Enabling-Condition Interactions and Finding Good Plans*

Dana S. Nau†
University of Maryland
College Park, Maryland 20742
nau@cs.umd.edu

Overview

In AI planning research, the best-known goal interaction is the deleted-condition interaction, in which the side-effect of achieving one condition is to delete some other condition that will be needed later. Enabling-condition interactions—in which the side-effect of achieving one condition is to make it easier to achieve some other condition—are not as well known. In this paper, I argue that enabling-condition interactions merit more attention than they have heretofore received.

This paper is organized as follows:

1. A definition of enabling-condition interactions.
2. The importance of finding “good” plans (rather than simply being satisfied with any plan we find), with examples from several planning domains.
3. How a number of planning strategies take advantage of enabling-condition interactions to produce better plans.
4. Some of the effects of enabling-condition interactions on the complexity of planning.
5. Concluding remarks.

Definition

An enabling-condition interaction is a situation in which some action invoked to achieve one goal $G_1$ also makes it easier to achieve another goal $G_2$. For example, in Figure 1, the action move(a, c, b) achieves the goal on(a, b), but it also has the side-effect of clearing c, making it easier to achieve the goal on(c, d). As another example, consider the following situation (based on (Wilensky, 1983)):

John lives two miles from a bakery and two miles from a dairy. The two stores are one mile apart. John has two goals: to buy bread and to buy milk.

If John goes to the bakery to buy bread, then this puts him closer to the dairy, making it easier for him to buy milk.

Finding “Good” Plans

Perhaps the most obvious case in which enabling-condition interactions affect planning is if we want to find a “good” plan (i.e., one whose length or cost is small, or whose efficiency is high) rather than being satisfied with any plan (no matter how inefficient) that achieves the goal.1 Until recently, the objective of finding “good” or “optimal” plans does not appear to have received much explicit discussion in the AI planning literature. However, it appears to have been an underlying motivation behind a number of existing planning strategies.

For example, Fig. 2 shows the well known “Sussman anomaly” of blocks-world planning (Waldinger, 1990, p. 127) (Sussman, 1975). The primary reason why this problem generated so much interest was that the best plan that previously existing planning procedures could generate for this problem was

\[
\text{move(b, TABLE, c), move(b, c, TABLE), move(c, a, TABLE), move(b, TABLE, c), move(a, TABLE, b).}
\]

1Actually, enabling-condition interactions affect planning even if we are not concerned with the goodness of a plan. I hope to discuss this more fully in a subsequent paper.
Consider the problem of finding a schedule for satisfying some set of orders for products that can be produced in a machine shop. For each order, there may be a set of alternative schedules for producing it, and each such schedule consists of a set of operations to be performed on various machines. An operation in a schedule is usually associated with a machine for carrying it out. If two or more operations require the same type of set-up, then doing them on the same machine may reduce the total time required—and thus reduce the total time required to complete all the schedules.

Multiple Database-Query Optimization

Let \( Q = \{Q_1, Q_2, \ldots, Q_n\} \) be a set of database queries. Associated with each query \( Q_i \) is a set of alternate access plans \( \{P_{i1}, P_{i2}, \ldots, P_{ik}\} \). Each plan is a set of partially ordered tasks that produces the answer to \( Q \). For example, one task might be to find all employees in some department whose ages are less than 30, and whose salaries are over \$50,000. Each task has a cost, and the cost of a plan is the sum of the costs of its tasks. Enabling-condition interactions occur if plans for two different tasks contain the same query, or if the result of evaluating one task reduces the cost of evaluating the other. A better overall plan can be produced by taking advantage of these interactions, and several research papers have been written on this topic (Sellis, 1988; Shim et al., 1991).

Planning Strategies

A number of planning strategies have been formulated to exploit enabling-condition interactions in order to produce better plans. Below are two examples.

Studies of human planning behavior show that people look for enabling-condition interactions when they are formulating plans, in order to make the plans more efficient. For example, consider the following excerpt from Hayes-Roth and Hayes-Roth's transcript of someone "thinking aloud" while planning a hypothetical day's errands (Hayes-Roth and Hayes-Roth, 1990, p. 254):

In section 6, the subject asks, "What is going to be the closest one?" This question indicates a strategic decision to plan to perform the closest errand next in the procedural sequence ... This planning strategy is one that Pollack (Pollack, 1992) has referred to as overloading: if you notice that an action you have selected in order to accomplish one goal also makes it easier to accomplish another goal, then use that action for both of these purposes.

The same basic idea has been incorporated into nonlinear planning systems, where it is sometimes called phantomization: SIPE (Wilkins, 1988), Nonlin (Tate, 1977), and Kambhampati's plan reuse framework (Kambhampati, 1990) are capable of recognizing an operator in the current plan satisfies another goal in addition to the one it was originally intended to achieve, and imposing constraints on the plan so that this operator will be used to achieve both goals.
A different but related strategy is action merging (Yang et al., 1990; Foulser et al., 1992; Yang et al., 1993). In this case, rather than simply recognizing that an existing action accomplishes an additional goal, the planner replaces an action that doesn't accomplish an additional goal with one that does (provided that this produces a lower total cost than would be incurred by using a separate action to achieve the additional goal). NOAH's "eliminate redundant precondition" critic (Sacerdoti, 1977) is a special case of this strategy.

Effects on Complexity
If a problem contains more than one enabling-condition interaction, then it can be difficult to determine which set of actions will produce the best plan. For example, if actions A and B both achieve goal G₁, and A also aids in achieving goal G₂, then we might prefer action A to action B—but if B also aids in achieving goal G₃, then it may no longer be clear which of A and B we should prefer. The difficulty of resolving such tradeoffs is illustrated in some of the repeated revisions that Hayes-Roth and Hayes-Roth's subject makes to his plan (Hayes-Roth and Hayes-Roth, 1990, p. 246-247). Below, I discuss several other cases where enabling-condition interactions increase the difficulty of planning.

Blocks World (The Usual Formulation)
The effect of enabling-condition interactions on blocks-world planning was analyzed formally in (Gupta and Nau, 1992). This paper showed that in the blocks world, finding an optimal plan is NP-hard—and that the NP-hardness is not due to deleted-condition interactions such as the Sussman anomaly, but instead due to a particular kind of enabling-condition interaction called a deadlock. For problems that do not contain deadlocks, there is a simple hill-climbing strategy that can easily find an optimal plan, regardless of whether or not the problem contains any deleted-condition interactions.

To see that deadlocks are a special case of enabling-condition interactions, consider the definition of a deadlock given in (Gupta and Nau, 1992). The set of blocks \{b₁, b₂, ..., bₚ\} is deadlocked in the state S if there is a set of blocks \{d₁, d₂, ..., dₚ\} such that the following three conditions hold:

1. In S, bᵢ is above dᵢ for i = 1, 2, ..., p.
2. In G, bᵢ is above dᵢ₊₁ for i = 1, 2, ..., p - 1, and bₚ is above d₁.
3. In S, none of b₁, b₂, ..., bₚ are in their final positions (if p > 1, then the other two conditions entail this condition).

For example, in Figure 3, in the initial state I there are two deadlocked sets of blocks:

1. In I, a is above c and d is above e. In G, a is above e and d is above c. Thus \{a, d\} is deadlocked.
2. a is above b in both I and G, and a is not in its final position in I. Thus \{a\} is deadlocked.

Figure 3: A problem in which two sets of blocks are deadlocked: \{a, d\} and \{a\}.
The above problem is similar to the problem of resolving multiple deadlocks in the more usual formulation of the blocks world. In both problems, if we make the wrong choice, then too many blocks must be moved temporarily to the table rather than moving them directly to their final positions, whereas the resulting plan will be longer than $L$.

The above problem is similar to the problem of resolving multiple deadlocks in the more usual formulation of the blocks world. In both problems, if we make the wrong choice, then too many blocks must be moved temporarily to the table rather than moving them directly to their final positions. However, the two problems are not identical. If no two blocks have the same name, then for a wrong choice to force us to move extra blocks to the table, we must have blocks which mutually block each others' progress—and this led to our definition of deadlock. But if more than one block can have the same name, then one can find other ways for a wrong choice to force us to move extra blocks to the table—and that is what Chenoweth did.

**Blocks World (Another Formulation)**

(Chenoweth, 1991) describes a more complicated version of blocks world planning, in which more than one block can have the same name. It is an open question how difficult deleted-condition interactions are in Chenoweth's version of the blocks world—but Chenoweth's NP-hardness proof for his domain again shows that enabling-condition interactions make the problem NP-hard.

Chenoweth's proof of NP-hardness is by reduction from 3SAT. Given a 3SAT problem with $m$ clauses and $n$ variables, he generates an MPBW problem in which $L = 3n + 5m + 1$. For each $i$ ($i = 1, \ldots, n$), there are two blocks named $u_i$, at the tops of two large stacks. For each $i$, one of the two $u_i$'s must be moved to the top of a block named $v_i$, and the question is which $u_i$ to move. If we move the wrong one, then later in the plan, we will have to move one or more blocks temporarily to the table rather than moving them directly to their final positions, whence the resulting plan will be longer than $L$.

The above problem is similar to the problem of resolving multiple deadlocks in the more usual formulation of the blocks world. In both problems, if we make the wrong choice, then too many blocks must be moved temporarily to the table rather than directly to their final positions. However, the two problems are not identical. If no two blocks have the same name, then for a wrong choice to force us to move extra blocks to the table, we must have blocks which mutually block each others' progress—and this led to our definition of deadlock. But if more than one block can have the same name, then one can find other ways for a wrong choice to force us to move extra blocks to the table—and that is what Chenoweth did.

**Domain-Independent Planning**

Enabling-condition interactions are important not only in the blocks world, but in domain-independent planning in general. As an example, consider Table 1, which was taken from (Erol et al., 1991; Erol et al., 1992). This table shows how the complexity of domain-independent planning with STRIPS-style operators depends on the nature of those operators.

In lines (11) and (12) of Table 1, plan existence can be determined in polynomial time, but the problem of finding an optimal plan is NP-complete. The reason for this is as follows. In these cases, if we are simply interested in finding a plan but do not care how good a plan it is, the restrictions allow us to plan for each subgoal separately, using backwards chaining. But if we are interested in a short plan (i.e., one of length $\leq k$) rather than just any plan, then we need to pay attention to enabling-condition interactions, because they make it possible to produce a shorter plan. It is not possible to detect and reason about these interactions if we plan for the subgoals independently; instead, we have to consider all possible operator choices and orderings, making it NP-hard to tell whether there is a plan of length $\leq k$.

Note also that throughout Table 1, whether or not negated preconditions are allowed does not affect the complexity of telling whether there is a plan of length $\leq k$. Again, what makes these particular problems difficult is how to handle multiple enabling-condition interactions—more specifically, how to choose opera-
tors that achieve several subgoals in order to minimize the overall length of the plan. For these problems, this task remains equally hard regardless of whether negated preconditions are allowed.

Conclusion

In AI planning research, the best-known goal interaction is the deleted-condition interaction, in which the side-effect of achieving one condition is to delete some other condition that will be needed later. Enabling-condition interactions are not as well known. In this paper, I have argued that enabling-condition interactions merit more attention, for the following reasons:

The goal of finding good plans (rather than simply being satisfied with any plan we find) is important in many planning situations, and has been an implicit motivation behind several existing planning strategies. But if we are interested in finding good plans rather than poor plans, enabling-condition interactions can dramatically increase the difficulty of planning. If this result is correct, then it suggests that enabling-condition interactions are just as important in planning as deleted-condition interactions. I hope to investigate this issue more fully in the future.

In addition, preliminary results suggest that even if we don’t care about the quality of the plan we find, enabling-condition interactions can still dramatically increase the difficulty of planning.

References


Foulser, David; Li, Ming; and Yang, Qiang 1992. Theory and algorithms for plan merging. Artificial Intelligence 57(2-3):143–182.


