A Critical Look at Knoblock's Hierarchy Mechanism

David E. Smith
Rockwell International
444 High St.
Palo Alto, CA 94301
de2smith@rpal.rockwell.com

Mark A. Peot
Rockwell International
444 High St.
Palo Alto, CA 94301
peot@rpal.rockwell.com

Abstract

Recently, Knoblock has advocated a mechanism for automatically constructing planning hierarchies. We show that Knoblock's method, and more generally, the principle of ordered monotonicity, can produce hierarchies that perform arbitrarily poorly. The reason is that Knoblock's technique addresses only one of two important factors in ordering clauses. We propose a technique for the other based on evaluating the number of potential solutions to different possible subgoals in a plan.

1 Knoblock's Method

Recently Knoblock [1, 2] has advocated a mechanism for automatically constructing fixed hierarchies to control planning search. The technique involves constructing a directed graph of potential conflicts between operators relevant to a goal. The graph is then broken into components and sorted to give the resulting hierarchy.

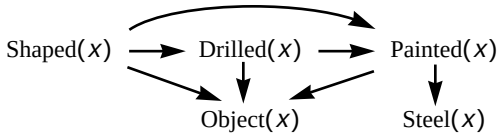
To illustrate Knoblock's approach consider a simplified machine shop example with the following operators for shaping, drilling, and painting an object:

Operator	Shape(x)
Precs:	Object(x)
Effects:	Shaped(x), $\neg$Drilled(x), $\neg$Painted(x)
Operator	Drill(x)
Precs:	Object(x)
Effects:	Drilled(x), $\neg$Painted(x)
Operator	Paint(x)
Precs:	Object(x), Steel(x)
Effects:	Painted(x)

Suppose that the goal is

$$\text{Shaped}(x) \wedge \text{Drilled}(x) \wedge \text{Painted}(x)$$

The graph that Knoblock would construct for this problem is:



According to Knoblock's technique, the planner should work on Shaped(x) before Drilled(x) or Painted(x). After expanding Shaped(x), the planner should work on Object(x) (because it is unaffected by any operators). It should then work on Drilled(x) before Painted(x), and so on.

2 The Problem

Suppose that the initial conditions are such that there are 100 pieces of stock in the machine shop, but only one of them is made of steel. In this case, Knoblock's approach would have the planner hunt through and build partial plans for many pieces of stock before finding one that is steel, and hence amenable to painting. In contrast, if the planner were to start work on Painted(x) followed by Steel(x), very little search would be required.

As this example illustrates, Knoblock's technique can perform arbitrarily poorly in comparison to the optimal fixed hierarchy for a problem. More generally, *Ordered Monotonic* (OM) hierarchies [1, 3] have this unpleasant characteristic.

The problem is that there are two different reasons why a conjunctive goal may be more difficult to solve than the two conjuncts taken independently:

1. Action interference,
2. Variable binding conflicts.

Knoblock's technique and OM attempt to address the first of these; they order clauses to minimize interference between actions. In fact, Knoblock's technique imposes more ordering constraints than necessary to accomplish this task. In the example above, all possible operator conflicts can be resolved by simple temporal ordering constraints among the actions in the plan. These ordering constraints are detected and resolved by a non-linear planning system. This is discussed further in [6].

3 Evaluating Clause Difficulty

In our machine shop example, the primary difficulty is related to variable binding conflicts; i.e. finding a variable binding for x that allows a solution to all three goal clauses. Knoblock's technique and OM have nothing to say about this.

One approach to this problem is to estimate the number of possible solutions to each clause and order the clauses to minimize the size of the resulting search space. For the example above, it is relatively easy to see how this might be accomplished. For the clause $\text{Shaped}(x)$, there is only one possible operator that applies and its precondition $\text{Object}(x)$ has 100 different solutions. As a result, there are 100 possible solutions to $\text{Shaped}(x)$. Similarly, $\text{Drilled}(x)$ has 100 possible solutions. For the clause $\text{Painted}(x)$, only one possible operator applies, which has two preconditions $\text{Object}(x)$ and $\text{Steel}(x)$. $\text{Object}(x)$ has 100 solutions, but $\text{Steel}(x)$ has only one, so the conjunction has at most one solution. This means that $\text{Painted}(x)$ has at most one solution.

The clause $\text{Painted}(x)$ therefore has the fewest possible solutions. If the planner starts with that clause only one solution will be considered for the other two clauses and a minimal amount of search is done.

The possibility of recursion among the operators, adds additional complexity to the problem of calculating the number of solutions for clauses. Techniques for dealing with this are described in [6].

4 Conclusion

To control search in planning, we need a much better means of estimating the difficulty of solving the goals and subgoals in a planning problem. Knoblock's technique and OM attempt to address one aspect of this problem; estimating action interference between subgoals. However, these techniques impose unnecessary and sometimes detrimental ordering constraints.

A second, and equally important aspect of problem difficulty is recognizing possible variable binding conflicts between goal clauses. Ordering clauses to minimize the size of the search space is a key to minimizing such conflicts. Doing this requires the ability to estimate the number of solutions possible for each different goal clause. We have given a hint as to how this might be accomplished and are currently implementing and evaluating this technique (see [6]).

Acknowledgments

Thanks to Craig Knoblock, Steve Minton, Qiang Yang and anonymous reviewers for comments and discussion. This work is supported by DARPA contract F30602-91-C-0031.

References

1. Knoblock, C., *Automatically Generating Abstractions for Problem Solving*, Report CMU-CS-91-120, Carnegie Mellon University, 1991.
2. Knoblock, C., Learning abstraction hierarchies for problem solving. In *Proc. 8th NCAI*, pages 923–928, Boston, MA, 1990.
3. Knoblock, C., Tenenberg, J., and Yang, C., Characterizing abstraction hierarchies for planning. In *Proc. 9th NCAI*, pages 692–697, Anaheim, CA, 1991.
4. Smith, D., Controlling Backward Inference, *Artificial Intelligence* 39(2):145–208, 1989.
5. Smith, D., *A Decision Theoretic Approach to the Control of Planning Search*, Report LOGIC-87-11, Department of Computer Science, Stanford University, 1988.
6. Smith, D., and Peot, M., *Ordering Clauses in Nonlinear Planning*, Technical Report, Rockwell International, Palo Alto Laboratory, 1992.