Predicting Dubiosity in CBR Systems

Oğuz Mülâyim and Josep Lluís Arcos

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

Abstract. To be able to guide the design and maintenance of Case-Based Reasoning (CBR) systems, we present a novel and domain independent method based on evolutionary techniques, for anticipating the performance of a system against a set of possible future problems by identifying low confidence regions in its case-base. Moreover, a simple experimentation is provided for illustrating the method.

1 Introduction

The techniques known as Case-Base Maintenance (CBM) (see [1] for CBM dimensions) have proved useful for improving the performance of CBR systems. Till today, most of the research on CBM is primarily focused on the removal of redundant or inconsistent cases while preserving the system competence [2, 3] and improving the system accuracy [4].

The common assumption of CBM techniques has been that analysis of the cases provided in a case-base (CB) is a good approach for estimating the performance of the system for future cases (a.k.a. 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 possible future problems different from the existing cases becomes an interesting issue. This preanalysis would give important clues about possible future system deficiencies and in turn –and most importantly– yielding repair opportunities in a proactive fashion.

In this paper, we propose a method inspired on evolutionary techniques to detect future problems for which a CBR system is not confident of its solutions. We call these future problems with low confidence solutions *Dubious Future Problems* (DFPs). In particular, given 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 [5–7] that attaches a confidence value to each solution proposed by the CBR system.

The search space for dubious problems is the space of all problems that can be generated according to the domain ontology. To find DFPs we use Genetic Algorithms (GA) as they have the advantage of scanning the search space in a parallel manner using fitness functions as heuristics and their implementations can be domain independent. In search for future system deficiencies, we believe that confidence measures can be used as effective heuristics as they indicate possible reasoning failures and/or lack of domain knowledge in the case-base.

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 [8]. 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.

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 DFPs.

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 problems which have no neighbour cases in the CB. Whereas, the dissimilarity between DFPs is preferred for obtaining diversity in the results of Exploration as a better basis for 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 steps incorporate two proximity limits in terms of similarity to an existing case or a future problem. These limits define the preferred regions in the problem space during the search for DFPs and their neighbours.

The following sections describe the details of the Exploration and Exploitation steps, respectively. Section 4 presents experimental results. Conclusions and future work are presented in Section 5.

2 Exploration of DFPs

The goal of the Exploration step is to identify an initial set of diverse dubious problems similar enough to the cases defined in a CB. The confidence threshold that defines a problem as dubious is domain dependent since the degree of being confident about a solution may vary for each domain or CBR system.

A graphical representation of the Exploration step is provided in Figure 1. The outer proximity 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. The inner limit is used also to draw a border around the found DFPs for the sake of diversity among DFPs.

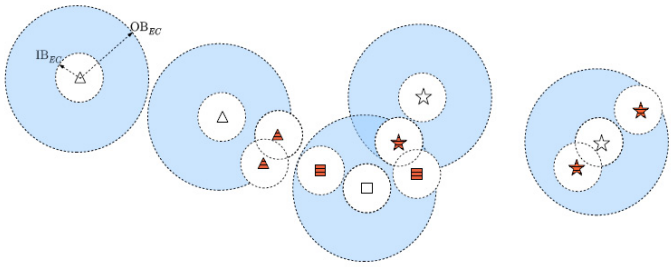

Fig. 1. Graphical representation of the Exploration step. Hollow shapes are existing cases (where different shapes refer to different classes); filled shapes are the encountered Dubious Future Problems; IB_{EC} and OB_{EC} are, respectively, inner and outer bounds.

The decision of the proximity limits depends on the answer to the question of how similar a problem should be to a case to be regarded as a relevant problem for a particular domain and application.

Throughout the execution of the GA for Exploration, we maintain a list of encountered Low Confidence Future Problems $\mathcal{LCFP}$. During the evaluation of a population, each time we come across a chromosome representing a different DFP we add it to the $\mathcal{LCFP}$ list.

The concepts used in the GA for the Exploration step are explained below:
Chromosomes: Each chromosome 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 ontology) 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 according to the domain ontology. In the existence of domain constraints, the RPG function generates valid problems that conform to those constraints. The size of the population depends on the problem space defined by the domain.
Fitness Function: The fitness of a chromosome is determined by two parameters: 1) the confidence value of the solution to the problem represented by the chromosome and 2) the similarity of the problem to the nearest case in the CB or to a previously found DFP.

The fitness function has to be adapted in each different domain or CBR system. However, the following guidelines for fitness can be used in Exploration: 1) The lower the confidence value is for a chromosome, the better candidate is that chromosome; 2) a chromosome in the preferred proximity of an existing case is a better candidate than a chromosome which is not in this proximity; and 3) the confidence factor of the fitness is more significant than the similarity factor since we are searching for dubiosity.

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 $SimilarityFactor$ takes into account the similarity to both cases and DFPs. $SimilarityFactor$ is calculated as follows:

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

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

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

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 guidelines defined above, $SimilarityFactor$ penalizes the chromosomes that are too close to either cases or future problems discovered in previous iterations. It should also be noted that for a desired chromosome our proposed function produces a fitness value which is lower than for a non-desired one.

Selection: We use a fitness-proportionate selection where each chromosome has a chance inversely proportional (since we are interested in chromosomes with lower fitness values) to its fitness value to be selected as a survivor and/or parent for the next generations.

Crossover: We use single-point crossover.

Mutation: One random gene value is altered for a given number of randomly chosen offspring chromosomes in the population.

Diversity Preservation: At each generation when the number of twins exceeds a given 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) and replaced by new problems generated by the RPG function.

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

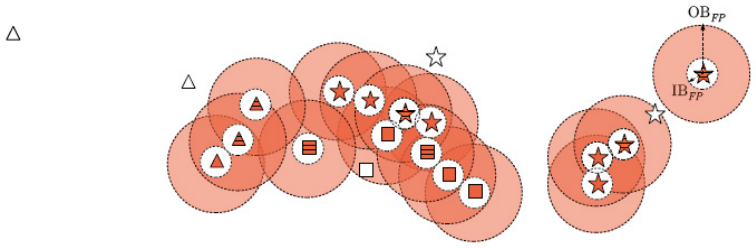

Fig. 2. Graphical representation of the Exploitation step. Hollow shapes are existing cases; filled shapes are the encountered and exploited Dubious Future Problems. IB_{FP} and OB_{FP} are, respectively, inner and outer bounds.

termination of the Exploration step. In the current implementation we adopted this last solution.
Termination: We let the population evolve for a certain number of generations.
Result: We obtain the list of DFPs $\mathcal{LCFP}$ maintained during GA execution.

3 Exploitation of DFPs

The goal of the Exploitation step is to explore the neighbourhood of the DFPs discovered in 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 this list with the members of the $\mathcal{LCFP}$.

For the Exploitation phase, the outer proximity 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 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 found 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, Termination: These concepts have the same definitions as the corresponding ones 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:

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

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.
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 Experimentation

We used the Zoology dataset (available from the UCI machine learning repository [9]) 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. 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 (e.g. `Hair`, `Feathers`, `Fins`) are boolean valued and 1 of them is an enumeration (`Legs`). 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.

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%.

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. Moreover, we kept the proximity for Exploitation narrower than Exploration.

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 [7] and the range for confidence was `[0.0, 3.0]`.

Trying different settings for inner and outer proximity limits for both cases and future problems, GA population sizes and number of generations for evolution, we wanted to see how the GAs evolved with $\mathcal{LCFP}$ and $\mathcal{LCPN}$ lists as their by-products. Furthermore, because of the random nature of GAs, for each setting we repeated the experiment several times to get an average value for the number of the members of the lists.

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

The results show that we may encounter a higher number of dubious future problems and their neigbours when: 1) the initial population is richer in size; 2) GAs evolve during more generations; 3) the area within proximity limits is wider; and 4) the threshold of low confidence is high.

Table 1. Experimental Results for Zoology 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; $\mathcal{LCFP}$: Number of the members of the $\mathcal{LCFP}$ list; $\mathcal{LCPN}$: Number of the members of the $\mathcal{LCPN}$ list.

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

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.

5 Conclusions

In this paper we presented a novel method for identifying possible future low confidence regions in the case-base of a given CBR system. We proposed an evolutionary approach for exploring the problem space by generating possible future problems and using a confidence measure supplied by the system as a heuristic in their evaluation. 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 where 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 characterize the regions identified by the discovered dubious future problems for providing a more abstract analysis tool.

The goal is to analyze neighbour dubious problems trying to characterize them according to a collection of patterns like holes, borders etc. These patterns may give hints for maintenance tasks allowing an improvement of the overall confidence of the system. 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. In particular, we plan to join the method described in this paper with a visualization method for case-base competence based on the solution qualities presented in [10].

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. This work has been done in the framework of the Doctoral Program in Computer Science of the Universitat Autònoma de Barcelona.

References

1. Wilson, D.C., Leake, D.B.: Maintaining case-based reasoners: Dimensions and directions. Computational Intelligence **17**(2) (2001) 196–213
2. Smyth, B., Keane, M.T.: Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In: Proceedings of IJCAI-95. (1995) 377–382
3. Smyth, B., McKenna, E.: Competence models and the maintenance problem. Computational Intelligence **17**(2) (2001) 235–249
4. Massie, S., Craw, S., Wiratunga, N.: When similar problems don't have similar solutions. In Weber, R., Richter, M., eds.: 7th International Conference on Case-Based Reasoning, ICCBR 2007. Volume 4626 of Lecture Notes in Artificial Intelligence. Springer-Verlag (2007) 92–106
5. Cheetham, W.: Case-based reasoning with confidence. In Blanzieri, E., Portinale, L., eds.: 5th European Workshop on Case-Based Reasoning, EWCBR-00. Volume 1898 of Lecture Notes in Artificial Intelligence. Springer-Verlag (2000) 15–25
6. Cheetham, W., Price, J.: Measures of solution accuracy in case-based reasoning systems. In Funk, P., Conzález-Calero, P.A., eds.: 7th European Conference on Case-Based Reasoning, ECCBR-04. Volume 3155 of Lecture Notes in Artificial Intelligence. Springer-Verlag (2004) 106–118
7. 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.: 6th International Conference on Case-Based Reasoning, ICCBR-05. Volume 3620 of Lecture Notes in Artificial Intelligence. Springer-Verlag (2005) 177–190
8. Michalewicz, Z.: Genetic Algorithms+Data Structures=Evolution Programs. 3rd edn. Springer Verlag, New York (1996)
9. Asuncion, A., Newman, D.: UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html. (2007)
10. Grachten, M., Garcia-Otero, A., Arcos, J.L.: Navigating through case base competence. In Munoz-Avila, H., Ricci, F., eds.: 6th International Conference on Case-Based Reasoning, ICCBR-05. Volume 3620 of Lecture Notes in Artificial Intelligence. Springer-Verlag (2005) 282–295