Merge Strategies for Multiple Case Plan Replay *

Manuela M. Veloso

Computer Science Department, Carnegie Mellon University
Pittsburgh, PA 15213-3891, U.S.A.
mmv@cs.cmu.edu, http://www.cs.cmu.edu/~mmv

Abstract. Planning by analogical reasoning is a learning method that consists of the storage, retrieval, and replay of planning episodes. Planning performance improves with the accumulation and reuse of a library of planning cases. Retrieval is driven by domain-dependent similarity metrics based on planning goals and scenarios. In complex situations with multiple goals, retrieval may find multiple past planning cases that are jointly similar to the new planning situation. This paper presents the issues and implications involved in the replay of multiple planning cases, as opposed to a single one. Multiple case plan replay involves the adaptation and merging of the annotated derivations of the planning cases. Several merge strategies for replay are introduced that can process with various forms of eagerness the differences between the past and new situations and the annotated justifications at the planning cases. In particular, we introduce an effective merging strategy that considers plan step choices especially appropriate for the interleaving of planning and plan execution. We illustrate and discuss the effectiveness of the merging strategies in specific domains.

1 Introduction, Related Work, and Motivation

Case-based planning and derivational analogy have been of interest to several researchers, who continue to investigate the singularities of using case-based reasoning in a planning framework. Many advances have been made in this context, building upon the pioneering CHEF [Hammond, 1986] and derivational analogy [Carbonell, 1986] work. CHEF showed how to explain plan failure and reason about failure for case indexing and retrieval. Derivational analogy introduced and showed the need to be reminded of the solution derivation rather than only of the final solution.

Several efforts have been following this line of research. Naming a few systems that address core planning problems helps to motivate this work and the interest in the area. PRIAR [Kambhampati and Hendler, 1992] notably formalizes and demonstrates the use of dependency links for plan reuse in hierarchical planning. Prodigy/Analogy [Veloso, 1994] develops the full derivational analogy approach and contributes an extensive analysis of the impact on planning efficiency of using the combination of case-based and state-space nonlinear planning. SPA [Hanks and Weld, 1995] is a simple and elegant interpretation of case-based plan adaptation, using SNLP as a plan-space planning approach. Using this same base-level planning approach and also building upon the

* This research is sponsored as part of the DARPA/RL Knowledge Based Planning and Scheduling Initiative under grant number F30602-95-1-0018. Thanks to Michael Cox and the anonymous reviewers for their comments on this paper.

Prodigy/Analogy approach, CAPLAN [Muñoz-Avila *et al.*, 1994] has been significantly extending the indexing and retrieval techniques and applying the paradigm to realistic domains, such as manufacturing. Similarly, DerSNLP [Ihrig and Kambhampati, 1994] is another successful implementation of derivational replay in SNLP. Several other systems, provide specific strong approaches to aspects of the case-based planning paradigm. For example, Dejà-vu [Smyth and Keane, 1995] shows how retrieval can use a prediction of the adaptation cost, PARKA [Kettler *et al.*, 1994] demonstrates massively parallel effective invocations to case memory during planning, and [López and Plaza, 1993] views medical diagnosis as a reactive planning task.

One of the interesting and less explored (or explained) aspects of the case-based planning paradigm is the use of multiple plans during the adaptation phase. In complex planning situations with multiple goals, a single past case that is similar to the complete new situation may not be found. However, several planning cases may be found that cover independent subparts of the new planning situation in a complementary way. Planning can then be viewed as the process of merging and adapting these multiple complementary planning cases. The effective use of multiple cases in planning is a challenging issue and is the focus of this work. This paper reports on our work in Prodigy/Analogy investigating and developing different plan merging strategies for analogical replay. A few other systems, such as [Redmond, 1990], and ASP-II [Alexander and Tsatsoulis, 1991] have addressed the use of multiple plans, although not necessarily in the replay or adaptation phase. An interesting recent effort in the Nicole system [Ram and Francis, 1996] explores the use of multiple *alternative* planning cases during reuse, as opposed to multiple *complementary* plans as carried out in Prodigy/Analogy.

This paper introduces several strategies to merge multiple planning cases during the adaptation phase (i.e., the replay phase in Prodigy/Analogy). These strategies are a refinement of the ones briefly discussed in [Veloso, 1994] within a re-implementation of Prodigy/Analogy integrated with the new Prodigy4.0 planner [Veloso *et al.*, 1995]. In analogical derivational replay, the merge algorithms are dependent on the underlying generative planning algorithm. In Section 2, we briefly introduce the Prodigy4.0 generative planner as the substrate planner of Prodigy/Analogy. We focus on explaining two main decision points of the planner and on showing the guidance that analogical replay can provide to improve planning efficiency. The remainder of the paper introduces different replay strategies for multiple plans. Section 3 sets the ground for the next sections by presenting serial replay as a simple strategy to replay a single case. Sections 4 and 5 present sequential replay and ordering-based interleaved replay, respectively. Section 6 introduces the novel choice-and-ordering-based interleaved replay. We illustrate our developed strategies in different domains. Finally, Section 7 concludes the paper with a summary of the contributions of the paper.

2 Improving Planning Efficiency: Replay of Planning Decisions

Planning is a complex task for which learning from past experience can improve planning performance along the three following dimensions: planning efficiency, task action model, and quality of plans generated [Veloso *et al.*, 1995]. Prodigy/Analogy is a case-based learning algorithm specifically designed to improve planning effi-

ciency. Therefore, it is important to understand what are the potential sources of planning inefficiency. Planning performance is dependent on the underlying planning algorithm and can vary with a great number of factors [Veloso and Blythe, 1994, Veloso and Stone, 1995]. Hence identifying universal causes of inefficiency for all domains and planning algorithms is not possible. We focus on explaining our planner's *decision points*, as opposed to its many other features, e.g., action representation, conditional planning, control knowledge, abstraction planning, or user interface.

2.1 Planning Decisions in Prodigy4.0

Prodigy4.0 combines state-space search corresponding to a simulation of plan execution of the plan and backward-chaining responsible for goal-directed reasoning. A formal description of Prodigy4.0 can be found in [Veloso *et al.*, 1995].

As opposed to the early state-space planners, such as Strips [Fikes and Nilsson, 1971] and Prodigy2.0 [Minton *et al.*, 1989], Prodigy4.0 performs a nonlinear state-space search by allowing the interleaving of subplans for different goals. At each point in its search, until the goal has been achieved, Prodigy4.0 faces both a set of goals that still need to be achieved and a set of plan steps (operators) already selected to achieve other goals. Some of these already selected plan steps may have all of its preconditions satisfied in the planner's state. When that is the case, Prodigy4.0 chooses between *applying* a plan step or continue planning for a pending goal, i.e., a goal that still needs to be planned for. Figure 1 shows these two main decisions.

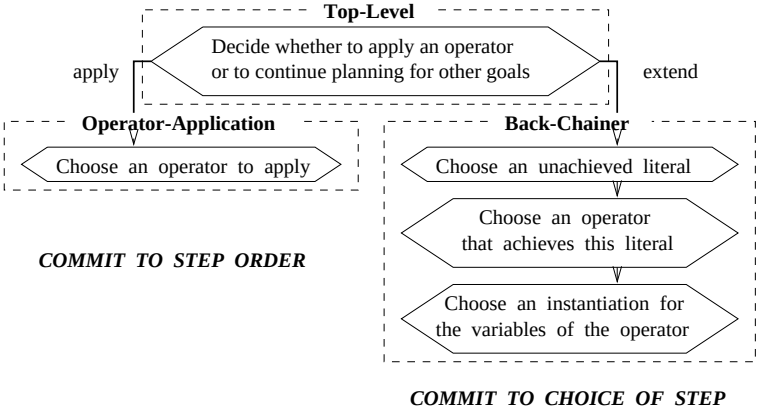

Fig. 1. Two Main Decisions while Planning: Step Order and Step Choice.

Applying an operator provides a new planning state. This decision is equivalent to a commitment in plan step order and therefore early simulation of plan execution while planning. Planning for a goal involves the selection of a specific step (instantiated operator) to achieve the goal. In some cases, selecting such a step can be easier, if updated state information can be taken into account. In a nutshell, we can view the planning search process depending on these two main choices, namely step ordering and choice of plan step. These two decisions are directly related to planning efficiency. For completeness, Prodigy4.0 can backtrack over all its choices, and eventually will generate

a solution plan, if one exists. However, the *planning efficiency*, i.e., the performance of the algorithm in its search, depends on the choices made during the search process.

2.2 Analogical Replay as Guidance to Planning Decisions

Analogical reasoning in Prodigy/Analogy achieves the integration of case-based reasoning and generative planning. It provides guidance for the planning choices and can therefore improve Prodigy4.0's planning efficiency. Essentially, Prodigy/Analogy introspects into the Prodigy4.0's planning episode after the search process to generate a plan, and generates a planning case by capturing several justifications for why choices are made. Case indexing includes the goal and the solution-relevant initial state (footprint). Retrieval compares new and past goals and problem states and returns a set of planning cases to *cover* the new planning situation. Analogical replay involves validating and replaying the retrieved case [Veloso, 1994].

In this section, we illustrate through a simple example the reduction in search effort that can be provided by analogical reasoning. Consider the planning domain introduced by [Barrett and Weld, 1994] and shown in Figure 2(a). We show a sample illustrative planning problem in this domain and the corresponding solution in Figure 2(b).

Operator:	A_i
preconds:	I_i
adds:	G_i
deletes:	$\{I_j, j < i\}$

(a)

Problem:
– Initial state: I_1, I_2, I_3, I_4, I_5
– Goal: G_2, G_3, G_4, G_1, G_5
– Solution: A_1, A_2, A_3, A_4, A_5

(b)

Fig. 2. Illustrative example: (a) Domain consists of N operators, each of the form A_i shown, $i = 1, \ldots, N$ [Barrett and Weld, 1994]; (b) Sample problem and solution.

This artificially-built domain can lead to a complex search, because there is a unique solution for each problem in an exponential search space. (It can be viewed as a search for a needle in a hay stack.) The complexity does not come from the choice of plan steps, as there is a single operator that can achieve each goal, e.g., goal G_3 is achieved (added) only by operator A_3. The complexity comes from finding the correct *step ordering*. When Prodigy4.0 uses the specific order in which goals are given and uses that ordering to eagerly commit to plan step orderings, it searches a large space.

Figure 3(a) shows part of the final branch from the Prodigy4.0's search tree while generating a solution to the problem introduced above. The trace shows a sequence of numbered choices of goals (represented in parenthesis), operators (represented within angle brackets), and applied operators (represented in upper-case letters in angle brackets). The interested reader can work out the problem and confirm the generation of the solution. (The trace further shows the depth of each search node – the first number on the line – and the number of alternative goals not explored at that search level – the information within the square brackets at the end of the line.) A major fact to notice, however, is that 305 nodes can be searched in this simple problem.

Prodigy/Analogy can store a solution derivation into a planning case. Figure 3(b) shows the trace of the search guided by a planning case that solves the same problem.

```
  2 n2 (done)
  4 n4 <*finish*>
  5   n5 (g2)  [g:4]
  7   n7 <a2>
  8   n289 (g1) [g:1]
 10   n291 <a1>
 11   n292 <A1>   [g:3]
 12   n293 <A2>   [g:3]
 13   n294 (g3)   [g:2]
 15   n296 <a3>
 16   n297 <A3>   [g:2]
 17   n298 (g4)   [g:1]
 19   n300 <a4>
 20   n301 <A4>   [g:1]
 21   n302 (g5)
 23   n304 <a5>
 24   n305 <A5>

Achieved top-level goals.
Solution:
        <a1>
        <a2>
        <a3>
        <a4>
        <a5>
#<PRODIGY result: t, 1 sol,
11.683 secs, 305 nodes>
```

(a)

```
  2 n2 (done)
  4 n4 <*finish*>
  5   n5 (g2)  "c5" 5   -- goal  (g2)
  7   n7 <a2>  "c5" 6   -- operator a2
  8   n8 (g1)  "c5" 289 -- goal  (g1)
 10   n10 <a1> "c5" 290 -- operator a1
 11   n11 <A1> "c5" 292 -- apply "c5" 291
 12   n12 <A2> "c5" 293 -- apply "c5" 7
 13   n13 (g3) "c5" 294 -- goal  (g3)
 15   n15 <a3> "c5" 295 -- operator a3
 16   n16 <A3> "c5" 297 -- apply "c5" 296
 17   n17 (g4) "c5" 298 -- goal  (g4)
 19   n19 <a4> "c5" 299 -- operator a4
 20   n20 <A4> "c5" 301 -- apply "c5" 300
 21   n21 (g5) "c5" 302 -- goal  (g5)
 23   n23 <a5> "c5" 303 -- operator a5
 24   n24 <A5> "c5" 305 -- apply "c5" 304
 End of current guiding case.
Achieved top-level goals.
Solution:
        <a1> (case "c5" 291)
        <a2> (case "c5" 7)
        <a3> (case "c5" 296)
        <a4> (case "c5" 300)
        <a5> (case "c5" 304)
#<PRODIGY result: t, 1 sol,
0.75 secs, 24 nodes>
```

(b)

Fig. 3. (a) Prodigy4.0 can search 305 nodes for a solution to the problem shown in Figure 2(b) (partial trace shown). This solution and its derivation is stored in a planning case,"c5"; (b) Analogical replay using "c5" guides planning for the same problem, finding immediately the solution with only 24 nodes searched. (The format of the traces was adapted for presentation.)

By replaying this planning case, all choices are revalidated successfully and search is completed avoided. This example illustrates the elementary case of search reduction provided by direct guidance. The search reduction is a result of the following two benefits provided by analogical replay:

- Proposal and validation of choices versus generation and search of possible alternatives operators and goal orderings.
- Reduction of the number of plausible alternatives – past failed alternatives are pruned by validating the failures recorded in the past cases.

In the most common and interesting situations of analogical replay, Prodigy/Analogy replays one or several cases in *similar* (and not the exact same) situation. We address next the complexity of the different replay strategies.

3 Serial Replay

The simplest replay situation involves using a single case and adapting it in a similar new situation. There are three kinds of possible differences between the new and the old situation, namely role (object), initial state, and goal differences.[2] Clearly, the most

[2] In [Wang and Veloso, 1994], we further consider situations in which the underlying available actions could be different between the past and the new situation. For the purpose of this paper, we assume a fixed set of actions.

common and challenging adaptation corresponds to the condition where we have a combination of different roles, initial states, and goals.

Consider the situation where the retrieval procedure returns a single case to cover the new planning roles, initial state, and goals. We introduce *serial replay* as the replay strategy with the following simple characteristics:

- A single case is retrieved.
- The past case is completely replayed before planning for any new goals.

This is clearly not a particularly involved replay technique to *merge* the past case into the new situation, as it postpones to the extreme reasoning about the differences between the case and the new situation. It represents a high degree of eagerness in terms of fully using the past case before considering the new situation. This technique has been implemented in a partial-order planner, DerSNLP [Ihrig and Kambhampati, 1994], as its replay strategy, where it showed to be appropriate.

Serial replay is potentially useful if the new goals correspond to minor changes to the retrieved case which can easily be performed *at the end* of the planning episode. The domain shown in Figure 4 illustrates this situation.

Operator:	A_i^1	A_i^2
preconds:	I_i	P_i
adds:	P_i	G_i
deletes:	$\{I_j, j < i\}$	$\{I_j, j \leq i\}$
		$\{P_j, j < i\}$

(a)

Problem:
– Initial state: $I_1, I_2, I_3, \ldots$
– Goal: $G_4, G_7, G_1, \ldots$

Solution: $A_1^1, A_2^1, A_3^1, \ldots, A_1^2, A_2^2, A_3^2, \ldots$

(b)

Fig. 4. Illustrative example: (a) Domain consists of $2N$ operators, N of the form A_i^1 and N of the form A_i^2 shown, $i = 1, \ldots, N$ [Barrett and Weld, 1994]; (b) Sample problem and solution.

We applied serial replay in this domain following the same experimental setup reported in [Ihrig and Kambhampati, 1994], namely, for each N-goal problem: (i) Solve and store an N-goal problem; (ii) Replay an N-goal case to solve a $N + 1$-goal problem using serial replay. We consistently achieved a large reduction in search space when comparing Prodigy4.0's eager step-ordering planning procedure against Prodigy/Analogy. An illustrative and representative sample of the results is shown in Table 1 for several 4-goal problems following different 3-goal case plans.

Serial replay backtracks in the case and returns to the case when the justifications (e.g. effects of the applied steps) are validated. The main advantage of replaying the $N - 1$ goal problem is the selection (with the inherent pruning of alternatives) of the right choices along all of the planning steps. Serial replay could be less efficient in situations that involve adaptation through the choices of plan steps instead of only step orderings and new step additions. In general, replay includes adding new steps, deleting old steps, and merging multiple planning cases.

Any 4-goal problem in Prodigy4.0 using eager step ordering (i.e., eager state changes) corresponds to searching $>$ 10000 nodes; (Using completely delayed commitments to step orderings for a problem with n goals corresponds to a search of $4(2n + 1)$ nodes.)

Problem	Guiding Case	Serial Replay	
		Nodes	Time (s)
p1-2-3-4	"case-p1-3-2"	58	1.95
p1-2-3-4	"case-p2-3-1"	58	1.98
p2-1-3-4	"case-p2-1-4"	65	2.3
p2-1-3-4	"case-p2-4-1"	65	2.3
p1-2-3-4	"case-p3-1-4"	75	2.566
p4-3-2-1	"case-p3-4-1"	75	2.666
p4-3-2-1	"case-p2-4-3"	85	2.933
p4-3-2-1	"case-p2-3-4"	85	2.867

Table 1. Sample table of results for solving 4-goal problems in the domain of Figure 4(a). The name of the problem and guiding case captures the goals in the problem and its order, e.g., p4-3-2-1 is a problem with goals G_4, G_3, G_2, G_1.

4 Sequential Replay

Adding new steps to a planning case is a common case adaptation found in analogical derivational replay. The reasons why steps need to be added to a planning case include the following two situations:

- New state misses precondition(s) of past applied step. For example, analogical replay finds a decision to apply a step with n preconditions; in the past case situation, there were $k < n$ preconditions true in the state; in the new state, there are m preconditions true, with $m < k < n$; extra planning is therefore needed to account for the extra $k - m$ unsatisfied preconditions.
- And the more interesting situation in which merging multiple plans requires adding new steps to combine individual cases.

Sequential replay is a technique that we developed to account for the situation where retrieval can provide information about the order in which multiple cases should be replayed. We developed this merge technique for the application of analogical replay to route planning [Haigh and Veloso, 1995, Haigh *et al.*, 1997]. The geometric characteristics of the map used by the retrieval procedure allow for the specification of an ordering between multiple planning cases. Sequential replay has the following features:

- Guiding cases are *ordered* by the retrieval procedure.
- Each case is replayed *sequentially* in the retrieved order.
- Merging occurs by planning *to connect* the planning cases.

The sequential replay algorithm attempts to validate each step proposed by the cases. Usually case steps are viable in the current situation, but two situations exist when a choice may not be viable:

1. when a case step is not *valid*, *i.e.*, when a choice made in the case is not applicable in the current situation, *e.g.* a route segment is closed for construction, and
2. when a step is *not reached yet*, *i.e.*, the next step is not yet in the set of adjacent reachable states from the current step, *e.g.* when there is a gap between cases.

Empirical results using a real map of the city of Pittsburgh [Haigh *et al.*, 1997] showed that the sequential replay is effective due to its three main features: its combination of retrieval of ordered situational-dependent similar past routing cases; its revalidation of the availability of the case segments used; and its ability to do extra planning when needed.

5 Ordering-Based Interleaved Replay

We introduce interleaved replay as a merge strategy that reasons about the decisions in the multiple planning cases before committing to a particular merge ordering. Interleaved replay does not have a pre-defined merge ordering and it considers the different past cases and the new situation in equal standing while making merging decisions.

Ordering-based interleaved replay is a strategy to merge planning cases that reasons specifically about plan step ordering commitments in the past cases. Ordering-based interleaved replay has the following features:

- Guiding cases are not ordered in any predefined order.
- Each case is replayed until a step ordering commitment is found.
- Planning is done for new goals until step ordering commitments are needed.
- Merging occurs by reasoning about the ordering constraints among different steps.
- As usual, new steps are added and old steps are deleted when needed.

Cases and new plan steps for new goals are ordered using the ordering dependencies illustrated shown in Figure 5.

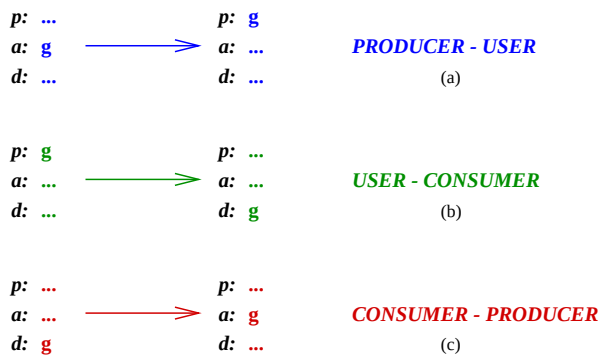

Fig. 5. Illustration of Plan Step Ordering Constraints. (p, a, and d represent the preconditions, adds and deletes of an operator, respectively.) For example, in (b) a step that needs a goal should precede a step that deletes that goal.

Deleting old steps occurs in the dual situations of the ones where new steps are found to be needed, namely:

- State provides precondition(s) of past applied step. For example, analogical replay finds a decision to apply a step with n preconditions; in the past case situation, there are $k < n$ preconditions true in the state; in the new state, there are m, preconditions true, with $k < m < n$; planning for for the $m - k$ unsatisfied preconditions in the

past case are not needed; replay deletes all the planning done dependent on the $m - k$ goals no longer necessary, by removing all the steps introduced for these goals; the annotated justifications at the case decisions provide the necessary linking information for this adaptation.

- And the more interesting situation in which merging multiple plans may identify repetition of steps in the different cases; analogical replay skips the repetition.

Figure 6 shows an illustration of the ordering-based interleaved replay procedure in the one-way rocket domain [Veloso, 1994].

```
 2 n2 (done)
 4 n4 <*finish*>
 5 n5 (at obj1 locb)               "case1" 5 -- goal (AT OBJ1 LOCB)
 7 n7 <unload-rocket obj1 locb r1> "case1" 6 -- operator UNLOAD-ROCKET
                                   "case1" 7 -- bindings OBJ1,LOCB,R1
  8     n8 (inside obj1 r1) case   "case1" 8 -- goal  (INSIDE OBJ1 R1)
 10     n10 <load-rocket obj1 r1 loca> "case1"  9 -- operator LOAD-ROCKET
                                       "case1" 10 -- bindings OBJ1,LOCA,R1
 11   n11 (at obj2 locb)           "case2" 5 -- goal  (AT OBJ2 LOCB)
 13   n13 <unload-rocket obj2 locb r1> "case2" 6 -- operator UNLOAD-ROCKET
                                       "case2" 7 -- bindings OBJ2,LOCB,R1
 14     n14 (inside obj2 r1)       "case2" 8 -- goal  (INSIDE OBJ2 R1)
 16     n16 <load-rocket obj2 r1 loca> "case2"  9 -- operator LOAD-ROCKET
                                       "case2" 10 -- bindings OBJ2,LOCA,R1
 17   n17 (at obj3 locb) unguided goal #<AT OBJ3 LOCB>
 19   n19 <unload-rocket obj3 locb r1>
 20     n20 (inside obj3 r1) unguided goal #<INSIDE OBJ3 R1>
 22     n22 <load-rocket obj3 r1 loca> [1]
 23     n23 <LOAD-ROCKET OBJ2 R1 LOCA> "case2" 11 -- apply "case2" 10
 24     n24 (at r1 locb) "case2" 12 -- goal  (AT R1 LOCB)
 26     n26 <move-rocket r1> "case2" 14 -- operator MOVE-ROCKET
                             "case2" 15 -- bindings R1
 27     n27 <LOAD-ROCKET OBJ1 R1 LOCA> "case1" 11 -- apply "case1" 10
 Goal (AT R1 LOCB): Goal causes a goal loop or is true in state.
 Marking all dependent steps to be skipped. Advancing case.
 28     n28 <LOAD-ROCKET OBJ3 R1 LOCA> apply unguided
 29     n29 <MOVE-ROCKET R1> "case2" 16 -- apply "case2" 15
 30   n30 <UNLOAD-ROCKET OBJ2 LOCB R1> "case2" 17 -- apply "case2" 7
 End of current guiding case.
 Switching to the last available case.
 31   n31 <UNLOAD-ROCKET OBJ1 LOCB R1> "case1" 17 -- apply "case1" 7
 End of current guiding case.
 31   n32 <UNLOAD-ROCKET OBJ3 LOCB R1> apply unguided
Achieved top-level goals.
Solution:
        <load-rocket obj2 r1 loca> ("case2" 10)
        <load-rocket obj1 r1 loca> ("case1" 10)
        <load-rocket obj3 r1 loca>
        <move-rocket r1> ("case2" 15)
        <unload-rocket obj2 locb r1> ("case2" 7)
        <unload-rocket obj1 locb r1> ("case1" 7)
        <unload-rocket obj3 locb r1>
#<PRODIGY result: T, 0.467 secs, 32 nodes, 1 sol>
```

Fig. 6. Illustration of Ordering-based Interleaved Replay. (The format of the trace was adapted for presentation.)

In this domain, objects can be loaded into and unloaded to a rocket carrier, which can only move once from the initial to the goal location. Two goals of a new 3-goal problem are solved guided by two cases and the third goal is unguided. The trace representation

is the same as used before. To note in the replay procedure are: the interleaved use of the two cases, case1 and case2, the deletion of repeated steps unnecessary steps after node n27, and the ordering-based switching points among the cases and the unguided goal (see nodes n11, n17, n23, n27, and n28).

6 Choice-and-Ordering-Based Interleaved Replay

Ordering-based interleaved replay reasons effectively about step-ordering commitments. As we noted earlier, however, step choice planning decisions may influence planning efficiency. Here we introduce choice-and-ordering-based interleaved replay, which goes beyond step orderings and reasons about the use of state information for step choices.

Figure 7 sketches the general reason why state information plays a role in step choices. Suppose an operator in the plan is applied and adds to the state some literal $p3$. (The effects of the operators, in particular if conditional or universal, are easily visible when an operator is applied.) Now the planner encounters a goal G for which there are 4 possible operators that can achieve it. If one of the operators needs the precondition $p3$, and the preconditions of the other operators are not satisfied in the state, then the planner may choose this operator because it does not need any further planning.

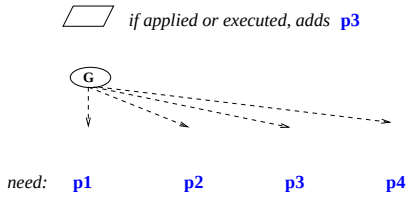

Fig. 7. State Information Guides Operator Choices

The choice of a plan step has been overlooked as a source of planning efficiency, but several concrete examples of its relevancy are introduced in [Veloso and Blythe, 1994]. Choice-and-ordering-based interleaved replay aims at using derivational analogy to provide guidance for this difficult planning decision.

We introduce a new dependency link to be annotated in a planning case: the *information-dependent step choice* link. This link is established if the choice of a plan step depends on the following condition: a previously applied or executed step adds (or deletes) information to the planning state that determines the choice of the plan step. This new kind of dependency link is in addition and contrasts to the ordering links based on interactions between preconditions and effects of planning operators (see Figure 5).

The choice-and-ordering-based interleaved replay merges cases by reasoning about the ordering and the information-dependent step choice constraints. The procedure that we are developing can perform the following functions:

- Consider effects of the result of operator application or execution and its impact in plan step choices.
- Use information-dependent and step ordering links to select merge order.
- Reconsider choices of plan steps, if justification for its selection is deleted.
- Use record of failed or untried alternatives as an opportunity for exploration.

This choice-and-ordering-based merge strategy is particularly appropriate for environments where planning and execution are interleaved and specifically in multi-agent

environments where multiple plans need to be coordinated. Execution can act as a source of information for the planner [Pryor and Collins, 1992, Stone and Veloso, 1996]. Choice-and-ordering-based interleaved replay aims at making use of the record in a planning case of the consequences of execution information in planning decisions.

7 Conclusion

This paper reports on our work in case-based reasoning applied to planning, in particular on analogical replay as an integration of generative and case-based planning. We have pursued extensive research in this area within the Prodigy/Analogy system. In this paper, we focus specifically on the issues involved in *replaying* multiple planning cases.

We introduce why the use of planning cases can reduce planning search and provide an improvement in planning efficiency. The contributions of the paper build upon this analysis for the introduction of several strategies that merge multiple planning cases.

We first introduce serial replay, in which a single case is replayed before planning for any new goals in the new planning situation. Serial replay is shown to be appropriate when the new situation is an extension of the past case. We then introduce sequential replay where multiple cases are merged according to a predefined ordering. New steps may be added and parts of the cases may be deleted to provide a suitable connection between the cases. For the general case where multiple cases are presented unordered to the replay procedure, we introduce ordering-based and choice-and-ordering-based interleaved replay. For these two merge strategies, the replay algorithm reasons about step orderings and information-dependent step choice commitments and constraints. Choice-and-ordering-based interleaved replay is built upon new information-dependent links that capture the dependency between the choice of plan steps and state information gathered from simulation or real execution. We briefly discuss that choice-and-ordering-based may be appropriate to multi-agent planning and execution environments. An extensive empirical analysis of the domain-dependent tradeoffs and suitability of the different merge strategies is part of our research agenda.

References

[Alexander and Tsatsoulis, 1991] Perry Alexander and Costas Tsatsoulis. Using sub-cases for skeletal planning and partial case reuse. *International Journal of Expert Systems Research and Applications*, 4-2:221–247, 1991.

[Barrett and Weld, 1994] Anthony Barrett and Daniel S. Weld. Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67:71–112, 1994.

[Carbonell, 1986] Jaime G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning, An Artificial Intelligence Approach, Volume II*, pages 371–392. Morgan Kaufman, 1986.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Haigh and Veloso, 1995] Karen Zita Haigh and Manuela M. Veloso. Route planning by analogy. In *Case-Based Reasoning Research and Development, Proceedings of ICCBR-95*, pages 169–180. Springer-Verlag, October 1995.

[Haigh *et al.*, 1997] Karen Z. Haigh, Jonathan Shewchuk, and Manuela M. Veloso. Exploring geometry in analogical route planning. *To appear in Journal of Experimental and Theoretical Artificial Intelligence*, 1997.

[Hammond, 1986] Kristian J. Hammond. *Case-based Planning: An Integrated Theory of Planning, Learning and Memory*. PhD thesis, Yale University, 1986.

[Hanks and Weld, 1995] Steve Hanks and Dan S. Weld. A domain-independent algorithm for plan adaptation. *Journal of Artificial Intelligence Research*, 2:319–360, 1995.

[Ihrig and Kambhampati, 1994] Laurie Ihrig and Subbarao Kambhampati. Derivational replay for partial-order planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 992–997, 1994.

[Kambhampati and Hendler, 1992] Subbarao Kambhampati and James A. Hendler. A validation based theory of plan modification and reuse. *Artificial Intelligence*, 55(2-3):193–258, 1992.

[Kettler *et al.*, 1994] B. P. Kettler, J. A. Hendler, A. W. Andersen, and M. P. Evett. Massively parallel support for case-based planning. *IEEE Expert*, 2:8–14, 1994.

[López and Plaza, 1993] B. López and E. Plaza. Case-based planning for medical diagnosis. In J. Romorowski and Z. W. Ras, editors, *Methodologies for Intelligent Systems (Proceedings of ISMIS'93)*. Springer Verlag, 1993.

[Minton *et al.*, 1989] Steven Minton, Craig A. Knoblock, Dan R. Kuokka, Yolanda Gil, Robert L. Joseph, and Jaime G. Carbonell. PRODIGY 2.0: The manual and tutorial. Technical Report CMU-CS-89-146, School of Computer Science, Carnegie Mellon University, 1989.

[Muñoz-Avila *et al.*, 1994] Héctor Muñoz-Avila, Juergen Paulokat, and Stefan Wess. Controlling a nonlinear hierarchical planner using case-based reasoning. In *Proceedings of the 1994 European Workshop on Case-Based Reasoning*, November 1994.

[Pryor and Collins, 1992] Louise Pryor and Gregg Collins. Cassandra: Planning for contingencies. Technical report, The Institute for the Learning Sciences, Northwestern University, 1992.

[Ram and Francis, 1996] Ashwin Ram and Anthony G. Francis. Multi-plan retrieval and adaptation in an experience-based agent. In David B. Leake, editor, *Case-Based Reasoning: experiences, lessons, and future directions*, pages 167–184. AAAI Press/The MIT Press, 1996.

[Redmond, 1990] Michael Redmond. Distributed cases for case-based reasoning; Facilitating the use of multiple cases. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 304–309, Cambridge, MA, 1990. AAAI Press/The MIT Press.

[Smyth and Keane, 1995] Barry Smyth and MArk T. Keane. Experiments on adaptation-guided retrieval in case-based design. In M. Veloso and Agnar Aamodt, editors, *Case-Based Reasoning Research and Development*, pages 313–324. Springer Verlag, October 1995.

[Stone and Veloso, 1996] Peter Stone and Manuela M. Veloso. User-guided interleaving of planning and execution. In *New Directions in AI Planning*, pages 103–112. IOS Press, 1996.

[Veloso and Blythe, 1994] Manuela M. Veloso and Jim Blythe. Linkability: Examining causal link commitments in partial-order planning. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 170–175, June 1994.

[Veloso and Stone, 1995] Manuela M. Veloso and Peter Stone. FLECS: Planning with a flexible commitment strategy. *Journal of Artificial Intelligence Research*, 3:25–52, 1995.

[Veloso *et al.*, 1995] Manuela M. Veloso, Jaime Carbonell, M. Alicia Pérez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[Veloso, 1994] Manuela M. Veloso. *Planning and Learning by Analogical Reasoning*. Springer Verlag, December 1994.

[Wang and Veloso, 1994] Xuemei Wang and Manuela M. Veloso. Learning planning knowledge by observation and practice. In *Proceedings of the ARPA Planning Workshop*, pages 285–294, Tucson, AZ, February 1994.

This article was processed using the LaTeX macro package with LLNCS style