Exploring Dubious Future Problems

Oğuz Mülâyim and Josep Lluís Arcos

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain
{oguz,arcos}@iiia.csic.es

Abstract. In Case-Based Reasoning systems, the election of the appropriate cases for the case base and the design of an accurate similarity measure are crucial issues. Case-Base Maintenance techniques provide a way to improve the case base mainly by identifying and deleting the cases that produce noise in a system. In this paper we present a novel method for identifying possible future problems with low confidence solutions based on evolutionary techniques. We call these problems dubious future problems. Moreover, some experiments are provided for illustrating the usage of the method.

Key words: Case-Based Reasoning, Case Base Maintenance, Confidence

1 Introduction

A Case-Based Reasoning(CBR) system solves new problems by reusing the solutions of similar previously solved problems. A case base representative of the problems to be solved and an accurate design of a similarity measure are crucial for fitting the CBR hypothesis that states *"similar problems have similar solutions"*.

With the increasing complexity of CBR systems, several techniques for the analysis of the case base structure and its relation with the similarity measure used in the retrieval phase have been proposed [1, 2]. The techniques known as Case-Base Maintenance (CBM) have demonstrated their capacity for improving the competence of CBR systems. Since in many applications the correctness of the cases cannot be assured (either because a part of the problem description may be unknown/incorrect or the solution wrongly labeled), existing CBM techniques are focused on improving the systems competence by means of deleting some cases of the case base [3–5]. For instance, in [5] a complexity measure for highlighting areas of uncertainty within the problem space is proposed.

The assumption of CBM techniques is that the analysis of the cases provided in the case base is a good approach for estimating the performance of the system in future cases. Nevertheless, new problems to be solved will be slightly different to the existing cases. Thus, the possibility of systematically assessing the performance of a system in a set of *possible* future problems becomes an interesting issue.

The use of confidence measures on the solutions provided by a CBR system provides an assessment about how sure the system is about the solution it proposes [6]. The importance of the availability of such a confidence measure is emphasised throughout recent research introducing possible confidence calculations, e.g. [7, 8].

With the importance of confidence in a CBR system in mind, we came up with the intuitive idea behind our proposal: If one could foresee the confidence of a CBR system for possible future problems, he/she could detect future fallacies and hence make improvements to leverage the overall confidence of the system in a proactive fashion. We call future problems with low confidence solutions *dubious future problems.*

In this work, we present an introspective method inspired on evolutionary techniques to detect problematic zones(in terms of confidence) in the problem space given an existing case base and a similarity measure. To effectively scan the problem space, we conducted the search for dubious future problems in two steps: First, we explored the problem space defined by the case base to find dubious future problems. Then, we carried out an exploitation phase to better locate these problems in the CB within their future neighbourhoods.

In this work we describe how this search is achieved assuming the availability of the following knowledge in a CBR system:

– a domain ontology associated with the case base;
– a similarity metric; and
– a confidence measure that attaches a confidence value for each solution provided by the CBR system.

The paper is organized as follows: Section 2 describes the details of the evolutionary method we are proposing and its two steps: Exploration and Exploitation. Section 3 presents experimental results. Conclusions and future work are presented in Section 4.

2 Exploration and Exploitation with an Evolutionary Approach

Given a domain ontology associated with a CBR system, we are interested in identifying possible future problems that are similar enough to the current cases and that the confidence on their solutions provided by the CBR system is low. Depending on the features that characterise a domain, the problem space can be too vast or even infinite. To tackle the task with an analytical approach by analysing the mathematical properties of the similarity measure could be a theoretical alternative but in real applications, where the similarity measure usually becomes a complex calculation, it cannot be performed easily. Genetic Algorithms [9, 10] (GA) have demonstrated their capabilities for exploring vast search spaces. They have the advantage of scanning the search space in a parallel manner using a fitness function as heuristics for search. Inspired by the Darwin's theory of evolution, solution of a problem is evolved through a GA.

A GA is an iterative algorithm that starts with an initial set of possible solutions (represented by chromosomes) called *initial population*. In each iteration the population is transformed (by means of genetic operators such as selection, crossover, and mutation) into a new generation hoping that each new generation will be better than the previous population. The chance of being selected for a chromosome as a survivor and/or a parent (of offspring chromosomes) is determined by its fitness, i.e. its suitability as a solution for the problem. A GA runs until some termination criterion is satisfied like passing a number of iterations or satisfying a fitness threshold.

In the search for dubious problems, the search space is the space of all problems that can be generated according to the domain ontology (i.e. in our case we will be using GA to find problems not the solutions). Moreover, since we are looking for dubious problems, the lesser confident the CBR system is about a solution the more preferable candidate will be the problem.

With a diverse initial population and an appropriate fitness function, dubious problems would evolve throughout generations of the running GA. However, as commonly seen in practice, GAs might have a tendency to converge towards local optima [9]. In our approach, 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 is to divide the search into two steps, namely Exploration and Exploitation of dubious future problems. In the Exploration step, the aim is to find dubious future problems 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 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 dubious future problems 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 dubious future problems encountered in the Exploration step for providing a more precise analysis of the low confidence local regions. In the Exploitation step, existing cases in the case base with low confidence solutions are also involved in the search.

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 dubious future problems and their neighbours. We will explain both limits in detail for each step in the next sub-sections. See Figure 1 for examples of proximity limits in the exploration step.

In many genetic algorithms, it is a good practise to maintain the diversity in the population in each generation or after a certain number of generations. This need may arise, for example, in the case where we have many similar chromosomes in a population due to successive crossover between similar parents or when we do not observe a significant improvement at the fitness of the chromosomes after a number of generations. Then, the diversity can be regained by reducing the number of similar chromosomes and by replacing a group of chromosomes with new ones respectively. 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.

2.1 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. For the Exploration step, the proximity limits mentioned above define the preferred region of the search for dubious problems. The outer limit defines the border for the less similar problems, while the inner limit 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 dubious future problems since we are looking for dubious future problems as diverse they are as possible in this step. A graphical representation of the Exploration step is provided in Figure 1.

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 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 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 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.

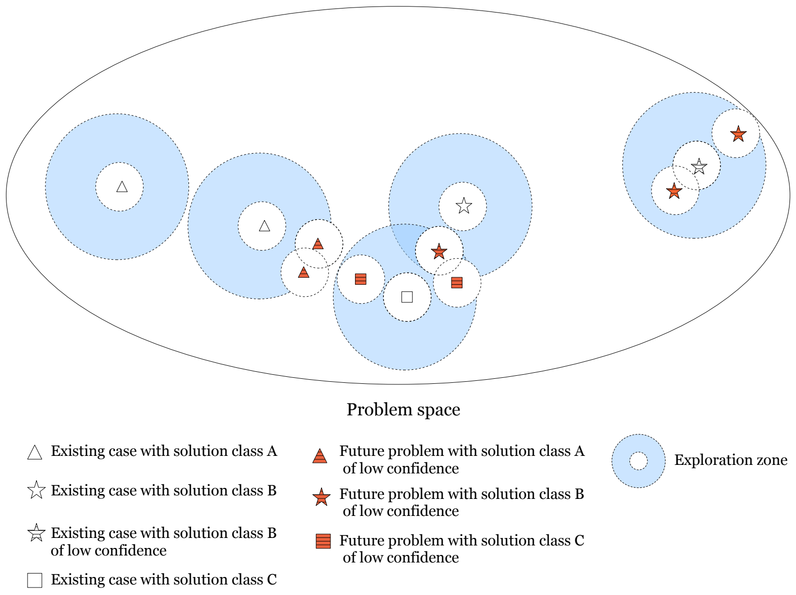
Problem space
Existing case with solution class A
Existing case with solution class B
Existing case with solution class B of low confidence
Existing case with solution class C
Future problem with solution class A of low confidence
Future problem with solution class B of low confidence
Future problem with solution class C of low confidence
Exploration zone

Fig. 1. Graphical representation of the Exploration step.

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 (supplied by the similarity metric) 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; $Confidence(c)$ is the confidence value supplied by the CBR application after solving c; and $SimilarityFactor$ is calculated as follows:

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

where $partSimEC(c)$ refers to the similarity of c to existing cases and $partSimFP(c)$ refers to the similarity of c to previously found future problems. $partSimEC(c)$ is defined as:

$$partSimEC(c) = \begin{cases} 1 - (OB + IB - maxSim(c, CB)) & \text{if } maxSim(c, CB) \geq IB \\ 1 - maxSim(c, CB) & \text{otherwise} \end{cases}$$

where $maxSim(c, CB)$ is the similarity value of c to the most similar case in the CB, IB is the inner proximity threshold and OB is the outer proximity threshold. $partSimFP(c)$ is defined as:

$$partSimFP(c) = \sum_{p \in FP} (similarity(c, p) - IB)$$

where $FP \subset \mathcal{LCFP}$ is the set of future problems to which c is more similar than the allowed value IB 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 that for a non-desired one.

Moreover, when a chromosome satisfies the confidence threshold and the proximity limits for existing cases and found future problems we add it to the $\mathcal{LCFP}$ list.

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 the number of twins is calculated and, if it exceeded the diversity threshold, the twins 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, they are 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 $\mathcal{LCFP}$.

2.2 Exploitation

The goal of the Exploitation step is to explore the neighbourhood of the low-confidence problems discovered in the Exploration step in order to improve the analysis of the case base.

If there exist any retained cases in the CB with low confidence solutions we also take them into account for exploitation. We name the list of those cases as Low Confidence Existing Cases ($\mathcal{LCEC}$).

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 initialise this list with the members of the union set $\mathcal{LCEC} \cup \mathcal{LCFP}$. That is, the members of this list are the problems and cases 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 defines the border for the less similar problems, while the inner limit 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 2.

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 pre-

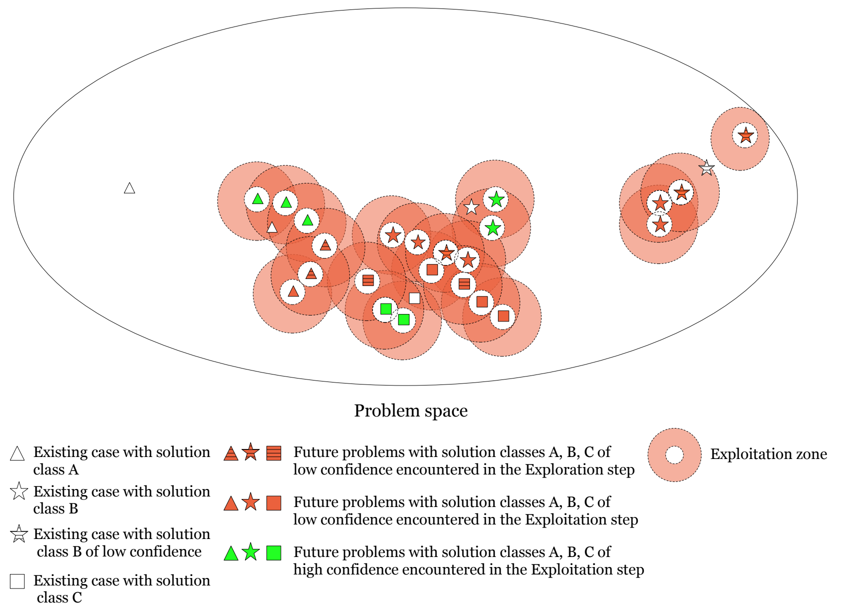
Problem space
Existing case with solution class A
Existing case with solution class B
Existing case with solution class B of low confidence
Existing case with solution class C
Future problems with solution classes A, B, C of low confidence encountered in the Exploration step
Future problems with solution classes A, B, C of low confidence encountered in the Exploitation step
Future problems with solution classes A, B, C of high confidence encountered in the Exploitation step
Exploitation zone

Fig. 2. Graphical representation of the exploitation step.

viously given in the Exploration step.

Initial Population: We partially feed the initial population with the union set of $\mathcal{LCEC} \cup \mathcal{LCFP}$ hoping to reproduce similar problems and we use the Random-Problem-Generator to reach to the wanted initial population size.

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:

$$Fitness(c) = \begin{cases} 1 - (OB + IB - maxSim(c, \mathcal{LCPN})) & \text{if } maxSim(c, \mathcal{LCPN}) \geq IB \\ 1 - maxSim(c, \mathcal{LCPN}) & \text{otherwise} \end{cases}$$

where $maxSim(c, \mathcal{LCPN})$ is the similarity value of c to the most similar problem in $\mathcal{LCPN}$.

When a chromosome satisfies the proximity limits for a member of $\mathcal{LCPN}$ or if it is a dubious future problem we add it to the $\mathcal{LCPN}$ list.

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 $\mathcal{LCPN}$ in which we both have dubious future problems together with their neighbours whether they are themselves dubious or not.

3 Experimentation

To carry out a simple, yet a complete experiment, we used the Zoology dataset available from UCI machine learning repository which has 100 examples and 7 solution classes. The domain ontology explicitly 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.

For both Exploration and Exploitation, in each generation :

- 40% of the population was selected as survivors to the next generation using Fitness Proportionate Selection(FPS).
- 60% of the chromosomes were selected as parents using FPS and Single-Point Crossover was applied between each pair to reproduce offspring.
- Mutation was applied to the 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.
- We evaluated each chromosome in the population using fitness functions defined in subsections 2.1 and 2.2 . During evaluation we maintained the list $\mathcal{LCFP}$ in the Exploration and $\mathcal{LCPN}$ in the Exploitation.
- GAs were terminated after a given number of generations which was 50 or 80.

We used a k-NN-Retrieve method with an Euclidean metric and $k = 3$. The Similarity Metric returned us similarity values in the range `[0.0, 1.0]`. This metric calculated the minimum similarity between two cases in the Zoology domain as `0.3536`, where the maximium similarity was `0.9683`. So, we chose the test range `[0.93, 0.99]` for the proximity limit values in our experiments. We kept the proximity for Exploitation narrower than Exploration since in the former step we are exploiting future problems to have a better idea of their future neighbourhood.

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 [8] 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.

In the Table 1 we give some of the experimentation results for different GA settings, confidence thresholds and proximity limits. Where,

- PopSize : Population size for both GAs;
- GenrCnt : Number of generations we allow for the evolution in both GAs;
- ThrsConf : Threshold for Low Confidence;
- EC-IB : Inner limit of similarity for existing cases;
- EC-OB : Outer limit of similarity for existing cases;
- FP-IB : Inner limit of similarity for future problems;
- FP-OB : Outer limit of similarity for future problems;
- $\mathcal{LCEC}$: Number of the members of the $\mathcal{LCEC}$ list;
- $\mathcal{LCFP}$: 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;
- 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 had 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 $\mathcal{LCFP}$ list 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.

Table 1. Experimental Results

PopSize	GenrCnt	ThrsConf	EC-IB	EC-OB	FP-IB	FP-OB	$\mathcal{LCEC}$	$\mathcal{LCFP}$	$\mathcal{LCPN}$
100	50	2.0	0.98	0.95	0.99	0.96	7	6	108
”	”	”	”	0.94	”	”	7	10	141
”	”	”	0.97	0.93	”	”	7	21	176
”	”	”	0.96	”	”	”	7	21	165
150	”	”	”	”	”	”	7	29	238
100	80	”	”	”	”	”	7	27	225
”	50	1.0	”	”	”	”	2	26	174
”	50	”	0.98	0.94	”	”	2	11	159
”	80	”	0.97	0.93	”	0.95	2	25	140
”	”	”	”	0.94	0.98	0.96	2	12	123
150	”	”	”	”	”	”	2	20	243
100	80	”	”	”	”	”	2	7	235

4 Conclusions

In this paper we presented a novel method for identifying future low confidence regions given an existing case base. We have proposed an evolutionary approach for exploring the problem space. We achieved the exploration by conducting a search in two steps: first, we explored the problem space defined by the case base to find dubious future problems; next, we carried out an exploitation phase to better locate the problems in the CB within their neighbourhoods.

We described the experiments performed with a classification dataset and provided some hints about how to tune the method parameters according to the systems designer interests.

We believe that the introduced method can be used in most, if not all, of the CBR applications in which it is possible to generate future problems using the domain ontology and evaluate them using the confidence and similarity measures provided by the CBR system.

The next step in our work is to characterise the regions identified by the dubious future problems for providing a more abstract analysis tool. The goal is to analyse neighbour dubious problems trying to characterise them according to a collection of patterns like holes, borders etc. These patterns may lead to an automatised maintenance or at least a collection of guidelines for a manual maintenance by a domain expert. For instance, a dubious problem found near existing cases of another class could be indicative of an erroneous reuse.

As a future work, we also plan to design a graphical tool for navigating through the problem space. We plan to join the method described in this paper with a visualisation method for case base competence based on the solution qualities presented in [11].

Acknowledgements. 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 de Catalunya under the grant 2005-SGR-00093.

References

1. Smyth, B., Keane, M.T.: Remenbering to forget: A competence-preserving case delection policy for case-based reasoning systems. In: Proceedings of IJCAI-95. (1995) 377–382
2. Wilson, D.C., Leake, D.B.: Maintaining case-based reasoners: Dimensions and directions. Computational Intelligence **17**(2) (2001) 196–213
3. Smyth, B., McKenna, E.: Building compact competent case-bases. In Branting, K., Althoff, K.D., eds.: Case-Based Reasoning. Research and Development. ICCBR'99. Number 1650 in Lecture Notes in Artificial Intelligence. Springer-Verlag (1999) "329–342"
4. Smyth, B., McKenna, E.: Competence models and the maintenance problem. Computational Intelligence **17**(2) (2001) 235–249
5. Massie, S., Craw, S., Wiratunga, N.: When similar problems don't have similar solutions. In Weber, R.O., Richter, M.M., eds.: Case-Based Reasoning. Research and Development: 7th International Conference on Case-Based Reasoning, ICCBR 2007. Number 4626 in Lecture Notes in Artificial Intelligence. Springer-Verlag (2007) 92–106
6. Cheetham, W.: Case-based reasoning with confidence. In Blanzieri, E., Portinale, L., eds.: EWCBR. Volume 1898 of Lecture Notes in Computer Science., Springer (2000) 15–25
7. Cheetham, W., Price, J.: Measures of solution accuracy in case-based reasoning systems. In Funk, P., González-Calero, P.A., eds.: ECCBR. Volume 3155 of Lecture Notes in Computer Science., Springer (2004) 106–118
8. Delany, S.J., Cunningham, P., Doyle, D., Zamolotskikh, A.: Generating estimates of classification confidence for a case-based spam filter. In Muñoz-Avila, H., Ricci, F., eds.: ICCBR. Volume 3620 of Lecture Notes in Computer Science., Springer (2005) 177–190
9. Michalewicz, Z.: Genetic Algorithms+Data Structures=Evolution Programs. 3rd edn. Springer Verlag, New York (1996)
10. Mitchell, T.M.: Machine Learning. McGraw-Hill Higher Education (1997)
11. Grachten, M., Garca-Otero, A., Arcos, J.L.: Navigating through case base competence. In Munoz-Avila, H., Ricci, F., eds.: Case-Based Reasoning Research and Development: 6th International Conference on Case-Based Reasoning, ICCBR 2005. Volume 3620 of Lecture Notes in Artificial Intelligence. Springer-Verlag (2005) 282–295