Planning for Complex Tasks: Replay and Merging of Multiple Plans

Manuela M. Veloso *
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
veloso@cs.cmu.edu

Abstract

This paper discusses the analogical/case-based planning approach developed in PRODIGY-/ANALOGY. The technique involves planning for complex problems by the multi-case reuse of simpler past plans that are conjunctively similar to a new planning situation. The paper demonstrates this effective incremental planning strategy by providing empirical results on the performance of PRODIGY/ANALOGY solving 1000 problems in a complex logistics transportation domain.

1 Introduction

Planning for complex tasks is a well-recognized difficult problem due to the size of the search space which in general increases exponentially with the problem complexity. On the opposite planning for simple problems is a rather tractable task. The interesting question is: How can a planning system solve efficiently complex problems given the fact that it can solve efficiently simple problems?

An answer to this question is to elaborately refine the knowledge available to the planner in such a way that the search is drastically reduced. On one hand, the developer may laboriously handcode and refine the domain representation until the planner is able to produce the desired plans for the complex problems. Another approach is to automate the learning process from the planning experience in simple problems and compile the acquired knowledge to reuse in future complex problems. In this paper we discuss the planning aspect of an analogical/case-based reasoning variation of this general learning strategy that we developed in PRODIGY/ANALOGY [Veloso, 1992]. The incremental learning and problem solving process has the following phases:

1. Let the planner solve simple problems.
2. Store the generated plans (annotated with their derivations) indexed by the goal interactions encountered and the relevant features of the initial state.
3. Let the planner solve complex problems by reusing multiple stored plans that are jointly similar to the new problem. Loop to steps 2 (and also 1, if needed).

The planning and machine learning characteristics of this strategy are, on one hand, its case-based planning approach by which past plans are reused to guide the planning process. Additionally, the past plans are retrieved when similar to the new situation rather than on an exact-match requirement. This paper focus on the discussion of the planning algorithm based on the replay (or reuse) of past plans. The power of the replay strategy in solving complex problems with a large number of goal conjuncts is in part due to the ability to merge the guidance from multiple past cases.

Further details on the replay strategy developed, as well as complete description of the other aspects of the analogical reasoning process, namely the generation, storage, and retrieval of cases, can be found in [Veloso, 1992].

2 Replay of multiple guiding cases

When the planner is given a new problem to solve described in terms of the initial state and the conjunctive goal statement, the system retrieves from the case library a set of similar past cases that cover the new problem solving situation.

A case in memory is the derivational trace of the planning episode of solving a problem. The system automatically identifies the sets of interacting goals by partially ordering the totally ordered solution found [Veloso *et al.*, 1990]. The connected components of the partially ordered plan determine the independent fragments of the case each corresponding to a set of interacting goals. Each case is multiply indexed by the sets of interacting goals.

*This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

The relevant literals of the initial state are foot-printed for each goal conjunct in the goal statement by goal regressing through the plan found. Several learning methods share the explanation provided by the subgoaling chain supplied by the underlying domain theory. In that sense, foot-printing is similar to explanation-based indexing techniques [Barletta and Mark, 1988, Hickman and Larkin, 1990] and chunking [Laird *et al.*, 1986]. A distinction between the methods is the level of generalization, abstraction, or scope of the explanation obtained. Foot-printing explains the episodic final solution while chunking explains each problem solving impasse. Explanation-based indexing, as used in [Barletta and Mark, 1988], uses goal regression to abstract domain features from the instantiated observables defining a solution episode. Foot-printing uses goal regression to reduce the set of instantiated features of the initial state.

Each case is retrieved as a guiding case for a set of interactions goals from the goal statement. Therefore each case *covers* a set of goals. At replay time, a case is *abandoned* when all the goals it covers are achieved. Until all the covered goals are achieved, the corresponding guiding case is always considered as a source of possible guidance and the problem solver keeps it active. The covered goals may be achieved by transferring all the steps of the guiding case or there may be local or global divergences. If a divergence is found, the guiding case stays active but suspended at the diverging step. The replay algorithm continues to test for additional transfer. When a local divergence is resolved the transfer continues successfully. If the divergence is global, the case remains suspended and the problem solver is not able to return back to it until all the covered goals are achieved by different means. At this point the suspended case is abandoned as its covered goals are achieved.

Figure 1 sketches the general picture of the replay mechanism following guidance from several past cases. The new planning episode is shown in the center of the figure with the steps being transferred (shown by arrows) from the past plans on the sides. The picture shows that: *(i)* the past cases can be reused only partially, as some steps are not transferred to the new situation; *(ii)* the cases can be merged in different orders; and *(iii)* the replay mechanism has the ability to replan for parts of the problem that are not covered by past cases (as shown by the unguided steps of the new plan).

The general replay mechanism involves a complete interpretation of the justification structures annotated in the past cases in the context of the new problem to be solved, and the development of adequate actions to be taken when transformed justifications are no longer valid.

We followed a satisficing paradigm in which the system is fully guided by its past experience. The syntactic applicability of an operator is always checked by simply testing whether its left hand side matches the current state. Semantic applicability is checked by determining whether the justifications hold (e.g., whether there is still a reason to apply this operator). For all the choice points, the problem solver tests the validity of the justifications (its semantic applicability, or rather its "desirability" in the new situation). In case the choice remains valid in the current problem state, it is merely copied, and in case it is not valid the system can replan at the particular failed choice or re-establish the failed condition.

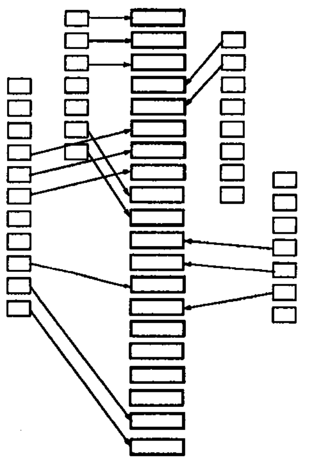
Figure 1: Replaying multiple cases

These two situations can be described in terms of their effect in how the cases are followed. When the justifications hold, the past choices are transferred to the new context. The cases are advanced to propose to the next potentially useful steps. When the justifications are not valid, then any of the two alternatives described above may correspond to the following actions in the guiding cases:

1. Suspend the guiding case if some extra planning work is needed. For example, this corresponds to the situation where an operator was applied in the past case, and now in the new problem, it cannot be applied yet, as one of its preconditions is not true in the state yet. The replay procedure diverges from the guiding case and replans for these new goals (it can recursively try to find another case that can guide the reachievement of the particular preconditions).
2. Advance the guiding case when some of the past planning work is not necessary. For example, this corresponds to the situation where the past case subgoals in a literal that is now already true in the state. The replay procedure tries to advance the case to the next step that can be replayed.

Deviations theoretically could lead to total divergence from the set of guiding cases. This does not occur however when the adequate foot-print similarity metric is used. The foot-printed initial state which is compared to the new situation captures the relevant features of the initial state in terms of the goals to be achieved and as a function of the solution to be replayed. While the case library is not rich enough in a diversity of cases, the retrieval procedure generally returns a smaller set of guiding cases rather than a larger set of not suitable ones.

Justification structures also encompass the record of past failures in addition to just the subgoaling links [Bhansali, 1991, Kambhampati, 1989, Mostow, 1989]. This allows both the early pruning of current alternatives that were experienced to have failed in the past, and the exploration of alternatives for which the past reason of failure does not exist in the current situation. Furthermore, the replay mechanism in the context of casual commitment as opposed to least commitment allows naturally to combine guidance from several past problem solving episodes. Replicated adapted decisions can be interleaved and backtracked upon within the totally ordered reasoning plan.

2.1 Advantages of replay

The replay functionality transforms the planner, from a module that costly generates possible operators to achieve the goals and searches through the space of alternatives generated, into a module that tests the validity of the choices proposed by the past experience and follows equivalent search directions.

In a nutshell and informally, the replay procedure provides the following benefits to the problem solving procedure:

- Proposal and validation of choices versus generation and search of possible alternatives.
- Reduction of the branching factor – past failed alternatives are pruned up front by validating the failures recorded in the past cases.
- Subgoaling links identify the subparts of the case to replay – the steps that are not in the subgoaling chain of active goals are skipped.

2.2 Merging strategies

The replay procedure works its reconstruction mechanism from a set of guiding cases as opposed to necessarily a single past case. Each guiding case suggests a guiding plan to a learned set of interacting goals. This enhancement constitutes a powerful technique to get guidance from complementary individual past cases. The replay of multiple cases proves to be highly useful for complex problems that may be solved by resolving minor interactions among simpler past cases. Following several cases however poses an additional decision making step of choosing which case to pursue. Resolving at this level of decision making may be seen as an instance of meta-level reasoning in a higher level of abstraction than the domain level decisions, such as which operator to apply, or which goal to pursue next in order to solve a user given problem situation. Although developing a learning method for meta-level decision making is beyond the immediate focus of this work, we discuss a few different merging strategies to merge the guidance from the several cases from the set of similar cases:

Serial: The cases are merged serially one after the other. The particular initial merging ordering of cases is randomly chosen. When all the steps of a case are reused or the case is abandoned then the next case in the serial order is returned and followed.

Round-robin: This is an opposite strategy to the serial one. The cases are maximally interleaved by following a step of each case at a time. The particular initial merging ordering of the cases is also randomly chosen. The system returns the next case in the merging ordering after the case that is linked to the current active search node or after the last guided search node, if the current node is unguided.

Exploratory: Finally this strategy merges the cases in a random order. The procedure returns a case arbitrarily chosen from the set of guiding cases.

It is interesting to briefly discuss these different merging strategies. The question to be addressed is twofold: Which of the merging strategies is more advantageous to help reduce the problem solving base search effort? And which of the merging strategies allows the learner to accumulate richer cases in terms of the interactions among goals? The learning question is of central relevance to our work as the purpose of our method is exactly to learn, e.g. new goal interactions, from planning experience. It is beyond the focus of this work to develop merging techniques to optimize the quality of the resulting plan [Nau *et al.*, 1990].

To debate the trade-offs raised by the two questions above, consider the two extreme situations in terms of goal interactions, namely:

A, where the set of goals covered by the different guiding cases are all independent from each other, and

B, where there are strong interactions among the goals covered by the different cases.

In terms of the expected reduction of the problem solving search effort, for situation **A** all the merging strategies are equivalent as the goals do not interact. On the other hand, for situation **B**, the merging strategy used produces fundamentally different results in search reduction. A serial strategy delays to an extreme the detection of goal interactions. A round-robin strategy may be able to spot the goal interactions rather early and contribute to avoid long undesirable serial search paths. This strategy provides the maximum benefits but only if the correct initial case ordering is selected. The exploratory strategy balances these two strategies by allowing cases to both be serialized or interleaved.

In terms of the accumulation of a wide variety of cases, the learner masters from a rich problem solving experience. Ideally the learner benefits most from an integral understanding of the complete search space as a result of its entire exploration by the problem solver identifying all the failing and succeeding paths. This situation however is not desirable in terms of problem solving efficiency. For both situations **A** and **B**, the problem solver ends up finding the correct solution after the necessary search. The learner captures and compiles the existing goal interactions. The issue is which of the strategies allows a richer exploration of the search space to learn from success and failures. The serial merging strategy is indifferent for situation **A** for both search reduction and learning usefulness. For situation **B** both the serial and the round-robin strategies are equally useful from the learning perspective, as they depend heavily on the initial case ordering. In a nutshell this discussion leads into the conclusion that the exploratory strategy secures the trade-off between handling situations **A** and **B** successfully both from the search reduction and learning utility point of views.

3 Empirical Results

The thesis reports extensive empirical results along a variety of dimensions. In this paper we focus on showing the cumulative planning times and the comparisons between the solution length of the plans for the system without analogy and with analogy. In particular, details about how the experiments were conducted and the cost of retrieval can also be found in the thesis. In the experiments reported, the merging strategy is fixed to be the exploratory one. The choices are picked up randomly from the set of available alternatives, if no additional guidance can be applied.

The problem solver in both configurations runs within a time limit. We ran a set of 1000 problems in a logistics transportation domain with 1-20 goals in the goal statement and more than 100 literals in the initial state.

The overall result of this planning and learning approach is that the analogical reasoner increased the solvability horizon of the base-level problem solver considerably. PRODIGY/ANALOGY can solve the complete set of 1000 problems with up to a running time limit of 350 seconds. Without analogy, i.e., by base search, the planner, NOLIMIT [Veloso *et al.*, 1990], solves only 458 problems out of the 1000 problems even when the search time limit is increased up to 350 seconds.

Previous comparisons between the performance of a problem solver before and after learning control knowledge [Minton, 1988, Knoblock, 1991, Etzioni, 1990] were done by graphing the cumulative running times of the two systems over a set of problems.[1] To follow this precedent we also graph the cumulative performance of the two systems.

Figure 2 (a) shows the cumulative running time for the set of problems (458) that were both solved by base search and by analogy. The curves are monotonically increasing because of the cumulative effect, and they are smooth because the problems are sorted according to their running time.

The graph shows a final factor of 3.6 cumulative speed up of the analogical problem solver over the base NOLIMIT. The maximum individual speed up is of a factor of approximately 38. The graph compares the running times for the solved problems. To make this comparison more similar to the ones performed previously in PRODIGY [Minton, 1988], we compute the cumulative running times accounting also for the problems not solved by the base level problem solver within the time bound of 350 seconds. Therefore for each unsolved problem, we add the running time until the time bound limit is reached, in the same way as it is done in [Minton, 1988]. Figure 2 (b) shows the curves obtained.

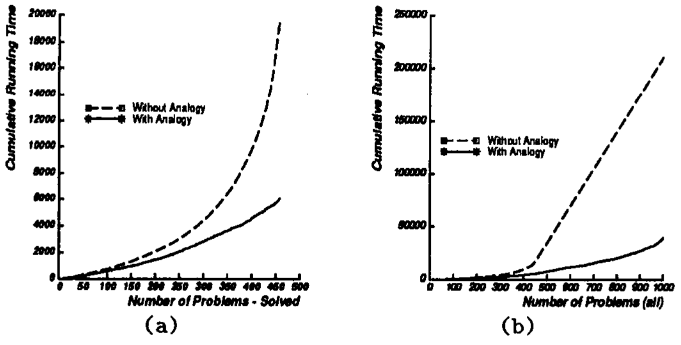

Figure 2: (a)Cumulative running time for the 458 problems from a set of 1000 problems solved both by base-level search (without analogy) and by derivational analogy (with analogy); (b) Cumulative running time for the complete set of 1000 problems. If a problem is not solved it is accounted for with the CPU time limit used of 350 seconds.

The 1000 problems solved by analogy correspond to a total of 39,479.11 seconds, while the total running time effort of the base level problem solver corresponds to 210,985.87 seconds. This represents a speed-up of a factor of approximately 5.3, and also means that the cumulative savings in running time for analogy is approximately 81.3%.

No direct comparison between earlier PRODIGY/EBL and current PRODIGY/ANALOGY is possible because the former used a linear problem solver whereas the latter used a nonlinear one. Moreover the complexity of the problems was substantially greater for PRODIGY/ANALOGY. These factors mitigate towards a larger overall search space for the current work and therefore more room for learning, as observed with respect to improved average running time and solvability boundary.

Another interesting issue is to compare the quality of the solutions produced by analogy and the ones returned by the base NOLIMIT. This study uses a measure of quality of plans which is based simply on the length of the solution.[2]

The study is done by finding the difference between the length in the solutions found by NOLIMIT and by analogy for each problem. Figure 3 shows a table summarizing the results found.

The immediate result from this table is that in 82.75% (36.68% + 46.07%) of the solved problems the analogical reasoner produces plans of no worst quality than the ones produced by base-level search. In terms of the total 1000 solved problems by analogy, in only 7.9% of the problems (79/1000) does analogy produce longer plans.

[1]The next sections consider the retrieval times in addition to the running time for the analogical runs.

[2]In [Pérez, 1992] Pérez proposes to research in acquiring control knowledge from an expert to guide the problem solver to achieve plans of higher quality according to several dimensions.

Difference in solution length: ANALOGY versus NoLimit	Number of Problems	Percentage of problems
Longer 1-6 steps	79	17.25%
Equal	168	36.68%
Shorter 1-13 steps	211	46.07%

Figure 3: Comparison in solution length between the analogical and the base-level problem solvers

Before we ran this comparison, we had not a clear feeling of what the outcome of this study would be. In fact we feared an eventually more balanced or even disadvantageous result for analogy. The reason for this expectation (which turned out to be ungrounded) is the exploratory strategy that we follow to merge the guidance from several cases at replay time. We chose to follow the principle that a learner benefits more from random exploration of its choices, if no preferences are available, than from following always a fixed exploration order. In particular this principle applies to the replay of multiple cases in the random interleave of the several guiding cases when no other preferred choice is known. Hence the exploratory merging strategy leads to novel explorations of the search space allowing the problem solver to encounter "surprising" successes or failures from which it can learn by enriching its library of problem solving experience. Though supported by this learning argument, it was not clear to us what were the effects of the approach in the quality of the specific final solution delivered by the analogical problem solver. The results in Figure 3 show the rewarding fact that the overall replay algorithm of multiple guiding cases produces solutions of equal or better quality in a large majority of the situations.

4 Conclusion

The paper advocates a strategy for efficiently planning in complex tasks. The technique is based on the reuse and merging of multiple past simple plans that are similar as a whole to the new problem solving situation. The method is implemented in PRODIGY/ANALOGY and the paper shows results that empirically validate the method.

Acknowledgements

The author would like to thank Jaime Carbonell for many helpful discussions and suggestions, and the anonymous reviewers for their comments.

References

[Barletta and Mark, 1988] Ralph Barletta and William Mark. Explanation-based indexing of cases. In *Proceedings of the First Workshop on Case-Based Reasoning*, pages 50–60, Tampa, FL, May 1988. Morgan Kaufmann.

[Bhansali, 1991] Sanjay Bhansali. *Domain-based program synthesis using planning and derivational analogy.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1991.

[Etzioni, 1990] Oren Etzioni. *A Structural Theory of Explanation-Based Learning.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1990. Available as technical report CMU-CS-90-185.

[Hickman and Larkin, 1990] Angela K. Hickman and Jill H. Larkin. Internal analogy: A model of transfer within problems. In *The 12th Annual Conference of The Cognitive Science Society.* Lawrence Erlbaum Associates, 1990.

[Kambhampati, 1989] Subbarao Kambhampati. *Flexible Reuse and Modification in Hierarchical Planning: A Validation Structure Based Approach.* PhD thesis, Computer Vision Laboratory, Center for Automation Research, University of Maryland, 1989.

[Knoblock, 1991] Craig A. Knoblock. *Automatically Generating Abstractions for Problem Solving.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991. Available as technical report CMU-CS-91-120.

[Laird *et al.*, 1986] John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

[Minton, 1988] Steven Minton. *Learning Effective Search Control Knowledge: An Explanation-Based Approach.* PhD thesis, Computer Science Department, Carnegie Mellon University, 1988.

[Mostow, 1989] Jack Mostow. Automated replay of design plans: Some issues in derivational analogy. *Artificial Intelligence*, 40(1-3), 1989.

[Nau *et al.*, 1990] Dana S. Nau, Qiang Yang, and James Hendler. Optimization of multiple-goal plans with limited interaction. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, pages 160–165, San Diego, CA, November 1990. Morgan Kaufmann.

[Pérez, 1992] M. Alicia Pérez. Learning from experts knowledge to improve the quality of plans. Thesis proposal, School of Computer Science, Carnegie Mellon University, 1992.

[Veloso *et al.*, 1990] Manuela M. Veloso, M. Alicia Pérez, and Jaime G. Carbonell. Nonlinear planning with parallel resource allocation. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, pages 207–212, San Diego, CA, November 1990. Morgan Kaufmann.

[Veloso, 1992] Manuela M. Veloso. *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1992. Available as technical report CMU-CS-92-174.