Novi Sad J. Math.
Vol. 38, No. 3, 2008, 219-226

CASE-BASED REASONING FRAMEWORK FOR GENERATING DECISION SUPPORT SYSTEMS[1]

Vladimir Kurbalija,[2] Zoran Budimac[2]

Abstract. Case-Based Reasoning (CBR) is a relatively new and promising technique of artificial intelligence. Using CBR, every new problem is solved by adapting the solutions of the previously successfully solved similar problems. The intention of our research is to develop a robust and general framework which supports generation of wide-range of CBR systems using different approaches. Presented framework integrates two previously developed CBR shells: *CaBaGe* and *CuBaGe*. *CaBaGe* (Case Base Generator) is a CBR shell for generating arbitrary decision support systems where the cases and the problems are represented as a set of values of some selected, most important attributes. *CuBaGe* (Curve Base Generator) is also a CBR shell in which both the problem and the previous cases are presented in the graphical manner using curves or time-series. Presented framework, which encompasses these two shells, inherits a number of advantages including: domain independence, incremental learning, platform independence, fast retrieval algorithm, generality, and robustness.

AMS Mathematics Subject Classification (2000): 68T20, 68T30

Keywords and phrases: Case-Based Reasoning, Knowledge Representation, Decision Support Systems

1. Introduction

Case-Based Reasoning (CBR) has become a successful technique applicable in different knowledge-based systems and different domains. This promising technique is based on the use of previous experience in a form of cases to understand better and to solve new problems in particular domain. The main idea arises from the fact that in many particular domains similar problems usually have similar solutions.

In the last several years, in the area of CBR technology at our Department, two independent shells dealing with different concepts and approaches of CBR have been developed: *CaBaGe* [9] and *CuBaGe* [10]. The main difference between them is the form of case representation. In the *CaBaGe* shell the case is represented as a set of values of chosen attributes, while in the *CuBaGe* shell

[1]This paper is partially supported by the project: Abstract Methods and Applications in Computer Science, Project no. 144017A, Ministry of Science and Technological Development of the Republic of Serbia (2006-2010)

[2]Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Serbia, e-mail: {kurba, zjb}@dmi.uns.ac.rs

the case is represented as a curve or as time series - set of appropriate points in some space.

The paper presents a unique and robust framework which supports usage of these two shells in order to analyze, compare, and implement different concepts of the CBR technique. This proposed framework will enclose a variety of case representation techniques and also different similarity measures defined for the corresponding case representation techniques.

The rest of the paper is organized as follows. The section 2 elaborates necessary concepts of CBR technique [3, 4, 8, 11, 12]. The shells *CaBaGe* and *CuBaGe* are briefly described in the third and fourth sections, respectively. In the fifth section some characteristics and possibilities for application of the proposed framework are discussed, while the sixth section concludes the paper.

2. Foundations of CBR

Case-Based Reasoning is a relatively new and promising area of artificial intelligence and it is also considered as a problem solving technique. It is used for solving problems in domains where experience plays an important role [4, 11, 12]. Generally speaking, CBR is applied for solving new problems by adapting solutions that worked for similar problems in the past. The main supposition here is that similar problems have similar solutions. The basic scenario for almost all CBR applications is: *In order to find a solution of an actual problem, one looks for a similar problem in an experience base, takes the solution from the past and uses it as a starting point to find a solution for the actual problem.*

In CBR systems experience is stored in a form of cases. The case is a recorded situation where the problem has been totally or partially solved, and it can be represented as an ordered pair $(problem, solution)$. The whole experience is stored in case base, and each case represents some previous episode where the problem was successfully solved.

The main problem in CBR is to find a good similarity measure - the measure that can tell in what extent the two problems are similar. In the functional way similarity can be defined as a function $sim : U \times CB \rightarrow [0, 1]$, where U refers to the universe of all objects (from a given domain), while CB refers to the case base (just those objects which were examined in the past and saved in the case memory). The higher value of the similarity function means that these objects are more similar [11].

The CBR system has not the only goal of providing solutions to problems but also of taking care of other tasks occurring when it is used in practice. The main phases of the CBR activities are presented in the *CBR-cycle* (see Figure 1) [1].

In the *retrieve* phase the most similar case (or k most similar cases) to the problem case, is retrieved, while in the *reuse* phase some modifications of the retrieved case has been done in order to provide better solution to the problem (case adaptation). As the case-based reasoning only suggests solutions, there may be a need for a correctness proof or an external validation. That is the

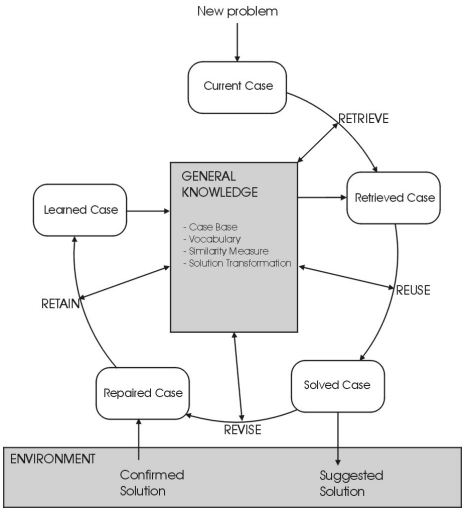

Figure 1: The *CBR-cycle* after Aamodt and Plaza (1994)

task of the phase *revise*. In the *retain* phase, the knowledge learned from this problem, is integrated in the system by modifying some knowledge containers.

The main advantage of this technology is that it can be applied to almost any domain. CBR system does not try to find rules between parameters of the problem; it just tries to find similar problems (from the past) and to use solutions of the similar problems as a solution of an actual problem. So, this approach is extremely suitable for less examined domains - for domains where rules and connections between parameters are not known. Furthermore, in more examined domains integration of CBR in classical rule-based reasoning systems brings some efficiency. The second very important advantage is that CBR approach to learning and problem solving is very similar to human cognitive processes - people take into account and use past experience to make future decisions.

3. *CaBaGe* - Case Base Generator

CaBaGe (Case Base Generator) is a general shell for generating an arbitrary decision support system based on CBR technique. In such shell previous cases as well as new problems (cases) are represented as a set of values of some selected, most important attributes. The shell was completely implemented in Java, mainly because it supports all concepts of object-oriented technology, but also because its main characteristic - platform independence.

The main advantage of this shell is that it is domain independent and can be used to instantiate appropriate decision-support system. For a particular domain of application, for which decision-support system (DSS) is instantiated, the user can select the most important and characteristic cases from the available set of previously solved cases. These cases are to be stored in an appropriate

form in the base of cases. According to that, the input to the DSS is a twofold one: the description of the case and the case base. Using these data, the DSS creates Case Retrieval Net (CRN) [11, 12] and it becomes capable to solve a new problem.

In the first input file - "Case Pattern File", the description of case is stored. Case pattern file contains the list of the attributes which represent essential features of the case, containing the name and the type of the attribute. The type of the attribute can be: *int*, *float* or *string*. Boolean type can be simulated, for example, with the string type where only two values ("Yes"/"No") are allowed. The number of the attributes is arbitrary, but all of them must be enlisted in the desired order.

The second file - "Case Base File", contains the list of, selected in advance, representative solved cases from the domain. Every case is described with the values of its attributes and with the final solution of that case. The final solution of the case is always listed at the end. Type of the solution can also be: *int*, *float* or *string* and it is determined dynamically, when all cases are parsed.

When the case base (i.e. appropriate text file) is created, the DSS generates a case retrieval net. The basic structure of CRN is given in Figure 2. Two main parts of the CRN are an array of attributes and a list of solutions. The array of attributes is created by using case pattern file, while the list of solutions is created from a case base file. Each value for each attribute represents one information entity, i.e. an ordered pair $(attribute, value)$. Each value (or information entity) contains a list of arcs to the solution nodes. The arc is pondered by its weight and has a pointer to the solution node; only the arcs whose weights are different from zero are saved.

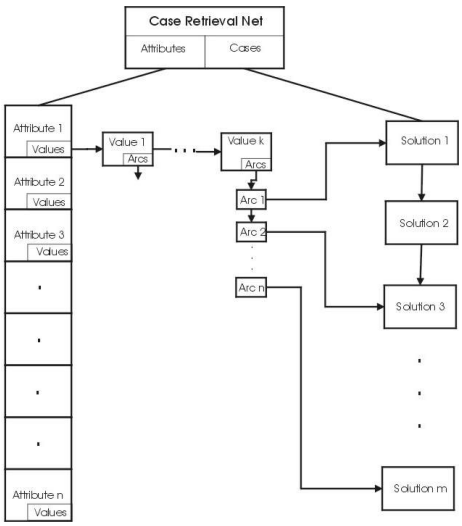

Figure 2: Structure of CRN

When the CRN is created, the DSS can be used for solving new cases (queries) from the domain. Since the new case (query) and the case from the case base have the same structure, the user has to enter the values of attributes in an appropriate input form. In order to better describe the problem, the user should enter all known values of the attributes although it is not necessary. The input form contains one more field for every attribute - importance. The importance is a value from the interval [0,1], and describes how much is the user sure in the validity of the value of the attribute he is entering. The queries where attributes have some additional information (like importance) are called *weighted queries*.

After entering the query, the DSS starts searching the case base in order to find a possible solution in the following way: *The information entities (attribute, value) that occur in the query are initially activated with the value of importance (weighted query). The activation is propagating through the arcs to the solution nodes, by multiplying the value of activation of the information entity node with the weight of the corresponding arc. Final activation of the solution nodes is calculated by summing all gained activations.*

CaBaGe shell is supposed to be used for generating wide range of decision support systems for different domains. This shell is successfully applied in medical domain, in diagnosis of Multiple sclerosis (MS) [6, 7, 9]. A set of attributes being important for diagnosis of MS (structure of the case) as well as data about previous patients (case base) are provided by medical experts from the Institute of Neurology, Novi Sad.

4. *CuBaGe* - Curve Base Generator

CuBaGe (Curve Base Generator) is another CBR shell in which both the problem and the previous cases are presented in a graphical manner [10]. The reasons for implementing such shell are that in many practical domains some decisions depend on behavior of time series, charts and curves [5]. The shell therefore analyses curves, compares them to similar curves from the past and possibly predicts the future behaviour of the current curve on the basis of the most similar curves from the past.

The main problem here, as almost in every CBR system, was to create a good similarity measure for curves, i.e. what is the function that can tell to what extent two curves are similar. Beforehand, it is necessary to choose an appropriate representation of the curve which is compatible with the desired similarity measure.

In many practical domains data are represented with the set of points - an ordered pair (x, y). Very often the pairs are (t, v) where t represents time and v represents some value in the time t. When the data is given in this way it can be represented graphically and when the points are connected they represent some kind of a curve. If the points are connected just with straight lines then they represent the linear interpolation, but if someone wants smoother curves then some other kind of interpolation with polynomials must be used. There was a choice between classical interpolating polynomial and cubic spline. The cubic

spline was chosen as the curve representation technique for two main reasons: its simple form and stability towards oscillation [10].

When the cubic spline is calculated for curves then one very intuitive and simple similarity (or distance - which is the dual notion for similarity) measure can be used. The distance between two curves can be represented as a surface between these curves. When the curves are given in simple, analytical form (such as polynomials) this surface can be easily calculated using the definitive integral. Furthermore, the calculation of the definitive integral for polynomials is a very simple and efficient operation.

Since *CuBaGe* is also implemented as a general shell, it could be used for a wide range of domains in which all cases can be represented by curves. The shell was successfully applied in a financial domain for prediction future payment process on the basis of previous payment and invoicing processes. However, this application of the *CuBaGe* shell is more complex and is beyond the scope of this paper. More information can be found in papers [13, 14].

Recently, as a next step of generalization, the developing of a new framework *FAP* (Framework for Analysis and Prediction) was started. This framework will include the *CuBaGe* as its special case, but also can include different curve representations, similarity measures and prediction algorithms. Currently, the *CuBaGe* is integrated in the *FAP*. The next step would be the integration of some well known curve representation techniques [5] such as: Discrete Fourier Transform, Discrete Wavelet Transform, Singular Value Decomposition, Piecewise Linear, and Piecewise Constant models, Adaptive Piecewise Constant Approximation, etc. The implementation of different similarity measures and prediction algorithms would make this framework robust and appropriate for detailed analysis of different approaches, as well as their combinations.

The framework is designed as a library of packages. The integration of newly implemented curve representations, similarity measures or prediction algorithms is designed to be as simple as possible, just by adding implemented class to a corresponding package.

5. Proposed Framework

The realized *CaBaGe* and *CuBaGe* are classical CBR shells [2]. They are realized for easy use by non-programmers. The user need not to know anything about the realization of particular shell. All he/she has to do is to prepare the input files in appropriate manner, in which the values for previous situations (cases) are stored. By using a prepared file with the information of previous cases and the file with data of a new case he/she can choose one of the proposed shells to generate a decision support system in a particular domain.

This proposed framework merges two independent CBR shells which support different approaches of the case representation and cover a wide range of possible domains of application.

This framework inherited numerous advantages of *CaBaGe* and *CuBaGe* shells:

- **Incremental learning.** The shells can gain the new knowledge simply by saving the successfully solved case in the base of previously solved cases (a textual input file).

- **Domain independence.** The shells calculate similarity measures just on the basis of previous cases from any domain. There is no need for some special domain knowledge (complex relations between attributes of the cases or values in the time series), nor for the expertise of domain experts.

- **Fast retrieval algorithm** The Case Retrieval Nets are used as a basic memory structure in *CaBaGe* since they have a very high retrieval performance. The solution is gained after just two iterations, no matter how big is the case base.

- **Generality and robustness** The architecture presented in the *CuBaGe* shell can be employed equally well in conventional time series (where all values are known), but also in some non-standard time series (sparse, vague, non-equidistant).

- **Platform independence.** The shells are implemented in Java, so they can work equally well on all kinds of platforms.

6. Conclusion

The framework could be useful for many reasons. Both of the shells support generation of a decision support system independent of the domain. So the resulting framework would be also independent of domain. Furthermore, the resulting framework would support generation of a greater variety of possible decision support systems: it will cover the domains where the situations could be specified with a set of attributes, but also the domains where the cases could be represented as curves or time series.

Additionally, the proposed framework could be useful as a good starting point and a core environment for future research. Some of possible utilizations could be done by:

- Developing new applications,

- Testing the performance of different approaches,

- Comparing various approaches of solving the same problem,

- Analyzing the results gained from testing and comparing...

References

[1] Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches, AI Commutations, pp. 39-58. 1994.

[2] An Object-Oriented Framework for the Design and the Implementation of Case-Based Reasoners, internet source: http://www-sop.inria.fr/aid/cbrtools/manual-eng/doc_web/Abstract98.html

[3] Budimac Z., Kurbalija V., Case-Based Reasoning - A Short Overview, Proc. of Second International conference on Informatics and Information Technology "Molika 2001", Molika, Macedonia, December 20–23, 2001, 222–234.

[4] Case-Based Reasoning at AIAI, http://www.aiai.ed.ac.uk/project/cbr/

[5] Chotirat Ann Ratanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, Gautam Das: MINING TIME SERIES DATA, chapter in Data Mining and Knowledge Discovery Handbook: Maimon, Oded; Rokach, Lior (Eds.), ISBN: 978-0-387-24435-8, Springer, 2005.

[6] Ivanović M., Kurbalija V., Budimac Z., Semnic M., Role of Case-Based Reasoning in Neurology Decision Support, Proceedings of the Fifth Joint Conference on Knowledge-Based Software Engineering "JCKBSE 2002", Maribor, Slovenia, September 11–13, 2002, 255–264.

[7] Kurbalija V., Ivanović M., Budimac Z., Semnic M., Multiple Sclerosis Diagnosis - Case-Based Reasoning Approach. Proceedings of Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS 2007), Maribor, Slovenia, June 20–22 2007, 65–71.

[8] Kurbalija V., Ivanović M., Case-Based Reasoning - A Powerfull Artificial Intelligence Approach, Journal of Electrical Engineering 12/s, Slovak Centre of IEE, Vol. 53 (2002), 20–24.

[9] Kurbalija V., Ivanović M., CaseBaseGenerator for Intelligent Systems. Novi Sad Journal of Mathematics, Vol. 35, no. 1 (2005), 25–40.

[10] Kurbalija, V., On Similarity of Curves - project report, Humboldt University, AI Lab, Berlin, 2003.

[11] Lenz M., Bartsh-Sporl B., Burkhard HD., Wess S., Case-Based Reasoning Technology: From Foundations to Aplications. Springer Verlag, Oct. 1998.

[12] Lenz M., Case Retrieval Nets Applied to Large Case Bases. draft, Lenz's homepage.

[13] Simić D., Budimac Z., Kurbalija V., Ivanović M., Case-Based Reasoning for Financial Prediction. Lecture Notes in Artificial Intelligence (LNAI), ISSN: 0302-9743, vol. 3533/2005, pp. 839–842.

[14] Simić, D., Kurbalija, V., Budimac, Z., An Application of Case-Based Reasoning in Multidimensional Database Architecture. In: Proc. Of 5th DaWaK, LNCS, Vol. 2737. Springer-Verlag (2003), 66–75.

Received by the editors October 30, 2008