we will show. This means that plan modification *is not uniformly as easy as plan generation*. Further, we show that these results also hold if we assume that the old and the new planning situation are similar.

Putting these results into perspective and relating them to practical approaches reveals that these approaches do not address the plan modification problem at all, although some authors claim otherwise.

2 Propositional STRIPS Planning

Like Bylander [1991], we define an instance of propositional planning as a tuple $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, where:

- $\mathcal{P}$ is a finite set of ground atomic formulae, the *conditions*,
- $\mathcal{O}$ is a finite set of *operators*, where each operator $o \in \mathcal{O}$ has the form $o^+, o^- \Rightarrow o_+, o_-$, where
  - $o^+ \subseteq \mathcal{P}$ are the *positive preconditions*,
  - $o^- \subseteq \mathcal{P}$ are the *negative preconditions*,
  - $o_+ \subseteq \mathcal{P}$ are the *positive postconditions* (add list), and
  - $o_- \subseteq \mathcal{P}$ are the *negative postconditions* (delete list).
- $\mathcal{I} \subseteq \mathcal{P}$ is the *initial state*, and
- $\mathcal{G} = \langle \mathcal{G}_+, \mathcal{G}_- \rangle$ is the *goal specification* with $\mathcal{G}_+ \subseteq \mathcal{P}$ the positive goals and $\mathcal{G}_- \subseteq \mathcal{P}$ the negative goals.

$\mathcal{P}$ is the set of relevant conditions. A *state* is a subset $S \subseteq \mathcal{P}$ with the intended meaning that $p \in \mathcal{P}$ is true in state S if $p \in S$, false otherwise. $\mathcal{O}$ is the set of *operators* that can change states. $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is the goal state specification, with the intended meaning that all conditions $p \in \mathcal{G}_+$ must be true and all conditions $p \in \mathcal{G}_-$ must be false. A plan Δ is a finite sequence $\langle o_1, \ldots, o_n \rangle$ of *plan steps* $o_i \in \mathcal{O}$. An operator may occur more than once in a plan. A plan Δ *solves* an instance Π of the planning problem iff the *result* of the application of Δ to $\mathcal{I}$ leads to a state S that satisfies the goal specification $\mathcal{G}$, where the result of applying $\Delta = \langle o_1, \ldots, o_n \rangle$ to a state S is defined by the following function:

$$\mathit{Result}(S, \langle\rangle) = S$$

$$\mathit{Result}(S, \langle o \rangle) = \begin{cases} (S \cup o_+) - o_- & \text{if} \quad o^+ \subseteq S \wedge \\ & \quad\ o^- \cap S = \emptyset \\ \bot & \text{otherwise} \end{cases}$$

$$\mathit{Result}(S, \langle o_1, o_2, \ldots, o_n \rangle) = \mathit{Result}(\mathit{Result}(S, \langle o_1 \rangle), \langle o_2, \ldots, o_n \rangle).$$

In other words, if the precondition of an operator is satisfied by a state, the positive postconditions are added and the negative postconditions are deleted. Otherwise, the state becomes *undefined*, denoted by $\bot \notin \mathcal{P}$.[1]

As usual, we consider *decision problems* in order to analyze the computational complexity of planning.[2] PLANSAT is defined to be the *decision problem* of determining whether an instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ of propositional STRIPS planning has a solution, i.e., whether there exists a plan Δ such that $\mathit{Result}(\mathcal{I}, \Delta)$ satisfies the goal specification. PLANMIN [Bylander, 1993] is defined to be the problem of determining whether there exists a solution of length n or less, i.e., it is the decision problem corresponding to the *search problem* of generating plans with minimal length.

Based on this framework, Bylander [1991; 1992; 1993] analyzed the computational complexity of the general propositional planning problem and a number of generalizations and restricted problems. In its most general form, both PLANSAT and PLANMIN are PSPACE-complete. Severe restrictions on the form of the operators are necessary to guarantee polynomial time or even NP-completeness.

3 Plan Modification in a Propositional Framework

Kambhampati and Hendler [1992] define the *plan modification problem* as follows (adapted to our framework of propositional STRIPS planning):

> Given an instance of the planning problem $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$ and a plan Δ that solves the instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, produce a plan Δ' that solves Π' by *minimally modifying* Δ.

We will call this problem MODGEN.

By "minimal modification of a plan" Kambhampati and Hendler [1992] mean to "salvage as much of the old plan as possible." Of course, the part of the old plan that has been salvaged should be *executable*, i.e., the preconditions of all operators should be satisfied. In order to guarantee this, we require that all operators are executable (see the definition of the function *Result*).

Turning the above specified search problem into a decision problem leads to what we will call the MODSAT problem:

> An instance of the MODSAT problem is given by $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$, a plan Δ that solves $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$, and an integer $k \leq |\Delta|$. The question is whether there exists a plan Δ' that solves Π' and *contains a subplan of* Δ *of at least length* k?

In order to fully specify MODSAT, we have to define the meaning of the phrase "Δ' contains a subplan of Δ of length k." For this purpose, we define the notion of a *plan skeleton*, a sequence of operators and "wildcards," denoted by "$*$." The length of a plan skeleton is the number of operators, i.e., we ignore the wildcards. A plan skeleton can be *derived* from a plan according to a *modification strategy* $\mathcal{M}$ by deleting and rearranging plan steps and adding wildcards. A plan skeleton can be *extended* to a plan by replacing each wildcard by a possibly

[1] This is a slight deviation from Bylander's [1991] definition that does not affect the complexity of planning. This deviation is necessary, however, to allow for a meaningful definition of the plan modification problem.

[2] We assume that the reader is familiar with the basic notions of complexity theory as presented, for instance, in [Garey and Johnson, 1979].

empty sequence of operators. Now we say that *plan Δ' contains a subplan of Δ of length k according to a modification strategy $\mathcal{M}$* iff a skeleton Γ of length k can be derived from Δ according to $\mathcal{M}$ and Γ can be extended to Δ'. In general, we will consider only *polynomial-time* modification strategies, i.e., strategies such that verifying that the skeleton Γ can be derived from the plan Δ is a polynomial-time problem. In the following, we will consider two different plan modification strategies that satisfy this constraint.

The first alternative we consider is to allow for deletions in the original plan and additions before and after the original plan. Supposing the plan

$$\Delta = \langle o_1, \ldots, o_i, o_{i+1}, \ldots, o_{j-1}, o_j, \ldots, o_n \rangle,$$

the following plan skeleton could be derived from Δ, for instance:

$$\Gamma = \langle *, o_1, \ldots, o_i, o_j, \ldots, o_n, * \rangle,$$

where Γ has length $i + n - j + 1$. The corresponding modification problem will be called MODDEL.

Another alternative is to allow for deletion of plan steps in the old plan and additions before, after, and in the middle of the old plan. Assuming the same plan Δ as above, the following skeleton plan of length $i+n-j+1$ could be derived:

$$\Gamma = \langle *, o_1, \ldots, o_i, *, o_j, \ldots, o_n, * \rangle.$$

The corresponding modification problem is called MODDELINS.

Finally, it should be noted that although the framework we have defined above deals only with linear plans, it can be easily modified to apply to nonlinear planning, as well. In particular, all hardness results will apply directly to nonlinear planning since linear plans are simply a special case of nonlinear ones.

4 The Complexity of Plan Modification

One almost immediate consequence of the definitions above is that plan modification cannot be easier than plan generation. This even holds for all restrictions of the PLANSAT problem. If PLANSAT$_\rho$ is a restricted planning problem, then MODSAT$_\rho$ shall denote the corresponding modification problem with the same restrictions.

Proposition 1 PLANSAT$_\rho$ *transforms polynomially to* MODSAT$_\rho$ *for all restrictions* ρ.[3]

However, plan modification is also not harder than plan generation in the general case.

Proposition 2 MODSAT *is* PSPACE*-complete.*

This proposition could be taken as evidence that plan modification is not harder than plan generation. However, it should be noted that the proposition is only about the general problem. So, it may be the case that there exist special cases such that plan modification is harder than generation. Such a case will not be found among the PSPACE- and NP-complete planning problems, however.

[3]Full proofs of propositions and theorems can be found in the full paper [Nebel and Koehler, 1992].

Theorem 3 *If* PLANSAT$_\rho$ *is a restricted planning problem that is* PSPACE*-complete or* NP*-complete, then* MODSAT$_\rho$ *is* PSPACE*-complete or* NP*-complete problem, respectively.*

Proof. PSPACE-hardness and NP-hardness, respectively, are obvious because of Proposition 1. Membership follows in case of PSPACE by Proposition 2. In case of NP, we initially guess (1) n ($0 \leq n \leq |\Delta| + 2$) possibly empty plans Δ_i such that $|\Delta_i| \leq |\Delta|$, (2) $2n$ states $S_1, \ldots, S_{2n}$, and (3) n polynomially bounded proofs that there exists plans from each state S_{2i} to state S_{2i+1} for $1 \leq i \leq n-1$. Since PLANSAT$_\rho$ is in NP, such proofs exist (in most cases, these proofs will be plans). Then we verify in polynomial time (1) that $S_1 = \mathcal{I}$ and S_{2n} satisfies the goal specification $\mathcal{G}$, (2) that $Result(S_{2i-1}, \Delta_i) = S_{2i}$, (3) that the plan existence proofs are correct, and (4) that $\langle \Delta_1, *, \Delta_2, *, \ldots, \Delta_{n-1}, *, \Delta_n \rangle$ is a skeleton of length k that can be derived from Δ. This is obviously a nondeterministic algorithm that runs in polynomial time. ∎

The converse of the above theorem does not hold, however. There exist cases when plan generation is a polynomial time problem while plan modification is NP-complete.

Theorem 4 *There exists a polynomial-time* PLANSAT$_\rho$ *problem such that the corresponding* MODDEL$_\rho$ *and* MODDELINS$_\rho$ *problems are* NP*-complete.*

Proof. The planning problem PLANSAT$_1^+$ defined by restricting operators to have only positive preconditions and only one postcondition can be solved in polynomial time [Bylander, 1991, Theorem 7]. Let PLANSAT$_1^{+,\overline{post}}$ be the planning problem defined by restricting operators to have (1) only one postcondition p, (2) the negated condition $\overline{p}$ as a precondition, and (3) any number of additional positive preconditions. From the specification of the algorithm Bylander [1991] gives for PLANSAT$_1^+$, it is evident that PLANSAT$_1^{+,\overline{post}}$ can also be solved in polynomial time. We will show that the corresponding modification problems MODDEL$_1^{+,\overline{post}}$ and MODDELINS$_1^{+,\overline{post}}$ are NP-complete.

For the hardness part we use a reduction from SAT, the problem of satisfying a boolean formula in conjunctive normal form. Let $V = \{v_1, \ldots, v_m\}$ be the set of boolean variables and let $C = \{c_1, \ldots, c_n\}$ be the set of clauses. Now we construct a MODDEL$_1^{+,\overline{post}}$ problem that can be satisfied iff there exists a satisfying truth assignment for the SAT problem.

The set of conditions $\mathcal{P}$ contains the following ground atoms:

T_i,	$1 \leq i \leq m$,	v_i = true has been selected
F_i,	$1 \leq i \leq m$,	v_i = false has been selected
S_i,	$1 \leq i \leq m$,	the truth value for v_i has been selected
E_i,	$0 \leq i \leq m$,	enable evaluation
C_j,	$1 \leq n \leq n$,	c_j evaluates to true.

Further, we assume the following set of operators $\mathcal{O}$:

		o^+,	o^-	$\Rightarrow$	o_+,	o_-
t_i	$\equiv$	$\{T_i\}$,	$\emptyset$	$\Rightarrow$	$\emptyset$,	$\{T_i\}$
f_i	$\equiv$	$\{F_i\}$,	$\emptyset$	$\Rightarrow$	$\emptyset$,	$\{F_i\}$
st_i	$\equiv$	$\{T_i, E_0, \ldots, E_m\}$,	$\{S_i\}$	$\Rightarrow$	$\{S_i\}$,	$\emptyset$
sf_i	$\equiv$	$\{F_i, E_0, \ldots, E_m\}$,	$\{S_i\}$	$\Rightarrow$	$\{S_i\}$,	$\emptyset$
e_i	$\equiv$	$\emptyset$,	$\{E_i\}$	$\Rightarrow$	$\{E_i\}$,	$\emptyset$
$pos_{i,j}$	$\equiv$	$\{T_i, E_0, \ldots, E_m\}$, if $v_i \in c_j$	$\{C_j\}$	$\Rightarrow$	$\{C_j\}$,	$\emptyset$
$neg_{i,j}$	$\equiv$	$\{F_i, E_0, \ldots, E_m\}$, if $\overline{v_i} \in c_j$.	$\{C_j\}$	$\Rightarrow$	$\{C_j\}$,	$\emptyset$

Assume the following initial and goal state:

$$\begin{aligned} \mathcal{I} &= \{T_1, \ldots, T_m, F_1, \ldots, F_m\} \\ \mathcal{G}_+ &= \{E_0, \ldots, E_m\} \\ \mathcal{G}_- &= \{T_1, \ldots, T_m, F_1, \ldots, F_m\}. \end{aligned}$$

The instance $\Pi = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ is, for example, solved by the following plan Δ:

$$\Delta = \langle t_1, \ldots, t_m, f_1, \ldots, f_m, e_0, \ldots, e_m \rangle.$$

Now consider the instance $\Pi' = \langle \mathcal{P}, \mathcal{O}, \mathcal{I}', \mathcal{G}' \rangle$ such that

$$\begin{aligned} \mathcal{I}' &= \mathcal{I} \\ \mathcal{G}'_+ &= \{E_0, \ldots, E_m, S_1, \ldots, S_m, C_1, \ldots, C_n\} \\ \mathcal{G}'_- &= \emptyset. \end{aligned}$$

It is obvious that the SAT formula is satisfiable if, and only if, the plan Δ can be modified by deleting at most m operators and adding some operators before and after the original plan Δ in order to achieve a new plan Δ' that solves Π'.

Membership in NP follows since $\text{PLANSAT}_1^{+,\overline{post}}$ is in NP. Using the same algorithm as described in the proof of Theorem 3 leads to a nondeterministic polynomial-time algorithm for $\text{MODDEL}_1^{+,\overline{post}}$ and $\text{MODDELINS}_1^{+,\overline{post}}$. ∎

5 Modifying Plans When the Situations are Similar

The results above could be considered as being not relevant for plan modification in real applications because we made no assumption about the similarity between old and new planning situation. The efficiency gains expected from plan reuse, on the other hand, are based on the assumption that the new situation is *sufficiently close* to the old one—which supposedly permits an easy adaptation of the old plan to the new situation. Besides the fact that this looks like a good heuristic guidance, there is the question whether small differences between the old and the new situation lead to a provable efficiency gain in terms of computational complexity. So it might be perhaps the case that modification is easier than planning if the goal specifications differ only on a constant or logarithmic number of atoms. Although this seems to be possible, there is the conflicting intuition that small changes in the planning situations could lead to drastic (and hard to compute) changes in the plans.

As it turns out, restricting the number of differing atoms does not lead to a different picture than the one presented in the previous section. First of all, Theorem 4 still holds for the restricted versions of the modification problems MODDEL and MODDELINS, where we require the old and new initial states to be identical and the old and new goal specification to differ only on one atom. We call these restricted versions of the modification problems MODDEL1G and MODDELINS1G, respectively.

Theorem 5 *There exists a polynomial-time* PLANSAT_ρ *problem such that the corresponding* MODDEL1G_ρ *and* MODDELINS1G_ρ *problems are* NP*-complete.*

Proof. The transformation used in the proof of Theorem 4 is modified as follows. A new atom B is added, which is assumed to be false in the initial state $\mathcal{I}$ and not mentioned in the old goal specification $\mathcal{G}$. The new goal specification $\mathcal{G}'$ is:

$$\begin{aligned} \mathcal{G}'_+ &= \mathcal{G}_+ \cup \{B\} \\ \mathcal{G}'_- &= \mathcal{G}_-. \end{aligned}$$

Finally, the following operator is added:
$\{E_0, \ldots, E_m, S_1, \ldots, S_m, C_1, \ldots, C_n\}, \{B\} \Rightarrow \emptyset, \{B\}$ ∎

Although this theorem confirms the intuition that small changes in the goal specification can lead to drastic changes in the plan, it does not rule out the possibility that there are some hard planning problems such that the corresponding modification problems are easy if the goal specification is only changed marginally. In order to rule out this possibility, we would need something similar to Proposition 1. Since there appears to be no general way to reduce PLANSAT_ρ problems to MODSAT1G_ρ problems, we will settle for something slightly less general. We will show that *generating* a plan by modifying a plan for a similar goal specification is at least as hard as the corresponding PLANSAT problem. Hence, instead of the decision problem MODSAT1G, we consider the search problem MODGEN1G. Further, in order to allow for a "fair" comparison between PLANSAT and MODGEN1G, we measure the resource restrictions of MODGEN1G in terms of the size of the planning problem instance—and ignore the size of the old problem.[4] Under these assumptions, it is possible to specify a Turing reduction from PLANSAT_ρ to MODGEN1G_ρ.

Theorem 6 *If* PLANSAT_ρ *is a restricted planning problem that is* PSPACE*-hard or* NP*-hard, then the corresponding* MODGEN1G_ρ *problem is* PSPACE*-hard or* NP*-hard, respectively.*

Proof. Using an algorithm for MODGEN1G_ρ, we can *generate a plan* by modifying it iteratively, starting with the empty plan and empty goal specification and continuing by adding step by step one goal atom. Since the size of the goal specification is linearly bounded by the problem instance, we would need only linearly many calls. Supposing that the theorem does not hold would imply

[4] This is necessary to rule out such pathological situations as the one where modifying an *exponentially* long plan appears to be polynomial while generating it is exponential.

that generating a plan under restrictions ρ is easier than PLANSAT$_\rho$, which is impossible by definition. ∎

It should be noted that we did not rely on any particular property of the MODGEN1G$_\rho$ algorithm. In particular, we did not make the assumption that the algorithm has to recycle a maximal reusable plan skeleton. Furthermore, the above theorems apply, of course, also to the modification problems that are restricted to have an one-atom-difference between the initial states.

6 Discussion

Of course, there arises the question of how the above results relate to practical plan modification systems. Kambhampati and Hendler [1992] investigate plan reuse and modification in the framework of the hierarchical planner and modification system PRIAR, which is based on NONLIN [Tate, 1977]. They use a large number of blocks-world examples in order to evaluate the relative efficiency gains provided by plan modification compared with planning from scratch. The average savings of runtime when plans were reused is given by the authors as 79%.

Hanks and Weld [1992] performed experiments on reusing blocks world plans with their system SPA. This plan generation and modification system is based on a lifted version of McAllester's and Rosenblitt's [1991] systematic nonlinear planning algorithm. In case of the SPA system, the savings turned out to be less drastic than in the PRIAR system. In fact, in the SPA system plan modification can be more expensive than plan generation in terms of runtime if the reuse candidate is not close enough [Hanks and Weld, 1992, p. 103], a situation that did not happen with similar input data in the PRIAR system.

While the relative savings appear to be different for the two approaches, in both cases there is a positive effect which increases when the difference between the new and the old situations decreases. Although this seems to run counter to our complexity results (in particular Theorem 6), these empirical findings do not contradict our results because the experiments were clearly not designed to explore worst-case situations, which complexity analysis is about. An interesting avenue of research would be to characterize the form of planning problems that can exploit plan-reuse techniques to improve the efficiency of the planning process.

What seems to be less easily explainable is, however, the discrepancy between the hope that reusing maximal subplans increases the efficiency of plan reuse and our findings. Our results imply that *conservative* plan modification introduces some combinatorics into the planning and reuse process. In particular, as a Corollary of Proposition 2 it follows that is not possible to determine efficiently (i.e., in polynomial time) a maximal reusable plan skeleton *before* plan generation starts to extend the skeleton.

Corollary 7 *It is* PSPACE*-hard to compute a maximal plan skeleton for* MODSAT *instances.*

In other words, plan generation and plan modification cannot be separated. For this reason, the planning process becomes actually more involved when recycling as much of the old plan as possible. Instead of searching for an arbitrary solution, a plan that contains a maximal subplan of the old plan has to be sought.

Kambhampati and Hendler [1992] mention *conservatism*, i.e., to "salvage as much of the old plan as possible," as an "important desideratum" for a plan modification capability, in order to "ensure efficiency." At a first glance, this seems to be indeed reasonable since it promises to minimize the additional planning effort. As we have seen, however, finding the maximal reusable plan skeleton is already as difficult as planning and is sometimes even more difficult than the corresponding planning problem (Theorem 4). Hence, "conservatism" seems to run counter to increasing planning efficiency.

Having a closer look at the PRIAR framework reveals that plan skeletons are derived in *polynomial time* [Kambhampati and Hendler, 1992, p. 197] by a process called "annotation verification." Hence, by Corollary 7, this process cannot by any means derive maximal applicable plan skeletons. Further, the authors do not give any arguments that they approximate such skeletons. In fact, the skeletons derived by PRIAR are not even guaranteed to be applicable. So, PRIAR does not seem to address the problem of "minimally modifying plans," contrary to what the authors claim.

In fact, *maximal* reuse of an old plan only seems to make sense in a replanning context if costs are charged for *not executing already planned steps*. So, it seems to be the case that the two motivations for plan modification, namely, *replanning* and *reuse* may not be as similar as one might think. While in plan reuse the *efficiency* of the planning process is the most important factor, in *replanning* the minimal disturbance of the old plan may be more important, leading to a more involved planning process.[5]

Plan modification in the PRIAR framework—and in other plan-reuse systems—seems not to be a *computational problem* that has to be addressed, but rather a *solution*, a heuristic technique. The "plan skeleton" that is reused is not the maximal applicable one, but the one that *the particular planning algorithm perhaps can exploit in generating a solution.* In other words, the old plan is used as an "entry point" into the search space of possible plans, as made explicit by Hanks and Weld [1992].

7 Conclusion

Improving the efficiency of planning systems by adding capabilities to modify existing plans has received some research interest recently. In analyzing the computational complexity of this problem, we showed that it is

[5] Kambhampati makes the same distinction in a later paper [Kambhampati, 1992]. Based on arguments concerning the search process of a planner, he also argues that *guaranteeing* that every step that could be reused is reused could be computationally expensive—a conjecture confirmed by Theorem 4.

as hard as planning and *sometimes modification is even harder than planning from scratch*. We showed also that these results hold under the restriction that the modification process has to account for only one changed atom in the goal specification. In particular, we showed that deriving the maximal reusable subplan is not easier than planning. Hence, we cannot hope for minimizing planning effort by first identifying the maximal applicable subplan which is then (minimally) extended by plan generation.

Relating these results to existing plan reuse and modification systems, it turns out that these do not address the modification problem at all, although some authors claim otherwise. In fact, in plan-reuse systems, plan modification is not attacked as a problem but considered as a heuristic technique. This means that instead of "using as much of the old plan as possible" these systems recycle "as much of the old plan as the particular planning algorithm will perhaps be able to use in solving the new problem instance." In fact, adopting the principle of *conservatism* in plan modification only seems to make sense in a replanning context where one wants to minimize the perturbation of the original plan.

Acknowledgements

We would like to thank Christer Bäckström, Tom Bylander, Subbarao Kambhampati, and the anonymous referees, who provided helpful comments on an earlier version of this paper. In particular, Tom's remarks and questions heavily influenced the paper.

References

[Bäckström and Klein, 1991] C. Bäckström and I. Klein. Parallel non-binary planning in polynomial time. In *Proc. 12th Int'l Joint Conf. on Artif. Intell.*, pages 268–273, 1991.

[Bauer *et al.*, 1993] M. Bauer, S. Biundo, D. Dengler, J. Koehler, and G. Paul. Phi—a logic-based tool for intelligent help systems. 1993. These proceedings.

[Bylander, 1991] T. Bylander. Complexity results for planning. In *Proc. 12th Int'l Joint Conf. on Artif. Intell.*, pages 274–279, 1991.

[Bylander, 1992] T. Bylander. Complexity results for extended planning. In *Proc. of the 1st Int'l Conf. on Artif. Intell. Plan. Sys.*, 1992.

[Bylander, 1993] T. Bylander. The computational complexity of propositional STRIPS planning. *Artif. Intell.*, 1993. To appear.

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

[Chenoweth, 1991] S. V. Chenoweth. On the NP-hardness of blocks world. In *Proc. of the 9th Nat'l Conf. on Artif. Intell.*, pages 623–628, 1991.

[Eiter and Gottlob, 1992] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell.*, 57:227–270, 1992.

[Eiter and Gottlob, 1993] T. Eiter and G. Gottlob. The complexity of logic-based abduction. In *Proc. 10th Symp. on Theo. Asp. of Comp. Sci.*, 1993. To appear.

[Erol *et al.*, 1992] K. Erol, D. S. Nau, and V. S. Subrahmanian. On the complexity of domain-independent planning. In *Proc. 10th Nat'l Conf. on Artif. Intell.*, pages 381–386, 1992.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[Gupta and Nau, 1991] N. Gupta and D. S. Nau. Complexity results for blocks-world planning. In *Proc. of the 9th Nat'l Conf. on Artif. Intell.*, pages 629–633, 1991.

[Hammond, 1990] K. J. Hammond. Explaining and repairing plans that fail. *Artif. Intell.*, 45:173–228, 1990.

[Hanks and Weld, 1992] S. Hanks and D. S. Weld. Systematic adaptation for case-based planning. In *Proc. 1st Int'l Conf. on Artif. Intell. Plan. Sys.*, pages 96–105, 1992.

[Kambhampati and Hendler, 1992] S. Kambhampati and J. A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artif. Intell.*, 55:193–258, 1992.

[Kambhampati, 1992] S. Kambhampati. Utility tradeoffs in incremental plan modification and reuse. In *AAAI Spring Symp. on Comp. Consid. in Supporting Incr. Mod. and Reuse*, pages 36–41, 1992.

[Koehler, 1992] J. Koehler. Towards a logical treatment of plan reuse. In *Proc. of the 1st Int'l Conf. on Artif. Intell. Plan. Sys.*, pages 285–286, 1992.

[McAllester and Rosenblitt, 1991] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proc. 9th Nat'l Conf. on Artif. Intell.*, pages 634–639, 1991.

[Nebel and Koehler, 1992] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. Research Report RR-92-48, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany, 1992.

[Nebel, 1991] B. Nebel. Belief revision and default reasoning: Syntax-based approaches. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. 2nd Int'l Conf.*, pages 417–428, 1991.

[Tate, 1977] A. Tate. Generating project networks. In *Proc. of the 5th Int'l Joint Conf. on Artif. Intell.*, pages 888–893, 1977.

[Veloso, 1992] M. M. Veloso. Automatic storage, retrieval, and replay of multiple cases using derivational analogy in PRODIGY. In *AAAI Spring Symp. on Comp. Consid. in Supporting Incr. Mod. and Reuse, Working Notes*, pages 131–136, 1992.