Case-Based Plan Adaptation: An Analysis and Review[1]

Héctor Muñoz-Avila[1] and Michael T. Cox[2]

[1]Department of Computer Science and Engineering
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015, USA
munoz@cse.lehigh.edu

[2]Intelligent Distributed Computing
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
mcox@bbn.com

Abstract: In this article we analyze the current state of case-based plan adaptation research. We include the traditional distinction between transformational and derivational analogy but add further dimensions to classify the field. We present six dimensions for categorizing various aspects of existing case-based plan adaptation algorithms. These dimensions are: the type of transformation, the role of the case, the case content, the use of case merging, the representation formalism, and the computational complexity of the algorithm. We use these dimensions as a framework to compare various systems. Our analysis clarifies some common misconceptions about plan adaptation.

1. Introduction

Case-based planning (CBP) is a problem-solving method that uses a library of cases, where a case associates a past problem and goal description with a plan that solves the problem by achieving the goal (Hammond, 1989). Given a new problem, CBP systems retrieve one or more cases whose problem is similar to the current one and adapt the plans contained in the retrieved cases to achieve the new goal. Case retrieval involves intelligent search of the case library to find cases that are adequate to solve the new problem. The result of the plan adaptation process is a plan that includes parts that are derived from the cases (e.g., plan steps copied form the cases) and parts derived by other means (e.g., first-principles planning). Although important relationships exist between the technology that measures similarity during case retrieval to assure adaptable solutions and the technology that performs effective adaptation during case reuse (e.g., Lopez de Mántaras *et al*., 2006), here we concentrate the analysis upon the adaptation process itself.

CBP was one of the earliest techniques developed by the case-based reasoning (CBR) community. For example, the CHEF system (Hammond, 1989; 1990) dates

[1] This research was in part supported by the National Science Foundation (NSF 0642882)

from the formative stages of CBR. CHEF constructs cooking recipes that are viewed as plans, because recipes constitute sequences of cooking steps such as adding ingredients. Since then, CBP has been a frequent research topic that includes a wide range of application areas such as manufacturing (Costas & Kashyap, 1993), military planning (Veloso, Mulvehill & Cox, 1997), route planning (Haigh, Shewchuk & Veloso, 1997), academic course scheduling (Grech & Main, 2004), and medicine (Salem, Nagaty, & El Bagoury, 2003; Schmidt, Vorobieva, & Gierl , 2003).

Here we present a retrospective on plan adaptation. Focusing on CBP for this study will prove valuable, because previous breakthroughs in CBP have impacted other CBR areas. For example, studies on derivational analogy (Carbonell, 1986), first made in the context of plan adaptation, have been used in tasks such as tutoring systems (Weber & Schult, 1998) that fall outside the immediate scope of planning research.

The purpose of this article is to present an overview of the state-of-the art in plan adaptation rather than an exhaustive literature review (for the latter see Cox, Muñoz-Avila, & Bergmann, 2006 and Spalazzi, 2001). Although we will continue to cite several CBP systems, our focus is the analysis of the state-of-the-art according to several dimensions that will give the reader an understanding of the main issues regarding plan adaptation. We use the following six dimensions in our analysis.

- **Adaptation type**. This dimension refers to the type of plan adaptation used: transformational or derivational analogy.
- **Case role**. This dimension indicates whether the cases contributes domain knowledge and/or control knowledge.
- **Case contents**. This dimension characterizes the contents of the case used during adaptation. Although this typically corresponds to the solution plan, we will show that other contents have been used.
- **Case merging**. This dimension refers to whether one or multiple cases are used to solve the new problem. When more than a single case forms a solution, alternative methods exist to merge the cases.
- **Representation formalism**. This dimension refers to the symbolic representation of the case contents. For example, some plans use a hierarchical representation while others use partially ordered plans.
- **Computational complexity**. We also provide additional perspective on formal results concerning plan adaptation complexity. We describe their impact on transformational and derivational adaptation methods.

The rest of the article continues as follows. In Section 2, we present a case study to illustrate the adaptation formula and discuss the formula's relation with transformational and derivational adaptation. In this section we also analyze some case-based planners according to our first dimension (i.e., adaptation type). Sections 3-7 present each of the remaining five dimensions and analyze them according to the adaptation formula. Finally, Sections 8 and 9 discuss future research directions and final remarks. Before continuing, we note that more than eighty publications exist on

the subject of CBP. Instead of reviewing all of them, our study presents a framework to categorize them.

2. Abstract Example

Before presenting the various dimensions, we briefly describe a simple example of plan adaptation in a travel domain. This example is reminiscent of how plan adaptation is performed in Prodigy/Analogy (Veloso, 1994), but we will keep it independent of any particular system. This example will help us to discuss the various dimensions subsequently.

A planning problem typically requires a system to achieve a set of specific goals. For example, if we are preparing a trip, we may specify as goals the destination location, and how much money we wish to retain. The problem also specifies conditions such as known routes between various locations, the starting location, the available means of transportation (e.g., bus, car, airplane) and costs of using the various means of transportation for the various routes.

A solution plan is a sequence of steps achieving the goals under the given conditions. In our travel scenario, a plan may first take a taxi from the starting destination to a local airport and then take a flight to another location. The plan will go on until the last step reaches the destination.

Solving a problem by plan adaptation involves selecting one or more existing plans and then modifying them to achieve the current problem goals. In our trip scenario, existing plans will indicate how passengers transit between various locations and the costs of these trips. Therefore, a generated plan will contain a route that is divided in various segments. Some of these segments may be derived from cases and other segments may be derived by other means (such as first-principles planning). Segments derived from cases may simply be copied as a whole without changes from the case or may be copied partially. For example, the original case was going from Cambridge to San Jose, but during the adaptation we only used an intermediate sub-segment from Cambridge to San Francisco, or the plan may have been transformed after copying it. So for example we may have modified the transportation means (e.g., we took the subway instead of the taxi, because it is cheaper). Additionally because we are low on money, we add a step to visit a bank machine. Figure 1 sketches a resulting plan.

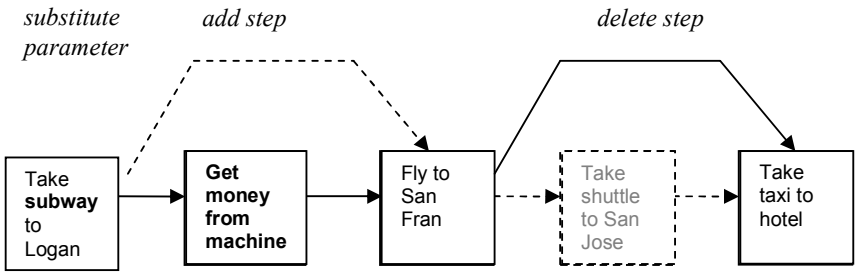

Figure 1 Sketch of a plan obtained by adaptation.

3. Adaptation Type

Case adaptation can be classified into two basic categories: transformational and derivational analogy. In transformational analogy, cases contain the solution plans for previous problems. These plans are reused in the new situation by making suitable changes where appropriate (Carbonell, 1983; van der Krogt & de Weerdt, 2005). *Transformation operators* are knowledge constructs prescribing how to transform existing plans into new ones. In the CBP system CHEF, for example, a transformational operator will indicate that raspberries will be a good substitute for strawberries when is the latter are not available. Therefore, applying this operator to a recipe (i.e., cooking plan) for strawberry soufflé will result in a raspberry soufflé recipe. Transformational analogy does not consider how the reused plan was originally obtained. Instead, it examines only the plan itself. In contrast, cases in derivational analogy contain a *derivational trace* rather than a plan as in transformational analogy. That is, the trace is a sequence of computational steps a planner followed to generate a plan. Derivational analogy provides more flexibility, because the planner can replay the derivational trace relative to the new problem. That is, no transformational operators are required. However, derivational analogy does requires the derivational trace to be known.

Figure 2 contrasts a plan and a derivational trace. The left side of the figure shows a simple plan in the logistics transportation domain (Veloso, 1994). The middle section lists a possible derivational trace for the plan. This trace corresponds to the one that a partial order case-based planner (e.g., CAPlan/CbC (Muñoz-Avila & Weberskirch, 1996), derSNLP (Ihrig & Kambhampati, 1994)) would have generated (see Section 5 for a description of partial order planners). The right side illustrates a portion of the rationale for the derivation (as adapted from Veloso, 1994, using Prodigy/Analogy). Rather than adapting the plan directly, the derivation and rationale allows the planner to reuse portions of the prior search.

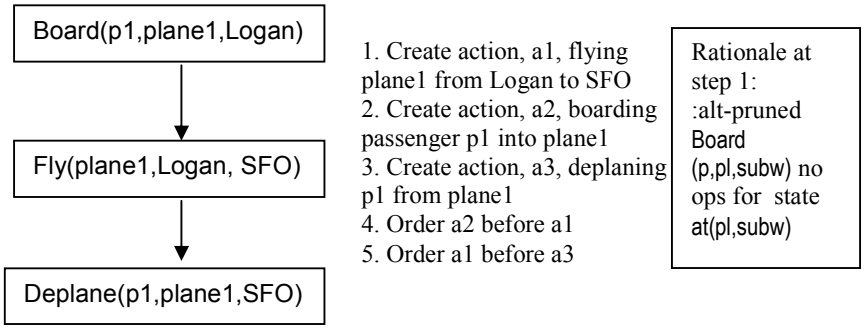

Figure 2 Example of a solution plan (left) and a derivational trace (right) generating the plan.

For example the prior planning may have tried to establish variable bindings for the Board operator using the subway as the location from which the person boarded the plane instead of the airport. The previous search had to backup from this choice, because the plane cannot use the subway as a destination for the Fly operator. To adapt this reasoning to the new situation, the planner can simply check the state to see if the plane is currently located in the subway, otherwise it can automatically avoid the unnecessary search by using an airport as a binding for where the boarding action takes place, not a subway.

Prodigy/Analogy typifies all derivational CBP systems, which includes derSNLP (Ihrig & Kambhampati, 1994) and CAPlan/CbC (Muñoz-Avila & Weberskirch, 1996) among others. Another system combining transformational in this category is CLAM (Melis & Whittle, 1999). CLAM constructs analogy-driven proof plans. Derivational analogy is used to reformulate the source plans and to apply case replay. CLAM uses first-principles to complete the unsolved goals. Other systems in this category include Paris and HICAP. Paris (Bergmann & Wilke, 1995) also uses first-principles planning. The distinguish characteristic of Paris is that it stores and reuses abstract plans. Abstract plans use actions that have a higher level of granularity than concrete plans. So for example, in blocks world, abstract plans may talk about piles of blocks rather than individual blocks as in the concrete plans. Another system in this category is HICAP, which is an interactive hierarchical CBP system (Muñoz-Avila et. al., 1999). Similar to adaptation by derivational analogy, HICAP extends the retrieved cases by using first-principles planning, although its cases are plan fragments rather than derivational traces. All systems mentioned so far uses first-principles to adapt the plan obtained from replay. But systems also exists using domain-specific plan adaptation rules (e.g., (Napoli & Lieber, 1994)). This is sometimes a misconception in that it is assumed that derivational analogy systems need to be domain-independent.

CHEF, the first CBP system, implemented transformational analogy. In this category is also included DIAL (Leake *et al.*, 1996). DIAL performs plan adaptation in two steps: first, it removes sources of conflicts in the case relative to the current situation, and second it uses transformational operators to transform the case. SPA (Hanks & Weld, 1995) also implements transformational analogy. However, what distinguishes SPA from other transformational systems is that it is domain independent. That is, the transformational operators are independent of any particular domain. SPA's adaptation algorithm is provably correct. In this category we include also the Priar system (Kambhampati, 1994). Priar is a hierarchical case-based planning system that laso implements domain-independent transformation rules. This category is particularly interesting because it is sometimes believed that transformational analogy systems are domain-specific. SPA and Priar show that this is not true.

Derivational analogy is frequently a preferred technique to transformational analogy, because it reduces the search space by pruning fruitless past choices, and because it substitutes revalidation of past rationale for search in the plan adaptation space. The technique is most appropriate when most past plans require much adaptation and when the cost of saving rationale is not too high. However, it presupposes that the derivational traces are provided. In situations where this is not possible, transformational analogy is a better choice since the plans themselves can be used for adaptation.

4. Case Role

Cases perform three potential roles; they can provide domain knowledge, search control knowledge, or both. Continuing with the travel example, the plan depicted in Figure 1 might be the only knowledge that we have about how to travel from Cambridge to San Jose. This is an example where the case provides domain knowledge. In a different situation we may have general knowledge about how to generate plans but the planner may need to search for right ordering between steps. In this situation the particular ordering of steps in the plan represents search control knowledge.

When the cases provide domain knowledge, then typically the adaptation is transformational. In this situation, transformational operators will allow the system to generate new knowledge about the domain by transforming the existing cases. The CHEF system is a typical representative of this category.

When cases provide control knowledge, the systems frequently perform derivational analogy. Planners such as Prodigy/Analogy, Paris, and derSNLP are in the same category in this dimension because they have a complete domain theory that allows them to generate plans from scratch. The role of the cases for these systems is to provide meta-knowledge about how to use the domain theory to solve those plans. Interestingly, although both Priar and SPA perform transformational adaptation, their

cases can be seen as providing control knowledge since both systems assume that a complete domain theory is available, and therefore, could generate solution plans even if no cases are available.

HICAP, like systems in the previous category such as Prodigy/Analogy, uses a first principles planner to extend/adapt the partial solutions obtained from the cases. However in HICAP, cases provide domain knowledge by filling gaps in the incomplete domain theories, available in HICAP. These cases can be seen as instances of an unknown complete domain theory. Under this perspective, cases may indicate alternative paths to a solution using this unknown domain theory. In this sense cases also provide search control knowledge.

In summary, the case role is determined by the scope of the domain theory available. If no domain theory is available, cases necessarily fill the role of providing domain knowledge. On the other hand if a complete domain theory is given, then cases fill the role of providing domain knowledge. Finally, if a domain theory only models the domain partially, cases can fill both roles of filling the gap for the missing knowledge and providing search control knowledge.

5. Case Contents

There are two basic kinds of contents that any case has: indexing and adaptation contents. The adaptation content typically consists of the solution, which for CBP corresponds to a plan. CBP systems such as SPA, Priar, and CHEF perform transformational analogy, and obviously need to store the solution plans. Other CBP systems (e.g., HICAP, Paris) that do not use transformational analogy also include plans. The travel example of Figure 1 illustrates one such a plan that can be stored in the case.

CBP systems that perform derivational adaptation include the derivational trace rather than the plan itself. This includes systems like Prodigy/Analogy, CAPlan/CbC, and derSNLP. Dial's memory search cases also fit into this category because they include the trace of the search process. The derivational trace for a travel plan shown on the right side of Figure 2 is stored as part of the case for derivational analogy systems.

A third category is CBP systems that include adaptation information in their cases. This includes Dial's adaptation cases, which include information for applying transformations. Prodigy/Analogy and CAPlan/CbC are also in this category because derivational traces contain *annotations*, which are used during the memory search process to prune the search space. Not all CBP systems that perform derivational analogy have such annotations (e.g., derSNLP). An example of such annotation is illustrated in Figure 1, where the step *Take shuttle to San Jose* is deleted form the plan. Table 3 summarizes this classification.

These three categories of case adaptation content, namely, the plan, the trace, and adaptation information typically require increasing levels of knowledge engineering effort. Plans have the lowest knowledge engineering effort, as they are presumably readily available if the decision was to use a CBR approach to problem-solving. Derivational traces require an additional knowledge engineering effort to obtain the sequence of decisions that led to a plan. This can be manually done as in the ELM system, where a domain expert enters the traces that can lead to pieces of LISP code. Alternatively, derivational traces can be generated automatically as in Prodigy/Analogy. However, this requires a domain theory and the corresponding automated planner that can reason with this theory. Having this domain theory available can require a significant knowledge engineering effort. Additional adaptation information can be added manually as in the DIAL system, or automatically as in Prodigy/Analogy, but for the latter a complete domain theory is once again required.

5. Case Merging

CBR often assumes a filter mechanism to select a single best case from a retrieval set. Yet for planning, a casebase may not contain any one solution for a novel goal set. Trivially a case may contain a plan to achieve a goal, G1, and a disjoint plan to achieve another goal, G2, such that neither plan is sufficient to cover the new problem of achieving G1 and G2. In such situations, a planner should retrieve both of the old cases and plan adaptation should be able to merge or combine the two solutions. In the travel example, the plan depicted in Figure 1 could be the result of merging two cases: one indicating how to travel from Cambridge to the airport in San Francisco, and the second indicating how to travel form the airport in San Francisco to the hotel.

When the goals that different cases achieve are independent, the manner in which a planner performs the merge matters little. CBP can adapt either plan first and then adapt the second, concatenating the result as the combined solution. A *sequential merge strategy* simply performs the adaptation in the retrieval order of the cases (Veloso, 1997). However step order is often crucial to assure correct plan behavior, especially when goals interact.

Veloso (1997) identified two additional merge strategies. They are the *ordering-based interleaved strategy* and the *choice-and-ordering-based interleave strategy*. Both strategies begin the merge with an arbitrary case in the retrieval set. The former strategy performs plan adaptation by replaying the plan derivation until it encounters a step ordering constraint. It then reasons about the relative ordering of steps in the current plan and in alternative cases. New steps are added and old steps deleted according to any single-case adaptation policy. The latter strategy takes into consideration operator choice as well as step ordering to determine the next partial case to use for adaptation. The Prodigy/Analogy system implements these merge strategies. All other systems choose a single best case from among the candidates returned during case retrieval and so need not perform case merging.

Case merging is most appropriate when planners must solve conjunctive sets of goals and when distinct subplans exist in the plan library that can be combined in multiple ways to solve different goals. Under these conditions few goal conjuncts will be solvable using a single case. Rather than replan large portions from first principles, the retrieval component should retrieve a set of cases that together cover the problem description. Merging these cases will prove more efficient than planning from scratch, if the subplans minimally interact.

6. Representation Formalism

Independent from the actual contents of a case is the choice of the formalism used to represent those cases. For the purposes of this discussion we concentrate on the following first-principles planning techniques, which determine the case representation:

- **Total order planning**. This is one of the earliest forms of planning. At any point in the planning process the system has a partial solution that consists of a totally ordered plan. An important advantage of this form of planning, which has been "re-discovered" recently, is that, by dynamically maintaining the state, better search control rules can be written that increase planning efficiency (Vidal & Regnier, 1999). The main drawback is that they cannot easily generate solution plans that require interleaved sub-plans. Figure 3 illustrates a total order plan.

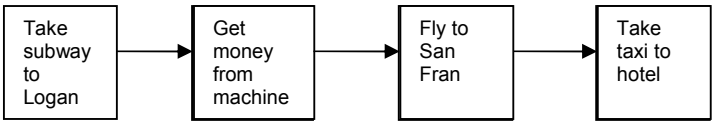

Figure 3 Total order representation of travel plan

- **Partial order planning**. This form of planning resulted in part as a way to handle interleaving between sub-plans. At any point of time a plan is maintained that is partially ordered. Partial order planning does not commit to any orderings unless it is necessary to solve conflicts. Their main disadvantage is that they can be very inefficient. Figure 4 illustrates the travel plan from Figure 1 as a partial order plan assuming only that they money has to be acquired sometime before arriving at the hotel.

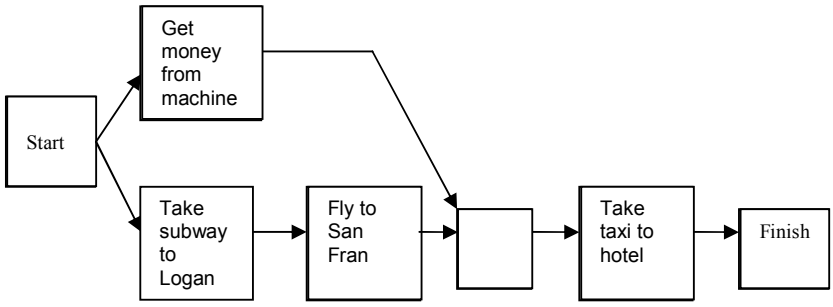

Figure 4 Partial order representation of travel plan

- **Hierarchical/Abstract planning**. Hierarchical planning is based on the idea of refining complex tasks into simpler ones. Abstract planning follows the same idea, except that the language used for expressing tasks at the higher levels of the abstraction may differ from the one used at the lower levels (Bergmann & Wilke, 1995). An important advantage of hierarchical planning is that it has been proven to be strictly more expressive than STRIPS planning, which has been used in both total and partial order planning systems (Erol *et al*., 1999). A disadvantage of hierarchical planning is that it complicates search control.

  Figure 5 shows a hierarchical plan in the travel domain we have been discussing. The top level task is to travel from Cambridge to San Francisco. This task is decomposed intro two subtasks: Take enough money and do the travel. The travel itself is decomposed into three subtasks: travel from home to Logan, Fly from Logan to SFO, the San Francisco airport, and travel from SFO to the hotel. Each of these are also decomposed until primitive operators appear at leaves such as boarding the plane.

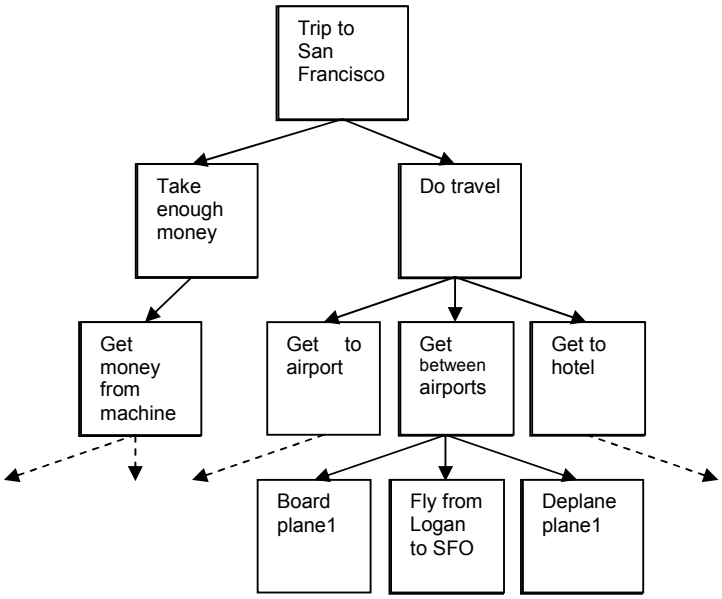

Figure 5 Hierarchical representation of travel plan

- **Planning graphs**. This is one of the newest forms of planning. A so-called planning graph is dynamically constructed that contains alternative levels of atom nodes and action nodes. Each atom node at level *i* represent several possible states at time step *i*. Action nodes at level *i* represent actions transitioning states from level *i* to states at level *i*+1. At each planning cycle a new level is added and a test is performed to determine whether a solution plan is contained in the current planning graph (Blum & Furst, 1997). Planning graphs have been shown to be more efficient than other forms of STRIPS planning but encoding domain-specific control rules into graphs is difficult.

As shown in Table 1, CBP systems have been proposed using each of these planning techniques and even combinations of those techniques such as in Paris and Priar. This table lists a new case-based planner named CPG (Gerevi & Serina, 2000) that combines both transformation and memory search. CPG's transformation process involves simply removing actions that are not applicable in the new situation. The bulk of its adaptation effort is done during memory search, where it identifies inconsistencies and revises the plan.

What kind of planner to used in which situation is a question that remains largely unsolved, and different, even contradicting answers have been given at different times. Total-order planning was the earliest form of automated planning. Then, partial-order planning was proposed and for some time was believed to be the best form of performing automated planning. Then came plan graph-based planning and other so-called neo-classical planners and those are currently believed to better than partial order planners. However, recently there has also been a resurgence of total order planners as they enable the definition of search control heuristics that can result in good performance. Hierarchical planners can encode search control knowledge in their domain theories that can led to good running-time performance. However, obtaining a domain description requires more effort compared to the other planners as these requires not only the collection of operators as the other planers but also a collection of methods indicating how to generate hierarchies.

Table 1. Classification of the representation formalism.

Planner	Total Order	Partial Order	Hierarchical	Plan Graph
Prodigy/Analogy [2]	✓			
CAPlan/CbC		✓		
Paris	✓		✓	
Priar		✓	✓	
CPG				✓

7. Complexity of Plan Adaptation

Nebel & Koehler (1995) proved that plan adaptation can be harder than first-principles planning. This result is frequently stated as evidence that plan adaptation is an inadequate problem solving strategy. However, a closer look at their proof reveals an assumption that does not hold for many CBP systems. It assumes that the new solution is obtained by *minimally* modifying the retrieved plan. The authors refer to this second assumption as a *conservative* approach.

Nebel and Koehler's result on plan adaptation does not hold for CBP systems whose plan adaptation strategy is not guaranteed to be non-conservative. In particular, adaptation by derivational replay, as implemented in Prodigy/Analogy, derSNLP, or CAPlan/CbC, is in the worst case as hard as planning from first principles with their respective first-principles planning systems (Au *et al*. 2002). This follows from the simple observation that, during a case replay step, a CBP system acts as an oracle for the underlying first-principles planning system; among the possible choices that the

[2] Prodigy4.0, the first-principles planning component of Prodigy/Analogy, can generate partial plans as well as totally ordered plans. However the system cannot adapt partially ordered plans.

underlying planning system can take, the oracle (i.e., the current step in the derivational trace) chooses one.

A more interesting question is that of the average-case complexity. We know of no good way to analyze the average-case performance theoretically, because there is no clear definition for the "average case"; it depends largely on the problem being solved and on how the planning algorithm decides what path to search next. For example, even if derivational replay reduces the size of a planning algorithm's search space, it is conceivable that for some planning problems and some planning algorithms, the algorithm might search a larger part of this reduced search space than it would have searched if derivational replay had not been used. A similar point is made for SPA system, which as explained before implements transformational analogy. For such reasons, average-case analyses of planning algorithms are typically done experimentally rather than theoretically. In all of the experimental studies that we know of, derivational replay has run significantly on the average faster than planning from scratch (Veloso, 1994; Ihrig & Kambhampati, 1996, Muñoz-Avila & Weberskirch, 1996). Recently, similar performance gains were reported for an algorithm based on transformational analogy (van der Krogt & de Weerdt, 2005).

8. Future Research Directions

As part of our study we have identified some issues on plan adaptation that have not been addressed sufficiently and that point toward new directions for research in this area. Note that much of the current research related to case adaptation in the CBR community is known by other names or associated with related research topics.

- **Interactivity.** Some initial work has been done to enable user interactions during plan adaptation. In Muñoz-Avila (1998), the user interactions are limited to deleting actions after the plan is generated. In such situations, the underlying CBP procedure generates alternative plans, if possible. In Cox & Veloso (1997), the user can make planning choices in the plan generation process that Prodigy 4.0 normally performs, but not in the derivational analogy process that Prodigy/Analogy performs. Research is needed to address the following outstanding issues: (1) The user can modify the plan *during* plan adaptation, and (2) the user can dynamically modify the facts defining the current situation. Both of these require addressing user interface problems such as how to present intermediate planning states. Cox (2004) reports preliminary progress in these issues.
- **Information gathering.** Planning for information gathering is a frequent research topic, in particular in connection with search over the Web. Current research concentrates primarily on plan generation. Algorithms need to be developed to adapt information gathering plans to solve new search problems.

- **Inference procedures.** Whereas many CBP systems have been built on top of first-principles planners, other inference techniques (e.g., constraint satisfaction or genetic algorithms) have rarely been used. Constraint satisfaction techniques could be useful for dynamic re-planning, in particular for interactive case adaptation, where the effects of the user interactions can be propagated to evaluate how they affect the current plan.
- **Plan recognition.** Adaptation can occur when using past cases to understand an observed plan execution. Plan recognition is the task of finding a match in a case library that corresponds to the current observations and using it to predict the subsequent behavior of the agent. When search fails to find an identical match, a case-based plan recognition system can choose a close one and adapt it to predict the behavior (Cox & Kerkez, 2006). Yet many undecided issues remain with such novel techniques such as how to control for the combinatorics of adapted operator variable bindings.
- **Quality issues**. Planning from scratch under quality constraints is known to be a very difficult problem (Perez & Carbonell, 1994). So far no work has been done that considers plan quality during plan adaptation. In the context of the travel planning example that we have used in this paper, the various planning steps might have different costs. Depending on how the plan is modified, plans with different costs may be generated for the same problem. The problem might become complicated if conflicting quality criterion must be considered. For example, using the plane might be a faster but more expensive than taking the train.

9. Concluding Remarks

In this paper we present an analysis of the state of the art for plan adaptation that transcends the traditional dichotomy between transformational and derivational analogy. In addition to this dimension, we categorize case-based planning systems among five other dimensions: the role of the case, the contents of the case, the use of case merging, the representation formalism, and the computational complexity.

First, we clarify a misconception that sometimes arises, namely, that transformational analogy systems are all domains-specific whereas derivational analogy systems are all domain-independent. We provide examples of domain-specific systems that perform derivational analogy and domain-independent systems that perform transformational analogy.

We also observe that when cases provide domain knowledge, they frequently perform transformational adaptation, and when cases provide control knowledge they frequently perform derivational adaptation. However, we note that SPA performs transformational adaptation even though cases provide control knowledge. We also cite examples of systems in which cases provide domain knowledge but the bulk of the effort is expended during memory search.

We identified three kinds of content for cases used by CBP systems: plans, traces, and adaptation information. Plans are contained in several CBP systems, including those performing transformational adaptation. Traces are contained in CBP systems performing derivational analogy. Some systems also include adaptation information to reduce the search effort.

CBP systems cover a wide spectrum of representation formalisms, namely, totally ordered plans, partially ordered plans, hierarchical and abstract plans, and planning graphs. Furthermore, some systems combine representations such as hierarchical and totally ordered plans.

Finally we examined previous results on the worst-case complexity of plan adaptation and observe that conservative adaptation is a rather restrictive requirement. We point to other results showing that plan adaptation with derivational analogy is not conservative. We also point out that a more interesting analysis would be on average-case complexity. But this has not been done to date and remains an open problem.

References

Au, T.C., Muñoz-Avila, H., & Nau, D.S. (2002) On the Complexity of Plan Adaptation by Derivational Analogy in a Universal Classical Planning Framework. In *Proceedings of the Sixth European Conference on Case-Based Reasoning*. Berlin: Springer.

Bergmann R., & Wilke W. (1995). Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*. 3, 53--118.

Blum, A., & Furst, M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90.

Carbonell, J.G. (1983) Learning by analogy: formulating and generalizing plans from past experience. *Machine Learning: An Artificial Intelligence Approach.* R.S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.). Tioga, Palo Alto, California.

Carbonell, J.G. (1986) Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. *Machine Learning*.

Cox, M. T. (2004). Mixed-initiative case replay. In the *Proceedings of the 17th International FLAIRS Conference* (pp. 166-171), Menlo Park, CA: AAAI Press.

Cox, M., & Kerkez, B. (2006). Case-based plan recognition with novel input. *International Journal of Control and Intelligent Systems*. *34*(2): 96-104.

Cox, M. T., Muñoz-Avila, H., & Bergmann, R. (2006). Case-based planning. *Knowledge Engineering Review. 20*(3): 283-287.

Cox, M. T., & Veloso, M. M. (1997). Supporting combined human and machine planning: An interface for planning by analogical reasoning. In D. B. Leake & E. Plaza (Eds.), *Case-Based Reasoning Research and Development: Second*

International Conference on Case-Based Reasoning (pp. 531-540). Berlin: Springer.

Costas, T, & Kashyan, P. (1993) Case-based reasoning and learning in manufacturing with TOTLEC planner. *IEEE Transactions on Systems, Man, and Cybernetics.* 23(iv) July/August

Cunningham, P., & Slattery, S. (1994) Modeling of Engineering Thermal Problems - An implementation using CBR with Derivational Analogy. *Proceedings of First European Workshop on Case-based Reasoning (EWCBR-93).* Kaiserslautern: Springer.

Erol, K; Hendler, J; & Nau D.S. (1994). HTN Planning: Complexity and Expressivity. In *Proceedings of the National Conference on Artificial Intelligence* (*AAAI-94*). AAAI Press.

Gereveni, A, & Serenia, I. (2000). Fast Plan Adaptation through Planning Graphs: Local and systematic search techniques. *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2000)*. Breckenridge, CO: AAAI Press.

Grech, A., & Main, J. (2004). Case-Base Injection Schemes to Case Adaptation Using Genetic Algorithms In *Advances in Case-Based Reasoning: 7th European Conference, ECCBR 2004*. Berlin: Springer.

Haigh, K. Z., Shewchuk, J. R., & Veloso, M. M. (1997). Exploiting Domain Geometry in Analogical Route Planning, *Journal of Experimental and Theoretical Artificial Intelligence. 9*: 509-541.

Hanks, S. and Weld, D. (1995). A domain-independent algorithm for plan adaptation. *Journal of Artificial Intelligence Research*, 2.

Hammond, K. J. (1989). *Case-based planning: Viewing planning as a memory task*. San Diego, CA: Academic Press.

Hammond, K. (1990). Explaining and repairing plans that fail. *Artificial Intelligence*, *45*: 173-228.

Ihrig, L.H., & Kambhampati, S. (1994). Derivational replay for partial order planning. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (AAAI-94). Seattle, WA: AAAI Press, 1994.

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

Leake, D. B, Kinley, A., & Wilson, D. (1996). Learning to Improve Case Adaptation by Introspective Reasoning and CBR. In *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. Menlo Park, CA: AAAI Press/MIT Press, Menlo Park, CA.

Lopez de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M. L., Cox, M. T., Forbus, K., Keane, M., Aamodt, A., & Watson, I. (2006). Retrieval, reuse and retention in case-based reasoning. *Knowledge Engineering Review, 20*(3), 215-240.

Melis, E. & Whittle, J. (1999). Analogy in inductive theorem proving. *Journal of automated reasoning*. 22: 2.

Muñoz-Avila, H. (1998). *Integrating Twofold Case Retrieval and Complete Decision Replay in CAPlan/CbC*. PhD Thesis. AG Richter. University of Kaiserslautern.

Muñoz-Avila, H., McFarlane, D., Aha, D.W., Ballas, J., Breslow, L.A., & Nau, D.S. (1999). Using guidelines to constrain interactive case-based HTN planning.

Proceedings of the Third International Conference on Case-Based Reasoning (pp. 288-302). Munich: Springer.

Muñoz-Avila, H & Weberskirch. (1996) Planning for manufacturing workpieces by storing, indexing and replaying planning decisions. *Proceedings of the International Conference on AI Planning Systems (AIPS-96)*. Edinburgh: AAAI Press.

Napoli, A., & Lieber, J. (1994). A first study on case-based planning in organic synthesis. *Proceedings of First European Workshop on Case-based Reasoning (EWCBR-93)*. Kaiserslautern: Springer.

Nebel, B. and J. Koehler, (1995) Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis, *Artificial Intelligence* (Special Issue on Planning and Scheduling), 76(1-2): 427-454.

Perez, A., & Carbonell, J. (1994). Control knowledge to improve plan quality. *Proceedings of the International Conference on AI Planning Systems (AIPS-96)*. (AIPS-94). Chicago: AAAI Press.

Salem, A.-B. M., Nagaty, K. A., & El Bagoury, B. (2003). A Hybrid Case-Based Adaptation Model for Thyroid Cancer Diagnosis. In ICEIS 2003, Proceedings of the 5th International Conference on Enterprise Information Systems (pp. 58-65). Angers, France, April 22-26.

Spalazzi, L. (2001). A survey on case-based planning. Artificial Intelligence Review 16: 3–36.

Schmidt, R., Vorobieva, O., & Gierl, L. (2003). Case-based Adaptation Problems in Medicine 2nd German Workshop on Experience Management. April.

van der Krogt, R.P.J. and de Weerdt, M.M.. (2005) Plan Repair as an Extension of Planning. In Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05).

Veloso, M.M. (1994). *Planning and learning by analogical reasoning.* Berlin: Springer.

Veloso, M.M. (1997). Merge strategies for multiple case plan replay. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference* (pp. 413-424). Menlo Park, CA: AAAI Press / The MIT Press.

Veloso, M. M., Mulvehill, A. M., & Cox, M. T. (1997). Rationale-supported mixed-initiative case-based planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference* (pp. 1072-1077). Menlo Park, CA: AAAI Press / The MIT Press.

Vidal, V. & Regnier, P. (1999). Total order planning is more efficient than we thought. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-1999)*. Orlando, FL: AAAI Press.

Weber, G. & Schult T. J. (1998). CBR for Tutoring and Help Systems. In: *Case-Based Reasoning Technology: From Foundations to Applications*. Springer.