LEARNING TO IMPROVE CASE ADAPTATION

Andrew Kinley
Computer Science Department
Lindley Hall 215
Indiana University
Bloomington, IN 47405

Submitted to the faculty of the Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Department of Computer Science

December 2001

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Doctoral Committee

David B. Leake
(Principal Advisor)

Michael Gasser

Edward Robertson

August 15, 2001

Andrew Kinley
Computer Science Department
Lindley Hall 215
Indiana University
Bloomington, IN 47405

Acknowledgements

Had it not been for the presence of countless individuals along the way, the long trip to the state of Indiana taken many years ago would have ended not long after I arrived. With the completion of this dissertation, it is these individuals who deserve a share of the credit for without them this work would never have begun.

From the beginning, the members of the Artificial Intelligence lab provided moments of inspirations and perhaps more importantly they provided moments of levity. These people included at various times David Wilson, Ryan Scherle, Raja Sooriamurthi, Susan Fox, Doug Eck, Cathy Rogers, Keiichi Tajima, and Doug Blank and several others along the way. To David Wilson I owe additional thanks for being the research partner we should all be lucky enough to find.

At Rose-Hulman where I have taught these past few years, several individuals have provided unwavering support for my actions and proven to be the best of colleagues. Mike Wollowski and Frank Young in particular deserve many thanks for their efforts on my behalf.

The friends I have made along the way made the long road exciting and fun and found numerous distractions to keep me from working until the wee hours of the night. Thanks to Juan Marchini, Kyle Wagner, David Kotchen, and Joe Lane for listening to my ideas and pretending to agree.

There would be no dissertation today if not for my advisor David Leake. David was tireless in his position as advisor. His countless critiques of my ideas and writing always made me push for something closer to perfection. Yet David also knew when to push and when to relax and wait for me to study a problem in greater depth. Thank you David.

To my family, I thank them for their patience, their encouragement and their love. I would not have made it here if it were not for the guidance and motivation of my parents. They allowed me to make my goals and were there to pick up the pieces if I didn't make it. I thank my wife Melissa for reading drafts, asking tough questions but

most of all for being there each day. My son Alexander provided the necessary smiles that made a day of difficulty vanish and reminded me of more important things.

This work was supported in part by the National Science Foundation under Grant No. IRI-9409348.

Abstract

Case-based reasoning (CBR) solves new problems by retrieving records of similar past problem solving episodes and adapting the prior solutions to fit the current situation. While the retrieval phase of CBR has been explored with success by past models, developing effective algorithms for automated adaptation remains an open problem. The central hypothesis of this research is that effective case adaptation knowledge can be learned and reapplied automatically by applying CBR to the adaptation process. In this model, adaptation knowledge is learned by storing the results of successful adaptations to be reused in solving similar future problems. If there is no relevant adaptation knowledge to reuse, general rule-based methods of adaptation are used to build the adaptation case base. These methods model the search for needed information as a planful process whose strategies can be captured and reused by case-based reasoning when similar situations are encountered in the future. In addition, as adaptation knowledge is acquired, methods for evaluating the similarity of past cases are refined to reflect the adaptability of a prior case to the new situation. This model is implemented in the DIAL system, a case-based planner in the domain of disaster response planning. DIAL analyzes new disaster situations and proposes response plans to address the problems arising from the disaster. In this research, two criteria are used to evaluate adaptation learning in the DIAL system: the efficiency of the solution process and the usefulness of its results. The efficiency of the solution process is examined through statistical evaluation of empirical results. Usefulness is defined as the system's ability to generate acceptable solutions. Through analysis of the results, the utility of this approach is measured and the contribution of the model is judged.

Contents

List of Figures

List of Tables

1

Introduction

This research introduces a new approach to automated case adaptation in a case-based planner. In this approach, case adaptation knowledge is learned by a system that stores derivations of solved case adaptation episodes. This knowledge is then reused to solve future similar problems.

A typical task for artificial intelligence systems is generating a sequence of steps to solve a specified problem. For example, building an efficient plan for transporting packages to and from specified points under specified constraints might be one such problem (Veloso, 1991). One approach to problems like these that has met with some success is *case-based reasoning*. Case-based reasoning relies on prior experience to guide the solution generation process. In fact, under ideal conditions, relevant prior experiences can be reused directly to solve new problems. However, in many situations past experiences only provide ballpark solutions and require additional modifications before being effectively reused. This plan modification process, known as *case adaptation*, has challenged researchers and a general purpose approach has not been demonstrated. This research introduces one method of automatically adapting problematic plans. Our system uses an internal case-based reasoning process to acquire *case adaptation knowledge* in an attempt to learn "how to adapt" by *derivational analogy* within a case-based planner.

Case-based reasoning is commonly used to store and generate plans for problems in a specified domain. What has not been done is to use a separate case-based reasoning process to address the case adaptation problem.

An internal case-based reasoning process to support case adaptation (Sycara, 1988) in a case-based planning system has been built as part of the **DIAL** system. Empirical evidence will show that this method is capable of speeding up the solution generation process and generating better final solutions than when not using this method.

This case adaptation learning process introduces several challenging new problems that will be addressed, including: how an internal CBR process can be integrated within a larger planning process, how different cases can be combined, how to select appropriate cases with changing system knowledge, and finally, how rules and cases can be combined to learn case adaptation.

This introductory chapter is divided into five sections. The first section begins by stating the primary thesis of this research. Section 2 introduces the key concepts from case-based reasoning. Section 3 supports our approach by outlining its advantages and examines related work. Finally, the last two sections expand on our thesis statement and provide questions to guide the dissertation.

An automated case adaptation process

This work addresses how knowledge can be acquired to support and guide the case adaptation process.

Case adaptation knowledge, the knowledge needed to identify and solve case adaptation problems, can be stored in cases. A *case* is a record of a prior problem solving episode. These cases can encapsulate the reasoning process used when solving previous case adaptation problems. Thus a system that learns case adaptation knowledge can begin with little or no knowledge about the requisite problem solving process. The reapplication of these reasoning traces when solving new *similar* problems should enhance a planning system's ability to solve future similar problems.

> **Thesis claim:** Effective case adaptation knowledge can be learned and reapplied automatically by applying CBR to the case adaptation process.

In this research, case adaptation knowledge is acquired by storing problem solving reasoning traces. This augments the system's knowledge and provides an implicit understanding of "how to adapt." In our system, this knowledge is stored as *adaptation cases*. The development and use of this knowledge has been guided by two principles:

1. System knowledge coded by hand is insufficient to handle the range of problems a system may encounter. Close attention must be paid to the methods of *knowledge acquisition* that support the reasoning processes.

2. Systems facing real-world problems must deal with unpredictable data. Over time, solutions that were once reasonable may no longer be applicable. The system must change to reflect the current state of its world.

Several criteria guide our evaluation of the effectiveness of the internal case-based approach to case adaptation. Our primary evaluation criteria focus on the overall improvement of the system when case adaptation is required. This improvement can be measured by asking two questions.

1. Is the problem solving competency of the case based planning system is extended? Are there problems that are solvable that would have unsolvable prior to the addition of case adaptation learning?

2. Does the system solve problems faster with the addition of case adaptation learning? With *speedup learning* the amount of time the system spends computing solutions is decreased.

A secondary set of evaluation criteria examine the affect of case adaptation learning on other aspects of the problem solving process. This acquired pool of knowledge can be used by other components to improve system performance on tasks not directly related to case adaptation. These components include *similarity assessment*, a rule base of strategies for searching the system's memory, and the organization of the system's case base.

We claim that the overall case-based reasoning process is improved by incorporating this internal learning process that happens to also apply the same CBR method.

1.1 Case-based reasoning

Case-based reasoning is the primary process driving this research. It acts as the central planning process and provides the mechanism for the automated case adaptation learner. This section identifies the key concepts and illustrates the basic operation of the case-based reasoning process. It begins by presenting several real world examples of case-based reasoning.

Consider each of the following scenarios taken from the author's experiences.

- As a student lacking in culinary skills, I would examine the contents of my refrigerator and then leaf through a cookbook searching for a recipe that closely matched the leftovers.

- A customer enters a video store searching for a movie. The clerk examines the customer's recent rental history and suggests a movie containing some of the same actors that were in the prior films the customer enjoyed.

- Searching for a bookstore in an unfamiliar city, I noticed a popular restaurant. I remembered how in my home town this same restaurant was in close proximity to a large shopping area. Using this knowledge, I traversed the immediate vicinity and found a mall containing a bookstore.

As the above examples indicate, reasoning by re-using past cases is a powerful and frequently applied way to solve problems for humans. This is supported with results from psychological research. Rose (1989) presented evidence for the existence of a case based process in human problem solving. Other researchers such as (Anderson, 1983) showed that humans use past cases as models when learning to solve problems. Experts also seem to have a preference for case reuse when solving new problems (Rouse & Hunt, 1984). It was therefore appropriate for researchers to explore a similar reasoning approach for computer problem solvers. This led to the development of case-based reasoning.

Case-based reasoning is solving new problems by reusing the solutions of past problem solving episodes.

The CBR process, sometimes described as "remember and adapt" or "remember and compare" (Kolodner & Leake, 1996), offers a straightforward algorithm to improve the overall problem solving process of a system. The basic solution generation process employed by case-based reasoning case be summarized as follows:

1. Create a new problem description

2. Search for a *similar* stored problem description.

3. Use the prior problem description and stored solution to generate a new solution for the presented problem.

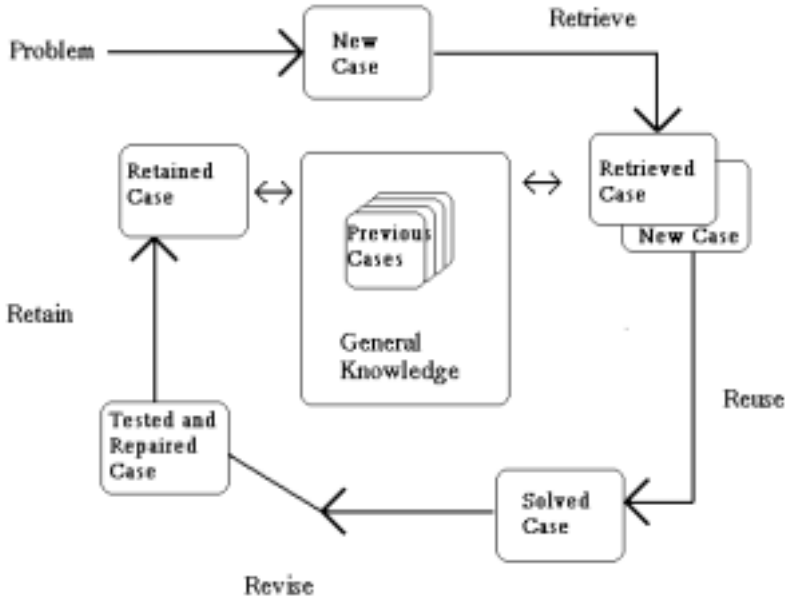

Figure 1.1: Traditional Model of CBR

Case base reasoning works succeeds because similar problems tend to reoccur, and the stored cases can easily provide solution to these problems.

Researchers have presented case-based reasoning as a combination of 4 separate and primary components (Aamodt & Plaza, 1994) as illustrated in figure 1.1. To best describe the function of and motivation for each component, they will be introduced in the context of a traditional CBR example. Following the example, each component will be examined and its contribution to the process stated.

CBR example

Perhaps the most familiar example of a case-based planner is Kris Hammond's CHEF system (Hammond, 1989). CHEF is a case-based planner that generates new cooking recipes. The system is given a set of constraints defining the problem to be solved and then uses case-based reasoning to determine a satisfactory solution.

Problem Description: A recipe is desired that satisfies the following constraints:

- Include beef in the dish

- Include broccoli in the dish
- Make a stir-fry dish

The first step is to **retrieve** the recipe. Using the constraints of the dish as features, the CHEF planner finds a recipe BEEF-WITH-GREEN-BEANS in its library of recipes. This recipe is an exact match of two constraints, in that it is a stir-fry dish and includes beef, and a partial match of a third constraint by way of generalization (both green beans and broccoli are vegetables).

The next step is to **reuse** the prior case in the new context. A new recipe is created using the beef and green beans recipe with the substitution of broccoli for the green beans. Next, CHEF tests each of the proposed recipes with a simulator program designed to provide useful feedback. Before the simulation, a set of expectations for the recipe is generated. Included in Hammond's example are the following expectations:

- The beef in now tender.
- The disk now tastes savory.
- The broccoli is now crisp.

When the simulator completes, one of these expectations is not satisfied. "The broccoli is now crisp" is not satisfied. Instead it is reported that in the proposed recipe "the broccoli is now soggy."

Following that the new plan must be **revised**. Assuming that the recipe is "in the ballpark" of a solution, CHEF attempts to determine the cause of the problem and attempts to repair it. Domain rules assist in determining that the problem was caused by the broccoli absorbing the liquid of the meat as they cooked.

One strategy, among many, used to repair problems is to split apart steps of the recipe. Splitting the steps in the current recipe creates a recipe with broccoli cooked separately from beef. Then a step is added to recombine the recipes after this subprocess is completed. The desired result is achieved with the proposed case adaptation. A second pass of the simulator verifies that this recipe has satisfied all of the expectations.

Finally, the successful plan is **retained**. The new recipe is stored in the system's case base for future reuse. Additional knowledge is recorded describing the new rule on how to solve this type of interaction problem. Both can be used to solve similar problems in the future.

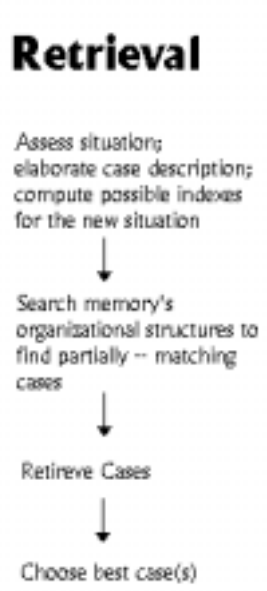

Figure 1.2: Model of Retrieval

CBR components

The CHEF example introduced the primary organization of the CBR process. The four components, retrieve, reuse, revise and retain are now described in greater detail. Each component is listed along side the key issues involved in its implementation as well as the unique knowledge necessary for that component's successful use.

1. **Retrieval:**

   The retrieval process, presented in figure 1.2 and adapted from Leake (Kolodner & Leake, 1996), receives as input a description of the current problem. When first encountering a new problem, a ballpark solution needs to be located in the memory of past cases. This process assumes two factors: first that a usable case exists somewhere in memory, and second, that cases, once located, can be evaluated on the basis of their similarity to the current problem.

   Once a case has been successfully retrieved, the next step is to apply the matching case to the new situation. This is the task of the reuse phase of case-based reasoning.

2. **Reuse:**

With the retrieved case, the system attempts to reuse the case in the current context. This requires it to identify any potential problems with the reuse of the plan. An evaluation subcomponent attempts to identify these potential problems so that they may later be repaired. Several different approaches to evaluation have been used by CBR systems. These methods include testing the proposed solution in the real world or a simulated world (Hammond, 1989), asking a user for feedback (Oehlmann, Sleeman, & Edwards, 1993), examining the outcomes of previous cases (Kolodner, 1988a) or accessing a rule base that provides information about well known problems. If the evaluation identifies no relevant problems, then the solution can be used and the new case will be retained. Otherwise, the solution is sent with a list of identified problems for revision. In practice, the boundaries between the reuse and revise phases are blurred until an acceptable solution is built.

3. **Revise:**

   The revision component performs case adaptations as necessary to repair the problems identified during evaluation. However, case adaptation can be difficult. To repair a problem, the problem must first be understood. For example, a student flunking an exam could have many potential causes: the student may not have studied, the exam was surprisingly difficult, or the teacher graded the student unfairly. In order to prevent the student from flunking in the future, it is necessary to *assess blame* to a cause so an appropriate solution can be generated.

   However, it is non-trivial to assign blame for the problems identified by the evaluator. One solution could cause a host of other problems, some far more difficult to solve than the initial problem. In fact, some problems may require knowledge that is outside the scope of the original problem domain. These types of problems often require that a separate knowledge base be created exclusively for the use of the case adaptation module.

   Due to these difficulties, case adaptation is often performed using static adaptation rules or by user intervention. Automated case adaptation has been considered by many to be an intractable problem. The problem is so acute that experts in both CBR research (e.g., Kolodner, 1991) and applications (e.g., Barletta, 1994; Mark et al., 1996) agree that it is not currently practical to deploy CBR applications with automatic case adaptation.

   However, research has begun to examine this problem in greater depth as methods are proposed to better understand the case adaptation process and the types of knowledge required to perform them (Hanney, 1997). In fact, one central idea of this thesis describes a mechanism for learning to find good case adaptations

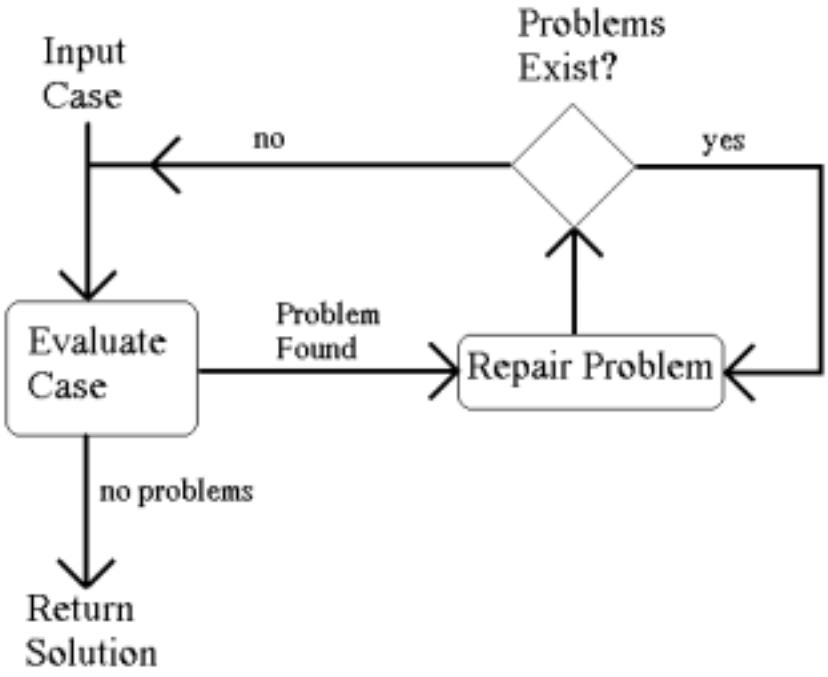

Figure 1.3: Evaluation/Adaptation Subprocess

for certain classes of problems.

> Case adaptation knowledge can be learned for certain classes of problems.

As repairs are made, the case adaptation component passes the repaired solution back to the reuse phase to ensure that other negative side-effects do not occur. If other problems are found, control is returned for additional revisions. This subprocess, shown in figure 1.3, continues until no additional problems are found. The final solution now can be used and is passed to the retain phase.

4. **Retain:**

   When the final solution is sent to be retained, it is ready to be placed in the case base. A key issue for retention is where to store the final solution in the case

base. A well organized case base can reduce retrieval time and thus decrease the total overhead of the CBR process. The retention process provides the mechanism for the system to learn from the problems that have been solved from scratch and gives the opportunity to reuse the solutions in future similar problem episodes.

1.2 Motivations for addressing the case adaptation problem

Case-based reasoning has proven effective in a wide variety of domains. For a survey of several such systems see Kolodner (1993) and Leake (1996a). However, case-based reasoning without case adaptation limits the types of problems that a system can solve, and case-based system developers have encountered difficulties implementing automated case adaptation. and some systems have chosen to avoid the issue entirely. As one CBR text observes (Kolodner, 1993), considerable domain knowledge may be needed to guide the case adaptation process. Further, the knowledge must be organized in a way to facilitate the case adaptation process. While this research does not dispute the necessity of such knowledge, it does address new ways of acquiring and reusing case adaptation knowledge.

Some other approaches to addressing the case adaptation problem have met with some success. Some previous case adaptation systems have relied on rule sets to support its process (Koton, 1988). This approach has limitations on how developers create the rules and subsequently determine the relevancy of the rules. Regardless of the number or type of rules created, gaps in the knowledge are likely to exist. In fact, even if rules alone were sufficient to solve the case adaptation problem, the creation of these rules would remain a difficult problem. New approaches that avoid these problems are needed.

Case-based reasoning provides a method for filling the gaps in knowledge left by rule based methods and their equivalent. By building case adaptation knowledge from past case adaptation episodes, a system can automatically acquire the knowledge that otherwise would be unavailable. CBR is a logical choice to use for this task as it is often applied to domains that are poorly understood or difficult to codify. Thus the CBR case adaptation process can succeed in developing the correct knowledge where other methods might have greater difficult.

This research has taken the view that much of the knowledge needed to perform case adaptation can be gathered through a meta-reasoning process. Case-based reasoning is perfectly suited to the task of acquiring this case adaptation knowledge and

thus helping to alleviate many of the knowledge acquisition issues. However, several other motivations exist for applying a case-based approach to case adaptation. The next subsections identify advantages of this approach and mentions other ways the problem has been studied.

Advantages of a CBR approach to case adaptation

A case-based reasoning approach to case adaptation provides several advantages for an automated solver. With case-based reasoning, a set of problem-solving episodes are stored and reapplied when similar problems are encountered. Each added case extends the problem solver's coverage of the solution space. However, despite this coverage, the problem solver may encounter certain types of problems for which its current knowledge is insufficient to find solutions. For these situations, additional knowledge is required to solve these problems. Case adaptation knowledge is exactly the type of knowledge needed.

If case adaptation knowledge is to be acquired then an internal learning mechanism needs to exist. The use of case-based reasoning for this internal process has many advantages. Many other types of learning require significant amounts of *a priori* knowledge about the organization of memory and thus would have difficulty in identifying any regularity that might exist. Case-based reasoning, however, has no such constraints on its knowledge sources. Instead it provides a useful paradigm to learn case adaptation knowledge since CBR has been shown to be successful in domains that are not fully understood and for which no complete theory on the organization of memory exists. The internal CBR process used to support case adaptation learning can begin to acquire useful knowledge that can be reapplied to future case adaptations from the first instances of learning.

Some researchers have proposed different internal learning processes such as an inductive approach (Hanney, 1997), case based estimate refinement (McSherry, 1998), genetic algorithms (Purvis & Athalye, 1997; de Silva Garza & Maher, 1999) or even decision tree (Shiu, Sun, Wang, & Yeung, 2000) approaches to case adaptation. These methods have had some success but do not result in a robust and dynamic mechanism for handling case adaptation in a changing environment or adequately address the knowledge acquisition issue. Others, such as Sycara (1988), have successfully used case-based reasoning to support a case-adaptation process. However, this method was limited in the ways it could reapply case adaptation knowledge. Specifically, it was given well-defined heuristics and rules on how to apply the stored case knowledge.

One possible limitation of using case-based reasoning is that given a retrieved case,

it is not always clear how that case should be appropriately reapplied. This is especially pertinent to our case-adaptation module which relies on accurately reapplying case adaptation knowledge to solve difficult problems. This issue must be addressed in any evaluation of our method.

Motivations from psychology

While there is sufficient evidence from case-based reasoning literature to suggest the feasibility of our approach, a similar process exists within humans and has been studied by psychologists. Humans tend to be analogical reasoners in certain problem solving situations. Several researchers have described experiments where humans change their approaches to problem solving when additional knowledge can be applied analogically (Chi, Feltovich, & Glaser, 1981; Suzuki, Ohnishi, & Shigermasu, 1992). The additional knowledge, the analogue to our case adaptation knowledge, aids humans in finding quicker and more accurate solutions than when compared to subjects without the additional knowledge. The knowledge used in these studies was most often experiential in nature and thus represents precisely the knowledge we suggest using to solve new case adaptation problems. While human methods may not always carry over to computer methods, this is reasonable evidence that such an approach is feasible and has been beneficial in at least one intelligent system.

1.3 Case based case adaptation

This section outlines the basic reasoning process required for case-adaptation. It begins by describing several of the knowledge acquisition issues and how they pertain both to case based reasoning and case adaptation. Next, our approach to automated case adaptation is presented in terms of case based reasoning. Finally, we highlight several advantages of acquiring case adaptation knowledge by suggesting areas beyond case adaptation where this knowledge can be used.

Knowledge supporting the CBR process

Learning by acquiring new cases is an integral part of the CBR process: each problem-solving episode itself provides a new case to save for future reuse. However, these cases form only one pool of knowledge that support learning in this framework. CBR systems rely on at least four types of knowledge:

- **The case base:** The stored solutions known to the system
- **The indexing scheme:** The features used to retrieve cases from the case base. These features are often predefined and fixed within the system.
- **Similarity criteria:** The method used for selection of candidates retrieved during the retrieval process. Like indexes, the criteria used have traditionally been predefined by the user.
- **Case adaptation knowledge:** The stored problem solving episodes, as described above, provide a rich source of knowledge that enables many types of learning.

These four types of knowledge exist in support of various aspects of the CBR process. For example, the indexing scheme supports the retrieval of relevant cases from the case base. While the case base changes as new solutions are accepted by the system, the other three knowledge sources are generally unchanging and defined from the outset of processing in the system. However, this dissertation proposes a new method for acquiring case adaptation knowledge. With the acquired knowledge, there is a potential to affect the other knowledge sources already in the system. One of our claims is that the stored plan cases can be reapplied more effectively if better case adaptation knowledge is available.

Case based reasoning can be further improved by augmenting the retrieval similarity criteria with the stored case adaptation knowledge. Stored cases can be selected for reuse based on how easily the system can subsequently adapt them. The interaction of different knowledge sources to augment and overcome the deficiencies of one another has previously been established and documented (Richter, 1995). However, the types of knowledge interactions available with the addition of learned case adaptation knowledge have not been examined. This dissertation describes some of these interactions and illustrates their affect on the overall CBR process.

Acquiring appropriate knowledge

The use of case adaptation knowledge by our automated case adaptation method involves two phases: the acquisition of appropriate knowledge, and the successful reapplication of that knowledge. This subsection examines the acquisition issue, while subsequent subsections address the reapplication issue. The knowledge acquisition process itself is affected by several decisions which guide our description: the learning methods used, how knowledge from different sources is integrated, and the effort expended during the acquisition phase.

This approach focuses on developing individual learning strategies for each type of knowledge required. When the specific knowledge is acquired it can then be integrated into the whole CBR system. This is not a unique approach. Learning methods already exist for refining indexing criteria (see (Kolodner, 1993)); learning methods have also been applied to case adaptation knowledge (Hanney, 1997; Sycara, 1988); and some CBR systems already combine multiple forms of learning (Hammond, 1989).

We introduce three questions in order to provide guidance in this document for addressing knowledge acquisition issue.

1. When building internal learning processes for knowledge acquisition, what determines an effective learning method?
2. How can these processes be used in order to maximize the gain of useful knowledge? Can they noticeably improve overall system performance?
3. What is the cost incurred by the knowledge acquisition phase? Are subsequent gains in speed or accuracy lost when the acquisition process is accounted for?

When knowledge is acquired in a CBR system relevant to a specific subcomponent, such as case adaptation, can this knowledge be integrated into the knowledge of other system components One of our claims is that knowledge acquired in one subcomponent may be reused in support of other subcomponents. When possible, this enhances the benefits of subcomponent knowledge acquisition. We will show that case adaptation knowledge can be used beyond solving case adaptation problems and will improve the operation of other aspects of the CBR process. In the following subsections, we examine how knowledge is acquired and reapplied for case adaptation using an internal case-based reasoning system.

Learning how to adapt

We have described several potential advantages of learning case adaptation knowledge. However, the internal learning process comes at the price of added system overhead as well as some form of training period before becoming effective. An obvious alternative would be to hand code the case adaptation knowledge in the form of rules that the system could apply when problems are encountered. This would be a preferred method if rules did not suffer from begin static and brittle:

- Static: Rules are fixed once created. Most rule sets are unable to adapt to changes in system knowledge. If system knowledge is changed, all of the rules may need to be recoded by hand. This can be both a tedious and error-prone process.

- Brittle: Even the best though out rules are not likely to provide complete coverage in anything but the simplest domains. As such, situations inevitable arise for which the rule set does not provide solutions.

In the place of rules, our approach acquires the needed knowledge with case-based reasoning. The case-based framework can circumvent many of the traditional problems of rule-based systems, and its learning can be applied immediately. Our approach works by beginning with a small set of domain independent rules that can be applied to the adaptation process. These rules are developed from the types of operations that can be performed directly on the system's memory, providing a comparatively unguided search of system memory. If a solution is found using this rule-based approach, then it can be stored as an adaptation case. The adaptation case can be reapplied using *derivational analogy* on future similar case adaptation problems. As this process continues, the case adaptation knowledge of the system will change from a primarily rule-based system to a primarily case-based system.

One obvious concern is whether the problems that plague rule sets affect our approach as our method begins with a foundation of rule-based knowledge. The rules used in this approach are not selected arbitrarily but rather are an exhaustive list of allowed search operations. These operations provide the foundation for all possible reasoning in the system. Our approach learns new ways to combine multiple rules to create task and domain specific patterns of higher reasoning. Further, as domain specific rules are augmented by new learning over time, they do not suffer from the problems of static pre-defined rules.

> This dissertation describes an automated case adaptation component that acquires knowledge through an internal case-based process.

Reapplying case adaptation knowledge

Once adaptation cases are built and stored, in order for the system to be effective, these cases must be reapplied on future problems. The reapplication of these cases to new problems presents some difficulties. The reuse of adaptation cases can occur in two different ways. These two methods provide a bridge between the adaptation problem and the desired solution. The first method, *transformational analogy*, directly applies the solution from a similar prior problem to the new situation. With this method the same overall solution structure is maintained and the relationships in the solution are kept. A second method of reapplication is *derivational analogy*(Carbonell,

1986; Veloso, 1994). Using this approach it is not the previous solution that provides the guidance for the new situation but instead it is the reasoning steps that were taken to achieve the past solution that are reapplied in the new context. Transformational analogy is akin to taking a recipe and substituting new but similar ingredients to form a new recipe. A derivational approach to cooking would examine how the original recipe was created and try to apply the same recipe creation strategies to solving the new cooking problem.

Consider for example how a student might approach studying for exams. On a mathematics exam a student might practice problems repeatedly until confidence is achieved. In other math related disciplines (such as a physics course), the same approach can be applied directly as a form of transformational analogy and the student can expect success. However, if the same student decides to study for a literature exam, that student should reevaluate this method. One approach would be to consider what topics were stressed in the class and focus on mastering those concepts. Another approach would be to ask other students who had previously taken the course. So rather than directly reapplying the approach used in the mathematics course, the student determines the overall strategy used to pass the exam. This reasoning process allows the student to prepare better for the literature exam than if literature word problems are sought after. When performing case adaptations on a plan, a choice must be made between a transformational or derivational approach to reapply.

Case adaptation and similarity

Another challenge for adaptation case reapplication is how to coordinate multiple learning processes. If, for example, the top level CBR process produces a set of plans that cannot be adapted using the stored case adaptation knowledge then there is no improvement in the system. To remedy this problem, one approach is to retrieve plan cases from the case base that will require the least amount of case adaptation.

Similarity and retrieval. The standard retrieval process for plan cases is driven by a similarity assessment process. This process uses surface features from the problem description to identify appropriate stored plan cases to reuse. Several candidate cases can be selected quickly using this approach, but none of these cases has any guarantee that it will be easily adaptable. When no case adaptation knowledge exists, this approach is justified. However, if case adaptation knowledge is stored in the system, similarity assessment can be refined.

Case adaptation knowledge and similarity. Case adaptation knowledge stored in adaptation cases can be exploited to evaluate each candidate plan case on the

basis of its expected repair cost. With a repository of adaptation cases containing the necessary statistical information, it is straightforward to score the adaptability of candidate cases. Thus as more case adaptation knowledge is acquired the more refined the similarity criteria can become when these two knowledge sources are coupled.

> The learning of case adaptation knowledge provides opportunities for learning in other areas of the system.

1.4 Thesis statement

This dissertation proposes the use of an internal CBR process to support case adaptation within the framework of a case-based planning system. The creation of this internal CBR process enables the system to acquire new and useful case adaptation knowledge which can be reapplied throughout the planning process.

> Learned case adaptation knowledge in the form of adaptation cases can support and contribute to a more robust and efficient case-based reasoning process.

Potential problems with this approach

The use of an internal case-based reasoning process to support automated case adaptation is a promising approach to this difficult problem. However, several different factors could limit the effectiveness of this approach or cause it to fail. Thus establishing our thesis requires demonstrating that the following potential pitfalls do not apply, or can be overcome.

- **The overhead of the internal CBR process could be larger than the time savings.** One of the primary goals of case adaptation learning is to *speedup* the planning process. However, if the cost of retrieving adaptation cases and managing the adaptation case base is greater than any savings, then this approach is not worthwhile.
- **Past adaptation solutions may not be easily reapplied to new problems successfully.** An assumption is made in this research that stored adaptation cases can be reapplied to future similar case adaptation problems. It is not clear that the prior solution derivations will generalize to new problems that are encountered.

- **Adaptation cases may not improve on the knowledge already stored in response plan cases.** Plan cases that are stored may implicitly cover the same knowledge that the adaptation cases attempt to store. A good retrieval component might select plan cases that already contain the appropriate solutions to the new problems assuming that the case base has sufficient coverage of the solution space. If the results from retrieval only systems are as good as the results when case adaptation learning is added, then this would eliminate the need for case adaptation knowledge.

- **Evaluation of problems in candidate plans may be imperfect and prevent quality case adaptations from being performed.** While not the primary focus of this research, the evaluation of candidate plans is central to the creation of acceptable new plans. Evaluation is a key component of the case adaptation process both in the identification of problems and the assessment of the proposed solutions. If the evaluation component fails in its task, the remainder of the system's performance is suspect.

- **The credit assignment problem could make the reapplication of adaptation cases difficult.** When reapplying a reasoning trace from past case adaptation episode, it can be difficult to determine why a given reasoning step was performed. When using this trace in a different context, the system makes determinations as to the best way to reuse the trace. If the system's assessment about the reasoning is flawed then it is difficult to accurately judge the affect of adaptation cases on the planning process.

- **Traces of prior problem solving episodes may be too specific to be applicable to new situations.** One premise of this research is that problems of a given type can be solved using the same reasoning steps independent of the context of the problem. While this may not hold true for every example, the reapplication of adaptation cases will be ineffective if it is not possible some of the time.

1.5 Central research questions

This dissertation will try to answer several central questions about the addition of the internal case-based adaptation process to case-based reasoning. Several of the general questions are asked here.

- Does the automated case adaptation component improve the CBR process in terms of efficiency and solution quality?

- What overhead is created by this case adaptation process?
- What is the relative contribution of each component in the CBR process and how does case adaptation interact with the other components
- Does the integration of several learning models and knowledge sources increase the overall efficiency of the system without incurring further overhead?
- What is the effect of an increased number of cases requiring processing and management during each problem solving episode and what are ways to manage these cases?
- How will this model scale up to a larger system and what are specific ways for which the utility problem can be addressed?

More specific questions that we will address include:

- Is the addition of a case adaptation learning component prohibitively expensive relative to any improvement to the overall CBR process?
- Does case adaptation learning provide an equivalent or synergistic mode of learning when compared with traditional case learning?
- Does the acquisition of case adaptation knowledge provide other opportunities for learning to occur in the CBR process?
- What role is played by the selection of the knowledge base to the success or failure of learning?
- How does this model of reasoning compare to other established models of case adaptation in terms of both efficiency of solution and coverage of the solution space?
- What is the benefit of including separate reasoning subprocesses such as case adaptation learning and similarity learning and how does the interaction of these subprocesses affect the overall reasoning process?

1.6 Thesis overview

This dissertation is organized into seven chapters. Chapter two illustrates the complete processing of the system through the use of an example. Chapter three

presents a detailed view of the case adaptation learning component. Chapter four describes adaptive similarity techniques that can exploit case adaptation knowledge. Chapter five lends empirical support to the validity of these approaches and offers some perspective to the entire process. The final chapter concludes this dissertation and places this research in the context of the current state of the art.

2

Overview of the DIAL System

This chapter introduces the DIAL system, the testbed system for the case adaptation learning process. The chapter will describe and illustrate the processing of DIAL through the use of a single extended example.

This chapter describes the overall processing of the testbed system DIAL that was used to examine the ideas presented by this research. A example from the system is traced to show how different learning methods combine to support the overall planning process. The chapter begins with a broad outline of the DIAL system, highlighting the primary components and identifying central areas of interest to this research. Following this, a second example provides a step by step illustration of the learning processes as they occur in the system. Each of the learning strategies used by DIAL is illustrated in the context of the system.

2.1 DIAL

DIAL, for **D**isaster Response Planning with **I**ntrospective **A**daptation **L**earning, operates in the domain of *disaster response planning* for natural and man-made disasters. Examples of such disasters include earthquakes, chemical spills, floods, forest fires, and "sick building syndrome," in which occupants of a building fall victim to problems caused by low air quality inside a building. Studies of human disaster response planning show that case-based reasoning plays an important role in response planning by human disaster planners (Rosenthal, Charles, & Hart, 1989). In fact, human disaster planners often are trained using a case-based process. They learn

the possible successes and failures of response planning by examining real world prior episodes.

DIAL starts with a library of domain cases—disaster response plans from previous disasters—and general (domain independent) rules about case adaptation and memory search. Like other case-based planners, it learns new plans by storing the results of its planning process. Unlike the architectures of some traditional planning systems, the DIAL architecture stresses the components that implement automated case adaptation and improve the overall case adaptation process.

When DIAL successfully adapts a response plan to a new situation, it stores not only the problem solving episode in the form of a *disaster response plan*, but also two types of case adaptation knowledge. *Memory search cases*, encapsulating information about the steps in the memory search process, and *adaptation cases*, encapsulating information about the case adaptation problem as a whole, are stored to be reused when similar case adaptation problems are encountered.

The entire DIAL system includes a schema-based story understander (that receives its input in a conceptual representation), a response plan retriever and instantiator, a simple evaluator for candidate response plans, and a case adaptation component to adapt plans when problems are found.

DIAL's basic processing sequence is as follows:

- A story is input to the system.
- Candidate *response plan cases* for similar problem situations are retrieved, using static similarity assessment techniques to retrieve a set of cases from similar prior disaster situations.
- A second similarity assessment process uses learned information about the difficulty of case adaptation to select the candidate case whose response plan is expected to be easiest to adapt.
- The response plan is evaluated and problems which need to be repaired are identified and packaged for case adaptation.
- Problems in the response plan suggested by the selected case are repaired by case adaptation. During case adaptation, DIAL learns by storing traces of the case adaptation process and of the memory search process used to find needed information. If its case adaptation attempt fails, DIAL can also learn by recording a trace of a user-guided case adaptation process.
- The resulting response plan case is stored for future reuse.

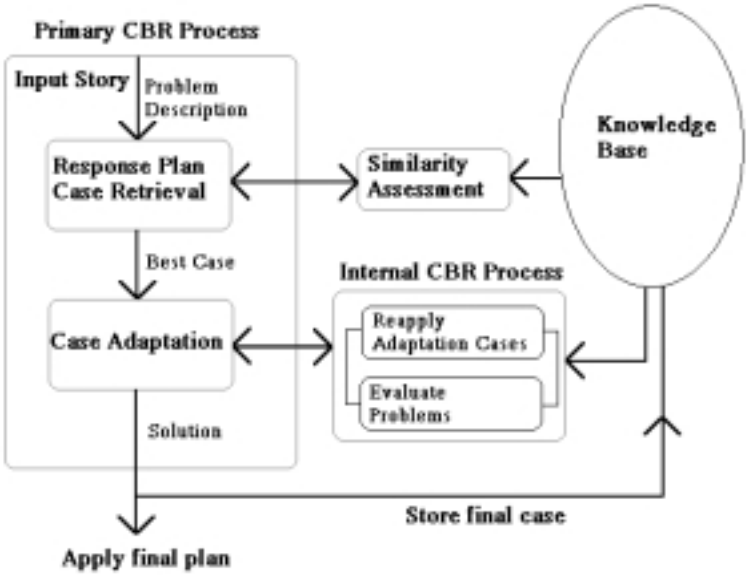

Figure 2.1: DIAL System Processing

Each step is discussed in the following sections. Figure 2.1 can be used as a guide to the flow of control of the system. This figure emphasizes the interaction of different knowledge sources in the case adaptation process. Both the case base and the rule base provide concrete information to support case adaptation while the similarity assessment, evaluation and case adaptation provide other forms of knowledge necessary to repair the current plan.

Story understanding component

The DIAL system contains a primitive story understanding component. It takes conceptual representations of news stories and extracts the basic information to be stored in a representation in the spirit of conceptual dependency (Schank, 1975). To illustrate, one of the examples processed by DIAL involves the following story: *At Beaver Meadow Elementary School in Concord, New Hampshire, students have been complaining of symptoms like unusual fatigue, eye irritation, respiratory problems, and allergic reactions from being inside the building.* When DIAL processes this story, a straightforward schema-based understanding process (Cullingford, 1978) identifies the disaster as an air quality problem. DIAL next attempts to retrieve and apply a response plan for a similar disaster.

Retrieval component

Important features, determined by pre-defined slots in a frame-based representation, are used as a primary index for retrieving a relevant response plan case. In DIAL, cases are assigned a score and ranked based on their feature similarity to the new situation. Important semantic features include but are not limited to: the type of disaster, the location of the disaster, and the primary victims of the disaster. The response plan retrieved for the Beaver Meadow story is the plan for the following factory air quality problem: *A & D Manufacturing in Bangor, Maine, has recently come under pressure from workers and union-representatives to correct perceived environmental problems in the building. Workers have been affected by severe respiratory problems, headaches, fatigue, and dizziness.*

Instantiation component

The response plan that has been retrieved is used as the primary guide in the development of a new plan for the current situation. A plan outline, based on the structure of the prior plan, is constructed after which various elements of the plan, including events and roles, are instantiated. Some prior role-filler values can be directly mapped into the new situation while other values are placed into the new plan without change after which the new plan is ready for evaluation. In our above example, a new plan is built with some fillers (e.g. the prior location (Bangor)) replaced by parallel fillers in the new situation (e.g. the new location (Concord)). Other values, such as the "worker's union," are left untouched and placed into the new plan to be examined by the evaluator.

Evaluation component

After instantiation, each response plan selected is evaluated to identify possible problems. Evaluation in the DIAL system is a two step process, DIAL makes a coarse grained assessment of the types of problems that might exist and identifies obvious problems, while a human user provides backup evaluation in the form of problem refinement. Each candidate role-filler value is analyzed and if the value is problematic, the user identifies the type of problem that caused the incompatibility. In this example, the response plan for A & D Manufacturing involves notifying the workers' union. DIAL's evaluator determines that the notification step does not apply to the current situation, because of a conflict with normative type restrictions on union members: elementary school students do not belong to unions. (The evaluation and

problem characterization process is similar to that described in Leake (1992b)). In the example, the user determined the union problem to be a role/filler mismatch.

Case adaptation component

DIAL's case adaptation component receives two inputs: an instantiated disaster response plan and a description of a problem in the response plan requiring case adaptation. DIAL next applies a suite of available methods to attempt to find a suitable solution to the problems identified by the evaluator. If possible, the system will make use of prior case adaptation knowledge to guide the new search. The solution to the Beaver Meadow School problem is found when the system determines that an authority relationship exists between the union and the workers and identifies a similar relationship which exists for the students. When it attempts to find alternate role-fillers with the same relationship, several possibilities arise including the student council, and the school administration. The user rejects these fillers but another alternative, the student's parents, is accepted as appropriate. This enables the system to replace weak knowledge (in the form of knowledge constraints) with specific alternatives from memory.

Storage component:

With a new response plan built, DIAL stores into its case base the entire new plan. In addition, a description of each case adaptation episode is stored as a case adaptation case. This storage process provides the system with an easier method of dealing with similar situations in the future, both on the plan level and at the case adaptation level.

DIAL's knowledge base

The knowledge used to solve problems was stored in a central hierarchical memory. This memory consisted of different concepts linked together via linguistic relationship. For example, the concept of *automobile* might be related to the concepts of *car* and *truck* via a parent/child relationship. This memory could be probed for needed information or searched to find concepts matching specified constraints.

2.2 Case adaptation learning

The previous section outlined the basic processing and architecture of the DIAL system within the context of the Beaver Meadow example. This section introduces the case adaptation process and the structures used to support that process in the frame of a second example.

This example processed a newswire report that an earthquake had occurred in Liwa, Indonesia. The DIAL system accepted the story and attempted to outline an appropriate response plan for the situation. No response plan that was currently in the response plan case base scored high with regard to similarity, however, the most appropriate response plan was one used for an earthquake in Los Angeles. This was selected primarily on the basis of the severity of the disaster and the lack of many available cases. The next subsection describes the representation used by the DIAL system to store this response plan case.

Plan representation

All plans in the DIAL system are stored as response plan cases. Figure 2.2 shows the system representation of the retrieved response plan from the Los Angeles earthquake disaster. Response plans are stored and retrieved from a central response plan case base that is seeded with a small set of preselected response plan cases. The four initial plans were hand coded and allow the system a starting point for the creation of new plans.

The response plan architecture consists of several components that include:

- **Index:** The index provides a concise description of the features of the disaster. It includes the location of the disaster, the type, the severity of the disaster and any other knowledge given by the initial disaster problem description.
- **Events:** Events are the set of actions to be performed in response to the disaster. Events might include mobilizing the police and developing rescue teams.
- **Role Fillers:** The fillers are individuals, or groups that perform the required roles of the events.
- **Constraints:** The constraints are *a priori* knowledge of what values or types of values are permissible as role fillers.

- **Statistical Information:** This information is used for external analysis of the system as a whole and is not directly applied during system processing. The data stored includes the time taken to retrieve the response plan, evaluate the response plan and adapt the response plan. While not shown in the printed representation this information is critical for proper evaluation of the approach described.

By examining the example in figure 2.2, several important features can be highlighted. The response plan structure begins with labels identifying the structure type and the name given to the plan. The name of the plan is computer generated and based on the location of the disaster and the type of disaster. Following this, the type of plan is given. This information can be used by the system to locate the appropriate plan, to store a new plan, or to find other plans of the same type. The roles and fillers of the plan appear next. The role names are given by the predefined response plans that form the initial case base. In this example, roles such as *police* and *residents* are placeholders for information that will be used in the events. The fillers for these roles use a non-standard representation for AI systems. The Wordnet knowledge base provides DIAL with access to a large set of interrelated concepts. Despite Wordnet's less optimal organization for a reasoning system, it is adequate and more practical than coding a complete knowledge base by hand. In fact, the flaws for reasoning of its organization makes it a more challenging basis for case adaptation learning (see section 2.6). The events include the list of participants in the action. These participants are specified with the same tokens used for the role names to allow for proper matching of filler values. When submitting the final plan to the user, each of the role place holders would be substituted with the corresponding and appropriate role filler.

Response plan creation

New response plans are created by reusing the information stored in past response plan solutions. One of the key issues in the reuse of past cases is the mapping of role fillers from the old situation to the new situation. This mapping can be non-trivial as new situations may require different types of knowledge to satisfy the specified constraints. Several approaches are used to support the mapping and selection of appropriate fillers for each required action including: examining the availability of resources, analyzing the constraints of the actions and applying the knowledge the system stores about groups and individuals. In the example response plan, the "national guard" was selected as the best filler for the prevent-looting action. However, in other situations using the same stored response plan as a starting point for solving a new problem, the "national guard" may be an unavailable resource and a different filler must be found to better match the current situation.

```
Response Plan                                          *structure type
    la-earthquake-response-plan                        *response plan name
    (earthquake-response-plan)                         *response plan type
    "Earthquake in Los Angeles"                        *printable name
    ((police "police")                                 *list of roles and common filler types
     (residents "resident")
     (location
        ((city "los_angeles")
         (state "california")
         (country "united_states")
         (continent "north_america")))
     (condition "catastrophic")
     (national-guard "national_guard")
     (fire-dept "fire_department")
     (relief-group "red_cross")
     (aid-group "united_states_government")
     (shelter "shelter"))
    ((rescue-survivors                                 *list actions and their participants
     ("rescue"
        (actor national-guard)
        (object residents)))
     (prevent-looting
        (("defend" (actor national-guard))))
     (build-shelters
        ("construct"
         (actor relief-group)
         (object shelter)))
     (provide-disaster-relief
        ("provide" (actor aid-group)))
     (remove-rubble ("excavate" (actor fire-dept)))
     (inspect-area ("inspect" (actor fire-dept)))
     (police-patrol ("patrol" (actor police)))))
```

Figure 2.2: Example of a response plan case

Response plan evaluation

Since the primary goal of the system is the creation of new response plans and the system accomplishes this by reusing stored plans, identifying problems with the prior plan in the new situation is a crucial precondition, however problem identification is not a research issue of this dissertation. DIAL examines the candidate plan using user defined rules that describe the appropriateness of fillers in different contexts. When the rules are insufficient for evaluation, the user can provide backup evaluation.

In the mapping of the Los Angeles plan to the Liwa disaster, a straightforward mapping from the old location to the new location is possible and does not cause any identifiable problems. However, the suggestion of using the Red Cross to provide aid and build shelters for the survivors of the disasters presents a problem. In most situations, this filler would be appropriate, however, the infrastructure in Liwa is unable to allow access to the affected area. This identified problem is then passed to the case adaptation component which attempts to repair this problem by finding a filler that can overcome this problem.

Methods of case adaptation

There are three primary methods of case adaptation in the DIAL system. Given inputs describing a candidate response plan and a problem to be adapted, the process performed by DIAL's case adaptation component is as follows:

1. **Case-based adaptation:** DIAL first attempts to retrieve an adaptation case that applied successfully to a similar previous problem. If the retrieval is successful, that case is re-applied and processing continues with step 3.
2. **Rule-based adaptation:** When no relevant prior case is retrieved, DIAL selects a transformation associated with the type of problem that is being adapted (e.g., role/filler mismatches, such as the mismatch between unions and students, are associated with substitution transformations: a mismatch can be repaired by replacing the role being filled or how the given role is filled). Given the transformation, the program generates a *knowledge goal* (Ram, 1987; Hunter, 1989) for the information needed to apply the transformation. E.g., for substitutions of role-fillers, the knowledge goal is to find an object that satisfies all the case's constraints on the object being replaced.

   The knowledge goal is then passed to a planning component that uses introspective reasoning about alternative memory search strategies (Leake, 1994a, 1995b) to find the information needed. This search process generates a memory search

plan whose operators include both an initial set of memory search strategies and *memory search cases* stored after solving previous case adaptation problems.

3. **Manual adaptation:** If both automated approaches to case adaptation fail, DIAL sends the problem to the user to be solved. The user guides the DIAL system through "how" the case adaptation should be performed, which is recorded and stored as an adaptation case.

The following subsections elaborate on the creation of knowledge goals, the memory search process, and the adaptation case representation using the Liwa earthquake story as an example.

Knowledge goal creation

In the Liwa example, there are no relevant cases in the system with which to perform case-based adaptation and consequently the system falls back on rule-based methods. To guide the search for information needed for a case adaptation, a *knowledge goal* is created. A knowledge goal stores the knowledge about the adaptation problem that is available from the current problem description and the instantiated plan. The knowledge goal is then used to provide guidance for a heuristic search of memory to find the required information. When problems are encountered during a search a knowledge goal will solve these sub problems by generating new knowledge goals reflecting the new problem.

The knowledge goal contains all available information relevant to beginning a search through memory. The key components of the knowledge are:

- **Slot:** The slot in the response plan to be adapted.
- **Current node:** The candidate filler suggested by the retrieved plan that was determined to be problematic.
- **Transformation type:** The type of transformation to be performed. In this research, we have focused specifically on substitution transformations. These are modifications where one filler creates a problem that is repaired by substituting a new filler.
- **Problem type:** The problem that requires case adaptation.
- **Constraints:** Any known restrictions on the solution.

The knowledge goal produced for the search of a replacement for the Red Cross in the Liwa response plan is given in figure 2.3.

Knowledge Goal

Name: "KG-14"	**unique goal identifier*
Slot: relief-group	**slot requiring new filler*
CurrentNode: ("actor" "person")	**initial solution*
Limit: 2000	**search limit*
Transformation: substitution	**transformation type*
Problem: means-of-lack-of-access	**problem-type*
Disaster: (earthquake ((city "liwa")	**current disaster*
(country "indonesia")	
(continent "asia")))	
Constraints:	**system designed filler constraints*
((has-abstraction? "organisation"))	

Figure 2.3: A system knowledge goal.

The memory search process:

The case adaptation processes in DIAL are guided by knowledge based searches of memory. This memory search process is driven by two different types of knowledge. The first set of knowledge involves manipulation of the knowledge goal, as the questions posed to memory by the knowledge goal may not be the precise questions needed to solve the current problem. The questions can be reformulated using *knowledge goal transformation rules*. For example, finding a replacement filler for the Red Cross is difficult. The Red Cross would in most circumstances be exactly the filler desired to provide aid and shelter to the survivors. Other charitable groups such as the Salvation Army will get the system no closer to a viable solution. Instead the knowledge goal can be transformed to search first for ways of overcoming the problem. When this problem is solved, the solution can be used as a guide for the original problem. The Red Cross cannot access the affected area in Liwa, so rather than searching for other groups, DIAL could transform the knowledge goal to find possible ways to access the area. One type of possible access during an earthquake is with helicopters. DIAL can use this knowledge to augment the original search and look for groups that can provide aid and also have helicopters.

The second type of memory search knowledge provided to the system is a suite of domain-independent *memory search operations* that depend on "weak methods" of

memory search. These operations are basic movements through the system's memory such as ascending or descending concept hierarchies to find related nodes. By repeatedly applying operations, a simple search of memory can be achieved that examines closely related concepts to an initial starting point and progressed towards distant concepts. With appropriate constraints, theses searches can wander to potential problem solutions. The reformulated knowledge goal described above suggests looking for fillers that can perform the same role as the Red Cross but also has the ability to use helicopters in the affected area of Liwa. With these constraints, one search of memory results in the discovery of the "military" as a possible filler.

In addition to providing the solution to the problem, the memory search process results in the step by step reasoning path that was followed. Traces of this search can be stored as *memory search cases* and made accessible for use during future memory searches.

This memory search process allows for a type of reactive planning similar to the RAPS system (Firby, 1989). The memory search process can respond to problems that arise during a specific search such as when necessary intermediate information cannot be found. As in the above example, the system can create sub-goals to circumvent problems encountered during the search. These sub-goals can be addressed as independent memory search problems for which the entire process can be repeated recursively.

Transformational case adaptations

The memory search process identifies solutions to problems represented in knowledge goals. However, these goals do not contain the necessary information about the type of transformation that must occur to apply the memory search solution to the new plan. Different types of transformations can be applied in different circumstances. In most cases, a substitution of the new filler can replace the value of the old filler directly. In other situations, additional information must be added to the new plan. For example, in order to substitute the military for the red cross, one of the subgoals generated was to find a means of transportation (and thus find individuals with that type of transportation). The means of transportation was added to the plan to satisfy this goal. Plan transformations must be performed carefully such that new types of conflicts are not created and that the original goal is still sufficiently satisfied.

Representation of cases learned from case adaptation episodes:

DIAL's *memory search cases* comprise the initial knowledge goal, a trace of knowledge-goal transformations and other memory search operations involved in the search process, a record of the search outcome (failure or success), the cost of the search in terms of primitive memory operations performed, and the resulting information found. Memory search cases are indexed under the knowledge goals that they satisfy, and can suggest search operations to attempt in the future. Memory search cases are accessible to the knowledge planning process for memory search, augmenting the initial library of built-in operators. For future searches, successful search cases that match the largest subset of the current knowledge goals are re-used. When the result of the stored search case does not satisfy current constraints, the search is continued by local search.

A memory search case contains:

- **An index:** The index for a memory search case includes the constraints of the original search and the starting location in memory.
- **The response plan:** The original response plan is referenced to provide a context for the solution the case provides.
- **The search path:** Each successful search results in a reasoning path that was followed to the solution. This trace of memory search operations is stored for future reuse.
- **Statistical information:** For later analysis, data about the amount of time, number of operations and nodes that were used during execution of the memory search is stored.

Figure 2.4 shows a memory search case taken directly from the successful solution to the Liwa Red Cross problem. Recall that the initial problem was identified because the Red Cross had no means to access the affected area of the earthquake and requiring instead a group with access. In this problem, the army was found as a plausible filler for the group to provide food and shelter to the survivors of the earthquake.

DIAL also packages *adaptation cases* including both the transformation used for the case adaptation and pointers to the appropriate memory search cases. Adaptation cases provide specific guidance about how to adapt cases to repair particular types of problems. Adaptation cases in the DIAL system contain all of the following information:

```
Memory Search Case
  "mem-search-case-504"                          *unique identifier for case
  index:
    (constraint:                                 *set of constraints used to find solution
       (has-abstraction? "group*))
    (original-node "red_cross")                  *original suggested filler
  response plan:
    (liwa-earthquake-response-plan               *plan and role of original
       relief-group)                                          solution episode
  search path:
    ((get-parent "organisation" )                *set of search steps taken
       (get-child "force")
       (get-child "military_service")
       (get-child "army" ))
  solution:
    ("army" (parent: "military_service"))        *actual solution filler
  statistics:
    (30170 1526 1406 1433 30460))                *statistical data
```

Figure 2.4: Example of a system memory search case

- **Index:** Adaptation case indexes are a description of the problem type that the case solves, with the response plan context and the slot being filled as secondary indices.
- **Transformation:** The type of transformation required by the case adaptation is stored. One type of transformation could be substituting for an invalid filler.
- **Memory Search Case:** The memory search case that stores the reasoning trace of the past solution is indexed by the adaptation case.
- **Type:** The type of adaptation represents the process used by the stored case to solve its case adaptation problem. These types are *weak-methods* for problems solved through undirected memory search, *strong-methods* for problems solved using learned case knowledge, and *manual-methods* for problems solved through user intervention.
- **Statistical Information:** As with other cases, this data is used to assess the

```
Adaptation Case
   "adaptation-case-15"                          *a unique name identifier
   15                                            *a unique numerical identifier
   adapt-case-index:
      means-of-lack-of-access                    *the type of problem solved
      (earthquake-response-plan                  *the originating plan for the
         liwa-earthquake-response-plan           identified problem
         relief-group)
      ((city "liwa")                             *the location of the disaster
         (country "indonesia")
         (continent "asia")
      "red_cross"                                *the proposed filler
   transformation: substitution                  *the repair required
   memory serach case:
      "mem-search-case-504"      *the memory case associated with the search
   solution: ("army" (parent: "military_service"))   *the final solution
   statistics:
      (30170 1526 1406 1433 30460)               *statistical data
      0                          *the number of times this case has been reused
   adaptation type:
      weak)                                      *type of solution episode
```

Figure 2.5: Example of a system adaptation case

relative cost of a particular adaptation case.

Figure 2.5 provides an example of an adaptation case solving the lack-of-access problem described earlier. This case represents all of the information needed to reapply it to new situations. In addition to the data described above, adaptation cases store additional information used to support system processes. One of these types of data is the number of times a given adaptation case has been reused successfully. This value is used by the system periodically to reduce the number of adaptation cases stored to limit the case base to cases with demonstrated relevance. Thus this mechanism prevents the total number of adaptation cases from exceeding the system's capabilities. With the case adaptation knowledge of the system represented as adaptation cases, new case adaptation can be performed without additional system knowledge.

Adaptation case organization:

DIAL's adaptation cases are organized by the problems they address using a vocabulary of problem types similar to those that guide case adaptation in numerous other CBR systems (e.g., Hammond, 1989; Leake, 1992). For example, if a candidate response plan is inappropriate because a role-filler is unavailable (e.g., a police commissioner may be out of town and unable to be reached in an emergency situation), the problem is described by the problem type FILLER-PROBLEM:UNAVAILABLE-FILLER, and that description is used as an index to retrieve adaptation cases for similar problems.

2.3 Learning methods and relationships

The DIAL system is supported by multiple learning processes that employ several different strategies. This section summarizes DIAL's learning processes and places them in the context of the strategies that are employed and the relationships between the processes. The strategies described included:

1. **Response plan learning:** The "baseline" learning method for DIAL is learning by case acquisition, the normal learning of case-based reasoning systems, with response plans reapplied by **transformational analogy** (Carbonell, 1983).

2. **Memory search learning:** When DIAL generates a memory search plan, it stores a trace of that plan as a *memory search case* for reuse by **derivational analogy** (Carbonell, 1986).

3. **Adaptation learning:** DIAL adapts cases by an introspective reasoning process deciding the transformations to apply and the knowledge goals to be satisfied. If DIAL cannot generate an acceptable case adaptation, it asks a user to guide the case adaptation process interactively. Traces of internally-generated case adaptations and of user case adaptations are stored as *adaptation cases* for reuse by **derivational analogy**.

4. **Similarity learning:** When similarity assessment is used to determine the case to adapt in a new situation, the goal is to select the stored case that will be easiest to adapt. DIAL's similarity assessment process uses prior experiences with case adaptation to estimate case adaptation costs for new problems by a **transformational analogy** process. Consequently, case adaptation learning and similarity learning are coupled; when adaptation cases are learned, that

learning provides not only knowledge to use during future case adaptation, but to use during similarity assessment as well.

To better comprehend how the different learning processes are related to one another and the type of interactions that may occur, four characteristics describing the learning processes have been identified. How these various characteristics manifest themselves in DIAL is summarized in table 2.1.

1. **Reasoning method:** Two different methods of reasoning are prevalent throughout the DIAL system. The first method uses rule-based reasoning as a set of predefined knowledge that can be used to support early learning in the system. The rules are carefully created to avoid many of the problems inherent to large rule-based systems. The second reasoning method used in DIAL is case based reasoning that drives the acquisition of new knowledge in the system.

2. **Reasoning type:** In support of each learning process, two different types of reasoning can be applied to solve problems. A *transformational CBR* approach generates new solutions by adapting prior cases while *derivational CBR* reapplies the reasoning of a past solution by following the steps taken to find new solutions.

3. **Level of reasoning:** The two primary learning processes in the DIAL system exist at two different levels. The main CBR process reasons about plans while the internal CBR process performs a type of meta-reasoning (about guiding the process for adapting plans to fit new situations).

4. **Control of reasoning:** A central issue in building multiple reasoning processes is how the interaction between individual learning processes exist. The main CBR process in the DIAL system sits in control of all other learning processes in the system. As such it can be termed a *master process*. Other processes in the system, such as the internal case based adaptation process, exist as subordinate processes to the master process. We will see that interactions can exist between a single subordinate process and the master process, and also between different subordinate processes.

The different types of cases in DIAL, plan case, memory search cases, and adaptation cases, can all be applied and reused independently of one another. That is, plan cases can guide plan generation and adaptation cases can guide case adaptation. However, these cases (as well as other sources of knowledge) can be viewed as complementary and able to support processes outside of their designated domain. DIAL

Task	Reasoning method	Reasoning type	Reasoning Level
Domain planning	CBR	Transformational	Task domain
Initial case adaptation	RBR		Internal processing
Subsequent case adaptation	CBR	Derivational	Internal processing
	RBR		Internal processing
Initial similarity assessment	RBR		Task domain
Subsequent similarity assessment	CBR	Derivational	Internal processing

Table 2.1: Characteristics of learning processes

attempts to exploit as many of the connections and interactions between the different learning processes in hope of each processes strengths overcoming the limitations of the others. We will show that the benefits of multiple learning processes far outweigh the costs associated with managing these interactions.

2.4 Overview of the different reasoning processes in DIAL

Several reasoning processes exist in the DIAL system that interact to support other reasoning processes. These processes can be categorized into three unique classes of reasoning.

1. **Transformational analogy**
2. **Derivational analogy**
3. **Rule-based search**

Figure 2.6 provides a schematic illustration of how each of the types of reasoning processes fit into this system's overall process.

Transformational analogy

While case-based reasoning is used throughout the DIAL system to support different reasoning process, there are different choices as to the type of knowledge the

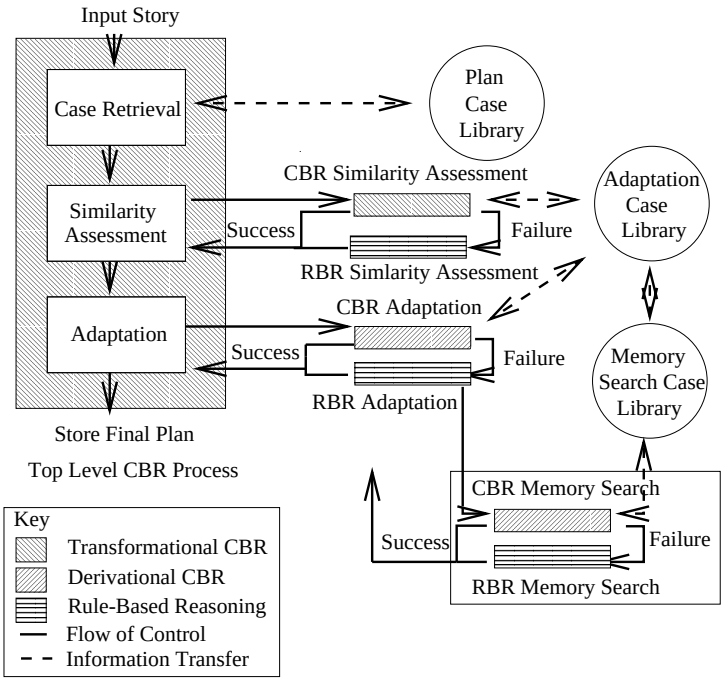

Figure 2.6: The integration of DIAL's different reasoning processes to support top-level transformational case-based planning.

cases should store. Transformational analogy (Carbonell, 1983), which modifies the solution of a problem, can be applied when a solution is known, but the rationale for that solution may not be. However, without information about the rationale, adapting the solution to fit new circumstances may be difficult.

For CBR tasks such as disaster response planning, derivations of solutions are not generally available, and planning from scratch is not satisfactory because domain theories are inaccurate and expensive to generate. However, examples of prior solutions are readily available from news stories and casebooks used to train human disaster response planners (e.g., Rosenthal et al., 1989). This favors a transformational approach to reusing disaster response plans. Thus the primary CBR process in DIAL is a transformational process.

Derivational analogy

Derivational analogy requires the rationale of prior problem-solving, but provides greater flexibility. When a reasoning process is replayed in new circumstances, the replayed decision-making process naturally takes into account the relevant features

of the new situation; adaptation of a derivation to fit new circumstances requires no additional mechanisms beyond those normally used for replaying cases.

On the other hand, derivational approaches can simplify the reapplication of a case to a new situation, and the rational for the CBR system's choice of particular steps during the adaptation of prior cases is available to a case-based adaptation component. This makes it possible to use derivational analogy as the basis of acquiring case adaptation knowledge from rule based memory search.

Rule based search

Rule-based search, using very general rules, enables initial case adaptation with minimal knowledge acquisition effort. However, general case adaptation rules are neither operation nor reliable. This supports using case-based reasoning when possible. Before cases can be successfully created a default reasoning process must be present to support this knowledge acquisition. A rule-base is easily developed and can reflect exactly the knowledge needed to search memory. Even after case-based reasoning becomes the primary reasoning process for case adaptation, the rule-based search provides a mechanism for filling in gaps in the case knowledge and providing a fallback reasoning method.

Comparison of the three approaches

In DIAL, transformational analogy provides the foundation of the system as the mechanism behind the case-based planning process. However, transformational analogy also supports some aspects of the case adaptation process as one simple method for reapplying the stored case adaptation knowledge. Derivational analogy, however, is more successful at case adaptation as it better encapsulates the reasoning process of the prior search. The cost of reapplying a derivation is much higher as the system has to asses or sometimes guess at the intent of each step in the reasoning trace. If a step of the trace suggests examining more specific instances of a concept (i.e. examining child nodes), it is an open question as the most appropriate way to focus of the first of the many possible concepts to consider next. Rule-based search supports acts as a subordinate process to each of the other two methods acting as both foundational support and as a guide to fill gaps in available knowledge.

Previous case-based adaptation systems store the solutions of a prior adaptation and reapply them by transformational analogy (Sycara, 1988). This is appropriate

when the derivation of the prior adaptation is not available. However, when derivations are available, derivational CBR is a natural means for providing flexible reuse (Veloso, 1994).

2.5 Integrating multiple processes

With several independent processes supporting different types of problems in DIAL, it is necessary to carefully manage how the system is integrated. DIAL provides a proof of concept for a system built of independent intelligent components. Riesbeck (1996) described a possible role for CBR supporting other systems by capturing the problems that occur and storing the generated solution to those problems. This is precisely what DIAL has accomplished. In this context, CBR is an *intelligent component* seamlessly integrated in a primary process and furnishes solutions to new case adaptation problems with no need to repeat the initial reasoning process.

This research applies the intelligent component technique to improving the performance of the CBR system itself. Each of the different reasoning subprocesses acts as a separate intelligent component providing the solutions to certain problems to other system components.

Three central questions provide guidance for examining the integrated reasoning components.

1. **System design:** How should the components' knowledge be represented and organized?
2. **Knowledge acquisition:** How much specialized knowledge must be provided to support component CBR processes, and how does this effort compare to hand coding rules for these processes?
3. **Effectiveness of learning:** How will the use of case-based components affect the overall efficiency of the top-level CBR system?

The first question has been addressed with the approach described in prior chapters by combining case-based and rule-base approaches. With this approach, the knowledge acquisition problem for case-based intelligent components has been controlled by intentionally limiting the components' initial knowledge to general knowledge with wide applicability, and using domain-independent "weak methods" as the foundation for its internal processing. This minimizes the burden of supporting the internal CBR process.

The key issue is how much gain can be achieved by using various intelligent subcomponents to support the primary system reasoning process. DIAL does not attempt to provide a general answer but demonstrates the feasibility of this type of approach. The potential value can be assess by how well this approach performs for a single sample system and task.

2.6 System knowledge sources

In case-based reasoning, the basic knowledge sources— plan cases and case adaptation knowledge—are overlapping in the sense that each can compensate for weaknesses in the other. For example, a large case library can compensate for limited adaptation knowledge, by providing cases that require less effort to adapt. Conversely, good adaptation knowledge enables successful reasoning with a smaller case library, by facilitating the reuse of existing cases. The internal CBR components make it possible for the system to learn either domain cases or adaptation knowledge (or both), learning multiple lessons from its experiences.

The interaction of DIAL's methods also helps each part to perform its processing. When the rule-based memory search process uses a case to suggest a search path, the case focuses its processing on a sequence of steps that—because it was useful in the past—might be expected to be useful again. In turn, DIAL's rule-based reasoning can be called upon by its internal CBR components. The case-based components of DIAL are intentionally limited to using very simple CBR processes, to simplify knowledge acquisition for these components. Consequently, reapplication of a single case may result in only a partial solution, which is then augmented by RBR.

General knowledge bases

The learning processes in DIAL each generate new knowledge for the system by solving problems and storing the solutions as cases. However, the success of each of the learning methods depends in some part on the organization of the system's general world knowledge. To this end, DIAL has been tested with two separate and independent general knowledge bases. The first knowledge base was hand coded for an initial set of tests in order to support the basic ideas of the system. This knowledge base included nodes for close to 2000 concepts. While this memory was sufficient for the tests that were run, a potential for bias exists whenever a knowledge base is built to correspond to the needs and requirements of a specific system. As DIAL's case adaptation mechanism was based on an abstract set of memory search operations

that were domain and knowledge base independent, there was concern that this bias might skew the results of any experiments that were constructed.

To compensate for this bias, many of the same experiments were performed using a second knowledge base built independently and externally of DIAL. The knowledge base selected for integration was the publicly available Wordnet system (Fellbaum, 1998). Wordnet is a hierarchical lexicon that provides relationships among over 100,000 different English words. Wordnet was designed to support various natural language tasks for computers and not specifically created to serve as a knowledge base. As such, many of the relationships one might find in a frame based memory (Minsky, 1975) or in the CYC ontology (Lenat & Guha, 1990), such as *color-of* or *is-relative-of*, are not present and consequently the operations that can be performed are limited to the scope provided by its authors. In addition, the Wordnet knowledge base was not designed to be extended by third parties in order to maintain system integrity. The result was a large knowledge base designed without many of the connections that DIAL had exploited in the hand-coded knowledge base.

The goals of the Wordnet lexicon differ substantially from those of traditional knowledge representation systems. Traditional knowledge representations, whether in the form of rules, cases or other common representations, are constructed to solve certain types of problems. In fact, these representations are carefully designed so that the appropriate new inferences can be drawn. Wordnet was not designed as a knowledge representation system but most closely resembles a supporting system for some forms of information retrieval. Wordnet, for example, could be used to support web based search engines. Its lexicon could help augment or improve engine queries by identifying shades of meaning or word choice alternatives that language provides. For example, Wordnet could take a search query term such as "military" and provide alternative terms that might produce desired results such as "armed forces" or "national guard." To make use of Wordnet as a knowledge base places the burden of knowledge organization and inference on the primary system. DIAL must overcome poor knowledge connections in the Wordnet memory and learn to construct new relationships externally to the knowledge base.

The basic operations supported by the Wordnet system are as follows:

- **Hypernyms:** A hypernym is the generic term used to designate a whole class of specific instances. If X is a hypernym of Y, then Y is a (kind of) X.
- **Hyponyms:** A hyponym is the specific term used to designate a member of a class. If X is a hyponym of Y, then X is a (kind of) Y.
- **Meronyms:** A meronym is the name of a constituent part, the substance of, or a member of something. If X is a meronym of Y, then X is a part of Y.

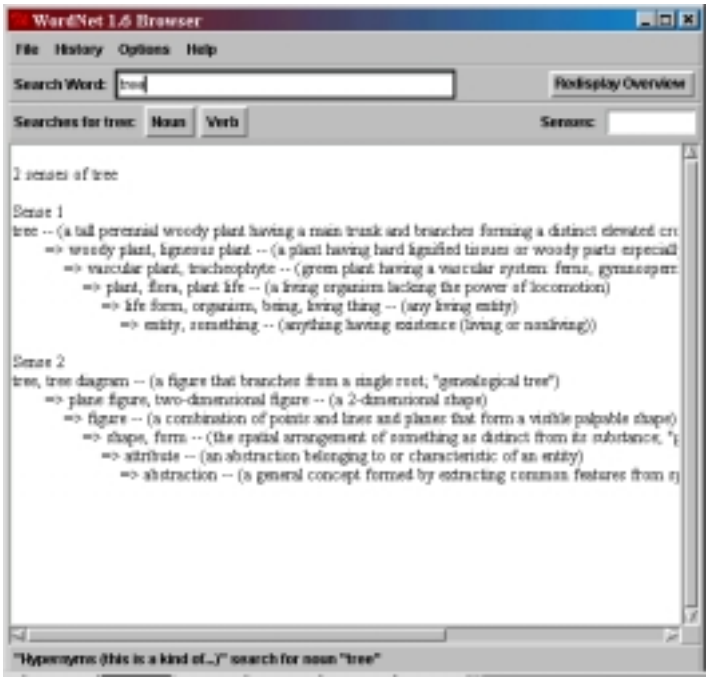

Figure 2.7: Wordnet interface

- **Synonyms:** Two equivalent terms in terms of their relationships to other classes are synonyms. If X is a synonym of Y, then X and Y are equivalent.

- **Antonyms:** Two terms that are opposites in a language. If something is X than it is not Y.

The DIAL system could make use of any of these operations, however, in practice hypernyms and hyponyms were the operations most often applied successfully. Meronyms occasionally proved useful and synonyms and antonyms were not used.

DIAL has its own internal interface to the Wordnet system. However, to gain a better understanding of what the output of an operation in Wordnet would be, a screen shot is provided in figure 2.7 of the graphical interface of the Wordnet system executing a hypernym on the concept "tree." This figure reveals one additional difficulty for DIAL to handle – multiple senses for a single linguistic term. The term "tree" has two separate meanings: a type of plant, and a type of graph representation. In DIAL's concept representation each sense of a word was tagged and thus each sense could be treated as a unique concept. Unfortunately, not all senses of words had such clear divisions of meaning and DIAL had to learn to navigate these gray boundaries.

2.7 DIAL example

The remainder of the chapter presents an actual transcript of the DIAL system handling a typical problem that requires case adaptation. Again the Liwa example is used but unlike the above example a slightly different outcome is seen in the particular situation as disaster examples were presented in a different order during this learning trial.

The example highlights the retrieval component of the DIAL system. The example begins with the presentation of the disaster to the system. The system uses adaptive similarity techniques described in a later chapter to select the response plan case that most closely matches the input example. To perform this similarity assessment an evaluation of each potential plan is made with the outcome being the selection of the ecuador-earthquake response plan. Comments are given in italic type within the example.

The initial disaster is presented to the system

```
Current disaster is
   (earthquake ((city "liwa")
               (country ("indonesia" noun 1 ()))
               (continent ("asia" noun 1 ()))))
```

Enter the retrieval component

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Retriever Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Selected 3 cases for similarity assessment
```

Similarity scores are given in parentheses. Lower scores indicate greater similarity.

```
The following cases have been ranked:
   Case:  neftegorsk-earthquake-response-plan (4)
   Case:  ecuador-earthquake-response-plan (5)
   Case:  la-earthquake-response-plan (5)

Using adaptive similarity techniques (if available)
```

Each part of the candidate plan is evaluated and a cost assigned

```
Evaluating the ecuador-earthquake-response-plan
------------------------------------------------------------------------
```

```
  slot            filler                 problem  type                cost
-------------------------------------------------------------------------
  police          "military_personnel"   no       n/a                    0
  residents       "resident"             no       n/a                    0
  location        "liwa"                 no       n/a                    0
  condition       "catastrophic"         no       n/a                    0
  national-guard  "military_personnel"   no       n/a                    0
  fire-dept       "fire_department"      yes      filler-does-not-exi ??
  relief-group    "red_cross"            yes      means-of-lack-of-ac ??
  aid-group       "red_cross"            no       n/a                    0
  shelter         "tent"                 no       n/a                    0
```

For each problem, the relevant adaptation cases are retrieved and compared

```
Best Adaptation Case:
   Name:  adaptation-case-9,
       Evaluation:  Ops:  4026 Nodes:  2250 CPUTime:  68510 RealTime:  69030

Best Adaptation Case:
     Name:  adaptation-case-5,
       Evaluation:  Ops:  150 Nodes:  78 CPUTime:  2350 RealTime:  2450

[some evaluation deleted]
```

The original plans are rescored with respect to their adaptability
Lower scores reflect greater adaptability

```
The problem based costs are:
     Case:  neftegorsk-earthquake-response-plan - (3)
     Case:  la-earthquake-response-plan - (6)
     Case:  ecuador-earthquake-response-plan - (2)
```

The final retrieved plan is passed on to the next component

```
The Selected RP is:  ecuador-earthquake-response-plan
```

The second transcript given below illustrates one part of the case adaptation process that has been described. In this example, two problems are identified with the proposed plan including one problem that is similar to the relief-group example

described above. This problem is solved using straightforward search methods culminating in adaptation cases and memory-search cases identical to the ones described in the earlier sections.

The second problem proposes the fire department as an appropriate filler for handling many of the tasks required by the plan but it is determined that no such filler exists in the context of the new situation. Case-based adaptation is employed to find a similar problem that existed earlier and determines again that the "army" is an appropriate and available substitution.

Fire department is the current filler and has a problem to be repaired

```
Adapter Module:

**Problem:  filler-does-not-exist-in-current-context
**Old-Value:  (fire-dept, "fire_department")
```

A knowledge goal is created to guide the case adaptation process

```
Created knowledge goal:  "KG-14"
```

First case-based case adaptation is attempted

```
Trying CBR Adaptation
```

The retrieved case is reapplied directly

```
Using Adaptation Case:
  Name:  adaptation-case-9,
  Evaluation:  Ops:  4026 Nodes:  2250 CPUTime:  68510 RealTime:  69030
```

The solution derivation for the fire dept problem is stored
as the following case

```
Storing AC:
  Adaptation Case:
      Name:  adaptation-case-14,
      Evaluation:  Ops:  31 Nodes:  67 CPUTime:  1010 RealTime:  1150
         ((has-abstraction?  ("group" noun 1 ())))
         ((has-abstraction?  ("group" noun 1 ())))
```

A second problem exists with the relief group filler

```
**Problem:  means-of-lack-of-access
**Old-Value:  (relief-group, "red_cross")
```

Another knowledge goal is created to guide case adaptation

```
Created knowledge goal:  "KG-15"
Trying CBR Adaptation
```

An adaptation case is reapplied but fails

```
Using Adaptation Case:
  Name:  adaptation-case-5,
  Evaluation:  Ops:  150 Nodes:  78 CPUTime:  2350 RealTime:  2450
Failure
```

Rule based search commences

```
Trying RBR Adaptation
  Solution Found:  "army"
```

The solution is stored for future reuse

```
Storing AC:
  Adaptation Case:
      Name:  adaptation-case-15,
      Evaluation:  Ops:  1526 Nodes:  1433 CPUTime:  30660 RealTime:  30940
         ((has-abstraction?  ("group" noun 1 ())))
```

A report is produced detailing all successful case adaptations

```
Adaptation summary
Type:  earthquake Location:  "liwa"
-------------------------------------------------------------------------
  slot            problem        solvedby   solution  ops    nodes time
-------------------------------------------------------------------------
  relief-group    means-of-lac   "army"     rbr       1526   1433  30660
  fire-dept       filler-does-   "army"     cbr       31     67     1010
```

The final plan can now be applied and stored

```
Storing liwa-earthquake-response-plan
```

2.8 Summary

This chapter highlighted the major features of the DIAL system. Two extended examples of DIAL's processing were given to motivate the various representations and processes used while solving problems. Subsequent chapters will focus of the primary aspects of DIAL's processing that relate to this research.

3

Adaptation Learning

This chapter gives a detailed description and analysis of the case adaptation learning component in the DIAL system.

The previous chapter examined the architecture and flow of control in the DIAL system. The next two chapters describe the primary learning processes that are integrated in the DIAL system and examine some of the specific issues and ramifications associated with them. This chapter describes an algorithm for learning case adaptation knowledge in the framework of our case-based planner. It begins by describing the case adaptation problem and its importance to the case-based reasoning process. Finally, a case-based model of acquiring case adaptation knowledge is presented and motivated with a case adaptation example.

3.1 The case adaptation problem

In the first chapter, case adaptation was described as one of four primary processes of the case-based reasoning process. However, many CBR researchers choose not to perform any case adaptation in their systems or limit the system to simple rule based methods to adapt proposed solutions. Avoidance of case adaptation is not an acceptable approach to handling the case adaptation question. In many real world situations, there is no exact solution to problems and each new situation mandates changes to prior solutions. Avoidance has been accepted only because automated case adaptation is difficult (Allemang, 1993; Leake, 1994b). Despite the practical benefits of retrieval-only advisory systems, successful use of advisory systems may require

considerable user expertise. Consequently, automatic case adaptation is important from a practical perspective.

There are numerous advantages to a automated case adaptation algorithm for case-based planning systems. For example, fewer plan cases could be stored because a good automated case adaptation system would handle the increased problems encountered effectively. A benefit of a reduced number of cases is that overhead due to case retrieval is decreased and a greater number of cases can be examined as candidates for reapplication. Additionally, the automated case adapter lessens the burden on human users to correct each small problem that arises in the creation of new plans.

Reasons against automated case adaptation

Two central issues provide the primary arguments against deploying automated case adaptation.

- Automated case adaptation may be not be computationally feasible.
- A case adaptation component that performs even moderately well would require time and knowledge resources that are unavailable or unreasonable. This section examines each of these issues and argues that automated case adaptation is not only possible but that its deployment is and practical.

Issues for automated adaptation

Automated adaptation is a hard problem for at least three reasons.

1. The knowledge needed to perform automated case adaptation can be different from the knowledge needed to create new plans. This is a central issue to a system such as DIAL. The knowledge needed by DIAL to form new plans requires representing knowledge of events and their relationships to one another. Case adaptation makes use of this information but requires specific knowledge of the various candidates needed to fill the roles of these events. Because of this, knowledge gained by the addition of new plans and the knowledge represented to support the creation of these plans are insufficient to perform the types of case adaptations required. Thus knowledge acquisition for case adaptation knowledge is a difficult task and separate from other knowledge acquisition in the system.

2. Context plays a pivotal role in determining the success or failure of a proposed case adaptation. A case adaptation used to solve an access problem in a flood in West Virginia may be very different from a case adaptation used to solve the same type of problem in Ecuador. Some possible case adaptations are available in one context and not in another so the automated case adaptation component has to understand and represent these inconsistencies by accounting for context in each case adaptation performed.

3. A solution to one case adaptation problem may affect the remaining plan, potentially invalidating other roles and fillers. It is difficult to assess the consequences of a single case adaptation. A case adaptation that results in an unsatisfactory plan has wasted substantial processing and moved no closer to a solution.

The second issue facing automated case adaptation is one of overhead cost versus the payoff of that cost. If a problem is solved by case adaptation, then the total cost incurred by this process should be less than if the problem had to be solved from scratch. The design of the automated case adaptation procedure must provide methods to reduce the added overhead to the system while achieving the greatest gain in speedup and accuracy of the solution. Experimental data can provide a means of analyzing and addressing the cost of the case adaptation process and any added overhead.

This research presents one method for acquiring and reusing case adaptation knowledge thus enabling good case adaptations to be performed autonomously by the CBR system. In addition, it is shown that this method can be an efficient and robust addition to the overall CBR planning process.

Background for case-adaptation learning

To successfully automate the case adaptation process, appropriate knowledge must be available to guide the process. Several different suggestions have been made as to the type of knowledge to use. One popular method has been to encode a set of rules. A rule could be used when certain system conditions are met. The rule would describe the appropriate action or set of steps needed to solve the problem. For example, a rule could read "If the red cross in unavailable and a lack of access problem exists then replace the red-cross with the local military organization." This rule quickly and effectively solves all problems of these types.

Rules themselves can take many forms. They could be abstract such as *add a new step to remove a harmful side-effect* as in the CHEF system or more specific such as *replace red-cross with army.* Abstract rules enable a system to function on a

relatively small rule set. However, abstract rules can be difficult to apply. An abstract description gives no clues as to when a certain rules in relevant to a situation or even if it would solve the problem if relevant. Of course, on way around this is to build a second set of rules to identify when to apply but that eliminates the benefits of abstract rules.

Other systems use specific knowledge to circumvent the problem of abstract rules. One example, the ROENTGEN system, attempts to generate X-ray treatment plans. In one instance, the retrieved plan administers the minimum X-ray dose required to destroy a tumor, but also has the bad side-effect of exposing the spinal cord to excessive radiation. With only the abstract rule given above, deciding which step to add in order to remove the bad side-effect may require considerable domain knowledge (in addition to the potential dangerous application of adding an additional harmful step). The abstract rule could be replaced by specific rules such as *add the step "rotate radiation sources" to remove harmful side-effect "excess radiation"* (Berger & Hammond, 1991).

However, neither abstract nor specific rules are sufficient. They demonstrate the operationality/generality tradeoff seen in explanation-based learning (e.g., (Segre, 1987)). Abstract rules have generality: a small set of transformations appears sufficient to characterize a wide range of case adaptations (Carbonell, 1983; Kolodner, 1993). However, abstract rules are difficult to apply. Specific rules, on the other hand, are easy to apply but have limited generality. In addition, defining such rules is difficult because of the specific knowledge that they require.

Kass (Kass, 1994) proposed one way to address the operationality/generality tradeoff. His approach uses hand-coded *case adaptation strategies* that combine general transformations with domain-independent memory search strategies for finding the domain-specific information needed to apply the strategies. However, coding the case adaptation strategies can require extensive knowledge of the CBR system's task, its domain, and the contents of its memory. This knowledge may not be available *a priori*. Thus in defining these case adaptation strategies, developers face the same problem of knowledge acquisition in imperfectly-understood domains that often impedes the development of rule-based systems in other contexts.

3.2 A proposal for case adaptation learning

The DIAL approach to case adaptation builds on the idea from Kass and related work (Leake, 1994a) of treating case adaptation knowledge as a combination of knowledge about general transformations and about memory search. However, instead of

relying on hand-coded memory search strategies, this approach builds memory search strategies as needed and remembers the strategies for future use. When presented with a novel case adaptation problem, it performs a planning process that reasons introspectively to determine the information required to solve the particular case adaptation problem and to decide which memory search strategies to use to find that information. This process guides the search for information needed to perform the case adaptation. Further, once a particular case adaptation is performed, the reasoning required to solve the problem can be stored as part of a case-based adaptation process enabling future similar case adaptations to circumvent this lengthy reasoning process.

Initially, little or no specific knowledge about case adaptation is stored. A few classes of transformations are maintained such as substitutions, additions, deletions and act on various levels of the generated plan. In addition, a small set of memory search operations exist to enable the system to search from the appropriate information in the knowledge base.

This part of the process is primarily rule based but differs from prior rule based systems by requiring little domain knowledge to find a solution. This rule-based process, however, enables the system to transform itself from the default case adaptation mechanism to a case based adaptation process.

From rule-based adaptation to CBR

> Successful sets of applicable rules can be stored as adaptation cases for later reuse.

After a case adaptation problem has been solved by reasoning from scratch, a natural question is how to learn from that reasoning. It might appear that explanation-based generalization (EBG) (e.g., (Mitchell, Keller, & Kedar-Cabelli, 1986)), would be the appropriate learning method, because it allows forming new generalizations that can aid in solving a wider range of problems. The memory search plan that found the needed information could be generalized and stored. However, using EBG to learn memory search rules is not practical (Leake, 1994a). For EBG to apply successfully to memory search rules, those memory search rules must provide a complete and correct theory of the contents and organization of memory. Unfortunately, the contents and organization of a specific memory are highly idiosyncratic (Kolodner, 1984; Schank, 1982) and thus hard to characterize precisely. Consequently, a chain of memory search rules that finds desired information in one instance is not guaranteed

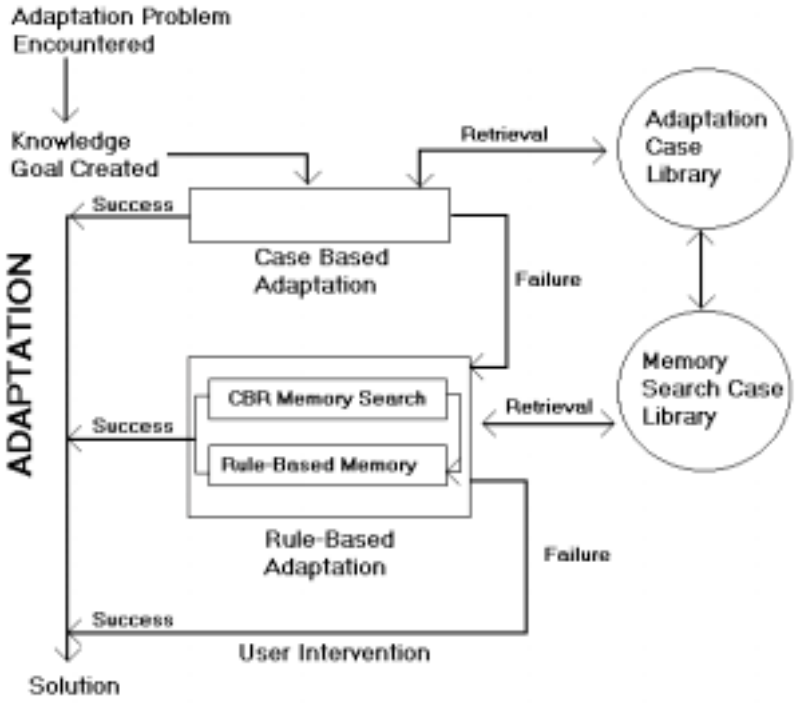

Figure 3.1: DIAL case adaptation process

to apply to other problems that appear to be within the scope of those same rules: explanation-based generalization may not yield reliable results.

In contrast, using case-based reasoning as the learning method for case adaptation knowledge makes it possible for learned knowledge to reflect the memory's idiosyncratic organization and its contents. Unlike abstract case adaptation rules, cases that package particular case adaptation episodes encapsulate the system's experience on specific case adaptation and memory search problems and reflect the system's specific task, domain, and memory organization. Consequently, DIAL applies CBR to learning adaptation cases. Thus this model acquires not only a library of problem-solving cases, but also a library of cases representing episodes of case adaptation. Case adaptation is realized in this system by three distinct mechanisms: case adaptation from scratch in response to novel case adaptation problems, case-based adaptation to re-use the results of previous case adaptation episodes, and manual case adaptation with the user providing guidance to lead the system through difficult or potentially intractable case adaptations. The following section presents an example of a case adaptation being performed from scratch and later being reused to solve a different case adaptation problem.

Figure 3.1 presents a high level model of this process. Greater detail on each facet of this process will be presented in the next section.

3.3 The case adaptation algorithm

The central case adaptation algorithm shown in figure 3.2 is given the knowledge goal encapsulating a description of the problem and the current problem situation. The knowledge goal, as described in the previous chapter, contains all the information available on the current problem requiring case adaptation. Other context information including the current disaster description and the proposed response plan are also given to the case adaptation component.

The case adaptation component has a hierarchy of sub-processes that are used to perform the case adaptation. Given the input to the case adaptation process, the case-based adaptation component is passed the knowledge goal and any adaptation cases deemed applicable by the evaluator. This process, described in more detail later, either is successful and returns a new knowledge goal containing the solution to the case adaptation problem along with other statistical information or it returns failure. When case-based adaptation is successful, the solution is applied to the response plan and the case adaptation component returns.

When the case-based component fails, either because no applicable adaptation-cases were available or a reapplication did not result in a solution, the rule-based adaptation component is called. The rule-based component applies weak search methods to the problem described by the knowledge goal. As with the case-based component, if the search is successful in discovering a solution, a new knowledge goal encapsulating the solution is returned to the adapter – otherwise a failure is signaled.

When both automated methods of case adaptation fail, the DIAL system relies on a manual case adaptation component that allows a human user to guide the system through a possible case adaptation.

> Three independent sub-components form the foundation for the automated case adaptation process.

The following subsections examine each of the three case adaptation sub-processes in more detail focusing on the algorithms used for each.

Input : A knowledge goal, the current response plan, the current disaster, and the problem-type
Output : The modified response plan with the knowledge goal satisfied for the given problem type

procedure **Start_Adaptation**(*knowledge_goal, response_plan, disaster, problem_type)*
1. *solution* ← **CBR_Adaptation**(*knowledge_goal)*
2. if (*solution)*
3. **Update_Response_Plan**(*solution, response_plan, knowledge_goal)*
4. return
5. *solution* ← **RBR_Adaptation**(*knowledge_goal)*
6. if (*solution)*
7. **Store_Adaptation_Case**(*solution, problem_type, disaster, response_plan)*
8. **Update_Response_Plan**(*solution, response_plan, knowledge_goal)*
9. return
10. *solution* ← **Manual_Adaptation**(*knowledge_goal)*
11. **Store_Adaptation_Case**(*solution, problem_type, disaster, response_plan)*
12. **Update_Response_Plan**(*solution, response_plan, knowledge_goal)*
13. return

Figure 3.2: General algorithm used for case adaptation

Rule-based case adaptation

The description of case adaptation in the DIAL system begins with the rule-based process because it is the basic fallback reasoning process for the adaptation component and the foundation for the case adaptation learning process. Therefore all automated case adaptation in the DIAL system has its roots in a rule-based process. The rule-based methods can be inefficient and sometimes ineffective but when successful provide both a solution and the exact information needed to create adaptation cases. The algorithm given in figure 3.3 illustrates the basic outline of the rule-based process. This algorithm contains methods to create new adaptation cases from the results of a successful search and stores these new cases in the case library.

The search method takes a list of operators that can be applied and the current knowledge goal describing the problem to be solved. The search expands concepts that most closely match the constraints from the knowledge goal first by applying the set of possible memory search operations to the current memory nodes being examined. While this search potentially is exhaustive in its examination of the memory space, it is limited to the guidance provided by the knowledge goal. Since this knowledge may not match the information needed by the search, the resulting search can be expensive.

As such, a limit on the number of nodes that can be examined was imposed on all rule-based searches to prevent . Once this limit is reached, the current search is stopped. Thus some problems may have solutions that exist but are unreachable in the search space given the time constraints.

The use of Wordnet as a knowledge base increases the cost of the search process as each node in the knowledge base may have hundreds of direct connections to other nodes. In tests of the rule-based methods, problems are solved more frequently than left unsolved but despite the system imposed limit, the time required to complete some of these case-adaptation is much higher than desired.

Currently in the DIAL system there are seven different operations making up the basic memory search rules of the system. The operations were selected on the basis of the organization of the memory being used and the representation of the knowledge goals. The operations available are as follows:

- **get-parent** – Selects the hierarchical ancestors of the current node. For example, the parent of the automobile node would be the vehicle node.
- **get-children** – The inverse of the get-parent operation, this selects all of the hierarchical descendants of the current node. One of the many possible children of the vehicle node is the automobile node.
- **goto-node** – Allows the system to jump from the current node to some other node in memory. Most often, the new node provides a new starting point for search to begin.
- **get-plan-fillers** – For some situations, fillers that have already been selected and are non-problematic in the plan can be used either as starting points for new searches or as fillers for newly created constraints. For example, a disaster problem might state that the police are already involved with the current disaster. This filler value might make a useful starting point for other problems with related constraints. This operation allows the searcher to immediately examine this area of the memory for other possible solutions.
- **value-from-constraints** – Takes a value or values from the constraints and uses them directly to guide search. For example, a new constraint may be added to a knowledge goal which defines an abstraction relationship with the goal filler. This abstraction might be used as a starting place for the new search to begin. A search may have a more restrictive constraint placed on it. For example, the group being searched is thought to be a military organization. This rule guides the searcher to move to the concept of military organization and begin the search from that point.

- **del-constraint** – Allows the search to remove a restrictive constraint from the knowledge goal. Constraints are normally assumed fixed in order to reduce the amount of searching required before failure is determined. By allowing deletion of some constraints, a wider search area can be examined. One such example of a restrictive constraint is when a specific type of organization such as the Red Cross constrains the solution. It may be possible that a different organization is more applicable to the new situation and the deletion of the constraint will free the searcher to examine these possibilities.
- **add-constraint** – For reasons similar to the deletion of a constraint, certain knowledge sometimes becomes available during a search that the addition of a constraint helps refine the search and eliminate many potential non-solutions from consideration.

These rules make the case adaptation search process effective and capable of solving a wide array of case adaptation problems. However, the cost to apply these rules in an ad-hoc manner with ill-specified constraints make them unsuited for general purpose case adaptation. The rule-based approach does make an excellent foundation for the internal case-based reasoning adaptation learner. This process forms the discussion in the next subsection.

Case-based adaptation

Figure 3.4 shows the algorithm used by the case-based adaptation component. The key aspect of this algorithm is the reapplication of the reasoning trace stored by the adaptation case. This trace coupled with the knowledge goal enables the case adaptation component to perform a directed search of memory on the current problem.

The same operations used by the rule-based process can be reapplied in two separate ways. The first approach reapplies the same operations to the knowledge goal that are described in the adaptation case's prior reasoning trace. As each operation is reapplied, a new set of knowledge goals are created as the result of the reapplication. For example, when applying the get-parent operators to a knowledge goal that is currently examining "groups of force" the resulting knowledge goals will include representations for the army, the national-guard and the police. The constraints are used to eliminate knowledge goals that do not apply to the current situation.

As the reapplication progresses and pruning occurs on the resultant knowledge goals, a set of potential solutions to the case adaptation problem are identified. These potential solutions are ranked for applicability and evaluated. It is possible at this

Input : A knowledge goal
Output : A structure containing the solution to the problem,
and statistical data regarding the case adaptation or failure if no solution found.

procedure **RBR_Adaptation**(*knowledge_goal)*
1. *solution* ← **Search(***operator_list, knowledge_goal)*
2. if (solution)
3. *new_adaptation_case* ← **Create_New_Case(***solution, adaptation_case)*
4. **Update_Case_Library(***new_adaptation_case)*
5. return *solution*
6. else
7. return failure

Input : An list of available operators and a knowledge goal
Output : A knowledge goal or knowledge goals reflecting the result of the search
procedure **Search**(*operator_list knowledge_goal)*
1. *kg_queue* ← **Make_Queue(***knowledge_goal)*
2. while (**Goal_Not_In_Queue(***kg_queue))*
3. if (**System_Timeout) exit**
4. for each *operator* ϵ *operator_list*
5. for each *kg* ϵ *kg_queue*
6. *new_kg_list* ← **Apply(***operator kg)*
7. *kg_queue* ← **Replace(***kg_queue kg new_kg_list)*
8. return **Get_Goal(***kg_queue)*

Figure 3.3: Algorithm for rule based adaptation and memory search

Input : A knowledge goal and an adaptation case
Output : A structure containing the solution to the problem and processing statistics

procedure **CBR_Memory_Search**(*knowledge_goal, adaptation_case)*
1. *solution* ← **Reapply_Derivation(***knowledge_goal,adaptation_case)*
2. if (*solution)*
3. *new_adaptation_case* ← **Create_New_Case(***solution, adaptation_case)*
4. **Update_Case_Library(***new_adaptation_case)*
5. return *solution*
6. else
7. return failure

Figure 3.4: Algorithm for reapplication of adaptation cases.

stage of reapplication that the desired solution has not been discovered despite the appropriateness of the derivation. To handle this situation, a limited local search is performed around the best memory nodes found from the reapplication. In situations where no acceptable solution is found, the case based case adaptation reports a failure and the case adaptation problem is passed to the rule-based component.

A second method of case adaptation uses the adaptation cases with straightforward transformational analogy and presumes the solution to the prior case adaptation problem can be reapplied directly without modification. Since this method has high utility with a low cost of application it is always attempted, and often solves trivial problems instantly.

If the case adaptation is successful, a new adaptation case is formed and stored in the case base. The overhead involved in this process is greater than with rule based methods alone, but because rule based methods may involve considerable backtracking in their search, a successful reapplication of an adaptation case may give a quick start to such a search. The success of the case adaptation approach can be measured by the time reduction in the processing of case adaptation problems including the additional overhead of maintaining the internal CBR process. It is critical in this research to accurately test the time savings of this method when compared with the additional system overhead.

Manual adaptation

In the event that neither automated method of case adaptation is successful, the DIAL system has a backup case adaptation component that provides for user

Input : A knowledge goal
Output : A structure containing the solution to the problem.

procedure **Manual_Adaptation**(*knowledge_goal)*
1. while (**NOT (Goal_Satisfied(***knowledge_goal)))*
    2. *knowledge_goal* ← **Select_Modification(***knowledge_goal)*
    3. **Search(***operator_list knowledge_goal)*
4. return *knowledge_goal*

procedure **Select_Modification(***knowledge_goal)*
1. repeat until done
    2. Select a modification to perform
        a) Add a constraint
        b) Delete a constraint
        c) Replace a constraint
        d) Continue as is
        e) Manual Modification
        Query Your Choice $\Longrightarrow$
    3. *knowledge_goal* ← **Apply_Selection(***knowledge_goal)*
4. return *knowledge_goal*

Figure 3.5: Algorithm for manual adaptation

intervention. Figure 3.5 shows the general outline of this manual case adaptation approach. Manual case adaptation provides a mechanism for handling difficult case adaptation problems for which the appropriate knowledge does not currently exist in the form of either cases or rules.

This method allows the user to interact with the system at different reasoning levels to repair an identified problem. In the simplest scenario, the user may suggest relaxing constraints or specify a new concept related to the unknown solution. In other instances, the user may guide the system through the reasoning needed to solve the problem. In either situation, the system captures a trace of the human user's reasoning – albeit within the constraints of the system's representations – that augments and improves on its built in reasoning. The derivation provided by the user becomes an adaptation case within the system's memory that could be reapplied in similar circumstances. The potential for new types of reasoning, otherwise unavailable to the DIAL system, is made possible by adding cases to the system from the results of the manual adaptation.

System example

To review the processing and interaction of the three case adaptation subprocesses a new example is presented from the DIAL system. This example begins with a reported chemical spill problem at the A&D manufacturing plant. An emergency plan is needed to handle the situation. One of the problems to solve is the appropriate way to handle the evacuation of the plant and process the numerous workers who will be unable to work until the problem is resolved. The candidate plan suggests that the employees act as their own representatives in this situation and the plant interact directly with them. A user identifies this solution as an inappropriate filler. It is not productive for a factory to attempt to process each individual directly. Case-based case adaptation finds no relevant cases to this problem and the system falls back on rule-based methods. A local search of memory, made by applying the rule-based operations and searching for groups related to the workers, discovers several possible solutions. With a user performing backup evaluation, the proposal suggesting that the union of the workers is selected as the most appropriate filler to be added to the plan. With the union added to the plan, the factory has a means of notifying employees of current situations and handling any additional employee concerns.

With the success of the search, a new adaptation case is created and added to the case base. This case encapsulates the set of steps from the memory search from the original worker concept to the more appropriate concept of the workers' union. While there are many potential paths this derivation could take (according to the order of application of various operators), one possible approach is discovering that workers have a relationship with the union and that the union acts in an authority role towards the workers. This case is now stored as a case adaptation for future use when other similar problems are encountered.

With the previous knowledge now stored in the system, a second disaster was presented to the system. The Beaver Meadow Elementary school reports an indoor air-quality problem. A candidate response plan suggested that the students be notified directly of evacuation plans and of plans to return to school. However, as before, the user identifies this as an inappropriate filler as elementary students are not able to make their own decisions in these matters. The adaptation component finds the adaptation case describing the workers-union solution and suggests that the same case adaptation be applied to the current situation. The case cannot be used transformationally in this example as students do not have unions. The derivation of the solution can be applied to this example and results in several groups who may have an appropriate relationship to the students. One of these fillers "the parents of the students" is selected by the evaluator as an appropriate solution and the plan is updated.

In some examples neither the case-based methods or rule-based methods produce any useful solutions in the time provided. In these situations the manual adapter allows the human operator to suggest the same types of relationships and operations to explore to focus the search efforts on relevant parts of the system's memory.

3.4 Issues for case adaptation learning

The addition of a case-based adaptation learner to a stand-alone case-based planner increases the complexity of the interactions between knowledge sources. When only the response plan case exists, it provides the sole guidance in determining the type of past case that will be reused. All other components of the system rely on the response plan case. However, when a second independent case base is introduced such as our adaptation case library, multiple knowledge sources are available to guide the creation of new solutions for some problems. The ideal approach would be to use both knowledge sources in a synergistic manner thus achieving maximal benefit. However, to achieve this synergy requires special consideration for how the two case bases interact with one another. Managing this interaction is another central theme of this research. We focus on this relationship in the next chapter. However, several potential problems can arise from the existence of multiple case bases.

- The cost of retrieval can be in the worst case $n + mp$ where n is the size of the response plan case base, m is the size of the adaptation case library and p is the number of identified problems. This assumes a linear search of each case base and that each response plan must examine every adaptation case at least once. The organization of the case bases is pivotal in reducing this retrieval overhead.

- Related to the previous point, maintenance of these case bases plays a key role in both reducing system overhead and finding relevant and useful cases with which to solve the disaster problems. The DIAL system does not directly perform an analysis of the two case bases to judge the coverage of the solution space and prevent the addition of redundant or useless cases. However, DIAL does implement a policy of forgetting where little used cases are removed from the case base when dormant for a length of time. This approach while conceivably removing cases with a high utility in the system when used assumes the savings in overhead from the removal of the case will outweigh the cost of reconstructing the case at a much later time.

- Directly linking adaptation cases to the response plans that initiate their creation is another issue of consideration. By attaching the adaptation cases to

the response plans the information needed to solve certain problems is directly available to the response plan that may need that knowledge. However, this can also be viewed as a limitation in the flexibility that is provided to response plans and adaptation cases. By permitting any set of adaptation cases to interact with any other response plan case provides the greatest maneuverability in the solution space.

Subordinate reasoning processes

Case adaptation learning is not a stand alone reasoning process. It is necessary to consider the affect and role of two subordinate reasoning processes that exist to support case adaptation learning: a rule-based system and a case-based system. Both of these processes are needed for the success of the case adaptation component. The rule-based system depends entirely on the predefined memory search operations. This rule system is unchanging during DIAL processing. The subordinate case base is constructed from learned memory search cases. These cases provide the trace of a successful memory search to a corresponding adaptation case. However, it is beneficial to distinguish between memory search cases and adaptation cases. Adaptation cases store how a certain case adaptation problem was solved. A memory search case holds only the information directly relevant to the search performed. Viewed separately, the memory search cases are then available as an independent knowledge source for the system. Memory search cases may suggest new relationships between concepts in memory, or could be used by the retrieval or storage components to guide the search for knowledge unrelated to case adaptation.

The benefits from these subordinate knowledge sources come without additional overhead as the cost is already encapsulated in its parent process. We will show the use of multiple independent processes to support a single task can be effective at reducing the difficulty of solving many different problems. This benefit will exist because the processes acquire different types of knowledge that can be shared between the them. This gain comes at little additional expense in the form of system overhead.

3.5 Conclusions

Automatic case adaptation is necessary to enable CBR systems to function autonomously and to serve naive as well as expert users. However, knowledge acquisition problems for the rule-based adaptation methods used in many CBR systems have proven a serious impediment to developing CBR applications that perform their own case adaptation.

This chapter described a framework for representing case adaptation knowledge and discussed how that framework is being used as the basis for a model of automatic learning of case adaptation knowledge.

The model combines reasoning from scratch and case-based reasoning to build up expertise at case adaptation. The motivation of this approach is to enable CBR systems to make the transition from case adaptation guided by general rules (which may be unreliable and expensive to apply) to case adaptation guided by adaptation cases that reflect specific case adaptation experience. Thus this method is a way for CBR systems to learn to become more effective at applying their existing cases to new situations.

The next chapter describes how this newly acquired case adaptation knowledge can be exploited to support other aspects of the case base planning process. It provides an algorithm for augmenting other knowledge in this system based on its ever changing knowledge on case adaptation.

4

Similarity Learning

This chapter introduces a method for reusing case adaptation knowledge to assess the similarity between stored response plans and a current problem description. Case adaptation knowledge can augment initial similarity criteria, improving overall system retrieval.

The previous chapter described an internal CBR process for case adaptation. This process built a repository of adaptation cases giving a foundation to solve other similar case adaptation problems. As these adaptation cases are acquired, a new type of knowledge is created in the system that was previously unavailable – experience based case adaptation knowledge. This chapter describes one method of reusing the case adaptation knowledge to refine similarity criteria in a process called *similarity learning*. The chapter begins by defining and discussing the motivations for similarity learning. Next it presents how DIAL reuses its own acquired case adaptation knowledge to improve similarity assessment.

4.1 Adaptability and similarity

One of the first steps in the case-based reasoning process is the retrieval of past similar cases to reapply to the new situation. Better retrieval results in better plans. Accurately assessing similarity is necessary to facilitate this retrieval process. One traditional approach to similarity involves comparing predefined features from the problem description with stored response plan cases and selecting the one with the most matching features. For example, two earthquakes in Los Angeles would be

similar because both involve the same type of disaster, and the location feature is the same for both disasters. For situations where past cases can be applied directly to the new situation, this similarity metric is effective. However, when most stored cases require some level of case adaptation, predefined features for similarity may not indicate the amount of effort required to adapt the cases. To better assess the cost of reapplying these cases a different method of computing similarity is proposed.

If the goal of retrieval is to reduce the total processing required of the system while still producing acceptable plans, the similarity metric should account for the work required to modify a stored case (Smyth & Keane, 1994; Leake, 1992a). After retrieval, the remaining work of a case-based system is to evaluate the proposed plan for problems and to adapt any identified problems. Therefore, if stored cases can be selected to reduce the number of problems or the amount of case adaptation required then overall system processing would be improved. Thus instead of selecting cases on the basis of predefined features a better method would select cases by comparing the adaptability of the different candidate plans. Cases that can be easily adapted are retrieved for reuse over cases that might be expensive. A plan might be adaptable either because there are few identified problems or because the problems identified require little additional effort to repair. By using a similarity metric not based on predefined features, more usable plans will be selected.

4.2 Reevaluating similarity

The addition of a case adaptation learning algorithm to the case-based planning system significantly changes the case landscape that must be navigated when new problems are encountered. Traditional case-based techniques have relied on the use of fixed similarity criteria and indexing to enable quick lookup of appropriate cases for a new problem. With the addition of a second case-base – in the form of adaptation cases – the issue of synchronizing similarity and case adaptation knowledge becomes important.

> Static similarity techniques must be reevaluated when case adaptation knowledge changes.

The four-step view of case-based planning dictates that the plan case is retrieved before the case adaptation component takes control. However, since similarity assessment should reflect the potential adaptability of the retrieved case, the two processes cannot remain independent. Thus, this research examined alternative methods for

assessing similarity. It focused on methods that gauged the adaptability of candidate response plans. Our discussion of similarity assessment methods begins by reviewing other research that provided inspiration for the similarity learning method used by DIAL.

4.3 Related research

Several researchers have described work on modifying similarity criteria to reflect changes in system knowledge and experience. This section examines some of the work that provided the inspiration and insight for the similarity learning approach used in DIAL.

Similarity assessment for case-based reasoning systems often relies on semantic similarity or other criteria that may not reflect the difficulty of case adaptation. Other researchers have suggested that case selection should reflect anticipated usefulness as directly as possible (Kolodner, 1988b; Smyth & Keane, 1998; Leake, 1992a; Keane, 1994).

There are several methods of judging the utility of selected cases. One approach is to refine similarity criteria to reflect the most relevant similarities. For example, explanation-based indexing (Barletta & Mark, 1988) and the Prodigy/Analogy (Veloso & Carbonell, 1994) system's "foot-print" similarity metric focus attention on goal-relevant features, in order to retrieve cases that refer to the prior problem situations with the most relevant similarities. Other approaches do failure-driven learning to refine similarity criteria after detecting retrieval of a case that is needlessly difficult to adapt (Birnbaum, Collins, Brand, Freed, Krulwich, & Pryor, 1991; Fox & Leake, 1995). All these approaches are worthwhile, but do not directly take adaptability into account.

Adaptability is the basis for similarity assessment in Smyth and Keane's (1996) Déjà Vu, a case-based design system which computes similarity as follows. First, it selects a set of case adaptation procedures to fit a stored plan to new needs. Next, it uses a fixed estimated cost figure for each procedure to estimate the cost of the entire case adaptation. Finally, it favors the plan with the lowest estimated case adaptation cost. This process requires all adaptation knowledge to be pre-specified and does not require any user intervention to detect problems. Experiments demonstrate that this improves performance compared to using traditional semantic similarity.

Déjà Vu assumes that an initial filtering stage retrieves a subset of the case library for fine-grained consideration. In Déjà Vu, this step checks all stored cases to select

those for which the system has case adaptation rules that potentially can adapt the features of the retrieved case to the new situation.

4.4 Learning new similarity criteria

This prior work strongly suggests that in order to facilitate later case adaptation, similarity criteria should reflect adaptability. Thus when new case adaptations are learned, similarity criteria should be modified to reflect changed case adaptation abilities. However, early versions of DIAL relied on static similarity criteria. Consequently, DIAL could select plan cases that are difficult or impossible to adapt while other plan cases exist with manageable adaptation requirements.

One straightforward method of assessing plan adaptability is to record system cost averages for typical problem types. However, knowing the cost of different problem types is only effective if these cost are known and are unchanging. In DIAL, the system's ability to perform different case adaptations is constantly changing as new case adaptation knowledge is acquired. Further, as the system has little initial case adaptation knowledge, it is impossible to accurately assess the costs of different types of problems prior to system execution. Instead a method achieving the same effect as the predefined costs for case adaptation is desired but also accounting for the system's changing case adaptation abilities. The next section focuses on the method used in DIAL to dynamically modify the similarity criteria to reflect plan adaptability and therefore account for both types of case knowledge found in the system.

4.5 The similarity learning algorithm

The linking of similarity assessment to the stored case adaptation knowledge was achieved by developing an adaptation algorithm involving the following sequence of steps.

1. First, a small set of potentially relevant cases from similar prior disasters is retrieved. This process uses coarse-grained semantic similarity criteria (based on the distances in memory between role-fillers in the problem descriptions) to retrieve a user-defined number of prior cases. This step is necessary to limit the amount of processing dedicated to retrieving adaptation cases. This filtering selects only those response plan cases that are relevant to the current problem description.

2. The system determines correspondences between old and new disasters and does an initial mapping of the prior response plans to the new situation, identifying roles for which this initial simple case adaptation fails.
3. Additional problems are identified by a combination of automatic stereotype-based problem detection (Leake, 1992b) and user feedback (possibly rejecting system mappings).
4. The similarity assessment process takes prior response plan cases and their problems as input, retrieves available information on the expected cost of adapting each problem, and estimates the total expected cost by summing the costs over the entire plan. This information is used to select the response plan case expected to be easiest to adapt, i.e. the case with the least total expected cost.
5. Problems in the response plan suggested by the selected case are repaired by the case adaptation component.

These steps are now examined in greater detail in the following subsections.

Filtering and selecting cases

The primary issue in the deployment of similarity learning is the mechanism used to filter both response plan and adaptation cases and select those most appropriate to the current situation. If the filtering does not occur, the system would need to score every system case for adaptability. Most of these stored cases would prove useless when examined and waste valuable processing time. Since one of the primary goals of using case adaptation knowledge to support case adaptation is speedup learning, any savings made in the case adaptation component would be lost in the retrieval component if filtering did not occur. Therefore the filtering must find a balance between eliminating potentially useful plans and the expenditure of unnecessary resources by the retriever. Fortunately, the case base is organized such that filtering can be done quickly and without significant danger from over or under filtration. This organization described in the example below was developed specifically to alleviate this problem.

To illustrate, a new example from the DIAL system is given. When presented with a new story about a flood disaster in West Virginia, DIAL retrieves a pool of response plan cases using static feature-based similarity measures. In practice, DIAL examines a set of predetermined fillers from the story and compares these fillers to values in the set candidate cases. The organization of the case base is divided along these same predetermined fillers giving the system the ability to select all cases with the same value in one retrieval of the case base. For example, the most commonly used fillers are the problem type, the location of the disaster and the severity of the

disaster. In the West Virginia example, the problem type is a flood of catastrophic severity that takes place in Wheeling, West Virginia in the United States. The case base is divided into separate sub-trees based on problem type and these trees are further subdivided by severity and then by geographical region. If after this retrieval of cases the number of selected candidate cases is large then the number is further reduced by applying semantic similarity techniques to each and selecting only the top five cases for further consideration. For the West Virginia problem, only three candidate response plans were discovered: one in Alaska, one in Georgia, and one in Oregon. After an initial mapping of the new situation on to the prior situation, these candidate cases are passed on to the next step of the similarity assessment sequence.

Problem detection

The second step of the similarity assessment sequence is the identification of problems that require case adaptation in each of the candidate plans. This is one potential bottleneck of the similarity assessment process. Problem identification is a critical step in the planning system that this dissertation describes, but is not the central focus of this research, and other researchers have provided adequate methods for handling evaluation.

Several researchers have described methods of identifying problems requiring case adaptation. These methods range from explanations of problems (Hammond, 1989) to pattern-based verification (Leake, 1992b). External information may also directly identify needed case adaptations. For example, a failure during plan execution could signal that additional case adaptation is necessary (Alterman, 1988).

DIAL relies on a combination of pattern-based methods—which can detect potential problems at minimal cost (Leake, 1992b)—and user feedback for information not available in the system's prior knowledge. In the disaster response domain, user input might reflect situation-specific information (e.g., that a road is impassable, or that a region is not under the jurisdiction of a particular agency). The DIAL model assumes that the planner's knowledge is incomplete, so the system may not be able to detect all problems internally.

For each candidate response plan, DIAL identifies problems detectable from its knowledge, and the user identifies and describes additional problems. For example, in the aftermath of the Oregon flood, the police forces maintained order. This response would not work for the new disaster in West Virginia, because the police force is not equipped to deal with the access problems posed by the flood. This results in a problem categorized as "lack of access" to the area. Re-applying the Oregon plan to

the new situation will require case adaptation to overcome this problem. DIAL also identifies additional problems in each of the candidate cases.

Scoring adaptability

Once each case is evaluated and the problems requiring case adaptation are identified, the plans are ranked based on their adaptability. Several different approaches of assessing adaptability were developed in this research and comparatively evaluated. These approaches are discussed in some depth in the next section. Once adaptability is scored, a cost value is determined based on the system's estimate of the amount of processing required to solve each of the identified problems. For each candidate plan, the costs for each problem are summed to arrive at a final score for the plan. This score reflects an estimate of the total amount of processing needed to solve all identified problems. Higher scores reflect larger costs and therefore less desirable plans. These costs are not absolute but are system predictions of relative cost. The accuracy of the prediction is dependent on the validity and utility of any stored case adaptation knowledge.

In the Oregon candidate plan this process would entail retrieving stored knowledge about "lack of access" problems in the context of a flood. Stored knowledge might suggest that this type of problem has a very high cost to solve, and therefore the candidate plan should be less desirable to reapply. When new system knowledge is acquired, this same type of problem may become easier to solve and in future similar situations this candidate plan might be highly desirable for reapplication. In other situations, there may be no available knowledge on this problem type and the system would have to rely on other methods to assess the difficulty of the problem.

4.6 Case adaptation knowledge and similarity learning

Several different approaches to similarity assessment using adaptation cases were implemented and compared in DIAL. Some metrics were selected as analogues to case adaptation based similarity metrics already suggested by researchers such as Smyth&Keane. Other methods were selected to exploit unique opportunities presented by the organization of knowledge in the adaptation cases. These methods are summarized in table 4.1.

Two of the approaches to similarity assessment are modifications of approaches

described by other researchers. These two methods do not directly apply any knowledge stored by adaptation cases and provide a useful base line criteria for assessing the performance of other methods. The first approach, the *problem count* method, assumes that all case adaptations have equal cost and therefore a plan with the fewest problems is the most adaptable. This method counts the number of problems in a candidate plan and thus requires no additional case adaptation knowledge. All case adaptations do not have the same base cost so the expectation was this approach would select cases with few problems but these cases may require significant processing time to solve. Nonetheless, the overhead of this method is negligible and could compensate for these expensive case adaptations.

The second benchmark method requires the system maintain *average costs* for solving case adaptations of each problem type. These average costs are stored by the system and updated whenever new case adaptations are performed. So unlike systems such as Déjà Vu, these averages are dynamic and not provided to the system at startup. This method should provide a fair indication of the general difficulty of different classes of problems. However, it does not accurately reflect the current knowledge that is stored in the adaptation cases. Early problems of a specific type could require substantial processing time to solve, but once adaptation cases are stored for these problem they may become straightforward and require little processing. With this method, the influence of those early time consuming problems will linger long after the addition of the new knowledge. Thus while neither benchmark approach was expected to improve system performance, they provide a starting point to compare other methods.

Three different methods were developed to apply the knowledge stored in adaptation cases to the process of similarity assessment. Each of these methods begins in the same manner by retrieving the most applicable adaptation cases for the identified problems. Adaptation cases are retrieved in two steps. The first step filters all the adaptation cases by problem type. So only adaptation cases describing "lack of access" problems are selected as candidates for the same type of problem in the response plan. Each selected adaptation cases is given a score based on how closely it matches the features of the current problem. The best matching case is then attached to this problem for reuse in the case adaptation component.

The differences in the three similarity methods using case adaptation knowledge stored in the cases is the type of knowledge used to score similarity. The three new methods are listed here followed by descriptions of each.

- *Prior adaptation cost*
- *Solution derivation length*

- *Reapplication costs and relevance*

The first method uses the prior cost of each case adaptation episode as a prediction of the difficulty of the new case adaptation problem. The prior cost is taken as the total number of memory search operations that were applied during the prior solution episode. These costs can then be added together to arrive at a numerical value that offers a prediction of the difficulty of the new case adaptations. The candidate plan with the least expected case adaptation cost is selected to form the foundation of the new solution. This approach avoids case adaptation problems that previously had a high cost associated with them. Using the prior case adaptation costs is similar to applying the average cost method but it uses more relevant information stored in an adaptation case that is similar to the current problem. However, the method is likely to overestimate the total cost required to adapt a response plan as some problems that were previously expensive will be less expensive when reapplying the adaptation case. Nonetheless, it does provide a good indicator for the cost of adapting a candidate plan with only weak search methods.

The second approach that applies stored knowledge in adaptation cases uses the length of solution derivations as a method of scoring the adaptability of a problem. A solution derivation from an adaptation case is the exact set of operations required to perform the prior stored case adaptation. If each selected adaptation case provided the exact solution to the new problem, then these operations could be reapplied directly and would solve the new problem. Thus a lower bound for the case adaptation cost for each candidate response plan can be determined. Again this is only an estimate of cost as adaptation cases generally provide inexact solutions to new problems and occasionally suggest no solution at all.

One problem with using solution derivations as a measure of cost is that the case adaptation selected may be an imperfect match to the new situation. This could result is additional processing time as the derivations could require adjustment or further case adaptation. To account for the selection of imperfect adaptation cases, a final similarity assessment method was developed called RCR (for Re-application costs and relevance) (Leake, Kinley, & Wilson, 1996). RCR estimates the cost of performing case adaptations by assessing both the prior derivation length and scoring how applicable the selected adaptation case is to the current situation. For each adaptation case that is selected for a problem in the candidate response plan, its similarity to the current situation is scored used coarse-grained feature similarity. This value is adjusted such that two identical situations would score a one and less similar situations would be given progressively larger scores. This score is multiplied by the length of the solution derivation from the adaptation case. This gives DIAL the ability to estimate what the future cost of correcting this problem might be and account for the relevance of the adaptation case to the current problem.

Methods	Description	Knowledge Source
Problem count	Counts the number of problems in each candidate response plan	None
Average problem cost	System keeps running tallies on the average cost of case adaptation for each problem type.	Stored system values
Prior case adaptation cost	Uses the previous cost of completing the similar case adaptation	Adaptation cases
Derivation cost	Counts the number of steps that were actually used to solve the similar case adaptation	Trace of solution from adaptation case
RCR	Used the derivation cost by scaled by a factor of how dissimilar the prior adaptation case is to the current problem	Adaptation cases

Figure 4.1: Summary of adaptability based similarity methods.

By making the refinement of similarity criteria a natural side-effect of case adaptation learning, better pairings between response plan cases and adaptation cases should be achieved. However, the final three methods have two potential drawbacks: either generating inaccurate similarity judgments (if the costs of the previous case adaptations retrieved turn out to be poor predictors), or imposing excessive computational overhead in retrieval and evaluation of cases. Both of these properties must be tested.

The benefits of a dynamic similarity process are numerous and include:

- There are few initial rules that have to be provided to the similarity process.
- The adaptation cases drive the types of response plans that are retrieved – some plans may not be selected because the requisite case adaptation knowledge does not exist.
- As new case adaptation knowledge is acquired, changes can be made to the similarity criteria. Problem situations that were previously difficult or impossible to adapt may now be solvable or even trivial.

By incorporating these methods, DIAL transforms from a rule based similarity

process to a primarily case-based similarity process. However, to understand the effectiveness of these new learned similarity methods three questions were posed:

1. Does the linkage between similarity and case adaptation knowledge created by similarity learning decrease the total system effort when case learning and case adaptation learning are used together?
2. How is the *overall* planning efficiency of DIAL affected by similarity learning and case adaptation learning and what is the comparative performance of the two?
3. How should the cost of similarity assessment be calculated? In this process, is there a clear division between the role of the retrieval/similarity system and the case adaptation system?

There are several potential drawbacks to implementing these similarity learning approaches. The next section provides a discussion of the most serious of these pitfalls.

Issues for similarity learning

The similarity learning algorithm is a promising approach to link the retrieval of response plans with the selection of relevant adaptation cases. However, some potential drawbacks can be identified. DIAL's filtering stage currently uses semantic similarity to retrieve a small set of candidate response plans for consideration. There is the potential of incurring a high cost when retrieving adaptation cases for each candidate response plan. For example, if ten candidate cases are selected and each has three problems requiring case adaptation then 30 different adaptation cases might need consideration. Further, the desire to retrieve even a single adaptation case could entail examination of several different cases to find the most applicable one. To compensate only five candidate cases are allowed in order to reduce this burden. The tradeoff of filtering out useful cases to achieve less overhead in adaptation case retrieval is a gamble. If none of the final set of cases is considered adaptable then little is gained by using the adaptation case knowledge. However, without this filtering the overhead could become prohibitively expensive and eliminate any system gains. As long as at least one comparatively adaptable case is in the final set of candidates, this tradeoff appears worthwhile. Filtering by semantic similarity seems reasonable, in light of evidence in prior CBR systems that semantic similarity can provide general estimates of similarity.

A second potential problem results from the selection of adaptation cases relying on static semantic similarity criteria – the same type of similarity DIAL attempts to

eliminate at the response plan level. Semantic similarity was replaced for response plans since the plans would later require case adaptation and thus this similarity should be based on the processes that will be applied to the plans. Unlike the response plans, selected adaptation cases are applied directly to the problems for which they were retrieved and do not require further processing before they can be used. However, it semantic similarity fails to find useful adaptation cases to reapply then extra processing cost will occur. We believe that similarity learning will improve the interaction of the case based planner and case adaptation component. Many of our concerns will be alleviated by the presentation of the test results.

These problems will be addressed by the system test results.

4.7 Conclusions

This chapter addressed the issue of how traditional similarity assessment methods can be modified when case adaptation knowledge is available. The approaches presented here couple similarity judgments directly to a case library containing the system's case adaptation knowledge. This case-based approach simplifies knowledge acquisition for similarity criteria, because similarity is learned from experiences with case adaptations rather than based on *a priori* analysis. Similarity learning provides fine-grained estimates of case adaptation costs, reflecting knowledge of individual prior problems, and provides a natural way to refine similarity criteria as new case adaptations are learned.

One open question is what is the appropriate information to reuse from adaptation cases to score the adaptability of response plans. This chapter presented several different methods for approaching this problem including using past costs, expected future costs and a method combining the relevance of a selected adaptation case to the amount of work required by the case adaptation.

Similarity criteria for selecting cases must change as case adaptation knowledge is learned; neither coverage of the case library, nor case adaptation abilities, can be judged in isolation from the other knowledge sources. In developing a combined method, how one type of learning affects the efficiency of one component of the CBR process is secondary to the efficiency effects of the learning on the CBR process as a whole. The close coupling of multiple processes and knowledge sources in CBR complicates the application of learning to each one, but also provides a new motivation for combined learning: Combined learning can enable a CBR system to better exploit the relationships between multiple types of knowledge.

The next chapter presents an empirical evaluation of the methods presented in

this and the prior chapter examining the hypotheses and questions that have been identified to this point.

5

Experiments

This chapter presents empirical evidence on the effect of case adaptation learning and similarity learning.

This chapter presents empirical evidence supporting the effectiveness of our case adaptation learning method. Using this evidence, questions can be answered about: the improvement of the case adaptation process, the improvement of the entire system process, and the role of similarity learning on the case adaptation process. The system is examined under various learning conditions in order to draw conclusions about the advantages and disadvantages of our approach.

The chapter is organized as follows: it begins by stating the hypotheses to test. Next, the experimental methodology and terminology for the experiments is summarized and explained. Finally, it presents the empirical results. These compare case adaptation learning and plan case learning, and the benefits of different similarity assessment methods.

5.1 Summary of system hypotheses

We have proposed the following hypotheses that will be addressed by the experiments:

1. A combination of plan case learning and case adaptation learning will result in acceptable plans generated in less time than with case learning alone.
2. As case adaptation knowledge is acquired, the cost of performing individual case adaptations will decrease.

3. Case adaptation knowledge will result in a greater coverage of the solution space, so fewer problems will be unsolvable.

4. The addition of a similarity learning improves the retrieval of both adaptation cases and plan cases. This process enhances the interaction between the two case based processes.

5. The initial cost of learning will increase with multiple learning processes, however, the ultimate benefits in terms of speed and quality of solution will outweigh this expense.

5.2 Methodology

The DIAL system was evaluated with a set of ablation studies to measure the contributions of the different learning components and knowledge sources. The system was tested using two different knowledge bases:

- **Hand-Coded Knowledge Base:** A knowledge base was built to support early testing of the DIAL system. The knowledge base contained the information needed to solve any problem encountered in the set of disaster examples presented. In addition to the necessary knowledge, a large number of irrelevant but related concepts were added to make standard search more difficult. However, it was limited to approximately 2000 different concepts and subject to the designer's bias.

- **Wordnet lexicon:** The Wordnet lexicon contains over 100,000 different English language concepts and is organized in a hierarchical model with linguistic relationships existing as the connections between concepts. This lexicon is sufficiently large that exhaustive search is not practical. Also, as it was built by research groups unassociated with the DIAL project, it does not exhibit any inherent bias for the types of problems DIAL attempts to solve.

A series of test examples was constructed from real world disasters as reported by the AP newswire and presented to the system for plan development. There were 22 disaster examples presented to the system and are listed in table 5.1. An expanded listing is included in appendix A.

During a single pass of the disaster examples the following steps occurred in our experiments.

Disaster Type	Disaster Location
Earthquakes	
	Los Angeles
	Liwa, Indonesia
	Ecuador
	Biak, Indonesia
	Neftegorsk, Russia
	Jiashi, China
	Yunhuan, China
	Rustaq, Afghanistan
	Bogota, Colombia
Floods	
	Wheeling, WV
	Bainbridge, GA
	Oregon
	Allakaket, AK
	Kabul, Afghanistan
	Izmir, Turkey
	Johannesburg, South Africa
	India
	Khartoum, Sudan
	Grand Forks, ND
	California
	Guizhou, China
Air Quality	
	Beaver Meadow School indoor air quality
	A&D Manufacturing indoor air quality
	Brookview Elementary indoor air quality

Table 5.1: Disasters processed by DIAL during experiments

1. The system learning parameters are set (as described below) to assign the desired system.

2. Next, each example disaster is fed to the system and processed until a satisfactory response plan is generated. This may involve several episodes of case adaptation and require user interaction.

3. Finally, statistical data on the problem solving costs and the results are gathered and stored for later analysis.

System parameters

The system has several independent learning parameters. These can be set to achieve a desired learning combination. The parameters available to the system are:

- ***case-learning* (boolean)** – This parameter enables learning in the system. If set to false, newly generated response plans are not stored in the case base, otherwise all generated response plans are stored and available for future reuse.
- ***adaptation-learning* (boolean)** – This parameter toggles the internal case adaptation learning process. If set to false, successful case adaptations are not stored for reuse, otherwise all generated adaptation cases are stored and available for future reuse.
- ***similarity-learning* (integer)** – This parameter determines the type of similarity assessment performed by the system. If false, the system relies on static similarity criteria. Otherwise the system attempts to apply the knowledge stored in adaptation cases to support similarity assessment. When no adaptation cases are available in the system, this parameter is ignored.
- ***similarity-type* (integer)** – This parameter allowed the user to specify which of the five different similarity methods would be used when similarity learning was activated

System measures of cost

Several different measures of cost were used to compare the performance of the system in the experiments. The first types of units were internal measures based on the architecture and construction of the testbed system. The number of concepts visited in memory was a type of internal unit. The second type of units was external

measures scored independently of the system. Processing time was an example of an external measure. Internal units were a preferred method of evaluating experiments as all overhead cost not specific to the system processing would be eliminated. For example, access times for the Wordnet database varied according to the speed and memory of the machine used whereas the number of accesses to the knowledge base remained constant. The following units of measure are used and recorded for each experiment.

- **Operations:** An operation in the system is the number of primitive search methods required to solve a case adaptation. These search methods correspond to the smallest movements permissible in the knowledge base. Moving from a concept in the system memory to a parent concept is one example of an operation.
- **Nodes:** Nodes are the number of unique locations visited in the knowledge base while searching for the solution. Each concept represented in the knowledge base corresponds to a single node. Nodes visited more than once in a search are only counted a single time. A single operation may lead to a single node or could lead to a hundred nodes. For example, the concept "tree" in the Wordnet knowledge base has over 150 different child nodes. By applying a single get-child operation to this concept, 150 new nodes are introduced. However, the concept of "oak tree" only has a single parent in the knowledge base and therefore one operation would yield only one concept. Thus it is possible for the number of nodes and operations to vary widely across a trial.
- **Time:** Time is the amount of actual CPU time spent searching for the solution to a case adaptation problem. Most of the results here are taken from a Sun Sparc Ultra 5 workstation.

Outline of experiments

For the set of experimental trials presented, up to 26 disaster response plans were created with each requiring between 0 and 6 case adaptations with a maximum number of 119 total adaptations performed in one of the trials. Small variations occurred based on the type of learning performed. One combination of learning parameters might require very few case adaptation episodes while other combinations (e.g., including case adaptation) learning might require several inexpensive case adaptations. The set of 26 response plans was selected for its manageability and because longer sequences of examples generally provided no additional learning improvements, although a complete scale up of the system is left as a future direction of work.

Our description of the system results follows the following organization:

- The hypotheses generated for the experiment are stated.
- The system results are presented.
- An explanation and analysis of the results provides perspective of the results and highlights anomalous data.

Additional analysis of the results and any conclusions drawn in terms of the system as a whole will be left for a final discussion of all of the data.

5.3 Value of case adaptation learning

As part of the initial assessment of the DIAL system, the direct contribution of case adaptation learning to the case-based reasoning process was examined. The first experiment was viewed along two distinct cross sections to understand the affect of case adaptation at both the plan level and the case adaptation level. The first analysis described examines the cost of individual case adaptations across various learning conditions. Based on these results, a second analysis of the same data is described which partitions the data in terms of the cost of response plan creation. From these results some conclusions can be drawn to support the use of case adaptation learning in certain types of systems.

For this set of experimental trials the system was examined under five different learning conditions.

1. **No Learning (NL)**: A baseline for the system where no learning occurred.
2. **Case Learning (CL)**: This is traditional CBR with response plan cases stored and reused but no adaptation cases are saved.
3. **Case adaptation Learning (AL)**: Only adaptations cases are stored and reused, completed response plans are discarded.
4. **Case adaptation Learning and Case Learning (AL+CL)**: Both response plan cases and adaptation cases are saved for reuse.
5. **AL + CL + RCR**: This is identical to the prior condition except the RCR method for similarity learning is added as described in chapter four.

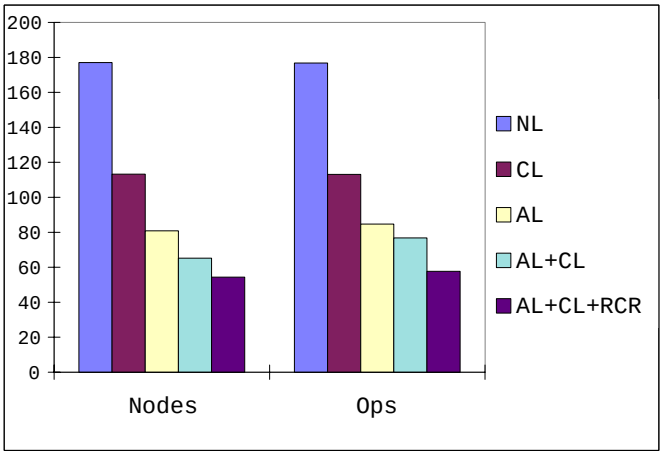

Figure 5.1: Average Cost of a single case adaptation on original KB

Case adaptation learning vs. plan case learning

The first experiment examined the direct contribution of learning to the cost of solving problems requiring case adaptation. The experiment was conducted using both knowledge bases. The results from the hand-coded knowledge base are replicated here from (Leake, Kinley, & Wilson, 1997) and are augmented here with results from Wordnet. This analysis should either support the use of case adaptation learning under different knowledge bases or provide insight as to the characteristics of a knowledge base that benefit case adaptation learning.

Hypothesis. Case adaptation learning alone was expected to result in a slightly higher processing cost than case learning alone. Without case learning, there are few starting points from which case adaptation learning can arrive at solutions. Without adequate case coverage of the space of possible case adaptation solutions, some problems will require much longer searches to complete. Additionally, the smaller coverage of the potential solution space could result in a much steeper learning curve for the case adaptation learner. Thus more effort would be expended early by the case adaptation learner and only over a time would this cost decrease. The combination of learning methods was expected to produce the best results as deficiencies in one learning method would be compensated by the other method. As the number of plan cases and case adaptation cases increased, the likelihood of identifying a solution increases due to a wider coverage of possible problems.

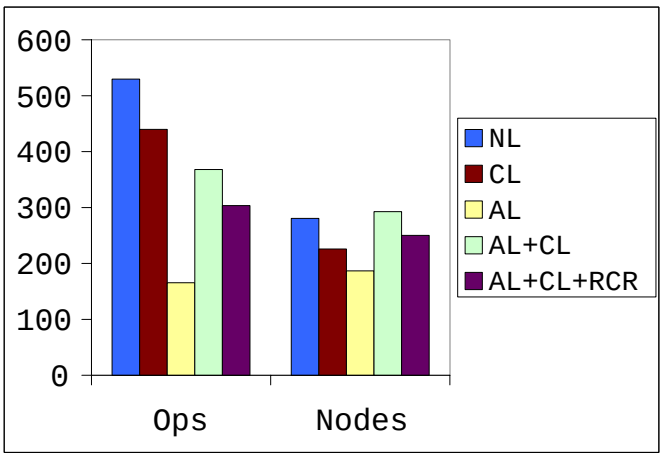

Figure 5.2: Cost of case adaptation with Wordnet

Results. Results from this trial are shown in figures 5.1 and 5.2. These results report the number of operations performed per case adaptation and the number of nodes visited per case adaptation over the course of the entire experiment. Fewer operations and nodes on average implies a better search. In figure 5.1, case learning performs better than only the no learning condition. As additional learning methods are combined, their interaction produces the best overall results for case adaptation. One surprising result from this data shows case adaptation learning without plan case learning to be superior to traditional plan learning alone.

Results from the same set of examples using the Wordnet knowledge base, shown in figure 5.2, provided several surprises. Case adaptation learning alone has a lower cost than all other methods, both in terms of the number of operations performed and the number of nodes visited. Also, the combination of case adaptation learning and plan case learning had a cost for the nodes visited measure comparable to the no learning case. It is interesting from this data that when counting the number of nodes visited, the combination of learning methods is much worse than either individual method. Also, the number of operations performed using only the case adaptation learning method seems out of place as it was almost twice as fast as the next best combination. These anomalies are described more in the following section. For a quantitative view of these same results, table 5.2 is provided.

	Avg Ops	Avg Nodes
No learning	529	280
Case Learning	439	225
Case adaptation Learning	165	186
AL/CL	367	292
AL/CL/RCR	303	250

Table 5.2: Average effort expended on case adaptation for the five learning situations.

Discussion. Despite some surprises, these results are encouraging. Case adaptation learning does reduced the amount of effort required to perform case adaptation on average for at least the examples given.. This result was consistent across both knowledge bases demonstrating the effectiveness of this approach is independent of the organization or content of its knowledge. However, the ineffectiveness of combined learning methods when using the Wordnet knowledge base particularly given the success of the adaptation learning method raised new questions not previously asked.

First, we expected that the combination of learning methods would improve the overall efficiency of the case adaptation process. With more plan cases available for retrieval and more adaptation cases available to support the case adaptation process, efficiency was expected to at least equal the other two methods. Instead, the two separate learning processes interfered with one another.

Case adaptation learning performed well above expectations. This is surprising as it was not designed as a stand alone learning method but as an integrated part of a larger CBR process. However, reducing the cost of case adaptation is only beneficial if the number of case adaptation problems that need to be solved does not increase. Thus the savings in cost of case adaptation alone is not enough to suggest overall improvement to the system.

To summarize, we can draw several conclusions from this data. Each conclusion is stated followed by a detailed discussion of the conclusion.

- Case adaptation learning reduces the system cost of performing case adaptations.
- Learning response plan cases does not reduce the cost of performing case adaptations. When case adaptation learning is not present, case adaptation took longer. However, when plan learning was present the number of case adaptation performed was reduced.

- The combination of multiple learning methods without similarity learning performs worse on average then any of the individual learning methods alone. This was an unexpected result.

The first conclusion is drawn straight from the presented data. Case adaptation learning does appear to reducing the effort required for case adaptation. Taken alone, this result is precisely what our hypothesis stated. However, these data also suggest that using case adaptation learning alone was significantly better than using case learning alone. This research does not attempt to make that claim. Instead, this data provide one perspective on the case adaptation process. When case learning is used, more plan cases exist to select from when new problems are encountered. Thus, retrieved plan cases are likely to be better matches to the problems can be found resulting in fewer total case adaptations. The few required case adaptations in this situation tend to require expensive search processes to solve. This ends up inflating the average case adaptation cost for each of these problem types. With case adaptation learning, adaptation cases are frequently retrieved to solve problems in the retrieved plan cases. Many of these case adaptations are simple and require little time to solve. A few case adaptations may be of the same difficulty as those handled in the case learning situation alone. However, the total number of case adaptations performed is high and thus reduces the system wide average for case adaptation.

A different, and perhaps more effective, perspective for analyzing the data is to examine case adaptation costs at the response plan level. The creation of a new response plan incorporates all aspects of the external CBR process and thus better reflects the total costs to the system. The next section will examine this same data from this new perspective to identify how case adaptation learning affects system processing as a whole.

Our final conclusion is that by combining the learning methods there actually appears to be a decrease in performance compared to each method alone. The combination of learning components does not guarantee better performance than using any single learning method. At least two factors contribute to this unexpected lack of synergy exhibited by DIAL in these results. These are system overhead and *learning interference*. Learning interference describes a situation where multiple learning methods are applied and tend to handicap each other. For example, both case learning and case adaptation learning proceed independently of one another with each adding new cases to their case bases when new problems of the appropriate types are solved. However, when plan cases and adaptation cases are selected, the adaptation cases are used to repair the problems identified in the plan case. If the plan case retriever selects plans for reuse that do not utilize available case adaptation knowledge, then case adaptation is performed at a sub par level. Ideally, both the case

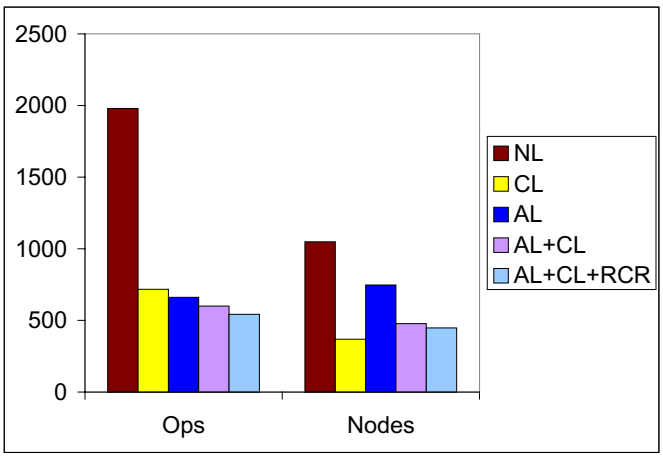

Figure 5.3: Average cost of case adaptation per response plan

learner and the case adaptation learner would store exactly the type of information most usable by the other method. The addition of similarity learning may assist the learning methods to better coordinate their actions in order to avoid this interference.

The effect of case adaptation on the system

A different perspective on the data is provided by examining at the level of the response plan as opposed to the level of the case adaptation. The response plan is a standard unit of processing to the DIAL system. The creation of a response plan encompasses all of the actions and processing the system can do. By viewing adaptation costs at the response plan level a more accurate view of the total system costs during processing may be achieved.

Hypothesis. Adaptation learning will reduce the cost of plan generation when costs are measured at a response plan level.

Results. The results, presented in figure 5.3, reflect the total cost of case adaptation for each response plan while using the Wordnet knowledge base. All of the learning combinations were superior to the no learning condition. However, the scale of this graph does not clearly distinguish between each of the learning methods. To

	Avg Ops	Avg Nodes
No learning	1979	1048
Case Learning	717	368
Case adaptation Learning	661	747
AL/CL	600	477
AL/CL/RCR	542	447

Table 5.3: Average effort expended on case adaptation for the the five learning situations per response plan.

provide for this information, table 5.3 reports the exact performance of each method. The table shows a small decrease in the number of operations required for each response plan with each added learning method. There is still a small difference in the number of operations performed by case adaptation learning and standard case learning but we make no claims of its significance. However, the number of nodes visited during case adaptations is substantially smaller for case-learning alone that for any other combination tested. This unexpected result will be analyzed further in the following discussion section.

Discussion. With the exception of the result for case-learning, the results support our claim that case adaptation learning contributes efficiency to the case-based reasoning process. However, the unexpected result requires additional discussion. There is a difference in the information gained from the reporting of the number of nodes visited and the number of operations performed. Nodes indicate the total number of concepts from memory that were placed into consideration for additional processing. If one area of the knowledge base is particularly dense with concepts, the number of nodes can quickly increase. In practice, this number is reduced by pruning repeated nodes and applying constraints to eliminate nodes from consideration. In contrast, the number of operations performed shows the number of decisions that were made with respect to the search process. Application of a single operation could result in the addition of multiple nodes to the pool of nodes to be considered. Further with case adaptation learning, each node that is encountered must be examined to ensure that the directed search follows the most appropriate node, whereas blind search can select nodes arbitrarily for expansion. This has the net affect of inflating the nodes visited number for case adaptation learning.

Case adaptation knowledge reduces the amount of effort required to solve certain types of case adaptation problems.

The effect of case adaptation learning on problem coverage

The results presented in the prior sections focused on case adaptation problems that were solved automatically by system methods. However, certain types of problems may be unsolvable if the coverage of the solution space by the stored cases is insufficient. These problems must eventually be solved by manual case adaptation often after substantial system resources have been expended attempting to perform the case adaptation. Thus the frequency with which the manual adapter must be used is a measure of the case adaptation ability of the system. In this analysis, the number of unsolved problems is examined – these are the problems that were attempted using automated system methods but required manual intervention to complete.

Hypothesis. Each learning method should require approximately the same number of manual adaptations.

As each trial begins with identical system knowledge and is presented with the same set of disasters for each learning situation, the set of reachable solutions should not vary widely in the system. One learning method might have the capability of solving a given problem faster but if a problem is unreachable with the given knowledge it will require manual case adaptation regardless of the learning method used. A few of the expected differences include case adaptation learning methods that would be able to reapply the knowledge from a prior manual adaptation. Thus an unreachable problem once solved will no longer be categorized as unreachable under those conditions. Therefore, case adaptation learning may record slightly fewer unsolvable problems. Accordingly, the no learning condition might require more manual case adaptations since new knowledge is never acquired and difficult problems remain difficult.

Results. The graphs for the two knowledge bases are presented in figure 5.4 and figure 5.5. As expected, the learning methods do better than the no learning condition in the coverage of the problem solution space. However, the two graphs differ between the case learning results and the case adaptation learning results. In the latter case, case adaptation learning actually requires a larger number of manual case adaptations than case learning. As expected, the combined learning methods require the least number of user interventions to solve difficult problems.

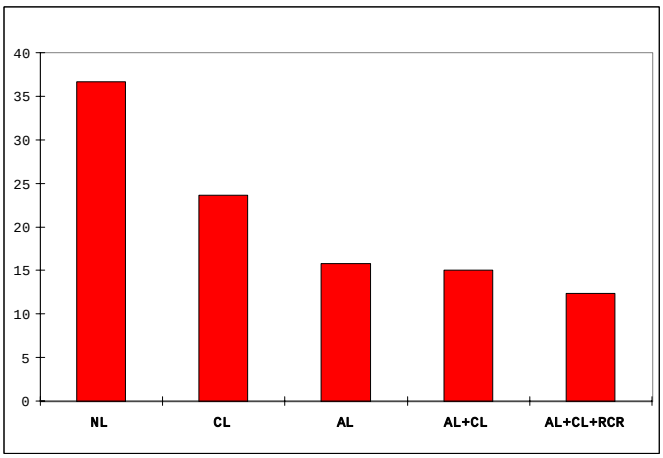

Figure 5.4: Number of unsolvable case adaptations using hand-coded knowledge base

Discussion. These results did not match the initial expectations described earlier. Case adaptation learning should benefit the most from the manual adapter and thus require fewer manual case adaptations over the course of the entire experiment. However, when using the Wordnet knowledge base, the opposite occurred. Two factors account for this disparity: the inability of the system to generalize from manual case adaptations, and the much larger size of the Wordnet knowledge base.

When case adaptation learning alone is used there is only a small pool of plan cases from which to adapt. Earlier, it was established that more case adaptations would be required when there are fewer plan cases because there is less diversity from which to create new plans. The appearance of new problems for which no applicable adaptation cases exist and with solutions that are unreachable using standard search techniques is more likely. Thus it is not surprising that more manual case adaptations may be required. However, compared to the total number of case adaptations that are needed when using case adaptation learning, the raw number of manual case adaptations is still small. Further, the case adaptation component has a strong preference for attempting to solve problems even if the adaptation cases selected are inappropriate for the given problem. This results in more time being spent by the adapter attempting to find a solution using adaptation cases that will not result in a satisfactory outcome. While this occurred infrequently, it does provide the first suggestion that case adaptation learning alone can lead to some potential difficulties when limited to a small set of plan cases from which to begin.

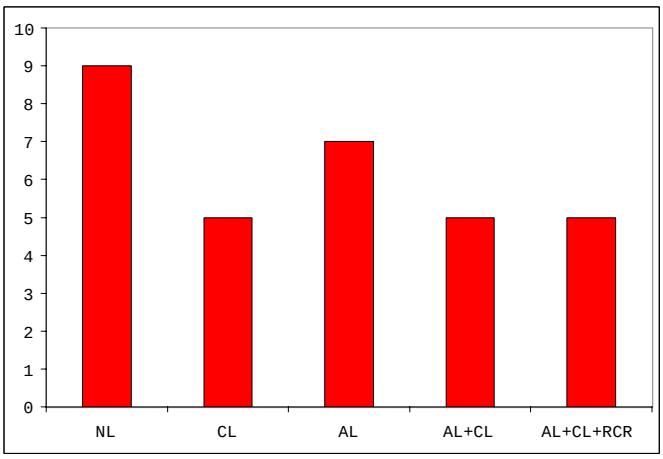

Figure 5.5: Number of unsolvable problems using Wordnet

This discussion prompts a question of whether unsolvable case adaptation problems can be avoided under any of the learning methods. While this question cannot be answered from this data, insights can be gained on the relative contribution of each learning method towards the avoidance of these difficult problems. It appears that plan case learning is a necessary component in the reduction of unsolvable problems that are encountered. This is likely due to all solutions to case adaptation problems initiating with values from a selected plan case. As the number of stored plan cases increases, a larger range of starting points exist on which to begin case adaptation. Without this variety the reach of adaptation cases is restricted. Even in situations where case adaptation learning is not used, the more plan case that exist allow the rule-based search methods a greater likelihood of finding acceptable solutions before reaching the imposed search limit. Thus one conclusion that can be drawn is that case adaptation learning is more effective if used in conjunction with case learning, at least in the reduction of unsolvable problems.

Perspective on the contribution of case adaptation learning

From the above data at least one strong conclusion can be drawn:

Case adaptation learning can be as effective as case learning and in some situations more effective.

The results demonstrate that the addition of case adaptation learning as a component of the CBR process can reduce the time needed to perform case adaptations. Several additional points can be drawn from the data as well:

1. **Storing generated plans for later reuse (plan case learning) alone reduces the number of case adaptation problems encountered**: This point is a fundamental property of case-based reasoning. In the trials using only plan case learning, some examples required zero case adaptations while others only needed trivial modifications for which local search of the knowledge base was sufficient. To some extent, case learning avoids the case adaptation problem. With appropriate selection of retrieved cases, the number of case adaptations that need to be performed is reduced, decreasing total case adaptation time. Of course a retrieval system that makes use of good similarity criteria to select the best plans is a necessary requirement.

   However, without a complete domain theory, finding sufficient similarity criteria is challenging. Further, the elimination of case adaptation problems may unnecessarily restrict the flexibility of the system to create new plans. The few case adaptations that are left for the case learning system to contend with can be problems that require a large amount of processing time to solve. Whereas other plans that have more initial problems may require less total case adaptation effort. Thus avoidance of case adaptation problems alone does not eliminate the need for case adaptation nor for building methods to handle case adaptation problems.

2. **Decreasing the difficulty of case adaptation problems is more beneficial than reducing the total number of case adaptations**: Case adaptation problems that require large amounts of processing time or manual case adaptation account for the largest portion of the total cost of case adaptation. These problems are so costly that solving one case adaptation of this type can cost more than all other case adaptations in a single plan combined. Therefore retrieving a plan containing a single unsolvable case adaptation will require more processing by the case adaptation component than finding any other stored plan that would contain no unsolvable case adaptations. An effective case base system employing case adaptation should reduce the total time spent solving case adaptations irrespective of the number of case adaptations being performed.

3. **Memory search learning augments the reachable areas of memory**: In traditional case-based reasoning systems, each case provides coverage for some area of the solution space. The larger this space, the larger the number of cases that need to be stored for sufficient coverage. The increase in cases is costly from the retrieval perspective as more cases require consideration when each new problem is presented. In addition, as cases are added there is a potential for redundancy in the case base as new cases overlap with existing cases. Using adaptation cases in conjunction with plan cases has the potential to extend the system solution space more than with only a single type of stored cases. Each adaptation case could be used to alleviate a certain type of problem in each of the stored plan cases. In other words, the addition of a single new adaptation case can augment the solution areas reachable for every available plan case. For example, if an adaptation case is added to the system that can solve certain *lack of access* problems. A stored plan case for a flood in Georgia that under reuse exhibits this same problem now might be solved without relying on unguided rule-based methods. In fact, all reused plans exhibiting this same problem might reapply this adaptation case towards solving this case adaptation problem. Thus the addition of even a few adaptation cases can improve the ability of the system to reapply its current set of stored plan cases.

5.4 Comparison of similarity criteria

To support the interaction of case-base planning and case-based adaptation learning we developed a new similarity method. In order to evaluate our adaptation knowledge based similarity method, we examined several different similarity methods.

The effect of differing similarity criteria on case adaptation performance

The first set of data is analogous to the case adaptation data given above. A set of trials was performed with varying similarity criteria in an attempt to support the benefit of the RCR approach. Five different similarity criteria were each used separately to score candidate response plan cases in order to decide which to adopt. The five criteria are:

- **Number of case adaptation problems:** The number of problems that require case adaptation in the candidate response plan are counted independent

of the adaptation cases that can be applied. In essence, this method weighs all case adaptation equally and gives preference to response plans that will require the fewest case adaptations.

- **Average problem cost:** This criteria was inspired by a similar method used by Smyth and Keane (1994). However, this method uses historical data from past case adaptations to rank cases on their expected cost of repair. In addition, this method determines the expected costs based on system stored averages for different problem types that are continuously updated as new problems are solved.

- **Length of stored derivation:** Scores are assigned by retrieving relevant adaptation cases and estimates the cost based on the length of the derivation path to be reapplied. The length of the derivation path is the number of independent primitive operations that need to be reapplied.

- **Prior cost of case adaptation:** Scores are assigned by summing all of the prior costs of case adaptations as given by the related adaptation cases. This method assesses the difficulty of a case adaptation based solely on weak search methods. The final solution path could be short but would score poorly if the amount of searching that occurred was high.

- **Reapplication cost and relevance (RCR):** Weights are assigned to each problem by assessing how closely a selected adaptation case matches the constraints of the current problem. Cost is assessed by multiplying the cost of reapplying the case by a value of how dissimilar the retrieved case is from the current problem. This method relies on some low-level semantic similarity methods as applied to the adaptation cases, however these methods mostly score feature values based on their absence or presence.

Hypothesis. The use of adaptive similarity methods (derivation length, prior cost, and RCR) should be superior to the semantic similarity method. Additionally, the RCR method should provide the greatest improvements to the cases adaptation process as it should select the most applicable adaptation cases for reuse.

Experiment. Previously, Smyth and Keane demonstrated the effectiveness of applying adaptability based similarity techniques to the selection of appropriate system cases. They argued that applying such techniques is superior to the traditional semantic similarity criteria. However, while this is an improvement on the traditional model, their technique does not account for the changing abilities of a system that learns to improve its case adaptation.

As adaptation cases are capable of storing large amounts of information describing previous case adaptations, it was unclear what information would prove the best indicator of case adaptability. The RCR method was expected to best assess adaptability as it scales the reapplication cost based on the relevance of the adaptation case to the current situation. Thus an adaptation case might be selected with a low reapplication cost but be scaled to reflect a higher cost if it is dissimilar and thus likely to be less useful than other more similar cases. With this method, the potentially expensive case adaptations would be avoided in favor of several cheaper case adaptations.

This experiment began after the system had been converted to use the Wordnet knowledge base. Thus the remainder of the results are provided only with Wordnet versions. Previous results have suggested that the greatest differences between the two knowledge bases were due to the size of the Wordnet knowledge base. Most of the instances in which different learning methods produced unexpected results were caused in part by the larger knowledge base. It was these results that were of interest to the research and thus, after these first experiments, we focussed exclusively on using Wordnet as our knowledge base.

Results. Figure 5.6 shows the results of a set of trials using each of the described similarity metrics. All trials were performed with both case and case adaptation learning present, and as such there is some overlap with results described earlier. These results illustrate that the RCR method does perform better for selecting easily adapted cases than using either the system average or a raw count of the number of case adaptation problems that exist in the response plan. However, RCR performed almost identically to both the prior cost of the case adaptations and the raw length of the prior derivations. These values are consistent for both the number of operations performed and the number of nodes visited.

Discussion. The results presented here show that in all situations tested, the prior problem cost and length of derivation similarity methods perform equally to the RCR method. An examination of the cases used in the system showed that there was substantial consistency between these methods in the cases that were created and subsequently reused. The primary reason for the convergence of these methods is believed to be the use of the Wordnet knowledge base. In the old hand-coded knowledge base of 2000 nodes, differences between concepts in the knowledge base were very small and the similarity metric required fine tuning to select the appropriate response plan, thus providing for the early successes with RCR. In the Wordnet knowledge base of close to 100,000 concepts, differences in concepts are large and any additional noise added by the different similarity metrics does not disrupt the selection of appropriate adaptation cases. Thus the overall performance between the methods that reuse information stored in adaptation cases remains about the same.

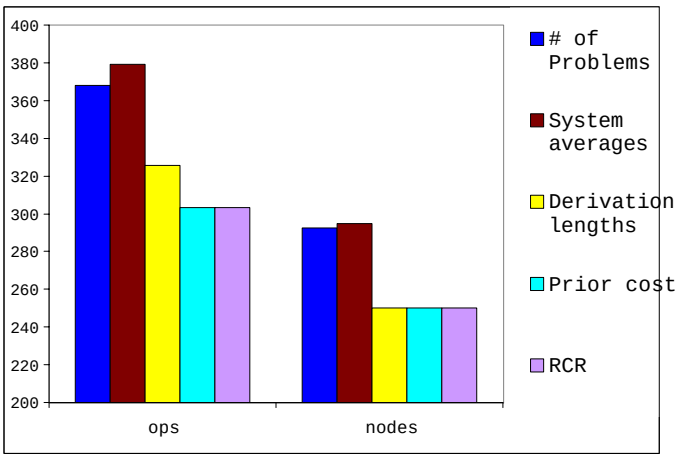

Figure 5.6: Average case adaptation cost for each similarity metric

This strongly supports the claim that case adaptation based similarity criteria is an effective method for selecting candidate plans from the case base.

In contrast, the methods that didn't use adaptation cases (number of problems, and average cost) performed poorly. Even more surprising was the approach of using system average costs for each problem type performed worse than a simple count of the number of problems present. It was believed that any method accounting for the case adaptation abilities of the system would perform better than systems that were blind to case adaptation difficulty. These results contrast with other work on applying case adaptation knowledge to similarity criteria (Smyth & Keane, 1994) but can be explained by the nature of case adaptation in this system. DIAL always attempts to avoid difficult case adaptation problems. However, this avoidance strategy can be flawed. If a difficult case adaptation problem is encountered, it is not always effective to avoid that problem. While this is a good approach from the response plan perspective, it is short sighted over the long term for the system. A one time expenditure of effort to solve a difficult problem type that is likely to reappear with some frequency can save processing time over the course of the system's processing life. Once the difficult problem is solved, the related adaptation case for the solution is stored for future reuse. When the same problem is identified in the future, it is no longer a difficult problem and can be solved by reapplication of the adaptation case. Thus solving difficult problems early can increase the types of problems that can be solved during later processing episodes.

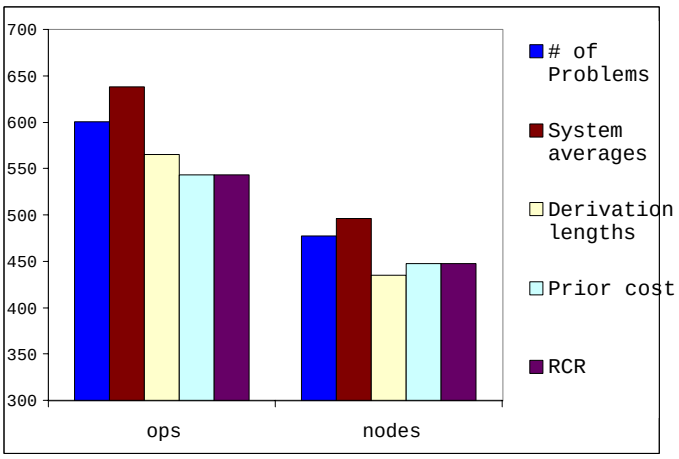

Figure 5.7: Average adaptation cost per response plan for each similarity metric

The effect of similarity learning at the response plan level

Increased benefits to the performance of individual case adaptation are only one part of the larger picture. Equally important is how the case adaptation costs affect overall performance during the creation of a response plan. The same set of data was examined for the average cost of case adaptation for each response plan that was created.

Hypothesis. Similarity methods that reuse knowledge from adaptation cases will result in reduced total case adaptation processing times when compared to static methods.

Results. The graph of this result is shown in figure 5.7 and shows several surprises. The first surprise was that the similarity metric using problem cost average performed worse than expected. Equally surprising, all three methods of similarity assessment that used knowledge from stored adaptation cases performed almost identically. This is contrary to our supposition that a more refined similarity method such as RCR would produce additional performance gains. However, all methods did perform slightly less that 10% better than method not using case adaptation knowledge.

Discussion. The use of case adaptation knowledge to support similarity assessment is reaffirmed by these results and the arguments laid out in the prior section continue to hold here. The inconsistencies between the results for the number of operations applied and the number of nodes visited require additional discussion. As the system acquires both plan cases and adaptation cases, some beneficial interactions are formed between pairs of cases. Certain adaptation cases, while widely applicable, may better address the problems that are likely to arise in specific response plan cases. When a disaster occurs and requires the creation of a response plan, there may exist multiple candidate cases that could reach the new solution with the use of appropriate adaptation cases. The similarity metrics that reused prior case adaptation knowledge selected different plan case and adaptation case pairs and therefore found different paths to the solution while the overall costs remained about the same. Thus some of the case adaptations among these three similarity metrics visit more nodes while others perform slightly more case adaptations but overall perform at about the same level. These differences reflect the types of adaptation cases selected by each method but overall the case adaptation cost remain approximately the same.

Similarity criteria based on case adaptation knowledge improves the selection of response plan cases.

The effect of similarity learning on unsolvable problems

Next, the number of problems that were unsolvable using system methods was examined. All similarity methods should prefer candidate plan cases that have the least expected case adaptation cost. Thus each method should find approximately the same number of unsolvable problems.

Results. The results in figure 5.8 show that there was only a small difference in the number of manual case adaptations performed. Particularly the method using system averages encountered two more problems that it could not solve automatically than any of the other methods. This should not be a surpise, however, as these averages reflect the past behavior of the system of certain types of problems and does not account for the learned adaptation knowledge. It is interesting to note that all the methods tended to encounter the same exact problems requiring user intervention.

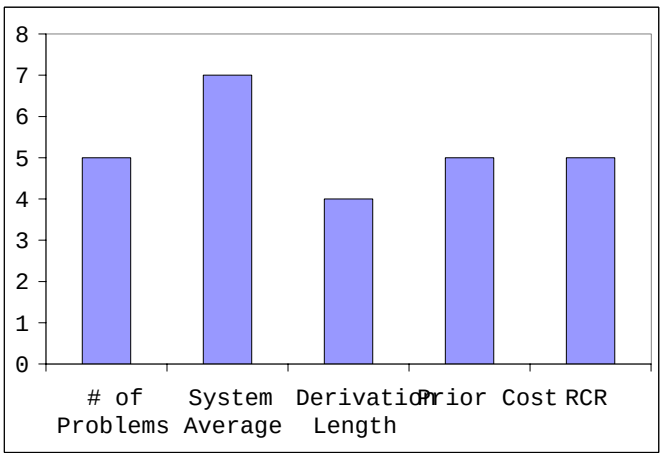

Figure 5.8: Number of unsolvable problems per similarity metric

5.5 Analysis of learning over time

The prior experiments validated the effectiveness of the case adaptation component on the examples that were presented to it. However, learning is an ongoing process in a CBR system and it is valuable to examine how the learning proceeds over the course of the presented examples. With most learning programs, the initial phase of examples is considered a training phase. During this phase it is likely that the system will perform worse than average on some case adaptation examples. Eventually the system should improve its performance once the requisite case adaptation knowledge has be acquired. However, in a dynamic environment, training is occurring continually and the system is adjusting its knowledge to reflect the current trends of its examples. Since the DIAL system begins with only a small amount of initial knowledge, the behavior of the system as learning progresses from the startup until some stabilization occurs is worth examination. This analysis breaks down the case adaptation average after each example over the course of the complete trial.

Results. Figure 5.9 and figure 5.10 show the progression of case adaptation cost over the course of the system trials. As expected, during the early portions of the trials, there is significant noise in the averages as the system attempts to learn and settle into a more stable state. Later, the averages begin to separate themselves although all methods seem to follow similar trends.

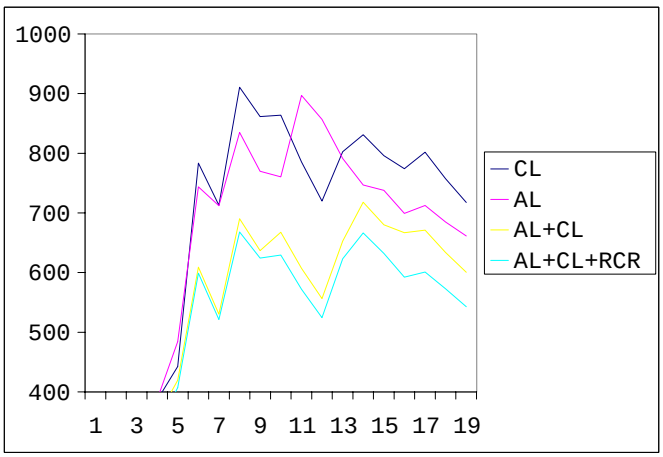

Figure 5.9: Average case adaptation cost per example

Discussion. These graphs provide the least amount of additional evidence regarding the effectiveness of our methods. It is difficult to assess what changes may occur over the course of hundreds of examples, a scope which was beyond the limits of this research. However, we will attempt to extrapolate some general meaning from this graph and what it suggests for the long term performance of our learning methods. Our research has been based on three primary ideas.

1. Case adaptation learning methods can improve system performance when case adaptation problems exist.

2. Similarity learning methods better couple response plan learning with case adaptation learning.

3. When case adaptation knowledge is reused successfully substantial performance gains are attained, and when attempted unsuccessfully there tends to be only a small additional cost.

Thus these graphs support these foundational claims but do not prove them. When an adaptation problem is encountered our learning methods will either solve the problem much faster than our baseline methods or will solve the problem with a cost similar to the baseline methods. And while this is not a universal rule in our system, it will trend our learning methods lower than the baseline methods over time.

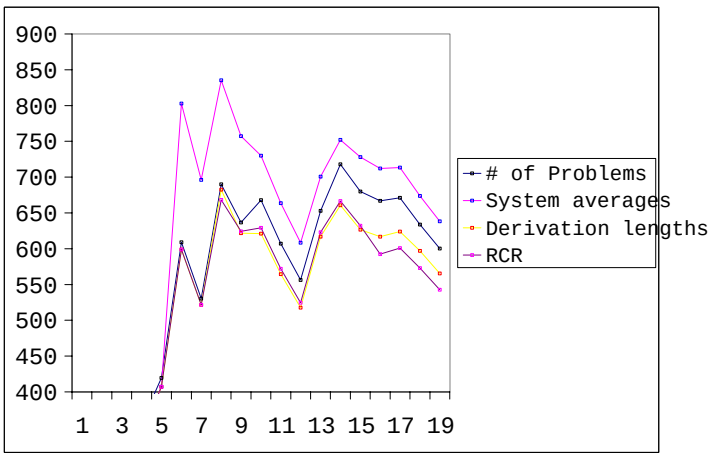

Figure 5.10: Change in case adaptation cost per example for each similarity metric

Perspective

One of the advantages of case-based reasoning over other learning algorithms has been that a case-based reasoning system is capable of solving new problems after storing only a single problem solution. The line graphs here provide evidence that this effect extends to the case-based case adaptation model. After only a few examples, the similarity metrics using stored case adaptation knowledge are substantially better than the static similarity assessment methods.

5.6 Overhead costs for case adaptation learning

Our discussion of case adaptation learning is incomplete without addressing the issue of added system overhead. With only a single case based reasoning process, the overhead of the system is dependent on the size of the case base and its organization. As the case base becomes large, the overhead to find and retrieve a similar case increases. However, our system employs two separate case based processes, and the possibilities for additional overhead are greatly magnified. In our system overhead can exist in any of the following forms:

- **Retrieving plan cases**: This is a direct lookup for a set of candidate plan cases corresponding to the current disaster type.

	Avg Processing Time	Avg Retrieval Time
No learning	30.4	.045
Case Learning	13.9	.088
Case adaptation Learning	15.8	.20
AL/CL	11.1	.25
AL/CL/RCR	17.3	.25

Table 5.4: Comparison of system processing times and average retrieval times per response plan episode.

- **Applying the similarity learning method**: This involves retrieving adaptation cases for each problem identified in the candidate plan cases.
- **Reapplying adaptation cases**: This is not new overhead but differs from traditional case based planning in that a suite of three different case adaptation methods are applied sequentially. This overhead is already accounted for in the presented results and will not be considered further.
- **Storing new adaptation cases**: This is a small amount of added cost to find the appropriate location in the case base to store all of the generated cases.

While this was a primary concern for the DIAL system, several organizational procedures were added to limit the amount of overhead in each of these areas. The added procedures included:

- Limiting the number of cases that can be examined
- Using a hierarchical organization for the case base.
- Forgetting unused or faulty cases

We expected overhead to exist but we did not believe that it would exceed the gains we have already described.

Results. Table 5.4 compares total retrieval times for the different learning methods alongside the time spent performing all other computations (including case adaptation). From this table is can be seen that retrieval times increase moderately as additionally learning and reasoning methods are added. However, all of the retrieval times are significantly smaller than actual system processing times and in this analysis can be ignored.

Discussion. These results are sufficient to suggest that overhead played no significant role in the experiments that were performed. However, this result does not suggest that overhead will never be a problem. While most research systems are small, an industrial case based system could have well over 10,000 cases (Kitano & Shimazu, 1996). A large system employing case adaptation learning would need to reexamine the issue of retrieval overhead before successful deployment. An astute observer may notice that the processing times per response plan do not resemble earlier results, this is one of the issues addressed in the next section.

> The overhead of multiple learning methods and knowledge sources does not reduce the overall effectiveness of the system.

5.7 Problems with case adaptation learning

While most of our results demonstrate the value of the case adaptation learning method, our approach is not without problems. This section examines some results that suggest limitations to the application of case adaptation learning.

The skewing effect of difficult problems

Performance of the system is improved on average when learned case adaptation knowledge is used to solve future adaptation problems,, however a closer examination of the examples processed suggests that our approach may not prove superior over traditional methods in all situations. Table 5.5 shows a breakdown of the different types of adaptations that were performed as a percentage of the whole for each method. The adaptation learning column numbers are skewed towards easier case adaptations for reason already discussed as this method performs case adaptation on almost every possible situation. The other columns are more instructive. The table is broken into four columns representing four different types of problem difficulties in the system. It should be noted that all learning methods saw the same basic problem set, but the level of difficultly of the problems for each learning method could vary.

- **Easy Problems:** Problems that are easy require fewer than 100 operations to complete. Normal blind search can find these solutions with little difficulty.

	Easy	Medium	Difficult	Very Difficult
No learning	41%	27%	11%	21%
Case Learning	32%	35%	16%	16%
Case adaptation Learning	64%	28%	5%	4%
AL/CL	42%	29%	18%	10%
AL/CL/RCR	53%	21%	18%	9%

Table 5.5: Breakdown of problem difficulty

- **Medium Problems:** Problems that are medium require fewer than 500 operations to complete. These are more challenging but can still be completed in an acceptable amount of time.
- **Difficult Problems:** Problems that are difficult required more than 500 operations but less than 1000.
- **Very Difficult Problems:** Problems requiring more than 1000 operations are unlikely to be solved without user intervention. In fact, we placed a cutoff value of 2000 operations on all of our searches. This larger value was chosen so as to allow any possible solutions that might exist to be examined.

Between case learning, the combined learning and the RCR method, the first two columns reflect a similar breakdown and account for about the same total cost to the system. Most of the difficulty that case learning faces (and accounting for at least some of the higher numbers it experienced) is due to the large number of very difficult problems that it must address. If these problems were removed from the system, the performance of case learning would fall much more closely in line with the other methods. These very difficult problems arise as context plays a large role in disaster response planning and the regularity that case based reasoning relies on is not always present. In a domain with greater regularity these types of problems would not exist and our automated case adaptation method might not be needed. However, most systems exhibit some level of irregularity and no set of rules will enable a case based system to address these problems without some form of case adaptation learning.

Processing time disparities

Table 5.4 showed that the actual processing time to solve each response plan using the RCR method was higher than any other method that employed learning. This is

in opposition to the improvement in both the number of operations and nodes that were reported earlier. However, this added cost is not unexpected. Some of this time cost is assumed in the added difficulty of performing single operations. Operations are not performed blindly when case adaptation learning is used but require assessing how the adaptation case states this operation should be applied. More of the cost is assumed by the evaluation component of the system. While we have not focused much of our discussion on the evaluation component it plays an important role in similarity learning. Once adaptation cases are retrieved for each candidate response plan, a substantial amount of additional processing much be performed to compute the applicability of the adaptation cases and to assess the future expected cost for each candidate plan. This processing cost cannot be easily alleviated but may not be a far reaching problem. The total time spent in all methods on each response plan is approximately the same, however if over the long term the system increases in competence and alleviates any need for human guided case adaptation than the lack of substantial time savings may not be relevant.

5.8 Perspective on the performance of the DIAL system

Several important questions were answered by the empirical analysis of the DIAL system. This section will highlight some answers to the questions that were raised by earlier chapters and provides additional discussion on conclusions that these results as a whole suggest for case-based reasoning and case adaptation learning. The discussion will combine all the results to suggest new ways of viewing knowledge acquisition from different sources and the sharing of knowledge during the CBR process.

Outcomes of learning

The data from our experiments has led to several observations and conclusions. This section discusses many of of the outcomes of our case adaptation learning process.

The effect of case adaptation knowledge on search. The first conclusion addressed is how acquired case adaptation knowledge affects the efficiency of the case adaptation process. The results illustrate that for certain sets of problems the acquisition and reuse of case adaptation knowledge reduces the overall burden on the system in terms of the amount of searching necessary. The data examined several learning interactions that exist in the system. The interaction of most interest at the start

of these experiments was the relationship between traditional case learning methods and the new case adaptation learning method. The initial chapter suggested that the addition of internal CBR processes to the primary CBR planning process can only improve the overall system. Case learning has already been demonstrated as beneficial in numerous systems (Hammond, 1989; Kolodner, 1993; Leake, 1996b)including DIAL. If the addition of case adaptation learning improved system performance by itself then it seemed reasonable that the inclusion of case adaptation and plan case learning would produce a synergy resulting in superior performance than with either method alone.

What occurred from even the earliest experiments was that case adaptation learning supported case learning. Problem solving in DIAL was centered primarily around the overall case-based planning process. One effect of the plan case learning is that there is a reduction in the overall number of case adaptation problems that need to be solved. Case adaptation learning working with the larger case learning system is then used to reduce the system effort required to solve any of the remaining problems. Thus some substantial efficiency gains were made over case based methods without automated case adaptation. However, this synergy did not immediately manifest itself. In fact, in some examples, the combination performed worse than using case adaptation learning alone. The discrepancy was attributed to a poor selection of plan cases from which to begin the case adaptations. These plans did not match the type of case adaptations that the system knowledge could most easily solve. Thus the case adaptation component was forced to attempt problems for which it did not have the most applicable case adaptation knowledge available. When the case adaptation based similarity criteria were included to allow the case adaptation component influence on the selection of plan case further improvement were seen.

The effect of the growth of multiple case bases. A central question of concern in the CBR community is the question of the utility of learning. The *utility problem* (Minton, 1988) states that as the amount of knowledge that is learned increases, the amount of time spent searching for the appropriate knowledge begins to outweigh the advantage of having that knowledge. In the DIAL system, learning occurs on several different levels and as such the utility question is of fundamental importance to the system. The problem was addressed by automated system management of the case base. Between problem solving episodes, the system examined the contents of all case bases and removed cases that are judged the least useful. This judgment is based the length of time a case has been present and the number of times it has been successfully reapplied.

System overhead. The overhead of employing the internal learning processes in the DIAL system was not greater than the amount of gains that are made by including

this learning. The key to keeping this overhead at a manageable level is to provide basic methods of maintaining the case bases. This maintenance is a process of limiting the number of cases that are stored such that only relevant and useful cases are kept over the long term. If cases are never used, then there may be little or no reason to retain this case when it is simply adding to the overhead of retrieval and storage.

Effect of similarity learning. Similarity learning proved to be an effective method of exploiting the potential synergy between case learning and case adaptation learning. Without similarity learning, the two learning methods did not have a well defined relationship in which to interact. The retrieval phase of the case based planning process worked to select cases that would minimize the difference between the problem description and the selected case. The reduction of differences does not relate to the difficulty of repairing the differences that remain in the selected plan case. The selection process in this scenario proceeds without ever considering the types of problems that the case adaptation component would prefer to solve. As the remaining processing time for the solution is dependent on the effectiveness of the case adaptation process, the similarity learning approach in effect queries the case adaptation component as to the solvability of different problems identified in candidate plans. With a similarity learning method that acts as bridge between the two learning components, the selection of a final plan case to reapply can be based on how effectively it can be solved in the case adaptation component.

Effect of interacting knowledge sources. When different components are allowed to independently stored knowledge, one component may lack some required knowledge that is stored by another component. An obvious solution is for the different components to share this knowledge in appropriate circumstances. It has already been described how the DIAL system shares case adaptation knowledge to improve the similarity assessment process. Other possible interactions include, augmenting the set of rule-based search methods by using store memory search cases that have been reused effectively on several occasions. Consequently problems that may be difficult to solve using a single knowledge source may become easier to solve when other knowledge sources share information, and the range of solvable problems may be increased.

Robustness of our method

We have presented evidence that our learning methods can outperform baseline methods through a range of examples. However, we have not provided evidence that these results are robust. How would the system perform under a different set of

examples of even perhaps a different ordering of the current problems? One way to establish robustness would be through statistical analysis of different problem sets. However, when experimenting with this approach it was realized that the types of examples provided to the system and the system's approach to removing unused cases can play a profound affect on the outcome.

Thus in place of statistical methods, we catalog different types of problem sets and how our methods would perform under these conditions.

- Example problems requiring little or no case adaptation knowledge. Our methods would produce no improvement over traditional methods. Learning case adaptation knowledge would not be of benefit for performance reasons.

- Example problems that provide for the occasional hard case adaptation but primarily populated with straightforward and easy case adaptations. Our methods could actually perform slightly worse under these conditions. Some adaptation knowledge would be acquired and the system would attempt to reuse that knowledge on future problems that could be solved easily with the case adaptation knowledge.

- Example problems requiring a mixture of typical hard problems and other types of problems. This is the type of domain that we performed our tests on, and is typical of many domains in case-based reasoning. Our approach would be most successful with these types of examples. Adaptation cases would be learned and reused regularly and subsequently could outperform methods that work from scratch each time.

- Example problems with predominantly hard case adaptations. It is unclear how our system would do under these circumstances. If only hard case adaptations arise then it is likely that there is little gained from the response plan learner and in fact the problems may exhibit some irregularity. Adaptation knowledge would attempt to tune itself to these problems by storing all solved problems, but it is unclear how successful reapplications would be.

Interesting future work could examine these types of domains. One difficulty of this is that there are few practical domain sets with these characteristics and it is a very difficult task to hand generate these sets without exhibiting creator bias.

5.9 Conclusions

This chapter focused on empirical data supporting the addition of both case adaptation learning and similarity learning in the DIAL system. Case adaptation learning reduced the total processing costs of the case-based reasoning process. It performed best when effectively used in conjunction with plan case learning, but also was competitive with plan case learning when each was examined individually. The data presented here provides the best argument for the inclusion of an automated case adaptation processes under the types of conditions described.

Similarity learning proved to be the key process for coupling case learning and case adaptation learning. The reuse of case adaptation knowledge to support the retrieval process produced the greatest reductions in processing costs. Several different approaches for reusing the case adaptation knowledge were examined. No substantial difference were found between the methods the directly reusing the knowledge in adaptation cases. This chapter has provided evidence and analysis to support many of the claims of this research.

6

Conclusion

This research has studied an approach to learning case adaptation knowledge as a supporting subprocess to a case-based planner. Our approach relied on a small set of predefined rules to provide a foundation for an internal case-based reasoning process. The internal process augmented the system's initial knowledge by acquiring case adaptation knowledge so as to better handle future case adaptation problems. Experimental evidence suggests that this learning can improve overall system performance. Further, we have shown that case adaptation knowledge can benefit more than the case adaptation process by integrating with other components of the larger system like the similarity assessment process. This chapter reviews our research results and returns to the original questions that were posed in order to evaluate the relevance and importance of this research.

The chapter begins by summarizing the information presented in this document. We then review prior work that formed the foundation for our research. Next, the chapter reviews several questions proposed in the early chapters and uses them to guide our overall evaluation of our method and system. Then, we describe the contribution of this research to the state of the art in case based reasoning and case adaptation. Following this, the chapter asks several new questions provoked by this research that open additional avenues for future work. Finally, the chapter concludes with a reflection on the contribution of this work and its importance to the field of case-based reasoning.

6.1 Document summary

The discussion of case adaptation began with an examination of case-based reasoning. Many CBR systems do not attempt to address the case adaptation question.

With only a few exceptions ((Cheetham & Graf, 1997), the common approach in applied CBR is to eliminate all case adaptation and provide expert users to perform case adaptations when necessary. While this approach is less than optimal, codifying the specialized case adaptation knowledge and applying the knowledge in an appropriate way has been out of reach. In fact, from our work, we conclude that it is unlikely that a general purpose case adaptation approach will be developed in the near future. However, we have shown that certain regularities can exist in the types of problems and solutions encountered in a specific domain. It is this regularity that motivated our approach to learning case adaptation. This idea laid the foundation for a case-based approach to case adaptation.

With our approach to case adaptation defined, the discussion continued in chapter two which developed the basic architecture supporting the case based case adaptation idea. The DIAL system was designed as a case-based planner to resolve problems created by natural disasters such as earthquakes or floods. Our design revolved around two guiding factors: the representation and reapplication of stored knowledge and the interactions between system components. Motivation for this architecture came from an extended example weaved into the system description.

Chapter three provided the explanation for our automated case adaptation algorithm. The algorithm begins with only a handful of general case adaptation rules. These rules are used to guide a slow and ineffective early case adaptation system. As problems are solved, the knowledge of how the problem was solved is encapsulated as adaptation cases resulting in new system case adaptation knowledge. The chapter described how this new knowledge can be reused by applying it to similar problems. If no case adaptation knowledge is available, the system can still rely on its slow and unguided search process or ultimately ask an expert user for assistance. The key to this algorithm is that, in a given domain, the types of problems encountered may be unknown during system development but the regularities that exist during actual processing can be exploited by the internal case based reasoning process. Therefore the resultant system will develop so as to solve case adaptation problems using a case base process whenever possible. In situations where current knowledge does not provide for a solution, the other methods exist to provide backup support.

One added benefit of acquiring case adaptation knowledge was how it could be applied by other components of the primary CBR process. This was the subject of chapter four. The similarity assessment process was improved by using the stored adaptation knowledge to rank candidate response plan cases. Previously this process relied on similarity rules that suggested plan features that should be examined. This is a reasonable approach when little advanced knowledge is known about the domain, however it is generally ineffective at selecting the best possible case for the system to process. Our key insight was that the best stored plan may not be the one that

on the surface appears most similar to the current situation. A better approach is to determine how easily the system can apply the stored case to the current situation. Case adaptation knowledge can give an estimate of the amount of added effort that must be applied to make the candidate plan case acceptable in the new situation. However, since a *score* is created from the case adaptation knowledge, several methods were proposed on how this score should be computed. These methods ranged from simply counting the number of problems that exist in each candidate plan to summing the past costs of each of the problems. By using the case adaptation knowledge to improve the similarity assessment process the relationship between the different CBR systems can be strengthened.

Chapter five presented the experimental results to support the basic claims of this research. The results showed that case adaptation learning reduced the overall case adaptation costs in the system. Our approach was effective while limiting overhead. Likewise, the addition of knowledge based methods for similarity learning improved case selection to the point that system costs decreased further. Our conclusions from these results was that case adaptation learning is a viable approach to take towards automated case adaptation. However, the results also suggested that this approach can only be effective under certain system conditions.

The discussion in this dissertation has attempted to present the benefits and disadvantages of our approach to case adaptation and adaptation learning. To assess this, our discussion turns to focus on how our work compares to the prior work on which it was built.

6.2 Comparisons to other research

There have been many sources of guidance and inspiration that helped form the foundation of this research. Much of this work has been cited throughout this document. However, explicit comparisons and differences have not always been presented. This section examines several prior efforts that provided the base for the work done with DIAL and how DIAL has expanded on the original idea.

Memory search:

The memory search process formed the basis for the entire case adaptation learning process. As such, the search process had to be accessible to explicit reasoning and learning. Thus our approach followed closely the models of the memory search process used by (Kolodner, 1984; Rissland, Skalak, & Friedman, 1994) in order to

increase the flexibility and effectiveness of memory search. Several researchers have designed models of memory search that were accessible to learning such as (Cox, 1994; Kennedy, 1995; Leake, 1995a). Our research followed closely along these prior models. The foundation provided by prior researchers on memory search allowed this work to directly move to examine the questions behind storing the results of memory search.

Case adaptation:

Our case adaptation process reapplies adaptation cases on new case adaptation problems. However, several different approaches to case adaptation have been previously implemented. For example, although CHEF (Hammond, 1989) had a static library of domain-independent plan repair strategies, it augmented that library with learned *ingredient critics* that suggested adaptations appropriate to particular ingredients. The PERSUADER system (Sycara, 1988) used a combination of heuristics and case-based reasoning to guide adaptation, searching memory for similar prior adaptations to apply. PERSUADER's approach handles case adaptation in a similar was to DIAL, however DIAL tries to infer the reasoning process previously used by focusing on derivational replay of cases. In both CHEF and PERSUADER, the adaptation information learned was domain and task specific. By using memory search cases as the basis for case adaptation in DIAL, more flexibility is achieved.

The use of CBR for case adaptation has also been advocated by Berger (1995), in the context of storing and re-using an expert's adaptations. One limitation of Berger's system was that the cases that were created were highly domain specific and not easily applied to new situations. An alternative approach to the case adaptation problem is to use derivational analogy, deriving a new solution by re-applying a prior solution process to new circumstances, rather than directly adapting the old solution itself (Veloso, 1994).

A number of methods have been proposed for facilitating the adaptation task. The difficulty of adaptations may be decreased by representing cases hierarchically (Aha & Branting, 1995; Goel, Ali, Donnellan, de Silva Garza, & Callantine, 1994; Marir, 1995; Redmond, 1992; Smyth & Keane, 1996), allowing cases to be reused at the most specific level of abstraction that can be easily applied to the new situation, or by combining relevant parts of multiple solutions, rather than by retrieving a single solution that must be adapted to fit (Ram & Francis, 1996; Redmond, 1992).

Support for interactive user adaptations has also been proposed (Bell, Kedar, & Bareiss, 1994; Smith, Lottaz, & Faltings, 1995; Sinha, 1994), but these methods make no attempt to store and reuse adaptation information. (Goel, Garza, Grue, Murdock,

Recker, & Govindaraj, 1996) presents the user with derivational traces of previous interactions to support interactive adaptation, although it does not store the new adaptations.

The INCA system (Gervasio, Iba, Langley, & Sage, 1998) relies on user interaction to correct problems in plans left after a rule based adaptation is performed. This is similar to the manual adaptation process in DIAL except DIAL stores a reasoning trace independent of the specific context.

One approach that closely matches our approach uses description logics as its underlying knowledge representation (Gomez-Albarran & et al., 1999). While our method makes no assumptions about the underlying representation, their specification in this area while, using a case adaptation method closely paralleling our own, demonstrates further possibilities for our approach.

Other adaptation learning methods have also been proposed. Bhatta and Goel ((1996)) present an approach to learning generic teleological mechanisms for use in model-based adaptation (Goel & Chandrasekaran, 1989). Some researchers have proposed inductive learning of adaptation rules (Hanney & Keane, 1997; Wilke, Vollrath, Althoff, & Bergmann, 1997). McSherry (1998) used a set of past cases to refine estimates of the value of new cases. Others have employed machine learning methods like decision trees (Shiu et al., 2000). Closest to our approach is Oehlmann's ((1995)) metacognitive adaptation, in which a planful approach is taken to generating and answering questions during adaptation and the process is reused.

6.3 Evaluation of case adaptation learning

Several questions guided this dissertation research. This section restates the central questions posed on case adaptation learning and summarizes how these questions were answered.

The first question asked by this research was if an automated case adaptation process driven by an internal case-based reasoning system can improve the overall performance of a planning system. The experiments showed a net decrease in processing time when case adaptation is performed with our method. However, the addition of case adaptation learning alone was not sufficient to substantially improve system performance. A modified similarity assessment method was required to substantially improve the interaction between the internal case adaptation process and the primary planning process.

Another question asked was how is the automated case adaptation process affected by the additional overhead required to perform the process. The addition of our

internal CBR process requires several potentially time consuming operations. The system must identify all potential problems in a suite of candidate plan cases and then retrieve adaptation cases corresponding to these identified problems. Next, the reapplication of adaptation cases on the selected plan must occur. In the event all of the identified problems are not solved the system is reduced to memory search to solve the problems. So while overhead was expected, out final results showed that this overhead was reasonable given the overall savings in processing time that was observed.

Finally, one important question that we examined was what the effect of having multiple case-based reasoning processes was. Would each process create a synergistic effect with the other or would the two processes ignore the benefits of the other? Based on our experimental evidence, it was clear that the two processes did not naturally combine with one another. In fact, the addition of new learning processes to a system is not necessarily sufficient to improve system performance. It was necessary to redefine the similarity criteria used to select candidate plans so as to account for the adaptation knowledge developed by the internal case adaptation component.

Our case adaptation learning method illustrates some of the potential benefits to a system where case adaptation knowledge exists in a changing environment. It also required us to define some criteria to evaluate multiple learning processes in a single system. We identified three issues that must be addressed to insure successful learning interactions.

- The **overhead** of the new process must be less than the improvements and savings to the actual reasoning process. DIAL managed its overhead effectively.

- When several intelligent components interact under a single reasoning process, the **control** of the different processes must be managed to achieve the maximal benefit from each subprocess. Control can exist at different levels as the primary process may pass control to the case adaptation component to manage its different subprocesses. In DIAL, the case-based planning process managed its various subcomponents, although different subcomponents such as the similarity assessment process and the case adaptation process were allowed to interact directly.

- When different learning processes each attempt to solve problems **limits** must be placed on them. Some problems are ill-suited for certain learning methods and the limits prevent them from wasting substantial system resources so that other methods may better address these problems. Each reasoning process in DIAL had a set limit of the number of steps that can be taken to solve a

problem. Without this limit, a reasoning process may follow an insufficient knowledge goal far away from any applicable solution.

6.4 Evaluation of similarity learning

Our work of the similarity learning process initially began as a way to repair a problem that arose between the interaction of two case based reasoning processes. Each process had a selection method that was effective for its own subsequent processing. However, the planning process never took into account the knowledge made available by the case adaptation process. This led to the creation of similarity learning method and has become one of the interesting results of this research.

It is clear from the results in chapter 5 that a similarity method based on the case adaptation capabilities of a system is a requirement. However similarity learning adds further overhead than that already imposed by the case adaptation process. To assess similarity based on predicted adaptation cost, each plan case that is to be considered must be evaluated for potential problems and then, for each of these problems, the appropriate adaptation case must be retrieved. Once the case bases become large, this similarity method would become unwieldy. However, by limiting this similarity method to only a small set of reasonable candidate plans (chosen by the original similarity method) this overhead was manageable. As such a tradeoff exists between selecting the best possible plan from the entire set and eliminating the overhead, our compromise produced useful results. Similarity learning acts as a bridge between the two case based reasoning processes and improves the interaction between the two.

6.5 Evaluation of DIAL

The section evaluates the different components of the DIAL system and identifies their contribution to the success of our method. The evaluation focuses on the contribution of different aspects of the research to the field of case-based reasoning. Each of the four components of the primary CBR process are integral to the success of our internal case-based case adaptation learner.

- **Contribution of the retrieval component:** The retrieval component provides the initial problem situation for the case adaptation component. It was shown that improper selection of retrieval cases can adversely affect the reuse of adaptation cases. It is necessary for the retrieval component to select response plan cases that will best make use of the stored case adaptation knowledge.

- **Contribution of the evaluation component:** Adaptation cases are indexed by the types of problems they solve, therefore the evaluation component is a primary determiner of the sets of adaptation cases that will be considered for reuse. The retrieval component could select the most applicable response plan from the case base, but if the evaluator incorrectly identifies the problems, then the case adaptation is likely to never be satisfactorily completed. The evaluator is a central part of the case adaptation process, and as such requires careful consideration before the total deployment of an automated case adaptation system. This research for reasons of scope has chosen not to explore the are of evaluation as related to the selection of adaptation cases, however this area is open for future exploration.
- **Contribution of the adaptation component:** For case adaptation learning to succeed, the adaptation component must be capable of solving case adaptation problems. When a problem is presented to this component in the context of a candidate plan, a solution to the case adaptation problem is found or the candidate plan will be rejected. In the DIAL system, the case adaptation component is composed of three different methods to solve these problems: case based reasoning, rule based reasoning using local search, and user guided adaptation. On successful completion of a case adaptation, this component facilitates the acquisition of the new case adaptation knowledge in the form of an adaptation case.

6.6 The success of the internal case-based approach

This work has successfully demonstrated the benefits of using an internal case-based process. When a primary system suffers from knowledge limitations or an ill-defined domain theory, case-based reasoning can be effective in overcoming some of the problems the system may face. Our approach is unique in that it uses case-based reasoning to cope with the limitations of a case-based reasoning system. We have already addresses how effective the internal case based reasoning process can be on the case adaptation task. However, three primary points emphasize the advantage of taking a case based approach.

1. Cased based case adaptation can be effective with only a few prior examples. Other learning methods such as inductive learners rely on large sets of examples to induce a general rule set.
2. While many other learning methods attempt to generalize from presented examples, case based reasoning simply stores each specific case. In the disaster

domain, different disaster locations each may have slightly different requirements so the possibility of generalizing across some of these features may be difficult. If each solution is stored as a case it can be directly reapplied.

3. While there is overhead in the application of the learned adaptation knowledge, the acquisition of the knowledge using case-based reasoning comes at little additional cost. Other learning methods may require substantial processing to identify the generalizations that need to be made.

6.7 Internal CBR as a general strategy

The use of case-based components that capture and reuse derivations of a system's own reasoning is a promising approach that can be applied within a broad range of systems to learn useful reasoning paths and operationalize general knowledge. The basic strategy is to augment AI systems by embedding within them case-based "intelligent components" (Riesbeck, 1996) that learn during normal processing. When similar problems arise in the future, the intelligent component furnishes a solution based on the stored case, to replace the initial reasoning process with CBR. Thus the intelligent component is seamlessly integrated with the initial system to improve its performance by building a case library covering actual problems the system encounters. DIAL's contributions to this area are both to investigate this general approach and to apply case-based components to capturing and reusing the *rationale* underlying the system's own reasoning processes.

Issues in applying case-based components included how the components' knowledge must be represented and organized, how much specialized knowledge must be provided to support component CBR processes (and how this effort compares to hand coding rules for these processes, given that the motivation for the component is to increase system performance while alleviating the knowledge acquisition burden), and the effects on overall performance.

It should be noted, however, that some of the specific methods depend on particular properties of the underlying system. For example, DIAL used derivational analogy for its case adaptation process. In order to use this type of CBR, it is necessary to have access to a derivational trace of the underlying process that can be captured and reused. Because DIAL uses a "planful" memory search process (Leake, 1995b), it is practical to capture a trace of that process for reuse. This would not be possible in systems with a more opaque reasoning process.

The internal CBR process for case adaptation also benefits from knowledge already used for the top-level CBR process as the basis of its indexing. Standard CBR

systems, such as DIAL's baseline CBR system with rule-based adaptation, must include some sort of indexing scheme to associate problems requiring adaptation to adaptation rules. The indexing scheme used for this purpose in DIAL's top-level CBR process is also used by its internal case-based adaptation component to index stored adaptation cases, decreasing the knowledge acquisition burden for this process. If case-based adaptation were added to a retrieval-only CBR system, for example, this information would not be available. Thus this strategy is appropriate for improving the performance of an existing adaptation component, but would be more difficult to use to provide an adaptation component starting from scratch. The aim of case-based intelligent components is to improve the performance of existing processes, rather than to provide entirely new capabilities.

6.8 Contribution of the research

This research has made significant contributions to the case based reasoning process. We summarize these contributions with the following five points.

1. An automated case adaptation component can be successfully implemented using an internal case based reasoning process.
2. Learned case adaptation knowledge can be used to refine similarity assessment criteria.
3. Combining rule based and case based knowledge can effectively overcome many of the limitations of each.
4. Case adaptation learning, while effective, is not a general purpose solution to the case adaptation problem.

6.9 Future Work

Several interesting questions arose during the course of this research that could be the subject of future work.

- We have argued that our approach to case adaptation learning improves the case based planning process and does so without significant overhead. However, these ideas have only been tested in the context of this research and not in a large scale application. Such large systems have different types of issues that must

be addressed including the possible size of the system case base and approaches to integrating new knowledge. It may also be difficult on such systems to find individuals with the appropriate expert knowledge to guide the system during its early learning phase. We believe that our method should scale into such a large application but its possible effectiveness is untested.

- The use of predefined rules and learned memory search cases creates an interesting relationship that was left unexplored. Although memory search cases are used to support the systems adaptation cases by storing the traces of successful memory traversals, this knowledge also identifies new relationships in the system's knowledge. For example, a memory search case storing a five step trace could be described as a new connection between concepts in memory that were previously five steps apart. The memory search cases that prove most effective could subsequently become new primitive rules in the system. So just as the predefined rules were used to develop the memory search cases, the memory search cases could be used to augment the predefined rule base.

- In our research, case adaptation learning only stored traces of successful searches. However, failed searches often contain useful knowledge. One failed search may teach the system to avoid certain areas of memory when encountering some types of problems. However, it is unclear how to assess a failed search or what portions of the search are relevant. Further, in the DIAL system unguided search of memory was a purely blind search that meanders in potentially several unrelated areas before finding a solution. How this information could be stored is unclear. A refinement of this idea may be to record application failures of adaptation cases. Thus an adaptation cases would not rely solely on the specified problem type for its selection but could assess its relevancy from its application history.

6.10 Final thoughts

This research has provided one method of learning to improve the case adaptation process of a case based reasoning system. In this venture, the research has been a success. However, the demonstration of the effectiveness of the internal CBR process only suggests that it could be equally effective in other systems. One open question remains as to how this method would handle the utility problems that might arise in a very large scale deployed reasoning system. Of the few large systems that exist, automated case adaptation might be a welcome addition.

Several other avenues of interest emerged directly from the initial question. The

integration of intelligent subcomponents proved a valuable tool to handle deficiencies in the system knowledge. By effectively using different forms of reasoning, an overall reasoning process emerged that was superior to all of its subparts.

Some major challenges remain in the field of case adaptation for CBR. One interesting area for future work would be to examine other methods of acquiring the initial case adaptation knowledge. One area touched on by the DIAL system by needing extensive additional work is acquiring this knowledge from observation of human operators. Human expert users sometimes do not understand their own reasoning process when making adaptations to problems they solve. Creating an interface to capture a trace of the actions expert user's perform while solving problems and determining a vocabulary that enables reapplication of these traces could provide the knowledge necessary to substitute automated adaptation systems in to large scale systems.

Our research has presented a case adaptation method that uses case based reasoning to learn the needed knowledge to perform automated case adaptation. Case adaptation is an important problem to study, but has received comparatively little attention in most research. This research has attempted to address this problem through CBR. Other approaches continue to be examined elsewhere. While no one method is likely to serve as a universal case adaptation procedure, each method adds to the understanding and importance of case adaptation knowledge. The case adaptation learning approach and its similarity learning counterpart are a step towards a better case based reasoning process.

A

Disaster Input to DIAL

The DIAL system processes disaster examples taken from news reports. DIAL is presented with a brief description of the disaster type, its location and the severity of the disaster. This appendix lists some samples of the input stories DIAL was able to process. For a complete listing of the processed stories, please contact the author. The examples include two containing the original news story and the system representations.

Flood disaster in Allakaket, Alaska

Original disaster newswire.

ANCHORAGE, Alaska (AP) -- Army helicopters were sent Sunday to evacuate residents of the village of Allakaket after the Koyukuk River surged to its highest level in 40 years.

''We're completely surrounded by water,'' Allakaket Mayor Agnes Bergman said.

No injuries were reported and nobody was in immediate danger, National Guard Capt. Mike Haller said from Anchorage. One house in Allakaket was uprooted and a neighbor said only the electric wires were keeping it from being swept downstream. The community of about 175 people is 180 miles northwest of Fairbanks.

The river has been rising because of heavy rains last week in Interior Alaska. It was not expected to crest until Monday at Allakaket. Most of the village's 35 to 40 homes were flooded with about 4 or 5 feet of water, Haller said.

Input representation of disaster given to DIAL.

```
(dial (flood (location
              ((city "allakaket")
               (state ("alaska" noun 1 ()))
               (country ("united_states" noun 1()))
               (continent ("north_america" noun 1())))))
             (condition (``catastrophic'')))))
```

Response plan created by DIAL.

```
#6(rp
   allakaket-flood-response-plan
    (flood-response-plan)
    "flood in allakaket"
    ((location ((city "allakaket")
                (state ("alaska" noun 1 ()))
                (country ("united_states" noun 1 ()))
                (continent ("north_america" noun 1 ()))))
     (relief-agency ("red_cross" noun 1 (("organization" 1))))
     (beneficiary ("resident" noun 1 ()))
     (volunteers ("prisoner" noun 1 (("unfortunate" 1))))
     (transport ("helicopter" noun 1 (("aircraft" 1))))
     (shelter ("school" noun 2 (("building" 1))))
     (build-object ("sandbag" noun 1 ()))
     (police-force ("police" noun 1 (("personnel" 1)))))
    ((build-shelters (("construct" verb 1 ())
                 (actor relief-agency)
              (object shelter)))
     (build-levees   (("construct" verb 1 ())
                      (actor volunteers)
                      (object build-object)))
     (evacuate-residents  (("evacuate" verb 1 ())
                           (actor police-force)
                           (object beneficiary)

       (vehicle transport)))
     (police-patrol
      (("patrol" verb 1 ()) (actor police-force))))))
```

Flood disaster in Kabul, Afghanistan

Original disaster newswire.

KABUL (Reuter) - Heavy flooding has killed more than 100 people across Afghanistan, relief workers said Monday as a Red Cross official warned of an ecological disaster in the war-devastated country.

At least seven provinces, covering more than a quarter of Afghanistan, have been affected by floods brought on by melting snow and heavy rains, aid workers said.

They said they had received reports of some 100 deaths but the toll was likely to be much higher. About 3,000 homes were destroyed or badly damaged and thousands of cattle and other livestock killed.

‘‘This is just the first symptom of a major ecological disaster affecting the whole of the Hindu-Kush (mountain range),’’ said Bob McKerrow, delegation head in Kabul of the International Federation of Red Cross and Red Crescent Societies.

At least 15 people are known to have died and 25 are missing. Hundreds of homes have been destroyed and hundreds of cattle killed, he said.

Aid groups are trying to assess the damage and provide relief but are being hampered by poor communications.

Input representation of disaster to DIAL.

```
(dial (flood (location  ((city ("kabul" noun 1 ()))
                (country ("afghanistan" noun 1 ())))
             (condition (‘‘serious’’)))))
```

Response plan created by DIAL.

```
#6(rp
   kabul-flood-response-plan
   (flood-response-plan)
   "flood in kabul"
   ((location  ((city ("kabul" noun 1 ()))
```

```
      (country ("afghanistan" noun 1 ()))
      (continent ("asia" noun 1 ())))))
   (relief-agency ("red_cross" noun 1 (("organization" 1))))
   (beneficiary ("resident" noun 1 ()))
   (volunteers ("military_personnel" noun 1 (("force" 4))))
   (transport ("helicopter" noun 1 (("aircraft" 1))))
   (shelter ("tent" noun 1 (("shelter" 1))))
   (build-object ("sandbag" noun 1 ()))
   (police-force
      ("military_personnel" noun 1 (("personnel" 1)))))
  ((build-shelters (("construct" verb 1 ())
            (actor relief-agency)
            (object shelter)))
   (build-levees   (("construct" verb 1 ())
                    (actor volunteers)
            (object build-object)))
   (evacuate-residents  (("evacuate" verb 1 ())
                 (actor police-force)
                         (object beneficiary)
                         (vehicle transport)))
   (police-patrol
     (("patrol" verb 1 ()) (actor police-force))))))
```

Flood disaster in Johannesburg, South Africa

Original disaster newswire.

JOHANNESBURG, Feb 15 (Reuter) - South Africa's heaviest rains for years have claimed at least seven more victims, most of them children, police said on Thursday.

The number of people believed to have died in a week of floods rose to 42, but weather forecasters said the worst of the downpours should have passed by late in the day.

Air force helicopters lifted stranded people to safety and rescuers set up tent cities for those whose homes had been washed away in the northeastern province of Mpumalanga, where a boy was reported drowned.

Tourism at the Kruger National Park wildlife reserve was washed out,

```
officials said.
```

Input representation of disaster to DIAL.

```
(dial (flood (location
              ((city ("johannesburg" noun 1 ()))
         (country ("south_africa" noun 1 ()))
       (continent ("africa" noun 1 ())))))
             (condition (``serious'')))))
```

Response plan created by DIAL.

```
#2(index
   ((city ("johannesburg" noun 1 ()))
    (country ("south_africa" noun 1 ()))
    (continent ("africa" noun 1 ())))
     #6(rp
   johannesburg-flood-response-plan
   (flood-response-plan)
   "flood in johannesburg"
   ((location
      ((city ("johannesburg" noun 1 ()))
       (country ("south_africa" noun 1 ()))
       (continent ("africa" noun 1 ()))))
    (relief-agency
      ("air_force" noun 1 (("military_service" 1))))
    (beneficiary ("resident" noun 1 ()))
    (volunteers ("air_force" noun 1 (("military_service" 1))))
    (transport ("helicopter" noun 1 (("aircraft" 1))))
    (shelter ("tent" noun 1 (("shelter" 1))))
    (build-object ("sandbag" noun 1 ()))
    (police-force
      ("military_personnel" noun 1 (("personnel" 1)))))
   ((build-shelters
      (("construct" verb 1 ())
       (actor relief-agency)
       (object shelter)))
    (build-levees
      (("construct" verb 1 ())
```

```
   (actor volunteers)
   (object build-object)))
(evacuate-residents
  (("evacuate" verb 1 ())
   (actor police-force)
   (object beneficiary)
   (vehicle transport)))
(police-patrol
  (("patrol" verb 1 ()) (actor police-force))))))
```

B

Sample Transcript of DIAL

The following is short transcript of the DIAL system presented with some disaster examples. At the start of this transcript, DIAL's case base stores a single response plan case for each disaster type, and no adaptation cases currently exist. The entire example encompassed seven disasters presented in sequence to the system. All learning methods are turned on in this example. I annotate the interesting portions of the transcript.

```
Chez Scheme Version 5.0
Copyright (c) 1994 Cadence Research Systems

> (load "loader.ss")
> (start)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
We input an initial description of the disaster for DIAL to
generate a response plna for.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Current disaster is
  (flood ((state ("oregon" noun 1 ()))
     (country ("united_states" noun 1 ()))
          (continent ("north_america" noun 1 ()))))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Retriever Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This is the start of the system and only one response plan for
flood disaster exists. So this plan is retrieved.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
Selected 1 cases for similarity assessment

The following cases have been ranked:
    Case: bainbridge-flood-response-plan (1)

Using adaptative similarity techniques (if available)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
The selected plan is evaluated for problems. In this example,
only two fillers are determined to have problems. We will follow
the relief-agency example.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Evaluating the bainbridge-flood-response-plan
-----------------------------------------------------------------
slot            filler             problem  type          cost
-----------------------------------------------------------------
location        "oregon"           no       n/a            0
relief-agency   "salvation_army"   yes      filler-dne     ?
beneficiary     "resident"         no       n/a            0
volunteers      "volunteer"        no       n/a            0
transport       "motorboat"        no       n/a            0
shelter         "school"           no       n/a            0
build-object    "sandbag"          no       n/a            0
police-force    "police"           yes      phys-unav      ?

The problem based costs are:
    Case: bainbridge-flood-response-plan - (2)

The Selected RP is:
   bainbridge-flood-response-plan

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Adapter Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

**Problem: filler-does-not-exist
**Old-Value: (relief-agency, "salvation_army")
```

```
Created knowledge goal: "KG-1"

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
We first attempt to find an adaptation case that can solve this
problem.  Currently no adaptation cases exist.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

No similar adaptation case available

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
We fall back on memory search to find a substitution for the
problem filler.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Trying RBR Adaptation
   Solution Found: "red_cross"

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This was a simple adaptation to solve (since both terms are close
to one another in memory). The solution to the problem along with
the reasoning trace are stored as an adaptation case.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Storing AC:
  Adaptation Case:
  Name: adaptation-case-1,
     Ops: 58   Nodes: 33  CPUTime: 850  RealTime: 880
((has-abstraction? ("organization" noun 1 ())))

**Problem: physically-unavailable
**Old-Value: (police-force, "police")

Created knowledge goal: "KG-2"

No similar adaptation case available

Trying RBR Adaptation
   Solution Found: "military_personnel"

Storing AC:
```

```
  Adaptation Case:
  Name: adaptation-case-2,
       Ops: 66   Nodes: 34  CPUTime: 1020  RealTime: 1080
   ((has-abstraction? ("group" noun 1 ())))

Adaptation summary
Type: flood  Location: ("oregon" noun 1 ())
-------------------------------------------------------------
 slot       problem     solvedby      solution  ops nodes time
-------------------------------------------------------------
 police-for phys-unav   "military_pe" rbr       66  34    1020
 relief-age filler-dne  "red_cross"   rbr       58  33    850

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Once all problems are solved. The response plan is returned to
the user and stored for future use.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

#####################################################################
Storing oregon-flood-response-plan
#####################################################################

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
A second flood disaster is entered into the system
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Current disaster is
  (flood ((city ("wheeling" noun 1 ()))
     (state ("west_virginia" noun 1 ()))
  (country ("united_states" noun 1 ()))
  (continent ("north_america" noun 1 ()))))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Retriever Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This time there are two response plan cases to be considered for
reuse.x
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
Selected 2 cases for similarity assessment

The following cases have been ranked:
    Case: oregon-flood-response-plan (3)
    Case: bainbridge-flood-response-plan (3)

Using adaptative similarity techniques (if available)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Each plan is evaluated for problem fillers
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Evaluating the bainbridge-flood-response-plan

------------------------------------------------------------
slot           filler             problem  type          cost
-------------- ------------------------------------------------
 location      "wheeling"          no       n/a           0
 relief-agency "salvation_army"    yes      context-mism  ?
 beneficiary   "resident"          no       n/a           0
 volunteers    "volunteer"         yes      context-mism  ?
 transport     "motorboat"         no       n/a           0
 shelter       "school"            no       n/a           0
 build-object  "sandbag"           no       n/a           0
 police-force  "police"            yes      lack-of-ac    ?

Evaluating the oregon-flood-response-plan
------------------------------------------------------------
slot            filler             problem  type         cost
--------------------------------------------------------------
location        "wheeling"          no       n/a          0
relief-agency   "red_cross"         yes      physically-u ?
beneficiary     "resident"          no       n/a          0
volunteers      "volunteer"         yes      context-mism ?
transport       "motorboat"         no       n/a          0
shelter         "school"            no       n/a          0
build-object    "sandbag"           no       n/a          0
police-force    "military_personn"  no       n/a          0
```

```
Best Adaptation Case:
  Name: adaptation-case-2,
    Ops: 66   Nodes: 34  CPUTime: 1020  RealTime: 1080

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Each plan is scored based on the current similarity criteria. The
plan with the lowest score (and thus the most similar) is
selected for reuse
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

The problem based costs are:
    Case: oregon-flood-response-plan - (2)
    Case: bainbridge-flood-response-plan - (3)

The Selected RP is:
   oregon-flood-response-plan

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Adapter Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

**Problem: physically-unavailable
**Old-Value: (relief-agency, "red_cross")

Created knowledge goal: "KG-3"

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
For the first problem, it is a relief-agency
physically-unavailable problem again and we have an adaptation
case designed for this specific instance which was retrieved. The
case is reused and solves the problem. The new problema and
solution are also stored as an adaptation case.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Trying CBR Adaptation

Using Adaptation Case:
  Name: adaptation-case-2,
    Ops: 66   Nodes: 34  CPUTime: 1020  RealTime: 1080
```

```
Storing AC:
  Adaptation Case:
  Name: adaptation-case-3,
    Ops: 62   Nodes: 97  CPUTime: 730  RealTime: 880

**Problem: context-mismatch
**Old-Value: (volunteers, "volunteer")

Created knowledge goal: "KG-4"

No similar adaptation case available

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Sometimes neither case based adaptation nor rule based adaptation
is successful at finding a solution. In these cases the system
resorts to a manual adaptation, where the user suggests different
constraints or approaches to searching for a solution
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Trying RBR Adaptation
  No solutions found.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Manual Adapter Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
The user goes through a menu driven list of choices to adapt the
filler.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Knowledge Goal:
  Name: "KG-4"
  Slot: volunteers,
  CurrentNode: ("volunteer" noun 2 ())
  Limit: 2000, Transformation: substitution
  Problem: context-mismatch,
  Disaster: (flood ((city ("wheeling" noun 1 ()))))
  Constraints:
    ((has-abstraction? ("person" noun 1 ())))
```

```
Select a modification to perform
    1) Add a constraint
    2) Delete a constraint
    3) Replace a constraint
    4) Continue as is
    5) Manual Modification
Your Choice ==> 1
What type of constraint to add?
  1) has-abstraction?
  2) has-part
  3) is-member
  4) has-specification
Your Choice --> 1

Enter the known abstraction: "force"
Senses of "force" are:
  Sense 1: force,
  Sense 2: force,
  Sense 3: force, forcefulness, strength,
  Sense 4: force, personnel,
  Sense 5: military_unit, military_force, force,
  Sense 6: violence, force,
  Sense 7: power, force,
  Sense 8: force,
  Sense 9: effect, force,
Which sense of the concept: 4
Any more modifications? y

Knowledge Goal:
  Name: "KG-4"
  Slot: volunteers,
  CurrentNode: ("volunteer" noun 2 ())
  Limit: 2000, Transformation: substitution
  Problem: context-mismatch,
  Disaster: (flood ((city ("wheeling" noun 1 ()))))
  Constraints:
    ((has-abstraction? ("force" noun 4 ()))
     (has-abstraction? ("person" noun 1 ())))
```

```
Select a modification to perform
    1) Add a constraint
    2) Delete a constraint
    3) Replace a constraint
    4) Continue as is
    5) Manual Modification
Your Choice ==> 2
Select a constraint to delete
  1) (has-abstraction? ("force" noun 4 ()))
  2) (has-abstraction? ("person" noun 1 ()))
Your Choice --> 2
Any more modifications? n

Storing AC:
  Adaptation Case:
  Name: adaptation-case-4,
    Ops: 4006   Nodes: 2008  CPUTime: 61860  RealTime: 62430

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Adaptation summary
Type: flood  Location: ("wheeling" noun 1 ())
-----------------------------------------------------------------
 slot       problem     solvedby      solution  ops nodes time
-----------------------------------------------------------------
 volunteers context-mis "military_pe" manual    4006 2008 61860
 relief-ag  physically- "military_pe" cbr       62   97   730
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Once the problems are solved the new case is stored.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


#####################################################################
Storing wheeling-flood-response-plan
#####################################################################

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Another problem is entered into the system. It is also a flood
example.
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Current disaster is
(flood ((city "izmir")
(country ("turkey" noun 2 ()))
(continent ("europe" noun 1 ()))))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Retriever Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
There are no 3 response plan cases that can be retrieved. We need
to select the most similar one.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Selected 3 cases for similarity assessment

The following cases have been ranked:
    Case: wheeling-flood-response-plan (5)
    Case: oregon-flood-response-plan (5)
    Case: bainbridge-flood-response-plan (5)

Using adaptative similarity techniques (if available)
Evaluating the oregon-flood-response-plan
-------------------------------------------------------------
slot            filler             problem  type          cost
-------------------------------------------------------------
location        "izmir"            no       n/a            0
relief-agency   "red_cross"        no       n/a            0
beneficiary     "resident"         no       n/a            0
volunteers      "volunteer"        no       n/a            0
transport       "motorboat"        yes      means-of-lack- ?
shelter         "school"           yes      innapropriate- ?
build-object    "sandbag"          no       n/a            0
police-force    "military_personn" yes      innapropriate- ?

Evaluating the bainbridge-flood-response-plan
-------------------------------------------------------------
slot            filler             problem  type          cost
```

```
------------------------------------------------------------
location        "izmir"            no       n/a            0
relief-agency   "salvation_army"   yes      filler-does-no ?
beneficiary     "resident"         no       n/a            0
volunteers      "volunteer"        no       n/a            0
transport       "motorboat"        yes      means-of-lack- ?
shelter         "school"           yes      innapropriate- ?
build-object    "sandbag"          no       n/a            0
police-force    "police"           no       n/a            0

Best Adaptation Case:
  Name: adaptation-case-1,
    Ops: 58   Nodes: 33  CPUTime: 850  RealTime: 880

Evaluating the wheeling-flood-response-plan
------------------------------------------------------------
slot            filler             problem  type          cost
------------------------------------------------------------
location        "izmir"            no        n/a           0
relief-agency   "military_personn" yes       innapropriate ?
beneficiary     "resident"         no        n/a           0
volunteers      "military_personn" yes       situation-mis ?
transport       "motorboat"        yes       means-of-lack ?
shelter         "school"           yes       innapropriat  ?
build-object    "sandbag"          no        n/a           0
police-force    "military_personn" yes       innapropriate ?


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
In this example, similarity costs are based exclusively on the
number problems identified in the plan. Thus in this situation,
two plans tie with 3 problems each, and a random one of the two
is selected.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

The problem based costs are:
    Case: wheeling-flood-response-plan - (5)
    Case: bainbridge-flood-response-plan - (3)
    Case: oregon-flood-response-plan - (3)
```

```
The Selected RP is:
   oregon-flood-response-plan

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Adapter Module:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
As before we attempt to adapt each problem filler to an
acceptable filler using case based, rule based and finally manual
methods as needed.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

**Problem: means-of-lack-of-access
**Old-Value: (transport, "motorboat")

Created knowledge goal: "KG-5"

No similar adaptation case available

Trying RBR Adaptation
   Solution Found: "truck"

Storing AC:
  Adaptation Case:
  Name: adaptation-case-5,
    Ops: 150   Nodes: 78  CPUTime: 2350  RealTime: 2450
((has-abstraction? ("vehicle" noun 1 ())))

**Problem: innapropriate-context
**Old-Value: (shelter, "school")

Created knowledge goal: "KG-6"

No similar adaptation case available

Trying RBR Adaptation
   Solution Found: "tent"

Storing AC:
```

```
  Adaptation Case:
  Name: adaptation-case-6,
    Ops: 288   Nodes: 151  CPUTime: 4650  RealTime: 4720
((has-abstraction? ("structure" noun 1 ())))

**Problem: innapropriate-context
**Old-Value: (police-force, "military_personnel")

Created knowledge goal: "KG-7"

No similar adaptation case available

Trying RBR Adaptation
   Solution Found: "police"

Storing AC:
  Adaptation Case:
  Name: adaptation-case-7,
    Ops: 64   Nodes: 33  CPUTime: 1010  RealTime: 1060
((has-abstraction? ("group" noun 1 ())))

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Adaptation summary
Type: flood  Location: "izmir"
-------------------------------------------------------------
 slot       problem     solvedby      solution  ops nodes time
-------------------------------------------------------------
 police-for innapropria "police"      rbr       64  33    1010
 shelter    innapropria "tent"        rbr       288 151   4650
 transport  means-of-la "truck"       rbr       150 78    2350
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

#################################################################
Storing izmir-flood-response-plan
#################################################################
```

Bibliography

Aamodt, A. & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, *7*(1), 39–52.

Aha, D. & Branting, K. (1995). Stratified case-based reasoning: reusing hierarchical problem solving episodes. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 384–390 San Francisco. Morgan Kaufmann.

Allemang, D. (1993). Review of the first European workshop on case based reasoning EWCBR-93. *Case-Based Reasoning Newsletter*, *2*(3). Electronic newsletter published by the special interest group AK-CBR of the German Society for Computer Science.

Alterman, R. (1988). Adaptive planning. *Cognitive Science*, *12*, 393–422.

Anderson, J. (1983). Acquisition of proof skills in geometry. In Michalski, R., Carbonell, J., & Mitchell, T. (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Tioga, Cambridge, MA.

Barletta, R. (1994). A hybrid indexing and retrieval strategy for advisory CBR systems built with ReMind. In *Proceedings of the Second European Workshop on Case-Based Reasoning*, pp. 49–58 Chantilly, France.

Barletta, R. & Mark, W. (1988). Explanation-based indexing of cases. In Kolodner, J. (Ed.), *Proceedings of a Workshop on Case-Based Reasoning*, pp. 50–60 Palo Alto. DARPA, Morgan Kaufmann.

Bell, B., Kedar, S., & Bareiss, R. (1994). Interactive model-driven case adaptation for instructional software design. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 33–38 Atlanta, GA.

Berger, J. (1995). Using past repair episodes. Unpublished manuscript.

Berger, J. & Hammond, K. (1991). ROENTGEN: a memory-based approach to radiation therapy treatment. In Bareiss, R. (Ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, pp. 203–214 San Mateo. DARPA, Morgan Kaufmann.

Bhatta, S. & Goel, A. (1996). From design experiences to generic mechanisms: model-based learning in analogical design. *Artificial Intelligence for Engineering Design*, *10*, 131–136.

Birnbaum, L., Collins, G., Brand, M., Freed, M., Krulwich, B., & Pryor, L. (1991). A model-based approach to the construction of adaptive case-based planning systems. In Bareiss, R. (Ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, pp. 215–224 San Mateo. Morgan Kaufmann.

Carbonell, J. (1983). Learning by analogy: formulating and generalizing plans from past experience. In Michalski, R., Carbonell, J., & Mitchell, T. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, pp. 137–162. Tioga, Cambridge, MA.

Carbonell, J. (1986). Derivational analogy: a theory of reconstructive problem solving and expertise acquisition. In Michalski, R., Carbonell, J., & Mitchell, T. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. 2, pp. 371–392. Morgan Kaufmann, Los Altos, CA.

Cheetham, W. & Graf, J. (1997). Case-based reasoning in color matching. In *Proceedings of the Second International Conference on Case-Based Reasoning*, pp. 1–12 Berlin. Springer Verlag.

Chi, M., Feltovich, P., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, *5*(2), 121–153.

Cox, M. (1994). Machines that forget: learning from retrieval failure of mis-indexed explanations. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 225–230 Atlanta, GA.

Cullingford, R. (1978). *Script Application: Computer Understanding of Newspaper Stories.* Ph.D. thesis, Yale University. Computer Science Department Technical Report 116.

de Silva Garza, A. G. & Maher, M. L. (1999). An evolutionary approach to case adaptation. In *Case Based Reasoning Research and Development*, pp. 162–172 Berlin. Springer Verlag.

Fellbaum, C. (Ed.). (1998). *Wordnet: An Electronic Lexical Database.* MIT Press, Cambridge, MA.

Firby, R. (1989). *Adaptive Execution in Complex Dynamic Worlds.* Ph.D. thesis, Yale University. Computer Science Department TR 672.

Fox, S. & Leake, D. (1995). Using introspective reasoning to refine indexing. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 391–397 San Francisco, CA. Morgan Kaufmann.

Gervasio, M., Iba, W., Langley, P., & Sage, S. (1998). Interactive adaptation for crisis response. In *Proceedings of the AIPS-98 Workshop on Interactive and Collaborative Planning*, pp. 29–36 Pittburgh, PA.

Goel, A., Ali, K., Donnellan, M., de Silva Garza, A., & Callantine, T. (1994). Multistrategy adaptive path planning. *IEEE Expert*, *9*(6), 57–65.

Goel, A. & Chandrasekaran, B. (1989). Use of device models in adaptation of design cases. In Hammond, K. (Ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, pp. 100–109 San Mateo. DARPA, Morgan Kaufmann.

Goel, A., Garza, A., Grue, N., Murdock, J., Recker, M., & Govindaraj, T. (1996). Explanatory interface in interactive design environments. In *Artificial Intelligence in Design '96*. Kluwer.

Gomez-Albarran, M. & et al., P. G.-C. (1999). Modelling the cbr life cycle using description logics. In *Case Based Reasoning Research and Development*, pp. 147–161 Berlin. Springer Verlag.

Hammond, K. (1989). *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, San Diego.

Hanney, K. (1997). Learning adaptation rules from cases. Master's thesis, Trinity College, Dublin.

Hanney, K. & Keane, M. (1997). The adaptation knowledge bottleneck: how to ease it by learning from cases. In *Proceedings of the Second International Conference on Case-Based Reasoning* Berlin. Springer Verlag.

Hunter, L. (1989). *Gaining Expertise Through Experience*. Ph.D. thesis, Yale University. Computer Science Department Technical Report 678.

Kass, A. (1994). Tweaker: adapting old explanations to new situations. In Schank, R., Riesbeck, C., & Kass, A. (Eds.), *Inside Case-Based Explanation*, chap. 8, pp. 263–295. Lawrence Erlbaum.

Keane, M. (1994). Adaptation as a selection constraint on analogical mapping. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 490–495 Atlanta, GA.

Kennedy, A. (1995). Using a domain-independent introspection mechanism to improve memory search. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*, pp. 72–78 Stanford, CA. AAAI Press. Technical Report WS-95-05.

Kitano, H. & Shimazu, H. (1996). The experience sharing architecture: a case study in corporate-wide case-based software quality control. In Leake, D. (Ed.), *Case-Based*

Reasoning: Experiences, Lessons, and Future Directions, pp. 235–268. AAAI Press, Menlo Park, CA.

Kolodner, J. (1984). *Retrieval and Organizational Strategies in Conceptual Memory.* Lawrence Erlbaum, Hillsdale, NJ.

Kolodner, J. (1988a). Extending problem solver capabilities through case-based inference. In Kolodner, J. (Ed.), *Proceedings of a Workshop on Case-Based Reasoning*, pp. 21–30B Palo Alto. DARPA, Morgan Kaufmann.

Kolodner, J. (1988b). Retrieving events from a case memory: a parallel implementation. In Kolodner, J. (Ed.), *Proceedings of a Workshop on Case-Based Reasoning*, pp. 233–249 Palo Alto. DARPA, Morgan Kaufmann.

Kolodner, J. (1991). Improving human decision making through case-based decision aiding. *The AI Magazine*, *12*(2), 52–68.

Kolodner, J. (1993). *Case-Based Reasoning.* Morgan Kaufmann, San Mateo, CA.

Kolodner, J. & Leake, D. (1996). A tutorial introduction to case-based reasoning. In Leake, D. (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, pp. 31–65. AAAI Press, Menlo Park, CA.

Koton, P. (1988). Reasoning about evidence in causal explanations. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 256–261 San Mateo, CA. AAAI, Morgan Kaufmann.

Leake, D. (1992a). Constructive similarity assessment: using stored cases to define new situations. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pp. 313–318 Hillsdale, NJ. Lawrence Erlbaum.

Leake, D. (1992b). *Evaluating Explanations: A Content Theory.* Lawrence Erlbaum, Hillsdale, NJ.

Leake, D. (1994a). Towards a computer model of memory search strategy learning. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 549–554 Hillsdale, NJ. Lawrence Erlbaum.

Leake, D. (1994b). Workshop report: the AAAI-93 workshop on case-based reasoning. *The AI Magazine*, *15*(1), 63–64.

Leake, D. (1995a). Combining rules and cases to learn case adaptation. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pp. 84–89 Mahwah, NJ. Lawrence Erlbaum.

Leake, D. (1995b). Representing self-knowledge for introspection about memory search. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*, pp. 84–88 Stanford, CA. AAAI Press.

Leake, D. (Ed.). (1996a). *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press, Menlo Park, CA.

Leake, D. (1996b). CBR in context: the present and future. In Leake, D. (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, pp. 3–30. AAAI Press, Menlo Park, CA.

Leake, D., Kinley, A., & Wilson, D. (1996). Acquiring case adaptation knowledge: a hybrid approach. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 684–689 Menlo Park, CA. AAAI Press.

Leake, D., Kinley, A., & Wilson, D. (1997). A case study of case-based CBR. In *Proceedings of the Second International Conference on Case-Based Reasoning*, pp. 371–382 Berlin. Springer Verlag.

Lenat, D. & Guha, R. (1990). *Building large knowledge-based systems : representation and inference in the Cyc project*. Addison-Wesley, Reading, MA.

Marir, F. (1995). Representing and indexing building refurbishment cases for multiple retrieval of adaptable pieces of cases. In *Proceedings of First International Conference on Case-Based Reasoning*, pp. 55–66 Sesimbra, Portugal.

Mark, W., Simoudis, E., & Hinkle, D. (1996). Case-based reasoning: expectations and results. In Leake, D. (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, pp. 269–294. AAAI Press, Menlo Park, CA.

McSherry, D. (1998). An adaptation heuristic for case-based estimation. In *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, pp. 184–195 Dublin, Ireland.

Minsky, M. (1975). A framework for representing knowledge. In Winston, P. (Ed.), *The Psychology of Computer Vision*, chap. 6, pp. 211–277. McGraw-Hill, New York.

Minton, S. (1988). *Learning Search Control Knowledge: An Explanation-Based Approach*. Kluwer Academic Publishers, Boston.

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: a unifying view. *Machine Learning*, *1*(1), 47–80.

Oehlmann, R. (1995). Metacognitive adaptation: regulating the plan transformation process. In *Proceedings of the Fall Symposium on Adaptation of Knowledge for Reuse*. AAAI.

Oehlmann, R., Sleeman, D., & Edwards, P. (1993). Learning plan transformations from self-questions: a memory-based approach. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 520–525 Washington, DC. AAAI.

Purvis, L. & Athalye, S. (1997). Towards improving case adaptability with a genetic algorithm. In *Proceedings of the Second International Conference on Case-Based Reasoning*, pp. 403–412 Berlin. Springer Verlag.

Ram, A. & Francis, A. (1996). Multi-plan retrieval and adaptation in an experience-based agent. In Leake, D. (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press, Menlo Park, CA.

Ram, A. (1987). AQUA: asking questions and understanding answers. In *Proceedings of the Sixth Annual National Conference on Artificial Intelligence*, pp. 312–316 Seattle, WA. Morgan Kaufmann.

Redmond, M. (1992). *Learning by Observing and Understanding Expert Problem Solving*. Ph.D. thesis, College of Computing, Georgia Institute of Technology. Technical report GIT-CC-92/43.

Richter, M. (1995). The knowledge contained in similarity measures. Invited talk, the First International Conference on Case-Based Reasoning, Sesimbra, Portugal.

Riesbeck, C. (1996). What next? The future of CBR in postmodern AI. In Leake, D. (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press, Menlo Park, CA.

Rissland, E., Skalak, D., & Friedman, M. (1994). Heuristic harvesting of information for case-based argument. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 36–43 Seattle, WA. AAAI.

Rosenthal, U., Charles, M., & Hart, P. (Eds.). (1989). *Coping with crises: The management of disasters, riots, and terrorism*. C.C. Thomas, Springfield, IL.

Ross, B. (1989). Some psychological results on case-based reasoning. In Hammond, K. (Ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, pp. 144–147 San Mateo. Morgan Kaufmann.

Rouse, W. & Hunt, R. (1984). Human problem solving in fault diagnosis tasks. *Advances in Man-Machine Systems Research*, *1*, 195–221.

Schank, R. (1975). *Conceptual Information Processing*, Vol. 3 of *Fundamental Studies in Computer Science*. North-Holland, Amsterdam.

Schank, R. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, Cambridge, England.

Segre, A. (1987). On the operationality/generality tradeoff in explanation-based learning. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* Milan, Italy. IJCAI.

Shiu, S., Sun, C., Wang, X., & Yeung, D. (2000). Maintaining case-based reasoning systems using fuzzy decisions trees. In *Proceedings of the Fifth European Workshop on Case-Based Reasoning*, pp. 285–296 Trento, Italy.

Sinha, A. (1994). Providing design assistance: a case-based approach. Artificial intelligence in management laboratory PITT-AIM-47, University of Pittsburgh.

Smith, I., Lottaz, C., & Faltings, B. (1995). Spatial composition using cases: IDIOM. In *Proceedings of First International Conference on Case-Based Reasoning*, pp. 88–97 Berlin. Springer Verlag.

Smyth, B. & Keane, M. (1994). Retrieving adaptable cases: the role of adaptation knowledge in case retrieval. In Wess, S., Althoff, K., & Richter, M. (Eds.), *Topics in Case-Based Reasoning*, pp. 209–220 Berlin. Springer Verlag.

Smyth, B. & Keane, M. (1996). Design à la Déjà Vu: reducing the adaptation overhead. In Leake, D. (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* AAAI Press, Menlo Park, CA.

Smyth, B. & Keane, M. (1998). Adaptation-guided retrieval: questioning the similarity assumption in reasoning.. *Artificial Intelligence*, *102*(2), 249–293.

Suzuki, H., Ohnishi, H., & Shigermasu, K. (1992). Goal-directed processes in similarity judgment. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pp. 343–348 Bloomington, IN. Lawrence Erlbaum.

Sycara, K. (1988). Using case-based reasoning for plan adaptation and repair. In Kolodner, J. (Ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, pp. 425–434 San Mateo, CA. Morgan Kaufmann.

Veloso, M. (1991). Variable-precision case retrieval in analogical problem solving. In Bareiss, R. (Ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, pp. 93–106 San Mateo. DARPA, Morgan Kaufmann.

Veloso, M. (1994). *Planning and Learning by Analogical Reasoning.* Springer Verlag, Berlin.

Veloso, M. & Carbonell, J. (1994). Case-based reasoning in PRODIGY. In Michalski, R. & Tecuci, G. (Eds.), *Machine Learning: A Multistrategy Approach*, chap. 20, pp. 523–548. Morgan Kaufmann.

Wilke, W., Vollrath, I., Althoff, K.-D., & Bergmann, R. (1997). A framework for learning adaptation knowedge based on knowledge light approaches. In *Proceedings of the Fifth German Workshop on Case-Based Reasoning.*