Evaluating the Effectiveness of Derivation Replay in Partial-order vs State-space Planning

Laurie H. Ihrig & Subbarao Kambhampati*
Department of Computer Science and Engineering
Arizona State University, Tempe, AZ 85287-5406
email: laurie.ihrig@asu.edu, rao@asu.edu

From: AAAI Technical Report WS-94-01. Compilation copyright © 1994, AAAI (www.aaai.org). All rights reserved.

Abstract

Case-based planning involves storing individual instances of problem-solving episodes and using them to tackle new planning problems. This paper is concerned with derivation replay, which is the main component of a form of case-based planning called derivational analogy (DA). Prior to this study, implementations of derivation replay have been based within state-space planning. We are motivated by the acknowledged superiority of partial-order (PO) planners in plan generation. Here we demonstrate that plan-space planning also has an advantage in replay. We will argue that the decoupling of planning (derivation) order and the execution order of plan steps, provided by partial-order planners, enables them to exploit the guidance of previous cases in a more efficient and straightforward fashion. We validate our hypothesis through a focused empirical comparison.

Introduction

Recently it has been demonstrated that partial-order planners, which search in a space of plans, are more flexible and efficient in plan generation (Barrett and Weld 1994; Minton *et al.* 1992). The aim of the current work is to analyze the possible advantage that a plan-space planner would have in derivation replay, which is one component of a form of *case-based planning* called *derivational analogy* (Carbonell 1986; Veloso 1992). We are motivated by a desire to see whether the known advantages of PO planning over state-space planning in plan generation, also make the former a more efficient substrate for replay.

Derivational analogy includes all of the following elements (Carbonell 1986; Veloso 1992): a facility within the base-level planner to generate a trace of the derivation of a problem solution, the indexing and storage of the solution trace in a library of cases, the retrieval of a case in preparation for solving a new problem, and finally, a replay mechanism by which the planner can utilize a previous derivation as guidance to a new search process. The storage and retrieval aspects of DA remain the same whether we use plan-space or state-space planners. In particular, solutions to the storage problem such as those proposed in (Veloso 1992) and (Kambhampati 1994) can be used for this purpose. Only the contents of the trace and the details of the replay component depend on the underlying planner. Thus, in our evaluation we focus on the automatic generation and replay of the solution trace.

We will start by comparing plan-space and state-space planning, two approaches to generative planning which vary as to how they derive their solutions. Then we will analyze the relative advantages of doing replay within plan-space vs state-space planners. One of the difficult decisions faced by the replay systems is that of deciding when and where to interleave from-scratch effort with derivation replay (c.f. (Blumenthal and Porter 1994)). In general, there are no domain-independent grounds for making this decision. This makes *eager replay*, i.e., replaying the entire trace before returning to from-scratch planning, the most straightforward strategy. We will show that, for replay systems based on state-space planners, eager replay inhibits effective transfer in problems where the plan-step order is critical to the solution. We will argue that the decoupling of planning (derivation) order from execution order of steps, provided by plan-space planners, allows effective transfer to occur with eager replay in more situations, and thus provides for a more efficient and straightforward replay framework. We will validate this hypothesis by comparing plan-space replay to replay systems implemented on two different state-space planners. We will end with a discussion of the applicability of our results in the context of a variety of extensions to the basic replay strategy.

Plan-space vs State-space Planning

State-space planners derive plans by searching through a space of world states. Each time that a transition is made into an adjoining state, the operator that facilitates that transition is added to the end (or, in the case of a backward-search from the goal state, to the beginning) of the plan that is so far constructed. This means that steps must be added to the plan according to their execution order.

As an example, consider the state-space planner TOPI (Barrett and Weld 1994). TOPI does simple backward search in a space of states, adding steps to the plan in reverse order of their execution. A trace of TOPI's derivation of a solution to a

*This research is supported in part by National Science Foundation under grant IRI-9210997, and ARPA/Rome Laboratory planning initiative under grant F30602-93-C-0039. Thanks to Manuela Veloso for helpful clarifications regarding Prodigy/Analogy. Thanks also to Suresh Katukam for his comments. Interested readers are encouraged to read a companion paper in Proceedings AAAI-94 describing our experiments in greater detail.

```
                         Goal : (AT-OB OB2 AP1)
          Initial : ((IS-A AIRPORT AP1) (IS-A AIRPORT AP2))
                 (IS-A AIRPORT AP3) (AT-PL PL1 AP3)
                         (AT-OB OB2 AP2) ...
Name : G1                                 Name : G4
Type : START-NODE                         Type : ESTABLISHMENT
Name : G2                                 Kind : NEW-STEP
Type : ESTABLISHMENT                      New Step: (LOAD-PL OB2 ?P1 ?A2)
Kind : NEW-STEP                           Open Cond: ((INSIDE-PL OB2 ?P1) 1)
New Step: (UNLOAD-PL OB2 ?P1 AP1)         Name : G5
Open Cond: ((AT-OB OB2 AP1) GOAL)         Type : ESTABLISHMENT
Name : G3                                 Kind : NEW-STEP
Type : ESTABLISHMENT                      New Step: ((FLY-PL ?P1 ?A3 ?A2)
Kind : NEW-STEP                           Open Cond: ( (AT-PL PL1 ?A2) 3))
New Step: (FLY-PL ?P1 ?A2 AP1)                       Key to Abbreviations:
Open Cond: ((AT-PL ?P1 AP1) 1)            PL=PLANE,AP=AIRPORT,OB=OBJECT

               Final Plan: (FLY-PL PL1 AP3 AP2) Created 4
                     (LOAD-PL OB2 PL1 AP2) Created 3
                      (FLY-PL PL1 AP2 AP1) Created 2
                    (UNLOAD-PL OB2 PL1 AP1) Created 1
                  Ordering of Steps: ((4 < 3) (3 < 2) (2 < 1))
```

Figure 1: An Example Solution Trace for DerTOPI

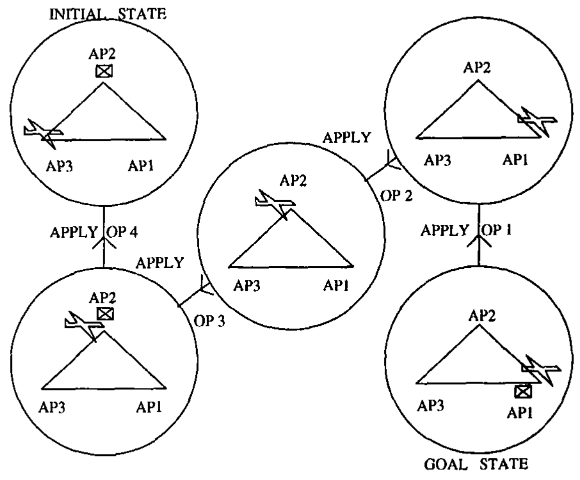


Figure 2: Moving Through State Space

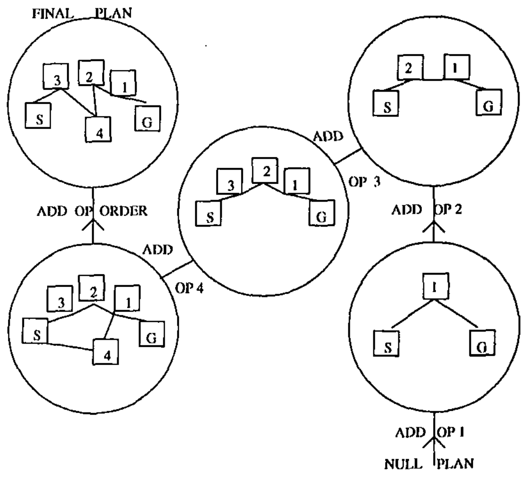


Figure 3: Moving Through Plan Space

problem from the logistics transportation domain of (Veloso 1992) is provided in Figure 1. This domain involves the movement of packages across locations by various transport devices. The trace corresponds to a simple problem in which there is an airplane P1 at airport AP3, a package OB2 at airport AP2, and the goal is to transport this package to destination AP1. Figure 2 shows graphically how TOPI would derive a plan for this problem by stepping through a sequence of world states, applying an operator to the plan at each transition.

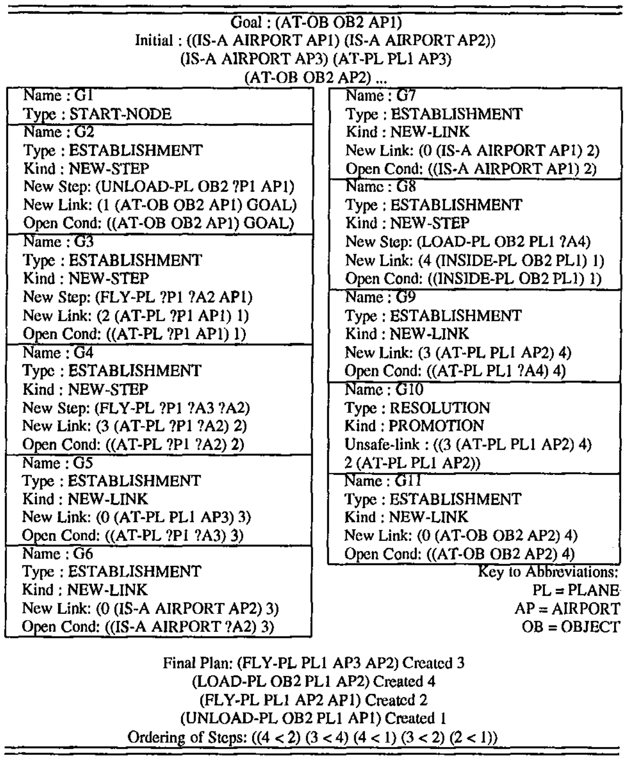
Plan-space planners derive their plans by traversing a space of partly-constructed plans. The plan-space planner moves through this space by successively modifying the currently active plan, starting with an empty plan. Plans are refined by adding constraints which include new steps as well as new orderings between steps. Figure 4 provides a trace of SNLP's decision process in arriving at a solution to our example problem taken from the logistics domain. SNLP (McAllester and Rosenblitt 1991; Barrett and Weld 1994) is a causal-link partial-order planner. A path through plan-space which corresponds to the derivation contained in Figure 4 is displayed graphically in Figure 3. This figure serves to illustrate the PO planners' least commitment strategy when it comes to step-orderings. Orderings are added as required by the subgoaling structure, since steps that contribute conditions must precede the steps that require these conditions. Step-orderings are also added to resolve conflicts between steps, for example, when one step is deleting the contribution of another. Notice in Figure 3 that this means that new steps may first be added in parallel to an existing step sequence. Further step orderings may then be added which accomplish the interleaving of the new action into the existing plan segment. It is this ability to interleave new steps into the plan which gives the PO planner an advantage in replay.

Relative Advantages of PO vs. State-space Planning in Supporting Replay

The previous section characterizes the differences between the state-space and plan-space planning formalisms. In this section, we will look at the ramifications of these differences on the efficiency of replay.

As mentioned earlier, replay involves applying the decisions taken in a previous planning episode to the context of a new planning problem. Since the new problem may contain goals and initial conditions which are different from that of the previous one, it is likely that the result of replay is a plan which only partially solves the new problem. Thus, the plan that is returned by the replay procedure may still contain open conditions, either because a prior decision that was taken in solving a subgoal is not valid in the new context, or because there are top level goals that are not covered by the previous trace. Such partial plans resulting from replay will then have to be extended into a complete solution for the new problem. It is in this situation that the plan-space planners show more flexibility over state-space planners.

In particular, as discussed in the previous section, plan-space planners can extend a partial plan by interleaving steps anywhere in the plan. In contrast state-space planners can only add steps to either the beginning or the end of the partial plan. Thus, if the extra goals of the new problem require steps



```
Goal : (AT-OB OB2 AP1)
Initial : ((IS-A AIRPORT AP1) (IS-A AIRPORT AP2))
(IS-A AIRPORT AP3) (AT-PL PL1 AP3)
(AT-OB OB2 AP2) ...

Name : G1
Type : START-NODE

Name : G2
Type : ESTABLISHMENT
Kind : NEW-STEP
New Step: (UNLOAD-PL OB2 ?P1 AP1)
New Link: (1 (AT-OB OB2 AP1) GOAL)
Open Cond: ((AT-OB OB2 AP1) GOAL)

Name : G3
Type : ESTABLISHMENT
Kind : NEW-STEP
New Step: (FLY-PL ?P1 ?A2 AP1)
New Link: (2 (AT-PL ?P1 AP1) 1)
Open Cond: ((AT-PL ?P1 AP1) 1)

Name : G4
Type : ESTABLISHMENT
Kind : NEW-STEP
New Step: (FLY-PL ?P1 ?A3 ?A2)
New Link: (3 (AT-PL ?P1 ?A2) 2)
Open Cond: ((AT-PL ?P1 ?A2) 2)

Name : G5
Type : ESTABLISHMENT
Kind : NEW-LINK
New Link: (0 (AT-PL PL1 AP3) 3)
Open Cond: ((AT-PL ?P1 ?A3) 3)

Name : G6
Type : ESTABLISHMENT
Kind : NEW-LINK
New Link: (0 (IS-A AIRPORT AP2) 3)
Open Cond: ((IS-A AIRPORT ?A2) 3)

Name : G7
Type : ESTABLISHMENT
Kind : NEW-LINK
New Link: (0 (IS-A AIRPORT AP1) 2)
Open Cond: ((IS-A AIRPORT AP1) 2)

Name : G8
Type : ESTABLISHMENT
Kind : NEW-STEP
New Step: (LOAD-PL OB2 PL1 ?A4)
New Link: (4 (INSIDE-PL OB2 PL1) 1)
Open Cond: ((INSIDE-PL OB2 PL1) 1)

Name : G9
Type : ESTABLISHMENT
Kind : NEW-LINK
New Link: (3 (AT-PL PL1 AP2) 4)
Open Cond: ((AT-PL PL1 ?A4) 4)

Name : G10
Type : RESOLUTION
Kind : PROMOTION
Unsafe-link : ((3 (AT-PL PL1 AP2) 4)
2 (AT-PL PL1 AP2))

Name : G11
Type : ESTABLISHMENT
Kind : NEW-LINK
New Link: (0 (AT-OB OB2 AP2) 4)
Open Cond: ((AT-OB OB2 AP2) 4)

Key to Abbreviations:
PL = PLANE
AP = AIRPORT
OB = OBJECT

Final Plan: (FLY-PL PL1 AP3 AP2) Created 3
(LOAD-PL OB2 PL1 AP2) Created 4
(FLY-PL PL1 AP2 AP1) Created 2
(UNLOAD-PL OB2 PL1 AP1) Created 1
Ordering of Steps: ((4 < 2) (3 < 4) (4 < 1) (3 < 2) (2 < 1))
```

Figure 4: An Example Solution Trace for DerSNLP

that have to be interleaved into the plan produced through replay, the PO planner is more likely than the state-space planner to successfully extend the plan resulting from replay.

As an example, suppose that the trace contained in Figure 4 is replayed for the problem that requires the additional goal, (AT-OB OB3 AP1), and OB3 is initially on the old route taken by the airplane. Notice that the optimal way of dealing with the extra goal requires interleaving two steps -- to load OB3 into the plane and unload it from the plane -- at appropriate places into the current plan. Replay systems based on a partial-order planner such as SNLP can accomplish this by first adding the steps in parallel to the existing steps, and then ordering them with respect to other steps in the process of resolving conflicts.

This interleaving of new steps is not possible if replay is based on state-space planners since they can extend a partial plan only by adding steps to the beginning or end. In this particular example, the state-space planners can extend the partial plan to deal with the additional goal, but do so by making the plane return to pick up the second package and then deliver it, thus resulting in a less optimal solution. In other cases, the inability to interleave the plan may make the state-space planner backtrack over the partial plan resulting from replay (This will happen, for example, when solving problems from the ART-MD-NS domain described below).

To summarize, when replay is based on a state-space planner and step order is critical for the success of the plan, eager replay may not allow for effective transfer. Partial-order plan-space planners do not suffer from this problem. In particular, since the PO planners decouple derivation (planning) order of plan steps from their execution order, an eager replay strategy will not mislead them as much. It is therefore more likely that all of the previous advice that is applicable to the new problem is fully utilized. This reasoning leads us to believe that plan-space planners will exhibit greater performance improvements through replay when interacting subgoals make step order critical for a plan's success. The next section describes an empirical evaluation of this hypothesis.

Empirical Evaluation

An empirical analysis was conducted in order to test our hypothesis regarding the relative effectiveness of eager replay for PO planners. For a more detailed description of these experiments, see (Ihrig and Kambhampati 1994). We implemented an eager-replay strategy on the state-space planner, TOPI (DerTOPI), and the plan-space planner SNLP (DerSNLP). With this strategy, the search process is interrupted to replay the entire derivation trace before returning to from-scratch planning. We also implemented eager replay on a second state-space planner, NOLIMIT, since it recently served as a substrate for a comprehensive derivational analogy system (Veloso 1992). Although NOLIMIT differs from TOPI in several respects[1], it is also a state-space planner in the sense that it extends a current partial plan by adding steps to the end of the plan. To facilitate fair comparisons, the three planning methods, SNLP, TOPI, and NOLIMIT, were (re)implemented on the same substrate.

ART-MD-NS Domain: Experiments were run on problems drawn from two domains. The first was the artificial domain, ART-MD-NS, originally described in (Barrett and Weld 1994) and shown in the table below:

ART-MD-NS ($D^m S^2$):
A_i^1 <u>*precond*</u> : I_i <u>*add*</u> : P_i <u>*delete*</u> : $\{I_j \mid j < i\}$)
A_i^2 <u>*precond*</u> : P_i <u>*add*</u> : G_i <u>*delete*</u> : $\{I_j \mid \forall j\} \cup \{P_j \mid j < i\}$)

Conjunctive goals from this domain are *nonserializable* in that they cannot be achieved without interleaving subplans for the individual conjuncts. For example, consider the problem that contains the conjunctive goal $G_1 \wedge G_2$. The subplan for achieving this goal would be: $A_1^1 \rightarrow A_2^1 \rightarrow A_1^2 \rightarrow A_2^2$. This plan has to be interleaved with steps to solve the additional goal G_3. The plan for the new conjunctive goal $G_1 \wedge G_2 \wedge G_3$ is $A_1^1 \rightarrow A_2^1 \rightarrow \underline{A_3^1} \rightarrow A_1^2 \rightarrow A_2^2 \rightarrow \underline{A_3^2}$.

Logistics Transportation Domain: The logistics transportation domain of (Veloso 1992) was adopted for the second set of experiments. Initial conditions of each problem represented the location of various transport devices (one airplane and three trucks) over three cities, each city containing an airport and a post office. Four packages were randomly distributed

[1]NOLIMIT is a means-ends analysis planner like PRODIGY and STRIPS, attempting goals by backward-chaining from the goal state. Applicable operators (operators whose preconditions are true in the current state) are added to the end of the plan and the current state is advanced appropriately. Unlike STRIPS, NOLIMIT can defer step addition in favor of further subgoaling. It does this by adding relevant operators to a list of potential operators before actual placement in the plan.

Phase	ART-MD-NS *(depth-first, CPU limit: 100sec)*						Logistics *(best-first, CPU limit: 550sec)*			
	DerSNLP		**DerTOPI**		**DerNOLIMIT**		**DerSNLP**		**DerTOPI**	
	replay	scratch	replay	scratch	replay	scratch	replay	scratch	replay	scratch
One Goal										
%Solved	100%	100%	100%	100%	100%	100%	100% (3.5)	100% (3.5)	100% (5.0)	100% (3.5)
nodes	30	90	30	60	30	120	617	946	46	507
time(sec)	.73	.68	.45	.47	.63	2.5	15	19	11	49
Two Goal										
% Solved	100%	100%	100%	100%	100%	100%	100% (5.8)	100% (5.8)	97% (6.2)	63% (5.6)
nodes	257	317	180	184	347	296	1571	2371	15824	8463
time(sec)	2	2	5	4	8	12	50	51	6216	3999
Three Goal										
% Solved	100%	100%	100%	100%	100%	100%	100% (7.9)	100% (7.9)	0%	0%
nodes	395	679	549	462	1132	662	6086	7400	-	-
time(sec)	7	4	16	11	34	34	230	262	-	-
Four Goal										
% Solved	100%	100%	100%	100%	100%	100%	100% (10.0)	100% (10.0)	0%	0%
nodes	577	1204	1715	1310	5324	1533	9864	24412	-	-
time(sec)	35	43	227	96	368	100	497	1264	-	-

Table 1: Performance statistics in ART-MD-NS and Logistics Transportation Domain (Average solution length is shown in parentheses next to %Solved for the logistics domain only)

over airports. So as to make step order critical, problems were chosen from this domain to contain subgoals that interact. Problems represent the task of getting one or more packages to a single designated airport.

Whereas each problem in ART-MD-NS has a unique solution, in the logistics domain there are many possible solutions varying in length. However, optimal (shortest) solutions can only be found by interleaving plans for individual goals. This difference has an important ramification on the way eager replay misleads state-space planners in these domains. Specifically, in the ART-MD-NS domain, state-space planners will have to necessarily backtrack from the path prescribed by eager replay to find a solution. In the logistics domain, they can sometimes avoid backtracking by continuing in the replayed path, but will find inoptimal plans in such cases.

Results

The results of testing are shown in Table 1. Each table entry represents cumulative results obtained from a sequence of 30 problems corresponding to one phase of the run. Problem size was increased by one goal for each phase. A library of cases was formed over the entire run. Each time a problem was attempted, the library was searched for a previous case that was *similar* (See (Ihrig and Kambhampati 1994)). If one was found, the new problem was run both in scratch and replay mode, and the problem became part of the 30 problem set for that phase. If there was no previous case that applied, the problem was merely added to the library.

The first row of Table 1 shows the percentage of problems correctly solved within the time limit. The average solution length is shown in parentheses for the logistics domain (Solution length was omitted in ART-MD-NS since all the problems have unique solutions.) The subsequent rows of Table 1 contain the total number of search nodes visited for all of the 30 test problems, and the total CPU time. DerSNLP was able to solve as many or more of the multi-goal problems than the two state-space planners both in from-scratch and replay modes, and did so in less time. Our implementation of DerNOLIMIT was not able to solve any of the multi-goal problems in the logistics domain within the time limit, and this column is therefore omitted from the table.

In the ART-MD-NS domain, replay resulted in performance improvements for DerSNLP which increased with problem size (See Table 1). Comparative improvements with replay were not found for the two state-space planners in the multi-goal phases. In the logistics domain, not only did DerTOPI fail to improve performance through replay, it also experienced an increase in average solution length. In contrast, replay in DerSNLP led to performance improvements (without increasing the solution length). These results are consistent with our hypothesis that state-space planners will be misled by eager replay when step order is critical.

Analyzing the Generality of Experimental Results

The experiments reported in this paper concentrated on the relative effectiveness of state-space and plan-space planners in supporting eager replay. Some implemented replay systems, such as REMAID (Blumenthal and Porter 1994) and PRODIGY/ANALOGY (Veloso 1992) utilize more complex forms of replay strategies to support derivational analogy. This raises a question as to the applicability of our experimental results to such frameworks. We will address this issue below.

Interrupting Replay : Some replay systems interrupt replay of the trace to interleave from-scratch effort (e.g. (Blumenthal and Porter 1994)). It would seem as if such strategies could offset the inflexibility of state-space planners in replaying cases when step order is critical. Consider again our example problem from the logistics domain. Recall that we have a plan that corresponds to transporting a single package to a designated airport, and the new situation requires us to transport another package to the same destination, and this package lies somewhere along the plane's route. One way a state-space planner can produce an optimal plan without backtracking over the result of replay is to interrupt the replay in the middle, do from-scratch planning to deal with the additional goal, and resume replay. However, the optimal point in the derivation for inserting the new steps depends on the problem description, since it will depend on where the new package is located on the old route. The early work on derivation replay that is rooted in state-space planning has thus been forced to focus a good deal of attention on the problem of determining *when* to plan for additional goals (Blumenthal and Porter 1994). In general, there are no domain-independent grounds for deciding when and where the from-scratch problem-solving effort should be interleaved with replay. This leaves the planner with heuristic guesses, which may turn out to be incorrect, leading to backtracking once again. In contrast, as we have shown, the partial-order planner can exploit the previous case with eager replay without the need to interrupt replay of the trace.

Multi-pass Replay: Other replay systems, such as PRODIGY/ANALOGY use a multi-pass strategy in replay. Rather than abandon the case after it has been replayed once, such systems keep the case and replay it again as necessary when the plan that resulted from replaying that case is backtracked over. This too could sometimes help state-space planners in improving their ability to exploit a previous trace. In our example above, this will involve replaying the old trace once, and when the planner fails to extend it to deal with the new goal, backtracking, working on the extra goal, and then replaying the case again. Veloso (Veloso 1992) discusses an interesting mechanism for keeping track of the extent to which the case has been replayed, and moving the pointer up appropriately whenever backtracking occurs.

Although this is an intuitively appealing idea (and could in fact be adapted to replay systems based on any planning framework), it is not guaranteed to completely overcome the inflexibility of state-space planners in replay. In particular, unless the planner has domain specific control information, it may be forced to backtrack many times before finding the right place to insert the steps corresponding to the new goals. We thus believe that while the multi-pass replay heuristic may sometimes improve the effectiveness of replay in state-space planning, it still does not eliminate the essential advantages of plan-space planners in supporting reuse.

Multi-case Replay: A feature of PRODIGY/ANALOGY is its ability to use multiple cases in guiding replay. Although our empirical study involved only replay of a single case for each problem, we believe that the results can also be extended to multi-case replay. When we have multiple cases, in addition to the decision as to how to interleave replay with from-scratch planning, we also have the decision as to *how to interleave the replay of the individual cases.* When step order is critical, the state-space planner will have to attempt both types of interleaving to successfully exploit the guidance of previous cases. Once again, the plan-space planner will have less need to do either of these, and can thus not only replay the cases in any order, but also employ eager replay on individual cases.

Before concluding this section, we would like to make some observations about the apparent disparity between our results and the previous work on DA rooted in state-space planning which has demonstrated significant performance improvements with replay (Veloso 1992). We believe the main reason for this may be the strong presence of interacting goals in our multi-goal problems, coupled with the fact that we used vanilla planning algorithms without any sophisticated backtracking strategies or pruning techniques. Although these techniques may improve the performance of state-space planners, these will be of benefit to PO planners as well. Moreover, we have shown that the shift to plan-space replay lessens the need for backtracking over the replayed path.

Finally, our results from experiments in the logistics domain indicate that even partial-order planners may be misled by eager replay in some instances. This happens when the previous case may have achieved one of the goals using some step s_1 and the new problem contains a goal g_n which cannot be achieved in the presence of s_1. However, in such cases, state-space planners will also be misdirected. Thus our hypothesis is only that plan-space planners are *less likely* to be misled (and thus more likely to exploit the previous case) through eager replay.

Summary

In this paper, we described the differences between plan-space and state-space approaches to planning. We developed a testable hypothesis regarding the relative advantage of plan-space planning over state-space planning in derivation replay. We supported this hypothesis with the help of a focused empirical study. We then discussed the generality and applicability of our hypothesis in the context of a variety of extensions to the basic replay strategy.

References

Barrett, A. and Weld, D. 1994. Partial order planning: evaluating possible efficience gains. *Artificial Intelligence* 67(1).

Blumenthal, B. and Porter, B. 1994. Analysis and empirical studies of derivational analogy. *Artificial Intelligence.* Forthcoming.

Carbonell, J. 1986. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, Ryszard; Carbonell, Jaime; and Mitchell, Tom M., editors 1986, *Machine Learning: an Artificial Intelligence approach: Volume 2.* Morgan-Kaufman.

Ihrig, L. and Kambhampati, S. 1994. Derivation replay for partial-order planning. In *Proceedings AAAI--94.*

Kambhampati, S. 1994. Exploiting causal structure to control retrieval and refitting during plan reuse. *Computational Intelligence Journal* 10(2).

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

Minton, S.; Drummond, M.; Bresina, J.; and Philips, A 1992. Total order vs partial order planning: factors influencing performance. In *Proceedings KR-92.*

Veloso, M. 1992. *Learning by analogical reasoning in general problem solving.* Ph.D. Dissertation, Carnegie-Mellon University.