Reactive and Proactive approaches for Introspective CBR

Mehmet Oğuz Mülâyim

Memòria del Treball de Recerca del Programa de Doctorat en Informàtica realitzat a

IIIA, Institut d'Investigació en Intel.ligència Artificial
CSIC, Consejo Superior de Investigaciones Científicas

Dirigit per Josep Lluís Arcos

Escola Tècnica Superior d'Enginyeries
Departament de Ciències de la Computació

Universitat Autònoma de Barcelona

Bellaterra, octubre de 2008

Abstract

This work investigates applying introspective reasoning to improve the performance of Case-Based Reasoning (CBR) systems, in both reactive and proactive fashion, by guiding learning to improve how a CBR system applies its cases and by identifying possible future system deficiencies. First we present our reactive approach, a new introspective reasoning model which enables CBR systems to autonomously learn to improve multiple facets of their reasoning processes in response to poor quality solutions. We illustrate our model' s benefits with experimental results from tests in an industrial design application. Then as for our proactive approach, we introduce a novel method for identifying regions in a case-base where the system gives low confidence solutions to possible future problems. Experimentation is provided for Zoology and Robo-Soccer domains and we argue how encountered regions of dubiosity help us to analyze the case-bases of a given CBR system.

Acknowledgements

First of all, I would like to thank to Dr. Josep Puyol-Gruart for his kindest concern and dedicated efforts that kept me alive throughout all the bureacratic procedures that I had to overcome before flying to Barcelona. Without his guidance I wouldn' t be a part of neither the university nor the IIIA.

I owe many thanks to Dr. Josep Lluís Arcos, for his friendly and encouraging tutoring and for the inspiring discussions that we had along the thorny path of scientific investigation. I' ve really learned a lot from him.

And finally, a hearty thanks to all the IIIA folks for all those days that I have spent with you. It has been a privilege for me to be and feel like a member of this big, warm and joyful family.

This research has been partially supported by the Spanish Ministry of Education and Science project MID-CBR (TIN2006-15140-C03-01), EU-FEDER funds, and by the Generalitat of Catalunya (2005-SGR-00093).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The application of artificial intelligence (AI) technologies to real-world problems has shown that it is difficult for developers to anticipate all possible eventualities. Especially in long-lived systems, changing circumstances may require changes not only to domain knowledge but also to the reasoning process which brings it to bear. One way to tackle this issue is to endow the sytems with *introspective reasoning*, metareasoning by a system about its own internal reasoning processes.

Introspective reasoning has been an active field of research in psychology, artificial intelligence and cognitive science for a long time. Metareasoning techniques provide a promising basis for self-improving systems (see Anderson and Oates (2007), Cox (2005) for recent reviews). As described by Cox and Raja (2007), the metareasoning approach incorporates a meta-reasoning layer, with monitoring and control capabilities over the reasoning process, to adjust that reasoning process as needed. Metareasoning layer can make these adjustments in a *reactive* way in response to detected reasoning failures or it may try to avoid such failures by anticipating them in a *proactive* manner.

This work specifically investigates applying introspective reasoning to improve the performance of Case-Based Reasoning systems, in both reactive and proactive fashion, by guiding learning to improve how a case-based reasoning system applies its cases and by detecting possible future system deficiencies.

Case-based reasoning (CBR) is a problem-solving methodology that exploits prior experiences when solving new problems, retrieving relevantly similar cases and adapting them to fit new needs (for an overview and survey, see Lopez de Mantaras *et al.* (2005)). Many CBR systems store each newly-solved problem and its solution as a new case for future use, enabling them to continuously improve their case knowledge. Nevertheless, the success of a CBR system depends not only on its cases, but also on its ability to use those cases appropriately in new situations (which depends on the similarity measure and the case adaptation mechanisms). Consequently, it is desirable for CBR systems to improve the processes by which they bring their cases to bear. Therefore, a thorough introspection carried out by a CBR system should take into account both its reasoning processes and its case-base.

As it will be introduced shortly in the next chapter, previous research on introspective CBR has shown that metareasoning can enable a CBR system to learn by refining its own reasoning process. That work has tended to apply the introspective approach only to a single aspect of the CBR system, for example, to adjust the indices used for retrieval or learning how to improve case adaptation. However, this approach does not always guarantee achieving an overall improvement of the system since reasoning components are closely interconnected and focusing on a single process may miss opportunities (Leake and Wilson 2008). Thus, developing an introspective reasoning model that enables CBR systems to autonomously learn to improve multiple facets of their reasoning processes has been the motivation for the first part of our research which is presented in Chapter 3. This part forms our reactive approach.

On the other hand, when we look at the research on improving the performance by maintaining the case-base (e.g. by removing redundant cases) we see the work which is known as Case-Base Maintenance. The common assumption of case-base maintenance techniques has been that analysis of the cases provided in the case-base is a good approach for estimating the performance of the system for future cases. This assumption is known as the *representativeness assumption*. Nevertheless, new problems are expected to be slightly different from the existing cases. Thus, the possibility of systematically assessing the performance of a system in a set of problems different from the existing cases becomes an interesting issue. Hence, finding a method to analyze the performance of a CBR system against possible future problems has become the motivation behind the second part of our research which is presented in Chapter 4. Consequently, this part forms our proactive approach.

Before going into the details of our research we introduce Related Work as a background. Later we present our two approaches, then we discuss the outcomes in the Conlusions part where we also share our ideas for future directions.

Chapter 2

Related Work

As our work is based on CBR, we start with a brief definition of the reasoning cycle that a typical CBR system follows to solve a new problem and to learn from that experience.

The case-based reasoning process consists of four steps (see Figure 2.1):

1. *Case retrieval/similarity assessment*, which determines which cases address problems most similar to the current problem, to identify them as starting points for solving the new problem,

2. *Case adaptation*, which forms a new solution by adapting/combining solutions of the retrieved problems,

3. *Case revision*, which evaluates and adjusts the adapted solution, and

4. *Case retention*, in which the system learns from the situation by storing the problem and its solution as a new case for future use.

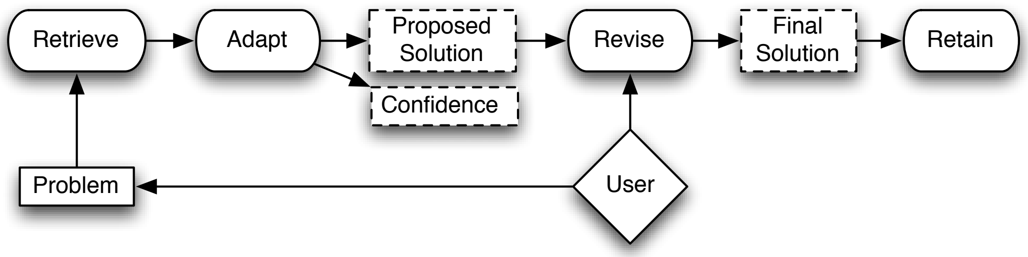

Figure 2.1: Case-Based Reasoning cycle with Confidence

In this document, we avoid further discussion of CBR since detailed information can be found elsewhere (Kolodner (1993), Aamodt and Plaza (1994), Leake (1996a), Lopez de Mantaras *et al.* (2005)).

Some CBR systems also attach *confidence* values to their solutions (see Figure 2.1), stating how confident the system is about a solution it proposes for a given problem (where a low confidence value points to a possible inaccurate solution). Cheetham (2000) shows the importance of the availability of such a measure for improving the usefulness of CBR systems. Cheetham and Price (2004) introduce a list of confidence indicators in CBR systems and give measures of solution accuracy. Delany *et al.* (2005) extend the research proposing a method that aggregates a collection of confidence metrics that can be used when a single confidence measure is not effective, as it is the case for spam filtering domain.

In general, confidence measures assist the user in determining when to trust a proposed solution. Nevertheless, since these measures provide no explanations of their assessments whatsoever, they are not very helpful for revealing the origin of an inaccurate solution, making their use difficult to guide repairs.

The rest of this chapter is divided into two parts. The first part gives a brief summary of research on Introspective CBR as a background for Chapter 3. The second part introduces some highlights of previous research on Case-Base Maintenance techniques as a background for Chapter 4.

2.1 Introspective CBR

Birnbaum *et al.* (1991) first proposed the use of self-models within case-based reasoning. Work by Cox and Ram (1999) develops a set of general approaches to introspective reasoning and learning, automatically selecting the appropriate learning algorithms when reasoning failures arise. This work defines a taxonomy of causes of reasoning failures and proposes a taxonomy of learning goals, used for analyzing the traces of reasoning failures and responding to them. In their work case-based reasoning is a vehicle for supporting introspective reasoning: CBR is used to explain reasoning failures and generate learning goals.

A number of studies apply introspective approaches to improve the performance of CBR systems. Leake (1996b) identifies the knowledge sources a CBR system uses in its reasoning process and the required self-knowledge about these sources. Moreover, he provides examples of refinement of retrieval knowledge using model-based reasoning and of acquisition of adaptation knowledge by search plans. Fox and Leake (2001) developed a system inspired by Birnbaum et al's proposal to refine index selection for case-based reasoners. Fox and Leake's work developed a declarative model for describing the expectations for correct reasoning behavior, and applied that model to detecting and diagnosing reasoning failures. When the introspective reasoner is able to identify the feature that caused the failure, the system's memory is re-indexed, resulting in a significant performance improvement. The DIAL system (Leake, Kinley and Wilson 1995) improves case adaptation using introspection. This research focuses on improving the performance of the system by storing the traces of successful adaptation transformations and memory search paths for future reuse. Likewise, Craw (2006) proposes an introspective learning approach for acquiring adaptation knowledge, making it closely related to our work. However, a key difference is that their learning step uses the accumulated case-base as training data for adaptation learning, in contrast to our approach of incrementally refining adaptation knowledge in response to failures for individual problems.

Arcos (2004) presents a CBR approach for improving solution quality in evolving environments. His work focuses on improving the quality of solutions for problems which arise only occasionally, by analyzing how the solutions of more typical problems change over time. Arcos's algorithm improves the performance of the system by exploiting the neighborhoods in the solution space but, unlike the model presented in this work at Chapter 3, learns only from success.

The REM reasoning shell (Murdock and Goel 2008) presents a meta-case-based reasoning technique for self-adaptation. The goal of REM is the design of agents able to solve new tasks by adapting their own reasoning processes. Meta-case-based reasoning is used for generating new task-method decomposition plans. Because the goal in REM is the assembly of CBR reasoning components, the meta-model is focused on describing the components in terms of their requirements and their effects. In contrast, our model is focused on describing the expected correct properties of the components and their possible reasoning failures.

2.2 Case-Base & Case-Based Reasoner Maintenance

Quoting from Leake and Wilson (1998) :

> "Case-base maintenance implements policies for revising the organization or contents (representation, domain content, accounting information, or implementation) of the case-base in order to facilitate future reasoning for a particular set of performance objectives."(p.2)

Case-Base Maintenance (CBM) has become an important issue when CBR applications have started to deal with larger case libraries with from thousands to millions of cases.

The first problem the CBR community faced was the so-called *Utility Problem* (Francis and Ram 1993, Smyth and Cunningham 1996). This problem occurs when the retrieval time increases as the case-base grows and eventually system efficiency degrades drastically. So, most of the existing CBM techniques have primarily focused on policies for controlling case-base growth. Smyth and Keane (1995) present a deletion policy which provides a means of ordering cases for deletion in terms of their *competence* (the range of problems a CBR sytem can solve) contributions.

Smyth and McKenna (1999) propose a technique which ensure the construction of initially competent case-bases by selecting training instances which are likey to contribute to performance. Smyth and McKenna (2001) give a comprehensive survey of their work on competence models and they show how these models allow them to build a competence map of a case-base which can be used in competence-guided editing and retrieval.

More recent research by Massie, Craw and Wiratunga (2007) introduces a complexity measure for highlighting areas of uncertainty within the problem space. Using this measure they identify and remove noisy (erroneous) and boundary cases (cases near class boundaries) to improve system *accuracy*.

These techniques proved useful for improving the performance of CBR systems. But they only focus on the case-base itself while they generally do not give insight about the CBR's reasoning processes (such as adaptation mechanisms). Wilson and Leake (2001) propose a framework for Case-Based Reasoner Maintenance (CBRM) for describing maintenance systems beyond CBM. The framework presents basic dimensions of CBRM policies in terms of data collection (e.g. synchronic vs diachronic or introspective vs nonintrospective), being proactive vs reactive, scope (broad vs narrow), integration (on-line, off-line), and timing (e.g. periodic, conditional, ad hoc).

According to this framework we may categorize our work as follows:

- Data Collection:

  Both our approaches are *synchronic* since they use snapshot information of the case-base instead of collecting data over time. Moreover, they are *introspective* since both of them analyze the internal state of the case-based reasoner system;

- Proactive vs Reactive:

  The model at Chapter 3 is *reactive* because the learning is triggered in response to a reasoning failure.

The method at Chapter 4 is *proactive* because it analyzes the case-base to anticipate possible future failures.

- Scope:

  The model at Chapter 3 has a *narrow* scope, because the applicability of the changes are limited to the neighborhood of the current problem.

  The method at Chapter 4 has a *broad* scope because it takes into account the whole case-base in search for future problems.

- Integration:

  Both methods are *off-line* since they do not intervene the active reasoning process.

- Timing:

  Our model at Chapter 3 can be considered as *conditional* because the learning is triggered by a poor quality solution.

  Our method at Chapter 4 is *ad-hoc* since, at the moment, it is initiated by the user.

Chapter 3

Using Introspective Reasoning to Improve CBR System Performance

This chapter describes our approach in which an introspective reasoner monitors the CBR process with the goal of adjusting the retrieval and reuse strategies of the system to improve solution quality. Novel aspects of this approach, compared to previous work on introspective reasoning for CBR, include that it applies a unified model for improving the two main stages of the CBR process, that a single failure may prompt multiple forms of learning, and that it performs internal tests to empirically assess the value of changes proposed by the introspective reasoner, to determine which ones should be retained.

The next section presents a detailed description of our approach and its implementation. The approach has been evaluated on problems from a fielded industrial application for design of pollution control equipment, for which we provide results in the following section. Before concluding the chapter, we put in context our model with respect to the metareasoning models discussed in (Cox and Raja 2007).

3.1 Introspective Reasoner

The goal of our introspective reasoning system is to detect reasoning failures and to refine the functioning of reasoning mechanisms, to improve system performance for future problems. To achieve this goal, the introspective reasoner monitors the reasoning process, determines the possible causes of its failures, and performs actions that will affect future reasoning processes.

To give our system criteria for evaluating its case-based reasoning performance, we have created a model of the correctly-functioning CBR process itself, together with a taxonomy of reasoning failures. Failures of a CBR system's reasoning process are modeled as conflicts between observed system performance and predictions from the model. These failures, in turn, are related to possible learning goals. Achieving these goals repairs the underlying cause of the failure.

Reasoning failures may be revealed by either of two types of situation: i) when the retrieval or the adaptation step is not able to propose a solution, or ii) when the solution proposed by the system differs from the final solution. Failures of the retrieval or adaptation steps are identified directly by contrasting their performance with model predictions. The second type of failure can be detected by monitoring the revision step. In CBR systems, the revision step often involves interaction with the user to determine the final solution. This interaction provides a feedback mechanism for assessing the "real" quality of the solution initially proposed.

For each of the four CBR steps (namely retrieval, reuse, revision and retention as illustrated in the bottom portion of Figure 3.1), the model encodes expectations, and the expectations are associated with learning goals which are triggered if the expectations are violated.

For example, the expected behavior of the similarity assessment step is to rank the retrieved cases correctly. If they are ranked incorrectly, the failure may be due to using an inappropriate weighting when similarity assessments along different dimensions are aggregated. Consequently, a possible strategy for solving the failure is to refine the weight model, and a corresponding learning goal is to learn new weightings.

Our model is domain independent, i.e., it is focused on the general case-based reasoning process for retrieval and adaptation, rather than on specific details of those processes for any particular domain. The model deals with three types of knowledge: indexing knowledge, ranking knowledge, and adaptation knowledge. To apply the model to any concrete application, domain-specific retrieval and adaptation mechanisms must be linked to the model.

Indexing knowledge determines the sub-space of the case-base considered relevant to a given problem. Ranking knowledge identifies the features considered most relevant to determining similarity, given a collection of retrieved cases. Adaptation knowledge defines transformative and/or generative operations for fitting previous solutions to a current problem.

Our approach is shaped by two working hypotheses. The first is that the system is initially provided with general retrieval and adaptation mechanisms, which apply uniform criteria to problems throughout the problem space. This is a common property of many case-based reasoning systems, but

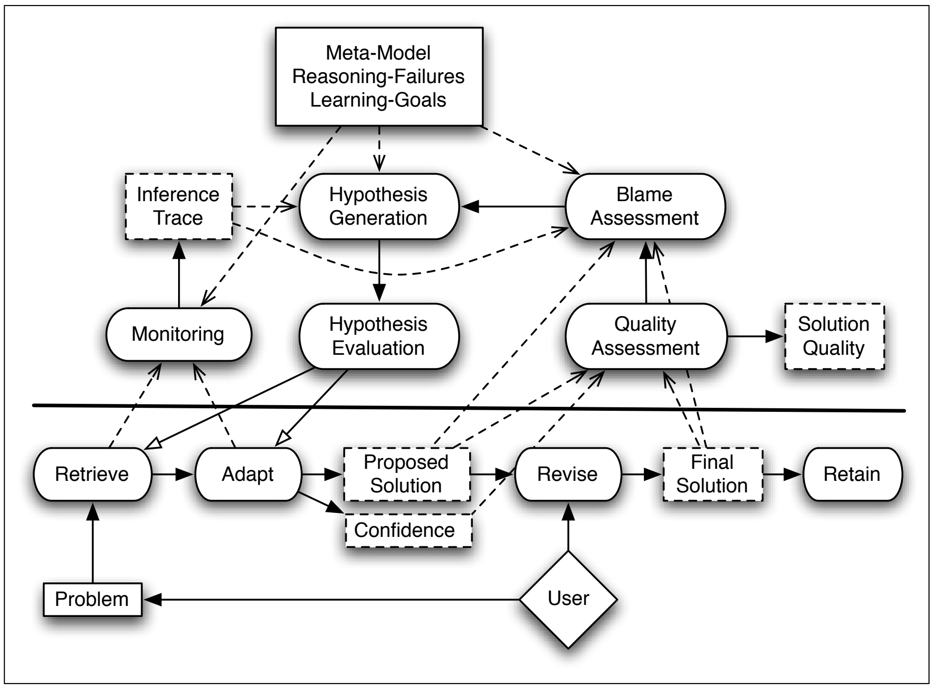

Figure 3.1: Introspective reasoner components. The horizontal line divides the CBR process (bottom) and the Introspective Reasoner (top).

experience developing CBR systems has shown that this uniform processing often results in sub-optimal processing, in turn resulting in the generation of low quality solutions. Consequently, one of the focuses of our approach is to address this problem: One of the learning goals of the introspective reasoner is to determine the 'real' scope of cases, to weight the different ranking criteria, and to refine the adaptation model for different problem space regions.

The taxonomy defined for the learning goals partially borrows from the taxonomy of learning goals proposed in (Cox and Ram 1999). Nevertheless, in our approach the learning goals are specifically oriented towards refining the CBR process. For example, determining the scope of cases is modeled in terms of differentiation/reconciliation goals, whereas improving the ranking criteria is modeled in terms of refinement/organization goals.

A second working hypothesis is that the CBR system is able to determine an internal estimate of confidence for the solution it provides for a new problem. Because this assessment will be domain-specific, it is not part of our general model. In the application we consider, the system always serves in an advisory role to an engineer, who assesses the system-generated solution before applying it. The engineer's assessment provides a natural source of feedback for judging whether the system's confidence value was appropriate.

Because we are not interested in reasoning about numeric confidence values, we deal with confidence using three linguistic labels: *low confidence*, *medium confidence*, and *high confidence*. The mapping to the numeric intervals that represent the linguistic values must be defined in each application. For instance, in our chemical application, due to the important safety constraints in the chemical processes, a high confidence is considered for values higher that 0.8 and low confidence has the threshold at 0.6.

The system's introspective reasoning is organized into five tasks:

1. the *monitoring* task, in charge of maintaining a trace of the CBR process;
2. the *quality assessment* task, that analyzes the quality of the solutions proposed by the system;
3. the *blame assessment* task, responsible for identifying the reasoning failures;
4. the *hypotheses generation* task, in charge of proposing learning goals; and
5. the *hypotheses evaluation* task, that assesses the impact of proposed improvements on solution generation.

Figure 3.1 depicts the introspective reasoning components. The horizontal line divides the CBR process (bottom) from the Introspective Reasoner (top). Rounded boxes represent inference processes; dashed boxes represent knowledge generated by inference; dashed lines show knowledge dependencies; black-tipped arrows show inference flows; and hollow-tipped arrows denote control relationships.

3.1.1 Monitoring

The monitoring task tracks the case-based reasoning process. For each problem solved by the CBR system, the monitor generates a trace containing: 1) the cases retrieved, with a link to the indexing knowledge responsible for the retrieval; 2) the ranking criteria applied to the cases, together with the values that each criterion produced and the final ranking; and 3) the adaptation operators which were applied, with the sources to which they were applied (the cases used) and the target changes produced (the solution features).

Note that this does not require that the adaptation step use only a single case, nor that all the retrieved cases must be involved in all adaptations; any such constraints depend on specific applications, independent of the general model. Similarly, our model distinguishes application of indexing criteria and ranking criteria as two sub-processes involved in the retrieval step, but it does not require that they be decoupled in the implementation being monitored. For instance, a K-nearest neighbor approach (Cover and Hart 1967) uses the value of K to determine the number of cases considered and uses the distance measure as a ranking criterion. Other approaches might use crude criteria for indexing and finer-grained criteria for case ranking.

3.1.2 Quality Assessment

When the user's final solution is provided to the system, quality assessment is triggered to determine the 'real' quality of the system-generated solution, by analyzing the differences between the system's proposed solution and the final solution. Quality assessment provides a result in qualitative terms: *low quality*, *medium quality*, or *high quality*.

Given the system's initial confidence assessment and the final quality assessment, the introspective reasoner fires learning mechanisms when there is a mismatch between the two. There are two main types of possible mismatches. When the confidence was high but the quality is demonstrated to be low, the reasoning failure points to the retrieval stage, because the confidence of a solution has a strong relationship with the coverage of the retrieved cases (Cheetham 2000).

On the other hand, when the confidence was low but the quality is demonstrated to be high, the unexpectedness of success may be either due to low coverage from cases (none of the system's cases appeared highly relevant) or due to bad ranking of the retrieved cases (the most relevant cases were not considered, due to a failure of the ranking polices to identify them). When the mismatch between the confidence and the quality assessments is small (i.e. high versus medium, medium versus high, medium versus low, and low versus medium) it may suggest a failure in the adaptation stage.

3.1.3 Blame Assessment

Blame assessment starts by identifying the source of the failure. It takes as input the differences between the solution and expected result, and tries to relate the solution differences to the retrieval or the adaptation mechanisms. The system searches the taxonomy of reasoning failures and selects those that apply to the observed solution differences.

For instance, when a final solution is radically different from the solution proposed by the system, the failure may be caused by the indexing knowledge,

Failure	Learning Goal
Missing Index	Create Index
Broad Index	Refine Index
Underestimated Weight	Adjust Weighting
Inappropriate interpolation	Change shape
	Increase slope

Table 3.1: Examples of types of hypotheses used by the Introspective Reasoner.

i.e. either the relevant precedents have not been retrieved or too many cases have been retrieved.

Search for applicable failures in the failure taxonomy uses the trace generated by the monitoring module. It starts by analyzing the index failures. There are three types of index failures: *wrong index*, *broad index*, and *narrow index*. When none of the retrieved cases have a solution close to the current solution, the wrong index failure is selected. A broad index failure is selected when many cases are retrieved and their solutions are diverse. On the other hand, when a small set of cases is retrieved, the narrow index failure is selected.

Ranking failures are identified by comparing the retrieval rankings with the solution differences they generate. Examples of ranking failures are *inappropriate ranking*, *overestimated weights*, and *underestimated weights*.

Adaptation failures are identified by linking the solution differences to the adaptation operators stored in the monitoring trace. When adaptation uses interpolation, adaptation failures originate in inappropriate interpolation policies.

Because the introspective reasoner will often not be able to determine a unique failure origin, all the possible causally-supported failures are chosen, resulting in multiple types of learning goals from a single failure.

3.1.4 Hypothesis Generation

The fourth reasoning stage, Hypothesis Generation, identifies the learning goals related to the reasoning failures selected in the blame assignment stage. Each failure may be associated with more than one learning goal. For instance, there are multiple ways of solving overestimated weights. For each learning goal, a set of plausible local retrieval/adaptation changes in the active policies is generated, using a predefined taxonomy.

Table 3.1 shows some of the types of hypotheses generated to explain failures in retrieval and adaptation stages. The changes must be local because their applicability is constrained to the neighborhood of the current problem. For instance, when a refinement goal is selected for the adaptation knowledge, an adaptation is selected from a pre-defined collection of tuning actions depending on the nature of the adaptation. Specifically, when adaptations are related to numerical features the tuning actions are types of numerical interpolations. The two main changes in numerical features are related to the *shape* and *slope* of the interpolation curve.

3.1.5 Hypothesis Evaluation

The fifth reasoning stage, Hypothesis Evaluation, evaluates the impact of introducing retrieval/adaptation changes. Because the introspective reasoner does not have a complete model of the inference process, it is not possible for it to definitively predict the effects of changes. Consequently, before altering the CBR system, some empirical evidence about the impact of the change must be obtained. In our current design this is obtained by re-solving the problem, applying each proposed change and evaluating its impact. Retrieval/adaptation changes that improve the quality of the solution are incorporated into the CBR inference mechanisms.

Note that when the introspective reasoner provides a problem to the CBR system for testing purposes, the case retention step is deactivated.

3.2 Experiments

We have tested the introspective reasoner as an extension to a fielded industrial design application. We have developed a case-based reasoning system for aiding engineers in the design of gas treatment plants for the control of atmospheric pollution due to corrosive residual gases which contain vapors, mists, and dusts of industrial origin (Arcos 2001). A central difficulty for designing gas treatment plants is the lack of a complete model of the chemical reactions involved in the treatment processes. Consequently, the expertise acquired by engineers with their practical experience is essential for solving new problems. Engineers have many preferences and deep chemical knowledge, but our interactions have shown that it is hard for them to determine in advance (i.e. without a new specific problem at hand) the scope and applicability of previous cases. They apply some general criteria concerning factors such as cost and safety conditions, but other criteria depend on specific working conditions of the treatment process.

On the other hand, because engineers make daily use of the application system to provide the final solutions to customers, the system has the opportunity to compare its proposed solutions with the solutions finally delivered. Thus, we have the opportunity to assess the impact of the introspective reasoner on the quality of the solutions proposed by the CBR system.

3.2.1 Applying the CBR process

The inference process in this design application is decomposed into three main stages:

1. selecting the class of chemical process to be realized;
2. selecting the major equipment to be used; and
3. determining the values for the parameters for each piece of equipment.

The quality of proposed solutions is computed automatically, by comparing the proposed solution to the solution applied by the experts at these three different levels. Mismatches at earlier steps are more serious than at later ones. For example, except in the case of under-specified problems, a mismatch with the class of the chemical process would indicate a very low quality solution.

The retrieval and adaptation steps have been designed taking into account the three knowledge sources described in the previous section: indexing criteria, ranking criteria, and adaptation operators. Here the problem features are related to the detected pollutants, the industrial origin of the pollutants, and working conditions for the pollution-control equipment (flow, concentrations, temperature). Indexing criteria determine the conditions for retrieving cases. The main indexing criteria are related to the initially defined chemical relations among pollutants. Ranking criteria determine a preference model defined as partial orders. Initially, the preferences are homogeneous for the whole problem space. Throughout the experiments, the introspective reasoner automatically refines the initial model.

Reasoning failures originate from situations in which the criteria do not properly identify the main pollutants or critical working conditions. The consequences are manifested in solutions for which the proposed chemical process is not correct or there are inappropriate washing liquids, or by mismatches on equipment parameters.

3.2.2 Testing Scenario

The design application can solve a broad range of problems. However, to test the effects of introspective reasoning for learning to handle novel situations, it is desirable to focus the evaluation on sets of frequently-occurring problems which share at least a pollutant (minimal indexing criterion), in order to have reuse. On the other hand, it is necessary to have sufficient diversity—good performance on quasi-identical problems can be obtained by case learning alone, so does not generate opportunities for the introspective reasoner.

We decided to focus the evaluation of the system on problems with the presence of hydrogen sulphide, a toxic gas produced by industrial processes such as waste water treatment. From the existing application, we had access to the 510 such solved problems, ordered chronologically. We divided the problems into two sets: 300 initial system cases and 210 testing problems.

To evaluate the contribution of the introspective reasoner we performed an ablation study, comparing the performance of the system when presenting the problems sequentially for five different reasoning strategies. In addition to testing inputs in chronological order, we repeated the experiments ten times with random orders for the testing problems, to assess the sensitivity of learning to problem ordering. The tested reasoning strategies are the following:

- *No-Retain*, a strategy that solved the problems without introspective reasoning and without incorporating the solved cases into the case memory;
- *Retain*, which solved the problems without introspective reasoning and incorporating solved cases into the system (the only learning normally done by CBR systems);
- *Int-Retr*, which combined *Retain* with introspective reasoning only for the retrieval refinement;
- *Int-Adapt*, which combined *Retain* with introspective reasoning only for adaptation refinement; and
- *Int-Compl*, which combined *Retain* with introspective reasoning for both retrieval refinement and adaptation refinement.

3.2.3 Results

Figure 3.2 shows the results of the evaluation for chronological problem presentation (results for random ordering were similar). Results support that the storage of solved problems—case learning alone—improves the performance of the system, but also show that this policy is not sufficient because the number of high confidence solutions is increased but the number of low quality solutions is not decreasing (see second column in Figure 3.2).

A second conclusion from the results is that the main contribution of using introspection to refine retrieval knowledge is to reduce the number of low quality solutions (a 36.67 % reduction). In our design application

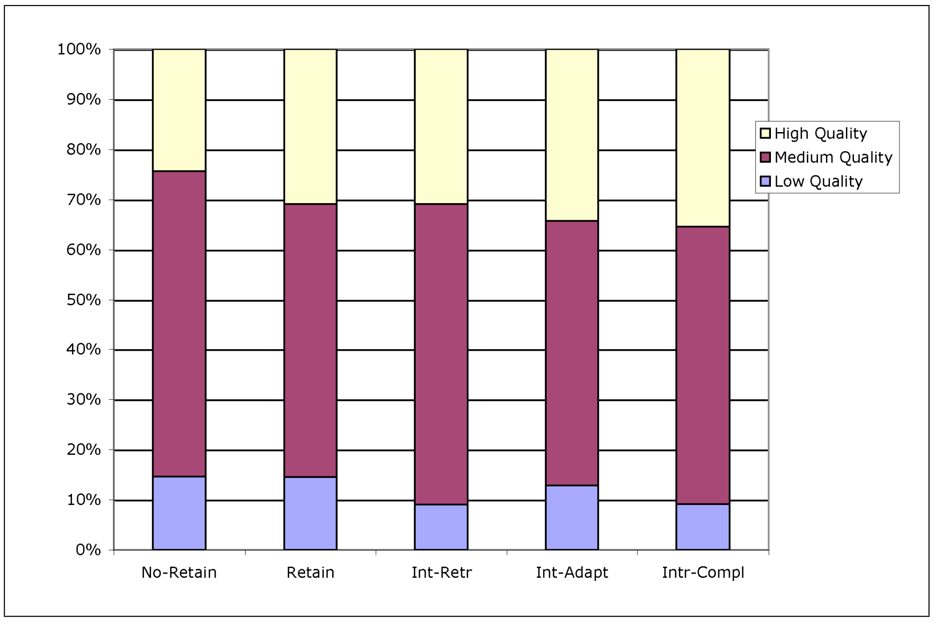

Figure 3.2: Average solution quality for all the strategies.

this improvement is achieved by providing more accurate ranking policies for determining the chemical process to be realized.

The main contribution of using introspection for refining adaptation knowledge (see fourth column in Figure 3.2) is an increase in the number of high quality solutions (a 12.5 % increment). In our task, learning more appropriate adaptation policies enables better determination of the different equipment parameters.

Interestingly, when introspection adjusts both retrieval and adaptation (last column in Figure 3.2), the improvement in the retrieval step has an indirect effect on the adaptation step, increasing the number of high quality solutions. An intuitive explanation is that better retrieval also facilitates the adaptation process. Thus, using both introspection strategies, the increase in the number of high quality solutions reaches 15.63 %.

Comparing the number of problems that changed their quality of solution, 12 % of the solved problems qualitatively increased their solution quality. Solution qualities varied, but the use of introspection did not decrease the solution quality for any problem. Moreover, the reduction in low quality solutions is statistically significant ($\rho < 0.05$), even though the increase of high quality solutions is not statistically significant. Consequently, we conclude that the number of problems whose solution quality was improved by the use of introspection is statistically significant.

Failures	Occ.	Prop.	Inc.
Indexing Knowledge	12	5	3
Ranking Knowledge	83	41	8
Adaptation Knowledge	74	56	12

Table 3.2: Summary of the number of times when learning goals are triggered. *Occ* stands for failure occurrences, *Prop* stands for hypotheses generated, and *Inc* stands for changes incorporated into the CBR process.

Table 3.2 summarizes the activity inside the Introspective Reasoner. Results summarize the experiments using both introspection strategies, reflecting learning goals triggered from the detection of 135 non-high-confidence solutions. Most activity was focused on ranking and adaptation failures, because these are the most difficult tasks. Note that not all the generated hypotheses were considered useful by the system (see third and fourth columns): revisions to the reasoning process were performed for 17 % of the instances for which learning goals were triggered.

This result illustrates that the introspective reasoner is dealing with partial understanding of the CBR process and that the introspective learner's hypotheses should be tested before being applied.

It is clear that the incorporation of the introspective reasoner entails a computational overhead. However, it does not interfere with normal system performance: the introspective reasoner is triggered only *after* a problem is solved and is a background process without user intervention. Most of the cost of introspective reasoning arises from hypothesis generation. Table 3.2 shows that the ratio between failures and hypotheses generated 0.6, because only failures highly explained by the model become hypotheses. Consequently, the number of hypotheses to verify is limited.

A risk of triggering metareasoning in response to individual reasoning failures is the possibility of treating exceptions as regular problems. In the current experiments, such situations did not arise, but in general we assume that the user is responsible for recognizing the exceptions. In addition, only taking action in response to clearly identified failures helps the system to avoid reasoning about exceptions.

Research on humans has shown that introspection may sometimes have negative consequences. Experiments reported in (Wilson and Schooler 1991) showed that, when people is forced to think about the reasons of a given decision, they focus only on plausible explanations in the specific context of the decision. This introspective process usually generates non-optimal explanations affecting negatively future decisions. However, such risks do

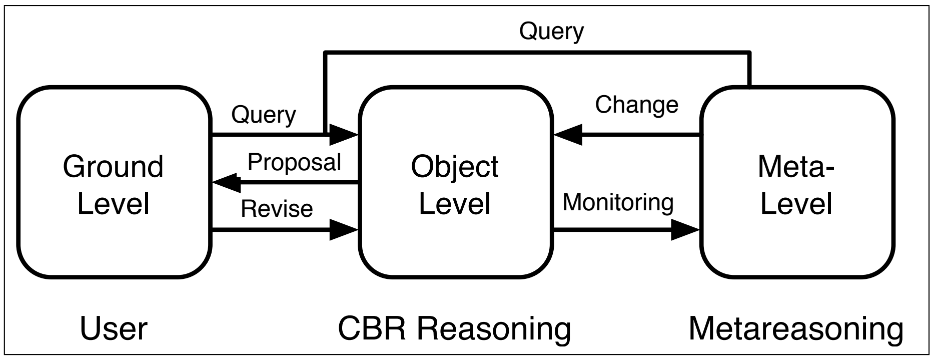

Figure 3.3: Relating our model with existing Metareasoning Models.

not apply directly to our approach. First, only the changes incorporated into the CBR process are affecting future decisions, i.e. not the exploration of plausible hypotheses. Second, the goal of the hypothesis evaluation process is to verify the effect of candidate changes on the system. Third, the changes incorporated only have a local effects.

3.3 Relationship to the Metareasoning Manifesto

Compared to the metareasoning models described by Cox and Raja (2007), our approach is closely related to the use of meta-level control to improve the quality of decisions. Taking as inspiration their 'Duality in reasoning and acting' diagram, our approach incorporates some revisions (see Figure 3.3).

First of all, at the ground level, our approach adds the user of the system. The role of the user is twofold: (1) she presents new problems to the system, and (2) provides a feedback by revising the solution proposed by the Object level. This second role is crucial since it allows to the Meta-level to estimate the performance of the Object level.

In our system, the Meta-level continuously monitors the Object level (the case-based reasoning process) and assesses the quality of the solutions proposed by the reasoner (using the quality assessment module). The user's final solution is used to assess the mismatch between system's expectations for its solution (the solution proposed at the object level) and the correct solution (the solution obtained from the ground level).

It is important to note the importance of the hypothesis evaluation step. Because the introspective reasoner cannot completely predict the effects of changing the reasoning level, the hypothesis evaluation phase acts as an on-line trainer. Thus, the Meta-level, analogously to ground level, has the ability to require the Object level to solve new problems (Top-most query arrow in Figure 3.3). Moreover, when the Meta-level is testing the performance of the Object level it can temporally deactivate the retention step (in our experiments this is achieved by activating the No-Retain policy).

The control of the object level is achieved by acting over three types of knowledge components used in the reasoning process at the object level: indexing knowledge, ranking knowledge, and adaptation knowledge.

Chapter 4

Understanding Dubious Future Problems

When we use CBR for solving problems, we count on the main assumption underlying this methodology (Watson 1999), viz. *similar problems have similar solutions*. Wouldn't it be nice if we could anticipate to what extent the CBR assumption holds for future problems? A positive feedback in this direction would increase the reliability of the system. Contrariwise, we would be aware of the need for carrying out the required design and maintenance tasks throughout our system to improve its future performance in a proactive fashion (Wilson and Leake 2001).

Indeed, this preanalysis would give us important clues about the future. For instance, we could discover deserted regions in our case-base where we do not have available cases to reason with, or we could encounter overcrowded zones in which we would have difficulty to classify our problem among cases of diverse classes.

Furthermore, this analysis would not only yield predictions about the case-base but it could also give us valuable insight about the reasoner itself helping us to verify the functioning of CBR mechanisms like retrieval and reuse in advance.

The question, of course, is how this preanalysis could be performed. One way for such an assessment is confronting the CBR system with possible future problems to detect deficiences beforehand. Though the idea sounds intuitive, the task of finding possible future problems that lead to system deficiencies is far from being trivial for most of the domains where the problem space is too vast or even infinite depending on the features that characterize a domain.

A common approach to attacking such a vast space is to use heuristics that guide the search. We believe that confidence measures can be used as effective heuristics to find system deficiences. A confidence heuristic would guide the search towards finding future problems with low confidence solutions (i.e. inaccurate solutions) thus indicating possible reasoning failures and/or lack of domain knowledge in the case-base.

In this chapter, we propose a method inspired on evolutionary techniques to detect problematic future problems in terms of confidence. We call these future problems with low confidence solutions *Dubious Future Problems* (DFPs). Later we extend this method for detecting and characterizing dubious regions in the problem space.

To effectively scan the problem space for finding dubious regions, we propose a method based on four steps: First, we explore the problem space to find dubious future problems. Then, we carry out an exploitation phase to better identify these problems by focusing the search on additional future problems in their neighborhoods. Next, to help the understanding of the regions where dubious future problems are located, we associate each DFP with a neighborhood pattern (e.g. hole, border). Finally, to focus on regions in the case-base that suffer from the same deficiency rather than dealing with individual problems, we group DFPs according to these patterns.

In Section 4.1 we present our evolutionary approach for scanning the problem space to find dubious future problems. The definitions of dubiosity patterns and the grouping algorithm are described in Section 4.2. In Section 4.3, first we give an introductry example of how to explore DFPs on a rather simple Zoology domain. Later, we give a complete example where we explore DFPs and group them by the patterns that they exhibit in a Robo-Soccer system. We interpret the results showing how they helped us to analyze given CBR systems.

4.1 Exploring Dubious Future Problems

Given a domain ontology associated with a CBR system, we are interested in identifying possible future problems that: 1) are similar enough to the current cases and, 2) that the confidence on their solutions provided by the CBR system is low. Thus, the exploration of the problem space to find DFPs requires only three knowledge components in a CBR system:

- a domain ontology (specifying at least the features and their data types used for defining cases);
- a similarity metric; and
- a confidence measure that attaches a confidence value for each solution provided by the CBR system.

In the search for dubious problems, the search space is the space of all problems that can be generated according to the domain ontology. As indicated above, this space can be too vast or even infinite depending on the features that characterize the domain. To find DFPs we use Genetic Algorithms (GA) as they have demonstrated their capabilities for exploring such vast search spaces. They have the advantage of scanning the search space in a parallel manner using a fitness function as heuristics and their implementations can be domain independent.

With a diverse initial population of possible future problems and an appropriate fitness function, DFPs will evolve as the GA runs, where the less confident the CBR system is about a problem's solution the more it will prefer to regard that problem as a DFP. However, as commonly seen in practice, GAs might have a tendency to converge towards local optima (Michalewicz 1996). In our case, this would result as getting stuck to a low confidence zone and generating problems only within that locality instead of scanning a wider region in the problem space. In many GAs, mutation is the trusted genetic operator to avoid this problem as it introduces diversity to the population, nevertheless it is usually not a guarantee.

Our approach to effectively search the problem space and to avoid local minima has been to divide the search into two steps, namely *Exploration* and *Exploitation* of dubious future problems. In the Exploration step, the aim is to find DFPs which are similar enough to existing cases and which are as dissimilar as they could be to each other. The similarity to existing cases argument is to avoid dealing with irrelevant (although possibly not unlikely) problems which have no neighbour cases in the CB. The confidence for a solution to a generated problem which has no similar neighbours would probably be very low, but since this would already be an expected result, it would not be of much interest to bring these problems to the expert's inspection. Additionally, the dissimilarity between DFPs is for the sake of obtaining diversity in the results of Exploration to achieve a richer set of future problems and their neighbours after the Exploitation step.

Successively, in the Exploitation step our objective is to find future neighbours of the DFPs encountered in the Exploration step for providing a more precise analysis of the low confidence local regions.

Both, Exploration and Exploitation steps, incorporate two proximity limits in terms of similarity to an existing case or a future problem. These limits define the preferred region in the problem space during the search for DFPs and their neighbours. We will explain both limits in detail for each step in the next sub-sections. We also added a *Diversity Preservation* feature to our GAs for both steps to keep the population's diversity at a desired level.

The following sub-sections describe the details of the Exploration and Exploitation steps.

4.1.1 Exploration

The goal of the Exploration step is to identify an initial set of dubious problems similar enough to the cases defined in a case-base. A problem is considered dubious when its confidence is lower than a given threshold. Since the minimum value for considering a solution as confident may vary in each CBR application, the decision about the confidence threshold is domain dependent.

For the Exploration step, the proximity limits mentioned above define the preferred region of the search for dubious problems. The outer limit OB_{EC} defines the border for the less similar problems, while the inner limit IB_{EC} defines the border for the most similar ones to an existing case in the CB. We also use the inner limit to draw a border around the found DFPs since we are looking for DFPs that are as diverse as possible in this step. A graphical representation of the Exploration step is provided in Figure 4.1.

The decision of the proximity limits depends on the answer of how similar a problem can be to a case to be regarded as a relevant problem for the domain and application. The similarity among existing cases may give an idea of the range of possible values for these limits. For example; if these two limits are chosen so that their sum is closer to the similarity value between two nearest cases of different classes, then preffered proximities will overlap thus giving us the possibility to discover borders for the classes in the CB.

Throughout the execution of the GA for Exploration, we maintain a list of encountered future problems with low confidence solutions $\mathcal{LCFP}$. During the evaluation of a population, each time we come across a chromosome representing a dubious problem we add it to the $\mathcal{LCFP}$ list.

The concepts used in the GA for the Exploration step are explained below:

Chromosomes: Each chromosome in our population represents a future problem where each gene is a feature of the problem. The value of a gene is thus one of the possible values (defined by the domain ontoogy) for the associated feature.

Initial Population: The initial population is formed by chromosomes generated by the *Random-Problem-Generator* function (RPG). RPG is a function able to generate a new problem by assigning random values for each problem feature. Values for problem features can be easily generated using the definitions of features in the domain ontology (feature definitions explicitly

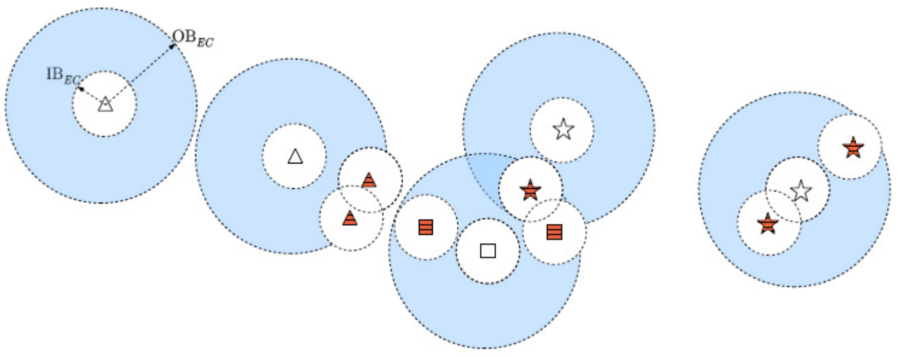

Figure 4.1: Graphical representation of the Exploration step. Hollow shapes are existing cases (where different shapes refer to different classes); filled shapes are the encountered Dubious Future Problems; IB_{EC} and OB_{EC} are, respectively, inner and outer bounds.

state the data type and the set of possible values for a feature). It should also be considered that in the existence of domain constraints, the Random-Problem-Generator function generates valid problems that conform to those constraints. Otherwise, generated future problems might be non-valid or irrelevant in the domain. The size of the population directly depends on the vastness of the problem space of the CB that is being worked on.

Fitness Function: The fitness of a chromosome is determined by two parameters: the confidence value of the solution to the problem represented by the chromosome and the similarity of the problem to the nearest problem in the CB. The fitness function has to be adapted in each different domain or CBR system. However, the following guidelines should be used in Exploration regardless of the domain or the application:

- The lower the confidence value is for a chromosome, the better candidate is that chromosome.
- A chromosome in the preferred proximity of an existing case is a better candidate than a chromosome which is not in this proximity.
- The confidence factor of the fitness is more significant than the similarity factor. This is not surprising since we are searching for dubious problems.

Our proposal for the fitness function definition is the following:

$$Fitness(c) = Confidence(c)^2 \times SimilarityFactor(c)$$

where c is the chromosome to be evaluated; $Confidence$ returns the confidence value supplied by the CBR application after solving c; and *Similarity-Factor* takes into account the similarity to both cases and DFPs. *Similarity-Factor* is calculated as follows:

$$SimilarityFactor(c) = partSimEC(c) + partSimDFP(c)$$

where *partSimEC* refers to the similarity of c to existing cases and *partSim-DFP* refers to the similarity of c to DFPs in $\mathcal{LCFP}$. *partSimEC* is defined as:

$$partSimEC(c) = \begin{cases} 1 - (OB_{EC} + IB_{EC} - Sim(c, CB)) & \text{if} \quad Sim(c, CB) \geq IB_{EC} \\ 1 - Sim(c, CB) & \text{otherwise} \end{cases}$$

where $Sim(c, CB)$ is the similarity value of c to the most similar case in the CB (i.e. the highest similarity); IB_{EC} and OB_{EC} are, respectively, the inner and outer bounds of similarity to the existing cases. $partSimDFP(c)$ is defined as:

$$partSimDFP(c) = \sum_{p \in FP} (similarity(c, p) - IB_{EC})$$

where $FP \subset \mathcal{LCFP}$ is the set of future problems to which c is more similar than the allowed value IB_{EC} and $similarity(c, p)$ is the similarity value of c to the problem p.

Following the previously defined guidelines, *SimilarityFactor* penalizes the chromosomes that are too close to either cases or future problems discovered in previous iterations (i.e. inside the radius defined by the inner threshold).

It should also be noted that for a desired chromosome (i.e. representing a dubious future problem which is in the preferred proximity of an existing case) our proposed function produces a fitness value which is lower than a non-desired one.

Selection: We defined a fitness-proportionate selection method. Fitness-proportionate selection is a commonly used and well studied selection mechanism where each chromosome has a chance proportional to its fitness value to be selected as a survivor and/or parent for the next generations. However, since we are interested in chromosomes with lower fitness values as explained above, to comply with our fitness function, selection of a chromosome was inversely proportional to its fitness value.

Crossover: We use single-point crossover as it is simple enough and widely used. Depending on the observed convergence of the GA, this method could easily be replaced by Two-Point or n-Point crossover methods.

Mutation: Generally, one random gene value is altered for a number of offspring chromosomes in the population. If a local minima problem is observed, more genes and/or more chromosomes can be mutated.

Diversity Preservation: We decided to use a diversity threshold that can be tuned for each application. Specifically, at each generation when the number of twins exceeds the diversity threshold, they are removed probabilistically using as probability their fitness value (i.e. twins with higher fitness have a higher probability to be deleted).

In our approach, the *validity* of a problem is another important issue. Due to the application of genetic operators in the evolution cycle, it is likely to reproduce offspring chromosomes which are non-valid. We may deal with these chromosomes basically in two ways: we may replace them with new valid chromosomes or we may let some of them survive hoping them to produce nice offspring in the following generations. In the former option, the replacement can be done in the Diversity Preservation. In the latter option, either a validity check can be incorporated into the fitness function reducing the fitness of non-valid chromosomes or simply non-valid chromosomes can be excluded from the $\mathcal{LCFP}$ after the termination of the Exploration step. In the current implementation we adopted this last solution.

Termination: The termination criterion for the GA can be reaching a number of generations or a number of dubious future problems. We let the population evolve for a certain number of generations.

Result: As the result of the GA we obtain the list of future problems with low confidence solutions $\mathcal{LCFP}$.

4.1.2 Exploitation

The goal of the Exploitation step is to explore the neighbourhood of the low-confidence problems discovered in the Exploration step. Similarly to the Exploration step, during the execution of the GA for the Exploitation step we maintain a list of Low Confidence Problem Neighbours $\mathcal{LCPN}$. We initialize

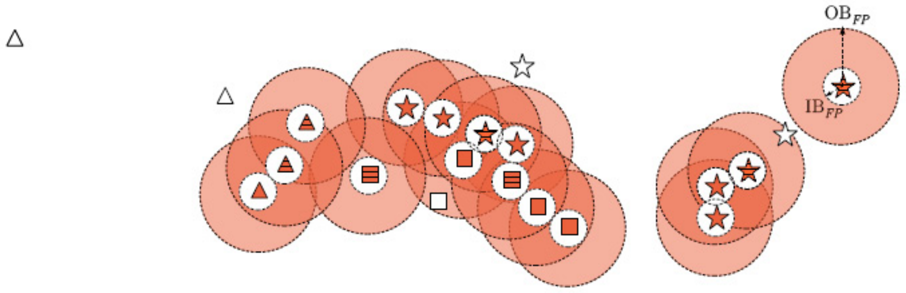

Figure 4.2: Graphical representation of the Exploitation step. Hollow shapes are existing cases; filled shapes are the encountered and exploited Dubious Future Problems. IB_{FP} and OB_{FP} are, respectively, inner and outer bounds.

this list with the members of the $\mathcal{LCFP}$. In other words, the members of this list are the dubious future problems that we want to exploit.

For the Exploitation phase, the proximity limits define the preferred region of the search for neighbour problems. The outer limit OB_{FP} defines the border for the less similar problems, while the inner limit IB_{FP} defines the border for the most similar ones to any member of the $\mathcal{LCPN}$. A graphical representation of the Exploitation step is provided in Figure 4.2. Notice that, comparing with the Exploration step, the proximity limits for Exploitation step are narrower since in this step we are looking for neighbours of the DFPs.

All DFPs satisfying the proximity limits are added to the $\mathcal{LCPN}$ list. The confidence threshold for dubiosity is the same value used in the Exploration.

The concepts used in the GA for the Exploitation step are the following:

Chromosomes, Selection, Crossover, Mutation, Diversity Preservation: These concepts have the same definitions as the corresponding ones previously given in the Exploration step.

Initial Population: We partially feed the initial population with the $\mathcal{LCFP}$ set hoping to reproduce similar problems. We use the Random-Problem-Generator to reach to the desired initial population size when needed.

Fitness Function: The fitness of a chromosome c in the Exploitation step depends only on its neighbourhood to any member of $\mathcal{LCPN}$. The fitness function is defined as follows:

$$Fitness(c) = \begin{cases} 1 - (OB_{FP} + IB_{FP} - Sim(c, \mathcal{LCPN})) & \text{if} \quad Sim(c, \mathcal{LCPN}) \geq IB_{FP} \\ 1 - Sim(c, \mathcal{LCPN}) & \text{otherwise} \end{cases}$$

where $Sim(c, \mathcal{LCPN})$ is the similarity value of c to the most similar problem in $\mathcal{LCPN}$; IB_{FP} and OB_{FP} are, respectively, the inner and outer proximity bounds of similarity to the previously found future problems.

Termination: We let the population evolve for a certain number of generations in Exploitation as well.

Result: At the end, the Exploitation step provides the list $\mathcal{LCPN}$ which contains dubious future problems found both in the Exploration and Exploitation steps.

4.2 Regions of Dubiosity

Exploration and Exploitation of DFPs give us a foresight of a possible bad performance of the CBR system. To inspect the underlying reasons of such a malfunction, the encountered DFPs may be presented directly for the domain expert's attention. Experts in turn may use this future map of the case-base to initiate maintenance tasks if needed. However, depending on their number, analysing DFPs manually may become a difficult task as domain experts would have to check each DFP together with its neighbours to reveal the system deficiencies.

To be able to assist the domain experts in the endeavour of analysing DFPs, we have defined six dubiosity patterns. Each DFP is tagged with a dubiosity pattern which indicates the possible reason of being classified as dubious. Furthermore, when we have a numerous list of DFPs, we propose a grouping algorithm for helping the expert to focus on regions in the case-base that suffer from the same deficiency.

4.2.1 Dubiosity Patterns

DFPs are good pointers to possible future system weaknesses as their solutions have low confidence values. But to identify the cause of the low confidence result, and thus, the needed policies for eliminating these weaknesses, DFPs themselves alone are not much of a help. This is because confidence measures, in general, do not provide detailed explanations of the judgement they make while attaching a confidence value to a solution. Thus, for analysing why DFPs were considered as dubious the expert should inspect the indicators of confidence used by the confidence measure of the CBR system.

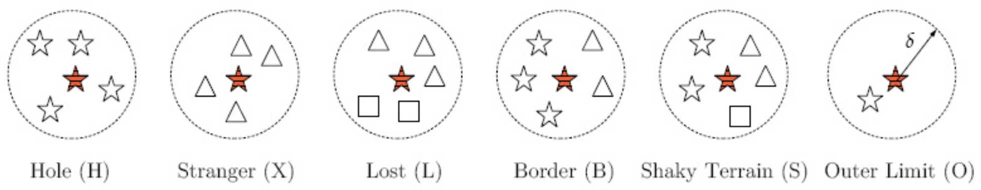

Figure 4.3: Graphical representation of DFP Patterns. Hollow shapes are cases and the filled one is a Dubious Future Problem. Each shape represents a different solution class. δ is the similarity threshold delimiting the neighbourhood of a DFP.

On the other hand, since usually the similarity measure plays an important role in confidence calculus (Cheetham and Price (2004), Delany *et al.* (2005)), looking at the neighbour cases of a DFP would give strong clues for analizing DFPs. For this aim, we have defined six dubiosity patterns according to the solution classes of a DFP and of its neighbour cases. Given a similarity threshold δ defining the neighbourhood, we say that a DFP exhibits a pattern of type (see Figure 4.3 for a graphical representation):

- **Hole (H)** when all of its neighbour cases are of the same class as the DFP;
- **Stranger (X)** when all of its neighbour cases are of the same class which is different from the DFP's;
- **Lost (L)** when there are at least two different groups of neighbour cases, according to their solution classes, where none of the groups is of the same class as the DFP;
- **Border (B)** when its neighbour cases can be grouped into two groups with different solution classes and one group shares the same class as the DFP;
- **Shaky Terrain (S)** when its neighbour cases can be grouped into at least three groups of different solution classes and one group shares the same class as the DFP. This pattern indicates regions where adding or removing a case might redraw borders for multiple classes.
- **Outer Limit (O)** when it has only one neighbour case sharing the class. This pattern may indicate outer limits for a particular class or it may point out isolated cases in the case-base.

After the Exploitation step, for each DFP in $\mathcal{LCPN}$ we check the solution classes of its neighbour cases together with its own and we associate a pattern to each DFP according to the above pattern definitions. The similarity threshold δ value should be coherent to the OB_{EC} value in the Exploration step since we were looking for similarity between DFPs and existing cases. We propose to choose a value slightly bigger than the OB_{EC} value for the δ.

4.2.2 Grouping Dubious Future Problems

Preliminary experiments for exploring DFPs have shown that depending on the features that characterize a domain and on the CBR inference mechanism we may end up with a lengthy list of explored DFPs. In one sense, a high number of DFPs is attractive since the more DFPs we encounter the more possible future deficiencies we are discovering. However, using this lengthy list to carry out maintenance tasks may turn out to be a tedious work in both manual and automated maintenance of a CBR system. Although the associated dubiosity patterns help us to analyse DFPs, each DFP still requires special attention to identify the needed maintenance tasks.

To overcome this overhead when we have too many DFPs to deal with, we propose to group the DFPs according to their patterns and the similarity of the DFPs among each other. Grouping DFPs in this way makes it easier to identify regions in the problem space that suffer from the same deficiency. Thus, any maintenance task that eliminates a common deficiency in such a region will probably make the CBR system more confident of its solutions for similar future problems that will fall into that region. We call these regions *Regions of Dubiosity.*

Given the list $\mathcal{LCPN}$ and a similarity threshold δ', the grouping algorithm performs the following two steps:

1. **Identification** of the Regions of Dubiosity by transitively grouping all DFPs that are neighbours at similarity δ'. This step forms different isolated regions. Each region is a graph where the nodes are the DFPs and edges connect two nodes when their similarity is, at least, δ'.

2. **Characterization** of the Regions of Dubiosity by grouping all the DFPs that share the same pattern and are directly connected. Thus, each subregion is a subgraph highlighting a pattern.

δ' should be coherent with (if not equal to) the OB_{FP} value in the Exploitation step since we are looking for similarity between DFPs for grouping them.

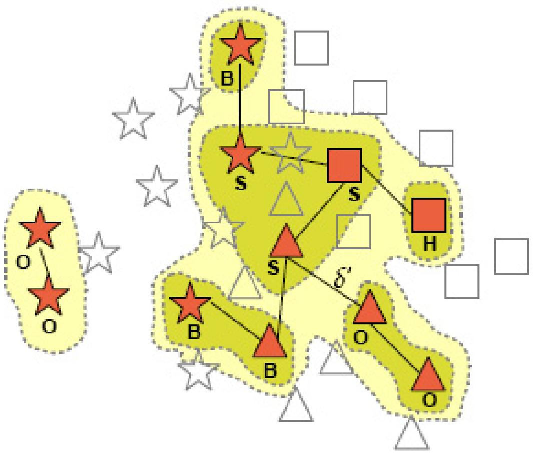

Figure 4.4: Regions of Dubiosity. Hollow shapes are cases and filled shapes are DFPs. Subscripts point out the patterns associated to each DFP.

At the end, the regions of dubiosity that we obtain from the above algorithm help us to identify the problematic zones in the CBR system. Moreover, each subregion of shared patterns serves to detect zones that suffer from the same deficiency, thus preventing us from having to deal with individual DFPs.

A graphical representation of an example for identifying Regions of Dubiosity is given in Figure 4.4. In the figure two different regions have been detected. The first one on the left only has two DFPs identified as outer limits. The big region on the right has three border sub-regions, one stranger sub-region, and one central shaky terrain sub-region. Connecting lines between two DFPs show that they are neighbours according to a given similarity threshold δ'. Note that region surfaces are only painted with the purpose of highlighting the dubiosity regions in the problem space.

4.3 Experimentation

We have conducted experiments on two different domains, namely Zoology and Robo-Soccer. With the first domain we aimed to carry out a simple, yet a complete experiment to get a better understanding of how to tune parameters, such as proximity limits and GA settings, to explore dubious future problems. The second part of our experiments was on a more complex domain where we analyzed a CBR application used for achieving team behavior at Robo-Soccer. Here we tried to analyse the case-base to reveal dubiosity regions and reasoning deficiencies if there were any. Details of the experiments on these two domains are given below.

4.3.1 Zoology Domain

For experiments on this domain, we used the Zoology dataset available from UCI machine learning repository (Asuncion and Newman 2007). The Zoology dataset has 100 examples and 7 solution classes. The domain ontology defines the data type and the set of possible values of each feature. In addition, we explicitly stated the constraints for the domain that restrict non-existing animals, e.g. an animal cannot have feathers and hair at the same time. Thus, we were able to generate valid future problems using this ontology.

In the genetic representation of the Zoology domain, each chromosome has 16 genes corresponding to the 16 features of each example in the data set. 15 of these features (`Hair`, `Feathers`, `Egss`, `Milk`, `Airbone`, `Aquatic`, `Predator`, `Toothed`, `Backbone`, `Breathes`, `Venomous`, `Fins`, `Tail`, `Domestic`, `Catsize`) are boolean valued and 1 of them is an enumeration (`Legs {0 2 4 5 6 8}`). We defined an additional `not-supplied` value for each feature to be able simulate non-complete problems. The RPG function generated random chromosomes using these features and their possible values.

Settings

The experimentation settings were the following: 40% of the population was selected as survivors to the next generation; 60% of the chromosomes were selected as parents to reproduce offspring; mutation was applied to a randomly chosen 5% of the offspring modifying a gene's value for each chosen chromosome; the diversity threshold for the twin chromosomes was 5% (we kept this amount of twins in the new generation and replaced the rest of them with new ones created by the RPG).

The similarity metric calculated the minimum similarity between two cases in the case-base as `0.3536`, where the maximium similarity was `0.9683`. Taking into account the similarities among existing cases, we chose the test range `[0.93, 0.99]` for the proximity limit values in our experiments. And we kept the proximity for Exploitation narrower than Exploration (see subsections 4.1.1 and 4.1.2).

The Reuse method returned the solution class with the highest confidence value. The confidence measure used was an implementation of the *Similarity Ratio Within K* introduced in (Delany, Cunningham, Doyle and Zamolotskikh 2005) and the range for confidence values was `[0.0, 3.0]`. We noted that in the domain:

- 2 cases had solutions with confidence values *confidence* $<$ `1.0`
- 7 cases had solutions with confidence values `1.0` $<=$ *confidence* $<=$ `2.0`
- 50 cases had solutions with confidence values `2.0` $<$ *confidence* $<$ `3.0`
- 41 cases had solutions with confidence values *confidence* $=$ `3.0`

So, a value of `2.0` for low confidence seemed a reasonable threshold for this domain.

Trying different settings for inner and outer proximity limits for both cases and future problems, GA population sizes and number of generations allowed for evolution, we wanted to see how the GAs evolved with the resulting $\mathcal{LCFP}$, $\mathcal{LCPN}$ lists. Because of the random nature of GAs, for each setting we executed the Exploration and Exploitation steps five times to get an average value for the number of the members of the lists.

Results

In the Table 4.1 we give some of the experimentation results for different GA settings, confidence thresholds and proximity limits where,

- Pop : Population size for GAs for both Exploration and Exploitation steps;
- Gen : Number of generations we allow for the evolution in both steps;
- Conf : Threshold for Low Confidence;
- IB_{EC} : Inner bound of similarity in Exploration(see Figure 4.1);
- OB_{EC} : Outer bound of similarity in Exploration(see Figure 4.1);
- IB_{FP} : Inner bound of similarity in Exploitation(see Figure 4.2);
- OB_{FP} : Outer Bound of similarity in Exploitation(see Figure 4.2);
- $\mathcal{LCFP}$: Number of the members of the $\mathcal{LCFP}$ list; and
- $\mathcal{LCPN}$: Number of the members of the $\mathcal{LCPN}$ list.

The results show that we may encounter a higher number of dubious future problems and their neigbours when,

- the initial population is richer in size;

Table 4.1: Experimental Results for Zoology Domain

Pop	Gen	Conf	IB_{EC}	OB_{EC}	IB_{FP}	OB_{FP}	$\mathcal{LCFP}$	$\mathcal{LCPN}$
100	50	2.0	0.98	0.95	0.99	0.96	6	108
”	”	”	”	0.94	”	”	10	141
”	”	”	0.97	0.93	”	”	21	176
”	”	”	0.96	”	”	”	21	165
150	”	”	”	”	”	”	29	238
100	80	”	”	”	”	”	27	225
”	50	1.0	”	”	”	”	26	174
”	50	”	0.98	0.94	”	”	11	159
”	80	”	0.97	0.93	”	0.95	25	140
”	”	”	”	0.94	0.98	0.96	12	123
150	”	”	”	”	”	”	20	243
100	80	”	”	”	”	”	7	235

- GAs evolve during more generations;
- the area within proximity limits is wider;and
- the threshold of low confidence is high.

For example, in the third and fourth rows of the table, we can observe that increasing the population size from 100 to 150, we obtained lists $\mathcal{LCFP}$ and $\mathcal{LCPN}$ with more future problems. The first and second lines give an example where we widened the proximity in exploration and had a richer $\mathcal{LCFP}$ and in turn after exploiting this richer list we got a richer $\mathcal{LCPN}$ as well.

It should be noted that although it is possible to get a richer list of future problems adjusting proximity limits, our aim is to find dubious problems within a reasonable neighbourhood of existing cases. So, these limits should be chosen carefully. The low confidence threshold is another crucial parameter because it is a matter of decision of up to which value we could regard the confidence of a solution as acceptable.

4.3.2 Robo-Soccer CBR system

In this second group of experiments we have performed the analysis of DFPs on a CBR system developed for the Four-Legged League (RoboCup) soccer competition (Ros *et al.* (2007), Ros (2008)). In RoboCup two teams of

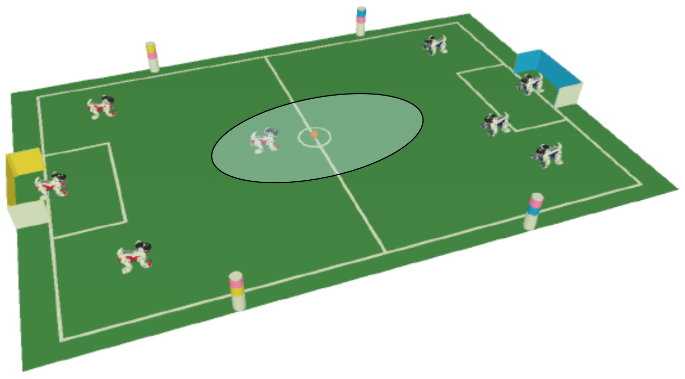

Figure 4.5: Four-Legged League (RoboCup) soccer

four Sony AIBO robots compete operating autonomously, i.e. without any external control (see Figure 4.5). The goal of the CBR system is to determine the actions (called *gameplays*) the robots should execute given a state of the game.

The state of the game is mainly represented by the position of the ball and the positions of the players (both teammates and opponents). The positions are constrained by the field dimensions (6 m long and 4 m wide). Moreover, since robots occupy a physical space in the field, a state of the game is considered valid whenever the distances among the robots are higher than their dimensions (30 cm long and 10cm wide).

The 68 cases stored in the system can be grouped into three main behaviors: cooperative behaviors (where at least two teammates participate); individualistic behaviors (only one player is involved); and back away behaviors (where the position of the opponents forces a player to move the ball back).

The confidence measure provided by the application took into account not only the similarity of the problem to the cases but also the actual distance of the current position of the players to the ball. Therefore, although all similar cases share the same solution, when the players are away from the ball the confidence of the solution is low.

The first goal of our experiments was to foresee whether there exist states of the game where the CBR system has difficulties in determining the best behavior, i.e. the confidence on the proposed solution is low. The secondary goal was to detect if there were any bad performing mechanisms of the CBR system.

Settings

Similarly to Zoology experiments, the settings were as follows: 40% of the population was selected as survivors; 60% of the chromosomes were selected as parents; a randomly chosen 5% of the offspring mutation was mutated by modifying a random gene's value for each chosen chromosome; and the diversity threshold for the twin chromosomes was 5%.

Taking into account the similarities among existing cases (minimum similarity `0.7481`, maximium similarity `0.9989`), we chose the test range `[0.93, 0.99]` for the proximity limit values. We again kept the proximity for Exploitation narrower than Exploration. The similarity threshold (δ in Figure 4.3) for associating patterns to DFPs was always a value slightly bigger than the OB_{EC} value in the Exploration step. Analogously, the similarity threshold for grouping DFPs (δ' in Figure 4.4) to form Regions of Dubiosity, was the same as the OB_{FP} value in the Exploitation step (see subsections 4.2.1 and 4.2.2). Finally, the test range for confidence threshold was chosen as `[0.3, 0.7]`.

We ran different experiments for analysing the sensitivity in identifying DFP regions by changing parameters such as the size of the initial population, the number of generations, the confidence threshold, and the proximity limits. Moreover, because of the random nature of GAs, for each setting we executed the Exploration and Exploitation steps several times to get an average value for the number of identified DFPs and regions of dubiosity.

Results

Throughout experimentation we have seen that we may encounter a higher number of DFPs when: the initial population is larger in size; GAs evolve during enough generations; the preferred proximity is wider or the threshold of low confidence was high. This was just the same case we had observed during the Zoology domain experiments.

In the Table 4.2 we provide some of the results of 63 experiments we did with different GA settings, confidence thresholds and proximity limits.

The results show (see Table 4.3 left) that the number of the DFPs encountered in the Exploration step is closely related to the number of the Regions of Dubiosity. The difference between these two numbers is due to linking of DFPs found in Exploration with other DFPs found in the Exploitation step. This happens when we reach a DFP previously found in the Exploration step while we are exploiting another DFP problem found in the same step. Therefore, when no linking is achieved between such DFPs, the number of the Regions of Dubiosity is equal to the number of the DFPs found in the

Table 4.2: Experimental results for Robo-Soccer domain.
Pop : Population size for GAs for both Exploration and Exploitation steps; Gen : Number of generations we allow for the evolution in both steps; Conf : Threshold for Low Confidence; IB_{EC} : Inner bound of similarity in Exploration OB_{EC} : Outer bound of similarity in Exploration IB_{FP} : Inner bound of similarity in Exploitation OB_{FP} : Outer Bound of similarity in Exploitation δ : Threshold for similarity of a DFP to existing cases for pattern attachment δ' : Similarity threshold for grouping DFP $\mathcal{LCFP}$: Number of the members of the $\mathcal{LCFP}$ list; $\mathcal{LCPN}$: Number of the members of the $\mathcal{LCPN}$ list; RD : Number of the found dubiosity regions; Patterns : Encountered dubiosity patterns.

Pop	Gen	Conf	IB_{EC}	OB_{EC}	IB_{FP}	OB_{FP}	δ	δ'	$\mathcal{LCFP}$	$\mathcal{LCPN}$	RD	Patterns
50	30	0.5	0.03	0.07	0.01	0.04	0.08	0.04	35	144	35	H X - B S O
”	”	”	”	”	”	”	”	”	61	160	58	H X L B S O
”	”	”	”	”	”	”	”	”	72	192	65	H X L B - -
”	”	”	”	0.06	”	”	0.07	”	35	125	35	H X - B - O
”	”	”	”	”	”	”	”	”	61	119	58	H X L B - O
”	”	”	”	”	”	”	”	”	66	231	62	H X - B - O
”	”	”	”	”	”	0.03	”	0.03	38	161	37	H X - - - O
”	”	”	”	”	”	”	”	”	34	97	32	H X L B - O
”	”	”	”	”	”	”	”	”	34	82	31	H X L B - O
100	”	”	”	0.07	”	0.04	0.08	0.04	111	341	93	H X - B - O
”	”	”	”	”	”	”	”	”	118	375	110	H X L B - O
”	”	”	”	”	”	”	”	”	108	296	92	H X L B - O
50	60	”	”	”	”	”	”	”	83	238	80	H X - B - O
”	”	”	”	”	”	”	”	”	94	396	92	H X L B - O
”	”	”	”	”	”	”	”	”	90	247	81	H X - B - O
”	30	0.7	”	”	”	”	”	”	105	283	95	H X L B S O
”	”	”	”	”	”	”	”	”	72	177	72	H X - B - O
”	”	”	”	”	”	”	”	”	77	171	74	H X L B - O
”	”	0.3	”	”	”	”	”	”	16	134	15	H X - - - -
”	”	”	”	”	”	”	”	”	3	16	3	- X L B - -
”	”	”	”	”	”	”	”	”	11	39	10	H X - - - O

Table 4.3: Average (Avg) and standard deviation (σ) of DFPs discovered in the Exploration ($\mathcal{LCFP}$) and Exploitation ($\mathcal{LCPN}$) steps in 63 experiments at Robo-Soccer Domain. RD shows the number of the Regions of Dubiosity created from DFPs. H, X, L, B, S, O are, respectively, the percentage of the number of the experiments in which *Hole*, *Stranger*, *Lost*, *Border*, *Shaky Terrain*, and *Outer Limit* patterns appeared.

	$\mathcal{LCFP}$	$\mathcal{LCPN}$	RD
Avg	59.26	183.21	55.95
σ	34.12	105.78	30.89

H	X	L	B	S	O
95%	100%	50%	85%	15%	85%

Exploration step.

Another interesting result is the analysis of the DFP patterns discovered in the experiments (Table 4.3 on the right summarizes the percentage of experiments where each pattern is detected). This analysis allows a better understanding of the regions of the problem space where the CBR system is not performing confidently:

– Holes in the soccer domain occured when although all the closer cases shared the same individualistic solution, the players were far from the ball. This was due to the provided confidence measure explained above. To obtain more confident solutions the neighbourhood of holes can be populated with new cases.

– Stranger DFPs were problems that proposed cooperative behavior but whose neighbours were individualistic cases. We saw that this was because of the design of the system for favouring cooperative behavior. If there is a close cooperative case to the problem, the application proposes cooperative solution even if there are more similar cases with different solutions. Stranger DFPs helped us to discover regions where the influence of the cooperative cases was excessive. To improve the confidence, we proposed to reduce that influence for similar regions.

– Lost DFPs were problems that proposed cooperative behavior in a region where their neighbours were individualistic and back away cases. This was again due to the excessive influence of a cooperative case nearby. The proposed maintenance task was the same as in the previous pattern.

– Border DFPs identified the regions where neither individualistic nor cooperative behaviors reach a significantly better confidence. The confidence in these regions could be improved by incorporating new cases into the case-base to be able to mark the borders better between these two classes.

– Shaky terrain DFPs does not seem to be significant in the Robosoccer

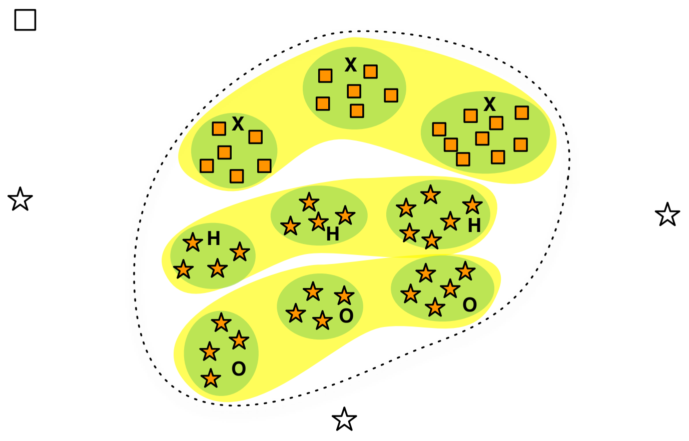

Figure 4.6: Visualizing a dubious region in the RoboSoccer problem. DFPs with individualistic solutions are represented as stars. Squares represent DFPs with cooperative solutions. H, X and O, respectively, indicate Hole, Stranger and Outer Limit patterns attached to the cases in the darker coloured regions of dubiosity.

domain due to the distribution of the cases in the CB. There are only three solution classes and back away behaviors are mainly close to individualistic behaviors. Hence, in only 15% of the experiments we encountered this pattern when the proximity limits were chosen to be too wide and the proximity of the cases of all three classes were overlapping in some regions.

– Finally, the encountered Outer Limit DFPs were either in the proximity of isolated cases in deserted regions of the CB or they were on the outskirts of the proximity of cases which were themselves at the border of a class.

In Figure 4.6 a visualisation of an example of a dubious region in the Soccer domain is provided. Hollow and filled squares represent respectively cases and DFPs with cooperative solutions. Analogously, hollow and filled stars represent respectively cases and DFPs with indivudualistic solutions. The visualisation of cases and DFPs was built using a force-directed graph-drawing algorithm where the repulsive force between two cases is proportional to their distance. The dotted line indicates the border of the dubiosity region. We have drawn dubiosity groups at two similarity levels by using two different values for δ' (dark and light colors in figure). Neighborhood lines have been omitted for facilitating the understanding of the figures.

Chapter 5

Conclusions and Future Directions

In this project, we have worked on applying introspective reasoning to CBR systems to improve their performance and have developed two different methods. The first method focuses on reasoning processes trying to identify and refine their deficiencies when the quality of a proposed solution proves to be poor. The method does this by dynamically assessing solution quality information from the user's revisions to the solutions proposed by the system. In that sense, the method can also be seen as a *reactive* maintenance in response to poor solution quality.

On the other hand, the second method takes steps in a *proactive* manner by pre-analyzing the performance of the system to find deficiencies against a set of possible future problems.

We have tested our methods on CBR applications dealing with three different tasks. These were design, classification and action selection tasks on three different domains, namely, gas treatment plant design, zoology and robosoccer.

Below we detail our conclusions for both methods and finally, we give some ideas that we have as future research directions.

5.1 Introspective CBR

In this work we have presented a new introspective model for autonomously improving the performance of a CBR system by reasoning about system problem solving failures. To achieve this goal, the introspective reasoner monitors the reasoning process, and when a poor solution is proposed it determines the causes of the failures, and performs actions that will affect future reasoning processes.

We have created a causal model of the correctly functioning retrieval and adaptation stages of CBR. Failures of a CBR system's reasoning process are modeled as conflicts between observed system performance and predictions from the causal model. The sources of these conflicts are identified and associated learning goals are fired, sometimes triggering multiple types of learning. As a result of the process, the CBR reasoning process is improved for future problem solving.

We have tested the introspective reasoner in a fielded industrial design application. Experiments show that the use of the introspective reasoner improved the performance of the system. Introspection-based refinements of retrieval knowledge reduced the number of low quality solutions; refinements to adaptation knowledge increased high quality solutions. Moreover, the combination of both was able to generate more high quality solutions.

Since our model of the CBR reasoning process is domain independent, it can be applied to other domains. The engineering effort for incorporating the metareasoning component to other domains would be concentrated on linking domain-specific aspects of the CBR reasoning process to the appropriate parts in the model (retrieval, adaptation, and revision models).

5.2 CBR Maintenance

We have introduced a novel method for identifying future low confidence regions given an existing case-base. The method is based on four steps: First, we explore the problem space to find dubious future problems, i.e. problems with low confidence solutions. Then, we exploit these problems to better locate them in the case-base within their future neighbourhood. Both steps use an evolutionary approach to scan the problem space of the domain. Next, to help the understanding of the regions where dubious future problems are located, we associate each problem with one of the six dubiosity patterns that we have defined. These patterns are based on the neighbourhood of future problems to the existing cases. Finally, we have proposed an algorithm for grouping dubious future problems according to these patterns to identify regions of dubiosity. We argued that these regions enabled us to focus on the regions in the case-base that suffer from the same deficiency rather than dealing with individual problems, thus facilitating maintenance tasks.

We described the experiments performed in Zoology domain and in a Robosoccer application. We have shown how DFPs associated with dubiosity patterns helped us to detect dubious regions in the case-base and to analyse bad performing mechanisms of the CBR system.

We believe that the proposed method is useful for improving the performance of CBR systems in a proactive fashion. The proposed method uses only the domain ontology for generating future problems and evaluates them by using the confidence and similarity measures provided by the CBR system.

Although this method seems to be for maintenance tasks, we note that it can be used for helping the sytem designer and knowledge engineer at the design step of the CBR system as well. Confronting the system with possible future problems can be an exhaustive test showing whether the case-base is seeded enough to cover the problem space of the domain and whether the reasoning mechanisms (e.g., similarity metric or adaptation mechanisms) is working as thought.

5.3 Publications

- Mulayim, O. and Arcos, J. L.: 2007, Exploring dubious future problems, *in* M. Petridis (ed.), *Twelfth UK Workshop on Case-Based Reasoning*, CMS Press, pp. 52–63.

- Arcos, J. L., Mulayim, O. and Leake, D.: 2008, Using introspective reasoning to improve cbr system performance, *in* M. T. Cox and A. Raja (eds), *Proceedings of the AAAI 2008 Workshop on Metareasoning: Thinking About Thinking*, AAAI Press, pp. 21–28.

- Mulayim, O. and Arcos, J. L.: 2008, Understanding dubious future problems, *in* D. Althoff, R. Bergmann, M. Minor and A. Hanft (eds), *Advances in Case-Based Reasoning: 9th European Conference, ECCBR 2008*, Vol. 5239 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 385–399.

5.4 Future directions

Surely introspective reasoning has many dimensions, each of them still subject to full time research. And our work is by no means an exception. The challenges lie in a wide range of problems to be addressed (a list of challenges together with opportunities can be found in (Leake and Wilson 2008)).

For example, for learning systems in general –and for introspective learning in particular– it is still an open question when to learn once a failure is detected. This issue is very relevant to our first method where the metareasoning component tries to learn immediately as a poor solution is encountered. Instead of learning from the failure instantly, for some situations it may be wiser to wait to gather more information or perhaps at the end simply not learning from the failure since it proves to be a mere exception.

Another important issue arises when there are multiple repairs to be done. Leake, Kinley and Wilson (1997) show that uncoordinated repairs of multiple knowledge sources (such as similarity assessment and adaptation knowledge for CBR process) may degrade performance. This concerns our introspective reasoner as the hypothesis generation component may propose multiple improvements in response to a failure. For now the improvements are proposed using a predefined taxonomy. Learning this taxonomy and how to coordinate the multiple improvements would make the introspective reasoner much more domain –and implementation– independent.

As for the second method we introduced, finding dubious future problems and dubiosity regions is only half way to autonomusly improve CBR performance. At the moment, the user has to deal with the dubiosity regions applying them appropriate maintenance tasks manually. As future work we plan to relate the dubiosity patterns to possible maintenance tasks as we did in the first method by using taxonomy of reasoning failures. Furthermore, we wish to design a graphical tool for navigating through the problem space. We plan to join our method with a visualisation method for case-base competence based on solution qualities presented in (Grachten, Garcia-Otero and Arcos 2005).

Our ongoing research is aimed at answering these issues and ultimately we wish to combine our two methods as a contribution to research on building more introspective Case-Based Reasoners.

Bibliography

Aamodt, A. and Plaza, E.: 1994, Case-based reasoning: foundational issues, methodological variations, and system approaches, *AI Communications* **7**(1), 39–59.

Anderson, M. L. and Oates, T.: 2007, A review of recent research in metareasoning and metalearning, *AI Magazine* **28**, 7–16.

Arcos, J. L.: 2001, T-air: A case-based reasoning system for designing chemical, *in* D. W. Aha and I. Watson (eds), *Case-Based Reasoning Research and Development*, Vol. 2080 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 576–588.

Arcos, J. L.: 2004, Improving the quality of solutions in domain evolving environments, *in* P. Funk and P. A. Conzález-Calero (eds), *Proceedings of the 7th European Conference on Case-Based Reasoning*, Vol. 3155 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 464–475.

Arcos, J. L., Mulayim, O. and Leake, D.: 2008, Using introspective reasoning to improve CBR system performance, *in* M. T. Cox and A. Raja (eds), *Proceedings of the AAAI 2008 Workshop on Metareasoning: Thinking About Thinking*, AAAI Press, pp. 21–28.

Asuncion, A. and Newman, D.: 2007, UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html

Birnbaum, L., Collins, G., Brand, M., Freed, M., Krulwich, B. and Pryor, L.: 1991, A model-based approach to the construction of adaptive case-based planning systems, *in* R. Bareiss (ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, San Mateo: Morgan Kaufmann, pp. 464–475.

Cheetham, W.: 2000, Case-based reasoning with confidence, *in* E. Blanzieri and L. Portinale (eds), *5th European Workshop on Case-Based Reason-*

ing, EWCBR-00, Vol. 1898 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 15–25.

Cheetham, W. and Price, J.: 2004, Measures of solution accuracy in case-based reasoning systems, *in* P. Funk and P. A. Conzález-Calero (eds), *7th European Conference on Case-Based Reasoning, ECCBR-04*, Vol. 3155 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 106–118.

Cover, T. and Hart, P.: 1967, Nearest neighbor pattern classification, *IEEE Transactions on Information Theory* **13**(1), 21–27.

Cox, M. T.: 2005, Metacognition in computation: A selected research review, *Artificial Intelligence* **169**(2), 104–141.

Cox, M. T. and Raja, A.: 2007, Metareasoning: A manifesto. technical report, bbn tm-2028.

Cox, M. T. and Ram, A.: 1999, Introspective multistrategy learning: On the construction of learning strategies, *Artificial Intelligence* .

Craw, S., Wiratunga, N. and Rowe, R. C.: 2006, Learning adaptation knowledge to improve case-based reasoning, *Artificial Intelligence* **170**(16-17), 1175–1192.

Delany, S. J., Cunningham, P., Doyle, D. and Zamolotskikh, A.: 2005, Generating estimates of classification confidence for a case-based spam filter, *in* H. Muñoz-Avila and F. Ricci (eds), *6th International Conference on Case-Based Reasoning, ICCBR-05*, Vol. 3620 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 177–190.

Fox, S. and Leake, D. B.: 2001, Introspective reasoning for index refinement in case-based reasoning, *ournal of Experimental & Theoretical Artificial Intelligence* **13**(1), 63–88.

Francis, A. G. and Ram, A.: 1993, The utility problem in case-based reasoning, *Proceedings of the 1993 AAAI Workshop on Case-Based Reasoning.*

Grachten, M., Garcia-Otero, A. and Arcos, J. L.: 2005, Navigating through case base competence, *in* H. Munoz-Avila and F. Ricci (eds), *6th International Conference on Case-Based Reasoning, ICCBR-05*, Vol. 3620 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 282–295.

Kolodner, J.: 1993, *Case-based reasoning*, Morgan Kaufmann, San Francisco, CA, USA.

Leake, D. B.: 1996a, *Case-Based Reasoning: Experiences, Lessons and Future Directions*, MIT Press, Cambridge, MA, USA.

Leake, D. B.: 1996b, Experience, introspection and expertise: Learning to refine the case-based reasoning process, *Journal of Experimental and Theoretical Artificial Intelligence* **8**(3-4), 319–339.

Leake, D. B., Kinley, A. and Wilson, D.: 1995, Learning to improve case adaptation by introspective reasoning and CBR, *in* M. M. Veloso and A. Aamodt (eds), *Proceedings of the First International Conference on Case-Based Reasoning*, Vol. 1010 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 229–240.

Leake, D. B., Kinley, A. and Wilson, D. C.: 1997, Learning to integrate multiple knowledge sources for case-based reasoning, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 1997*, Morgan Kaufmann, pp. 246–251.

Leake, D. B. and Wilson, D. C.: 1998, Categorizing case-base maintenance: Dimensions and directions, *in* B. Smyth and P. Cunningham (eds), *Advances in Case-Based Reasoning:Proceedings of the Fourth European Workshop on Case-Based Reasoning*, Vol. 1488 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 196–207.

Leake, D. and Wilson, M.: 2008, Extending introspective learning from self-models, *Proceedings of the AAAI 2008 Workshop on Metareasoning: Thinking About Thinking*, AAAI Press, pp. 143–146.

Lopez de Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M. and Watson, I.: 2005, Retrieval, reuse, revision, and retention in CBR, *The Knowledge Engineering Review* **20**(3), 215–240.

Massie, S., Craw, S. and Wiratunga, N.: 2007, When similar problems don't have similar solutions, *in* R. Weber and M. Richter (eds), *7th International Conference on Case-Based Reasoning, ICCBR 2007*, Vol. 4626 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 92–106.

Michalewicz, Z.: 1996, *Genetic Algorithms+Data Structures=Evolution Programs*, 3rd edn, Springer Verlag, New York.

Mulayim, O. and Arcos, J. L.: 2007, Exploring dubious future problems, *in* M. Petridis (ed.), *Twelfth UK Workshop on Case-Based Reasoning*, CMS Press, pp. 52–63.

Mulayim, O. and Arcos, J. L.: 2008, Understanding dubious future problems, *in* D. Althoff, R. Bergmann, M. Minor and A. Hanft (eds), *Advances in Case-Based Reasoning: 9th European Conference, ECCBR 2008*, Vol. 5239 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 385–399.

Murdock, J. W. and Goel, A. K.: 2008, Meta-case-based reasoning: self-improvement through self-understanding, *Journal of Experimental & Theoretical Artificial Intelligence* **20**(1), 1–36.

Ros, R.: 2008, *Action Selection in Cooperative Robot Soccer Using Case-Based Reasoning. Ph.D. Diss.*, Universidad Autonoma de Barcelona, Spain.

Ros, R., Lopez de Mantaras, R., Arcos, J. L. and Veloso, M.: 2007, Team playing behavior in robot soccer: A case-based approach, *in* R. O. Weber and M. M. Richter (eds), *Proceedings of the 7th International Conference on Case-Based Reasoning, ICCBR-07*, Vol. 4626 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 46–60.

Smyth, B. and Cunningham, P.: 1996, The utility problem analysed: A case-based reasoning perspective, *Proceedings of the Third European Workshop on Case-Based Reasoning*, Springer Verlag, pp. 392–399.

Smyth, B. and Keane, M. T.: 1995, Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems, *Proceedings of IJCAI-95*, pp. 377–382.

Smyth, B. and McKenna, E.: 1999, Building compact competent case-bases, *in* K.-D. Althoff, R. Bergmann and K. Branting (eds), *3rd International Conference on Case-Based Reasoning, ICCBR-99*, Vol. 1650 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 329–342.

Smyth, B. and McKenna, E.: 2001, Competence models and the maintenance problem, *Computational Intelligence* **17**(2), 235–249.

Watson, I.: 1999, CBR is a methodology not a technology, *Knowledge Based Systems* **12**(5,6), 303–308.

Wilson, D. C. and Leake, D. B.: 2001, Maintaining case-based reasoners: Dimensions and directions, *Computational Intelligence* **17**(2), 196–213.

Wilson, T. and Schooler, J.: 1991, Thinking too much: Introspection can reduce the quality of preferences and decisions, *Journal of Personality and Social Psychology* **60**, 181–192.