Integrating Hierarchical and Analogical Planning*

Billy Harris and **Diane J. Cook**
Computer Science and Engineering Department
University of Texas at Arlington
Arlington, Texas 76013
{wharris,cook}@cse.uta.edu

Abstract

Both hierarchical planning and analogical planning can separately reduce the amount of search a planner must perform to find a solution to a particular problem. We believe that these two methods can be integrated and that the combined planner will have an even smaller average search space than either method would produce alone. In this work, we present our ideas for integrating the two methods; focusing on the opportunities for beneficial interactions between analogical planning and hierarchical planning.

Introduction

Planning—determining a sequence of actions which accomplishes a particular goal—is a crucial part of intelligent behavior. Unfortunately, existing planners examine an exponential number of nodes to discover their solution. Planning researchers have explored many techniques to improve planning efficiency; some have even suggested discarding deliberative planning in favor of reactive planning. However, relatively little work combines two different methods of reducing deliberative search.

We describe two popular methods of reducing search; hierarchical planning focuses first on critical, difficult-to-achieve conditions and then focuses on less difficult conditions. Analogical planning identifies "similar" problems which the planner has encountered before and adjusts an old plan to cover the new situation. After summarizing each approach, we describe how the methods may be combined, yielding a hierarchical analogical planner. Hierarchical planning complements analogical planning by focusing the search for an adequate base case as well as providing efficient search if the planner must use from-scratch planning. Analogical planning complements hierarchical planning by finding existing cases which satisfy the abstract or "hard" conditions, allowing the hierarchical planner to avoid or reduce search for some or all of its abstraction levels.

Analogical Planning

Analogical or case-based planners use a library of previously generated plans. When confronted with a problem, analogical planners identify "similar" problems in the library. A base plan is retrieved and adjusted to account for differences between the original problem and the current problem. Often, the analogical planner incorporates generational planning to account for goals in the new problem left unsolved by the retrieved plan. The planner may decide to store its new solution in the plan library.

Plan Storage

Planners using *derivational analogy* store search traces identifying choices the planner made which led to the solution (Carbonell 1986). For new problems, the planner replays the trace and hopefully makes only minor additions accounting for the new situation. Derivational analogy has been used by both state-space and plan-space planners.

Plan Retrieval

An analogical planner must decide which of its stored plans is best suited for a new problem. This decision is often difficult; in addition to finding a suitable mapping between the old problem and the new problem, the old plan may not successfully solve its subgoals in a new context. Explanation-based learning is sometimes used to determine if a given case may be successfully adapted to a new situation (Ihrig & Kambhampati 1995).

Plan Adaptation

Ideally, analogical planners are able to find a valid solution even if the retrieved plan is not applicable to the new situation. Hanks and Weld have developed SPA (Systematic Plan Adaptor) to address this problem (Hanks & Weld 1995). SPA views the retrieved plan as a node–not the root–in a search tree. If the retrieved plan is applicable to the new situation, search continues "downward" as the planner adds steps corresponding to new goals not considered by the retrieved plan. If the retrieved plan is not applicable, the planner retracts some of the choices made for the original plan

and moves "upward" toward the root of the search tree. The root of the search tree represents the empty plan; the planner retracting to the root represents the need for from-scratch planning. SPA finds solutions even if it retrieves an inapplicable base plan.

Plan Generalization

Once the planner has generated a new solution, it must decide whether or not the new problem is sufficiently novel to warrant saving in the plan library. If so, the planner should generalize the solution so that it may be applied to as many situations as possible. Veloso uses explanation-based generalization to generalize plans (Veloso & Carbonell 1990).

Hierarchical Planning

Hierarchical planners divide a domain into abstraction levels. Hierarchical planners first find an "abstract" solution for part of the planning problem, then proceed to lower abstraction levels to refine its solution by incorporating additional details. In abstraction level 0, or the ground abstraction level, the planner considers all remaining subgoals.

Forming Abstraction Levels

Intuitively, domain predicates in the highest abstraction levels represent the most crucial or most difficult portions of the domain. Traditionally, users of hierarchical planners explicitly assign predicates into abstraction levels; however, some systems form abstraction hierarchies automatically (Knoblock 1994).

Abstract Plans

Hierarchical planners begin by searching only those predicates in the highest abstraction level. Thus, the planner considers fewer preconditions and possibly fewer plan operators in its abstract search space. Once the planner has an abstract solution, it adds additional subgoals and proceeds to a lower abstraction level. Modern hierarchical planners use "monotonic refinement" meaning that the planner may not undo work performed at a higher abstraction level (Woods 1991). Not all abstract solutions can be successfully refined; the planner can backtrack to a higher abstraction level and find a different abstract solution. Some domains have the "downward refinement property"; abstract solutions in these domains can always be refined if the problem has a solution at all.

Search Reduction in Hierarchical Planning

Hierarchical planners reduce the amount of search performed by dividing the problem into mostly independent search spaces. A conventional planner solving a problem with an s-step solution will explore a single search tree with a depth of at least s. In contrast, a hierarchical planner with n levels of abstraction hopes to explore n search trees, each of which has a depth of only s/n.

Hierarchical Analogical Planning

We believe hierarchical planning and analogical planning can be successfully combined to yield a hierarchical analogical planner. Such a planner would follow the same process as an analogical planner, but hierarchical planning would affect several steps. Similarly, the plan generation portion of the planner would have the same search characteristics as conventional hierarchical planning, except that each abstraction level would have a proposed solution ready for replay or revision.

Plan Storage

Like ordinary derivational-analogical planners, our system will store the sequence of choices the planner made on the search path to the solution. Since our planner is a hierarchical planner, this search trace will begin with choices involving the most abstract goals. In addition, the trace identifies when the planner has found an abstract solution and changed the abstraction level. The stored trace, as always, reflects the process used to generate the plan solution; in this case, the process is hierarchical planning.

Note that the choices are stored in the order that the planner made them. Since we use a partial-order planner, the order of the steps in the finished plan is independent of the order in which the steps were first added to the plan. Thus, the planner could, after performing replay, add a new step to the plan and constrain the step to occur between two steps added during replay.

Case Retrieval

Correctly identifying and instantiating a stored plan challenges analogical planners. If a new plan mentions n objects and a possible stored plan mentions k objects, finding the optimal mapping between the two plans may take up to $\binom{n}{k}$ steps (Hanks & Weld 1995). As explained in the next section, hierarchical planning can greatly reduce this cost.

In addition to reducing the cost of analogical mapping, hierarchical analogical planning can also help choose between two competing plans which both offer partial solutions to the current problem. If two different library plans each solve two goals but leave one unsolved, which one should we select? This choice is difficult for traditional analogical planners, but hierarchical analogical planners offer a heuristic. Assuming that conditions in a high abstraction level are more difficult to achieve than conditions in lower abstraction levels, our retriever should prefer a library plan which completely solves goals in a high abstraction level (but does not solve any in a low abstraction level) to a plan which partially solves goals in both high and low abstraction levels.

Plan Adaptation

Like conventional replay, hierarchical analogical planning uses the retrieved plan to solve portions of the

problem and uses generative planning to solve additional goals. Methods similar to the Systematic Plan Adaptor can be applied to each abstraction level and ensure that the planner will find a solution even if the retrieved plan does not apply to the new situation.

Specifically, SPA (Hanks & Weld 1995) views the retrieved plan as being one node (not the root) of a search tree. Thus, if the planner does not find a solution by refining the retrieved plan, it can perform backtracking over the choices made when the library plan was first performed. When SPA backtracks, it will identify an "exposed" choice to retract. A choice is exposed if no other plan choice depends on it. For example, if the planner adds a STACK operator to achieve "(ON A B)" and then adds an additional ordering constraint to satisfy the preconditions of the STACK operator, the ordering decision prevents the add-operator action from being exposed.

Our planner adapts the SPA approach to hierarchical planning. Specifically, we begin at the highest abstraction level by forming two copies of the portion of the library plan pertaining to the most abstract conditions of our new problem. One copy is refined normally to account for goals unmet by the library plan. The other copy is marked for retraction (SPA puts this plan in an "up" queue). If the refinement search does not progress toward a solution, the planner backtracks over the retraction node, undoing one of the choices made in the original library plan. The planner can continue backtracking until it reaches the original, empty plan corresponding to the root of the search tree.

On the other hand, the original retrieved plan may lead to an (abstract) solution. In this case, like ordinary hierarchical planners, our planner reduces the abstraction level. When this happens, the planner attempts to re-apply the plan choices made in the library plan which relate to the new (lower) abstraction level. Again, the planner makes two copies of the initial node of the lower abstraction level, allowing it to retract unrefineable choices made in this lower abstraction level.

Hierarchical planning assumes that conditions in the highest abstraction levels are the most difficult to achieve. If this assumption means that these conditions are the most likely to fail in new circumstances, then a hierarchical analogical planner can more quickly recognize when a library plan is unlikely to apply to the new situation. The hierarchical analogical planner focuses first on the most difficult portion of the plan and only attempts to reuse the "easy" part of the library plan if the abstraction portions successfully applied to the new situation. In contrast, a conventional analogical planner may expend considerable effort before realizing that the library plan is inapplicable to the new situation.

In addition, if the planner must use from-scratch planning to solve a problem, hierarchical planning can reduce search compared to non-hierarchical generative planning.

Plan Generalization

As before, if the planner decides that a new case is "sufficiently different" to add to its library, the planner will generalize the solved plan. Unlike before, the planner can generalize the solution for each abstraction level separately; after all, the planner may later wish to reuse an abstract solution and this type of reuse will usually be more applicable than reuse of the entire plan.

Since multiple plans may share the same abstract solution, the plan can be stored in a tree structure. This tree reduces the amount of match a hierarchical analogical planner must perform to identify an applicable plan.

Hierarchical Matching

Consider a conventional analogical planner trying to solve a problem in the Towers of Hanoi domain. After a few iterations, our planner may have built the small case file shown in Figure 1. Case 3, for example, shows that the planner knows how to solve the problem with an initial state in which the small disk starts on a particular peg and the medium and big disk start on the same peg (different from the peg used by the small disk) and with a goal state in which the medium and big disks are moved to the peg initially occupied by the small disk, which should end up on yet a third peg.

Now, suppose the planner is faced with a problem with the initial state of "(on-big peg-3) (on-medium peg-3) (on-small peg1)" and a goal state of "(on-big peg-1) (on-medium peg-1) (on-small peg2)." Our planner must consider each of the four cases and identify which case (and which bindings) will ensure the closest match. Specifically, our planner first finds those cases giving the best match for the goal state. If two or more cases are tied, the planner identifies which of the tied cases best matches the initial state. For each case, our planner will try to match the three goal conditions with the three goals of the new problem. After studying four cases, each of which has three goals needing matching, our planner will conclude that Case 3 gives the best match, with ?A matching peg-3, ?B matching peg-1, and ?C matching peg-2 .

In contrast, our hierarchical analogical planner stores its cases in a tree as shown in Figure 2. When our hierarchical planner tries to the same new problem, it begins by considering only the most abstract goal—(on-big peg-1). Our planner then performs matching on the most abstract portions of the case file. In our example, all of our cases share the same conditions at the highest abstraction level, so our hierarchical analogical matcher quickly decides that ?B should map to peg-1. The matcher then considers the case portions in the next lower abstraction level. Each child node contains the same goal (with ?B already bound to peg-1), so the matcher considers the initial states of each tied match. Based on initial states, the matcher prefers the right subtree and binds ?A to peg-3. Finally, the matcher considers the least abstract conditions, and prefers Case

```
  IS: (on-big ?A) (on-medium ?B) (on-small ?A)
  GS: (on-big ?B) (on-medium ?B) (on-small ?C)
Case: 1
```

```
  IS: (on-medium ?B) (on-big ?A) (on-small ?C)
  GS: (on-small ?B) (on-big ?B) (on-medium ?B)
Case: 2
```

```
  IS: (on-small ?B) (on-medium ?A) (on-big ?A)
  GS: (on-medium ?B) (on-small ?C) (on-big ?B)
Case: 3
```

```
  IS: (on-small ?A) (on-big ?A) (on-medium ?A)
  GS: (on-small ?B) (on-medium ?B) (on-big ?B)
Case: 4
```

Figure 1: Traditional Analogical Index File

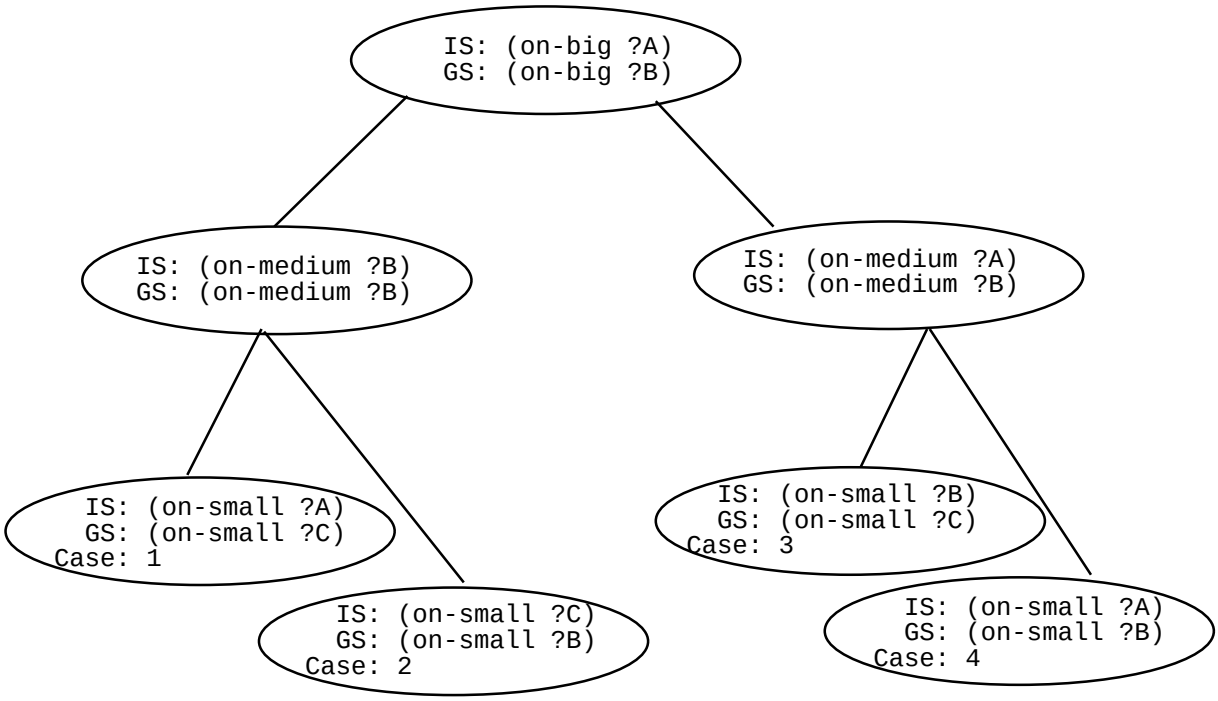

Figure 2: Hierarchical Analogical Index File

3 over Case 4 based on goal-mapping, with the final binding of ?C to peg-2.

Our hierarchical matcher never fully examined cases 1 or 2. These cases did not match as many abstract conditions as cases 3 and 4, so the planner did not consider them further. Our hierarchical matcher considered matches for the (on-big ?B) goal only once, while the conventional matcher had to match this goal for each case. Finally, for a given case, our conventional planner performed matching three times, each time between goals with only one condition; our conventional matcher, in contrast, performed matching only once with goals containing three conditions.

Conclusions and Future Work

We believe hierarchical planning and analogical planning can be combined beneficially. Hierarchical planning reduces the matching problem in case retrieval and helps organize the case library. Hierarchical planning offers a method of discriminating among plans which each offer partial solutions by preferring plans which solve abstract goals to plans which solve concrete goals. Analogical planning helps hierarchical planning by providing partial solutions for each abstraction level. Compared to a non-hierarchical analogical planner, a hierarchical analogical planner may more quickly recognize that a retrieved plan in inapplicable to the new situation; if the planners must perform from-scratch search, hierarchical planning can reduce the search space size.

We will implement our ideas using a modified version of the SNLP/SPA planner (Hanks & Weld 1995; McAllester & Rosenblitt 1991). We intend to compare the cost of conventional plan retrieval to hierarchical plan retrieval and compare the costs of analogical planning to (from-scratch) hierarchical planning.

Acknowledgements

This research was supported in part by the National Science Foundation, under grant GER-9355110.

References

Carbonell, J. 1986. Derivational analogy: A theory of reconstructive problem solving and expertise aquisition. *Machine Learning: An Artificial Intelligence Approach* 2.

Hanks, S., and Weld, D. 1995. A domain-independent algorithm for plan adaptation. *Journal of Artificial Intelligence Research* 319–360.

Ihrig, L., and Kambhampati, S. 1995. An explanation-based approach to improve retrieval in case-based planning. *Current Trends in AI Planning: EWSP '95.*

Knoblock, C. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68:232–302.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. *Proceedings AAAI-91* 634–639.

Veloso, M., and Carbonell, J. 1990. Integrating analogy into a general problem-solving architecture. *Intelligent Systems.*

Woods, S. 1991. An implementation and evaluation of a hierarchical non-linear planner. Master's thesis, University of Waterloo.