Evaluating Case-Base Maintenance Algorithms

Eduardo Lupiani*, Jose M. Juarez, Jose Palma

Department of Information and Communications Engineering
Artificial Intelligence and Knowledge Engineering Research Group
University of Murcia, Spain

Abstract

The success of a Case-based Reasoning (CBR) system closely depends on its knowledge-base, named the case-base. The life cycle of CBR systems usually implies updating the case-base with new cases. However, it also implies removing useless cases for reasons of efficiency. This process is known as Case-Base Maintenance (CBM) and, in recent decades, great efforts have been made to automatise this process using different kind of algorithms (deterministic and non-deterministic). Indeed, CBR system designers find it difficult to choose from the wealth of algorithms available to maintain the case-base. Despite the importance of such a key decision, little attention has been paid to evaluating these algorithms. Although classical validation methods have been used, such as Cross-Validation and Hold-Out, they are not always valid for non-deterministic algorithms. In this work, we analyse this problem from a methodological point of view, providing an exhaustive review of these evaluation methods supported by experimentation. We also propose a specific methodology for evaluating case-base maintenance algorithms (the $\alpha\beta$ evaluation). Experiment results show that this method is the most suitable for evaluating most of the algorithms and datasets studied.

Keywords: Case-Base Reasoning, Case-Base Maintenance, Evaluation

*Corresponding author

Email addresses: elupiani@um (Eduardo Lupiani), jmjuarez@um (Jose M. Juarez), jtpalma@um (Jose Palma)

1. Introduction

Case-Base Reasoning (CBR) is based on experience-solving old problems to find a solution to new problems [1]. In this way, given that a problem and its solution conform a case, when new problems are solved then new cases are created and stored in a case-base. Although the learning ability of CBR is an important advantage, this characteristic is not free of drawbacks. For example, the addition of new cases to the case-base could degrade a system's performance [19, 35, 47] because case-bases with many cases need a long time to retrieve cases similar to an input query. Some reasons for increase in retrieval time are the scalability of the data structures that represent the case-base, and the case descriptions for both problem and solution. Consequently, calculation of the similarity function between cases is complex too.

Instead of incrementing hardware power, the alternative way to tackle performance problems is to use Case-Base Maintenance (CBM), whose main purpose is to update the existing case-base in order to maintain problem-solving competence [38], defined as the ability to solve new problems within an acceptable interval of time.

Two different strategies can be considered in CBM. The first strategy focuses on optimisation of the software that implements the case-base, such as tuning the parameters of the retrieval step [10] or changing the data structure used to model the case-base. The second strategy is aimed at removing cases from the case-base according to a given deletion policy [27], deleting redundant or noisy cases. In other words, the goal is to obtain smaller case-bases with the same problem-solving competence. The resulting case-base may not only replace the original case-base, but it may also be used as an index to enhance the retrieval of similar cases [24, 45].

Algorithms that delete cases have been widely studied in the Machine Learning field [2, 6, 17, 26, 43], as well as in the CBR community [11, 12, 22, 28, 38, 39, 47]. All these works show that reducing the number of cases through a CBM process is an effective approach to decreasing the retrieval time.

When using CBM algorithms, the question that remains is whether the maintained case-base is better, equivalent to or worst than the original one. The straightforward approach to answering this question is to use classical Machine Learning evaluation methods, such as Hold-Out and Cross-Validation, to analyse and compare the outcomes of the same CBR system

obtained with these two case-bases. These methods divide the case-base into a number of training and test sets, and while the training set plays the role of the case-base, the test set contains descriptions of all the input problems to be solved. In such a way, every problem in the test set is solved using the cases within the training set, which enables the accuracy, sensitivity and specificity to be calculated. Whereas Hold-Out uses one training set and one test set, the Cross-Validation usually involves up to five or ten training and test sets in order to finally average the results and obtain more reliable results. Thus, once the CBM has built the maintained case-base from the original, the evaluation method uses both the original and the maintained case-bases and builds the training and test sets. However, CBM algorithms may drastically reduce the size of the original case-base, and there will not be enough cases in the maintained case-base to resolve all the problem from the domain; that is, the competence of the CBR system is reduced.

To avoid this inconvenience, the evaluations for the CBR systems are performed in a different way. This approach consists of using the CBM algorithm on the training set created, instead of the CBM algorithm on the complete case-base. This strategy can be used both in Hold-Out evaluation [4, 11, 24, 29, 34, 38] and in Cross-Validation [13, 33].

However, this evaluation strategy has its own drawbacks. Non-deterministic CBM approaches, such as those described in [4, 20, 24, 30], return different maintained case-base outputs when multiple executions are carried out using the same original case-base, and it is necessary to execute the CBM algorithm several times taking the training set as input in order to obtain an average and achieve more reliable evaluation results.

In this work, we propose a novel evaluation method designed to deal with deterministic and non-deterministic CBM algorithms. In order to demonstrate the suitability of our approach, we recount the exhaustive experimentation made to compare the results given by our method and by other known evaluation methods.

Although different evaluation strategies are used in many publications, providing correct results, these results are difficult to compare precisely because they are obtained by different evaluation methods: for instance, building a test set with 40% or 30% of the cases, or repeating the Cross-Validation up to 10 times or executing it only once. The present paper is intended to clarify the differences between the best known evaluation methods to reveal their advantages and disadvantages. Furthermore, a new evaluation method is presented to solve those disadvantages that arise when working with non-

deterministic algorithms.

Finally, a Java library implementing the best known CBM algorithm and the introduced evaluation method is published for free access.

The remainder of this work is structured as follows: the next section revises the related work. Section 3 introduces our α-β evaluation proposal for evaluating of CBM algorithms. Section 4 describes CELSEA, a *Java* library to perform CBM. Section 5 shows the experimentation results after performing a set of CBM algorithms with a set of experimental case-bases from the UCI repository [14]. Section 6 is a discussion of the results from section 5. Finally, in section 7, we give our conclusions.

2. Related Work

2.1. Case-Based Reasoning

Case-Based Reasoning (CBR) is a methodology that makes use of past experiences to solve new problems. In a CBR system the atomic unit of knowledge is the *case*, the knowledge of previously experienced, specific problem situations [1]. The representation of a case is basically a tuple composed of two descriptions that characterise a problem and its solution. When multiple cases are created to solve problems in a given domain, they conform the knowledge base from which the CBR system can perform its problem-solving process [1]. Within the CBR community, the knowledge base is known as the *case-base*.

Despite the fact that multiple alternatives may represent a case, the simplest case representation is based on a vector of attributes, which describes the problem, and an attribute describing the solution. Other complex representations are possible, such as workflows [16], signals [36, 37], time sequences [21] or graphs [7] among others.

The most commonly used implementation of a CBR system was proposed in [1]. Basically, it consists of a four-step process: (1) *Retrieve*: the system searches for those cases that have a problem description with high similarity to the input problem description, for instance using a k-nearest neighbour algorithm [9]; (2) *Reuse*: using the cases retrieved, the system builds a solution to solve the input problem; (3) *Revise*: the system checks whether the solution proposed in the previous step is correct and, finally, (4) *Retain*: a new case is created with the description of the input problem and the output solution, and this is stored in the case-base.

One of the most important characteristic of this CBR model is that its learning process is incremental and continuous, since new solved cases are added (retained) in the case-base, thus the learning step is integrated in the problem solving process itself.

2.2. The Case-Base Maintenance

The Case-Base Maintenance implements policies for revising the organization or contents (e.g. representations, domain content or accounting information) of the case-base in order to simplify the future problem-solving process subjected to a particular set of performance objectives [27]. In particular, according to [49], CBM aims to reduce the case-base size or to improve the quality of the solutions provided by the CBR system [38].

The CBM has been studied in depth in Machine Learning disciplines such as Instance-based Learning, Exemplar-based Learning, as well as in Case-based Reasoning. There are a wealth of approaches to perform CBM, such as those published in [6, 15, 33, 38, 43, 48, 49]. Using the k-nearest neighbour algorithm [9] to find the similar cases (neighbours) to a given case, these algorithms usually try to classify the cases within the case-base as redundant or noisy, and delete them according to a specific deletion policy. Whereas a case could be considered as redundant when most of its neighbours have the same solution, a noisy case has a solution completely different to its neighbour's solutions [32]. The task of identifying a redundant case is easier than determining whether a case is noisy. A case surrounded by cases with a different solution could be due to an error in the description, or it could simply be an exception [39]. For instance, in a case-base that contains cases describing birds, we can describe a swan as a bird with white feathers, although there are infrequent black swans as well. In this way, it is incorrect to delete those cases that describe a swan like a bird with black feathers.

Some authors have proposed different taxonomies of CBM algorithms in an attempt to enhance our understanding of them. For example, [38] classify the different CBM algorithms according to the following features: (i) *Case Search Direction:* to build a maintained case-base, a CBM algorithm starts with an empty case-base and continues adding cases to it from the original case-base until a termination condition is reached. This is known as *Incremental* CBM; otherwise, the CBM algorithm is known as *Decremental*. (ii) *Order sensitive:* when the case-base is sorted in some way and this order determines the sequence in which cases are processed,then the CBM is *Order Sensitive*; otherwise the CBM algorithm is known as *Order Insensitive*.

(iii) *Selection Criteria:* when the decision to include or not a case in the case-base relies only on the local case parameters. For example, considering the solution given to its neighbours, then the criteria is *local*, however when all the cases are involved then is *global*. [49] propose an additional feature: (iv) *Type of cases to retain:* here, there are two approaches: retain those cases that have at least one case within their nearest neighbourhood with different solution (border cases), or retain cases that have all their nearest neighbours with the same solution as themselves (central cases). In the first scenario, the maintained case-base has smooth borders between clusters of cases, while in the second, the CBM algorithm is aimed at deleting the redundant cases.

Other classifications can be made according to whether a CBM algorithm is deterministic or non-deterministic. Whereas a deterministic CBM generates the same maintained case-base from a given case-base, a non-deterministic CBM builds different maintained case-bases each time that the algorithm is executed. Most of the CBM algorithms in the literature are deterministic because their deletion policy is fixed and constrained by the order of cases in the original case-base, and because they always apply the same action (removal) when they classify a case as noisy or redundant.

The retrieval time is usually proportional to the case-base size, among other factors such as the number of features (attributes) and the data representation complexity of each attribute. Therefore, to achieve good performance the number of learned cases in a CBR system must be kept to a minimum. However, the exact number of cases is difficult to estimate [28].

2.3. The CBM Algorithms

The simplest CBM algorithm consists of a random deletion of cases until the case-base reaches a certain number of cases [31]. However, CBM algorithms generally try to select cases in such a way that the CBR system using the maintained case-base has better or equal problem-solving capabilities than when the original case-base is used. To summarize the CBM algorithms proposed in the literature, we propose a classification based on four families: Nearest Neighbour algorithms (NN), Instance-Based algorithms, DROP family, Competence and Complexity models.

Some algorithms are actually composite methods, whereby the maintenance task is divided into two steps. The first step is usually aimed at reducing the amount of noisy cases, and the second step is the maintenance process itself.

Algorithms from NN family select cases according to a nearest neighbour policy. One of the first attempts to reduce case-base size was the Condensed Nearest Neighbour (CNN) [17]. Starting with a random case of each existing solution as initial case-base, the algorithm uses the rest of them as test sets and classifies them using the selected cases with a k nearest neighbours classifier (K-NN). If a case is misclassified then it is added to the final case-base. The process stops when all the original cases are correctly classified. Although size reduction is possible, this algorithm does not check for noisy cases, due to which, the Reduced Nearest Neighbour Rule (RNN)[15] was introduced as an extension of CNN to remove noisy instances from the resulting case-base after the use of CNN. Each case of the final case-base is removed, and if no case from the original case-base is misclassified then the candidate is finally removed, otherwise the case is added again. The Edited Nearest Neighbour (ENN) removes misclassified cases according to the solutions of their three nearest neighbours [48]. When ENN is executed multiple times, taking each output as input of the next execution, then the method is called Repeated ENN (RENN). All-kNN consists of executing k times ENN, where each execution uses from 1 to k neighbours respectively to flag a case for no selection [46]. Some authors claim ENN and its variations are actually noise removal techniques [49].

The Instance-based Family consists of algorithms that represent cases as instances (a data structure with a vector of features and an attribute class), and could be understood as a simplified representation of a case. IB2 and IB3 algorithms modify the IB1 classifier to perform CBM [2, 3]. In the IB2 algorithm, if a case is misclassified by its nearest neighbours then this case is added to the final case-base. The IB3 algorithm includes a more restrictive condition to keep a selected case inside the final case-base by reducing noise. Shrink algorithm executes CNN and, finally, it removes from the resulting case-base those cases misclassified by their nearest neighbours [23].

The CBM algorithms DROP1, DROP2 and DROP3 belong to the DROP family [49]. All these methods introduce the concept of associate case, which is a case within the set of nearest neighbour cases with the same solution. In DROP1, a case is removed if most of its associates already in the maintained case-base are solved correctly by the CBR system with the case-base without it. In DROP2, a case is removed if most of its associates in the original case-base are solved correctly without it. DROP3 uses ENN to remove the noisy cases before executing DROP1.

The algorithms from the Competence Model and the Complexity Profil-

ing Family are distinguished from the aforementioned families because they explode the implicit information of the case-base to build a maintenance model. The *Competence Model*[42, 43] defines concepts as Coverage, Reachability and Relative Coverage [44]. For each concept, a new CBM algorithm is proposed. COV and RFD use Coverage Set and Reachability cardinality, respectively, to sort the case-base, and RC uses the Relative Coverage model. Finally, the sorted case-base is the input to CNN, which performs the maintenance. *Complexity Profiling* estimates the proportion of redundant and noisy cases, as well as the existing error rate in the case-base [32, 33]. The basis of this approach is a local complexity measure that provides the probability of finding another case in the nearest neighbourhood of a case with the same solution. Although the local complexity measure is useful for case discovery [32], it does not provide enough information to evaluate the case-base competence. Lastly, other algorithm as *Weighting, Clustering, Outliers and Internal cases Detection based on DBscan and Gaussian* (WCOID-DG) means [41], use the competence concepts to reduce both the storage requirements and search time and to focus on balancing case retrieval efficiency and competence for a case-base. WCOID-GM is mainly based on the idea that a large case-base with weighted features is transformed to a small case-base with improving its quality.

2.4. Evolutionary Approach to perform CBM

The main objective of the CBM algorithms is to find the smallest subset of cases that provides the CBR system with a good problem-solving capabilities, usually through the removal of irrelevant or redundant cases. This problem can be modelled as a multi-objective optimization problem: minimising the number of cases within the case-base and maximising the CBR system accuracy. In this sense, Evolutionary Algorithms (EA) have been recognised as appropriate techniques for multi-objective optimisation because they perform a search for multiple solutions in parallel [8]. Current evolutionary approaches for multi-objective optimisation include multi-objective EAs based on the Pareto optimality notion, in which all objectives are optimised simultaneously to find multiple non-dominated solutions in a single run of the EA. For example, in this works [4, 20, 24, 30], the authors use Evolutionary Algorithms to perform CBM. Such evolutionary approaches are non-deterministic CBM algorithms, because each execution generates a different maintained case-base from the original case-base. Nonetheless, the returned maintained cases-bases converge to a common set of cases that ap-

pear frequently within all the maintained case-bases. Therefore, whereas there are different cases distribution among the set of different maintained cases-bases, these case-bases are able to solve similarly the given problem domain.

2.5. Measures for Evaluating a CBR System

The accuracy of a given CBR system is related with its ability to solve the input problem with the right solution. Using a set of cases as test set and a confusion matrix is a common practice to analyse the accuracy of the CBR system. Given a case-base with the set of solutions $s_1, s_2, \ldots, s_n$, a matrix is composed with a column and a row for each solution. The table 2.5 shows an example of confusion matrix, where N is the number of cases in the test set. With all the matrix positions initialized to 0, every time a case, with solution s_j, from the test set is solved by the CBR system, an output solution s_i is returned, and the cell in the position $p_{i,j}$ is incremented by 1.

		Actual solution				
		$\mathbf{s_1}$	$\mathbf{s_2}$	$\ldots$	$\mathbf{s_n}$	Total
Output solution	$\mathbf{s_1}$	$p_{1,1}$	$p_{1,2}$	$\ldots$	$p_{1,n}$	$\sum_{i=1}^{n} p_{1,i}$
	$\mathbf{s_2}$	$p_{2,1}$	$p_{2,2}$	$\ldots$	$p_{2,n}$	$\sum_{i=1}^{n} p_{2,i}$
	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
	$\mathbf{s_n}$	$p_{n,1}$	$p_{n,2}$	$\ldots$	$p_{n,n}$	$\sum_{i=1}^{n} p_{n,i}$
					Total	N

Table 1: Confusion Matrix for a multi-solution classification.

Once all the cases in the test set are solved, then it is possible to calculate statistical measures such as *accuracy*, *error*, *recall* (sensitivity) and *specificity* (true negative rate), among many others. The formal definition of these measures are given in the expressions 1, 2, 3, 4 and 5. Whereas the statistical measures given in the expressions 2, 3 are defined by a particular solution s_i, they can be used to create average measures such as those given in the expressions 4 and 5.

$$accuracy = N^{-1} \times \sum_{i=1}^{n} p_{i,i} \tag{1}$$

$$recall(s_i) = p_{i,i} \times \left(\sum_{j=1}^{n} p_{j,i} \right)^{-1} \tag{2}$$

$$specificity(s_i) = \sum_{j \in (1,i] \bigcup [i,n)} p_{j,j} \times \left(\sum_{j,l \in (1,i] \bigcup [i,n)} p_{j,l} \right)^{-1} \tag{3}$$

$$recall = n^{-1} \times \sum_{i=1}^{n} recall(s_i) \tag{4}$$

$$specificity = n^{-1} \times \sum_{i=1}^{n} specificity(s_i). \tag{5}$$

Other relevant measures are the achieved reduction rate by the method, and the difference between the retrieval time before and after the appliance of the CBM algorithm [40]. Given M as the original case-base, and M^{σ} as the maintained case-base by the CBM algorithm σ, the reduction rate is computed as:

$$\text{Reduction rate} = \frac{|M^{\sigma}|}{|M|} \tag{6}$$

2.6. Evaluation of CBM

Once a CBM finishes its execution, the next step is to figure out whether the resulting maintained case-base is better, worse or equivalent to the original one. Generally, this is done through a comparison of evaluation results or statistical measures, for instance accuracy, false positive and the kappa index, which are given by the CBR system using the original and maintained case-bases.

Two of the most common evaluation methods are Cross-Validation and Hold-Out, which provide a good estimations of statistical measures, e.g. accuracy [18]. The basis of both evaluations is similar since they divide the case-base into subsets of cases, where at least one of them is the test set and the remaining became part of the training set. Usually, the training set conforms the case-base of the CBR system, whereas the test set are the input problems to solve. Whereas the Hold-Out divides the case-base just

into two to create the test set and training set, usually using 30% of the cases to create the test set, Cross-Validation divides the case-base into n subsets, known as folds. The number of folds decides the number of evaluations, thus 10 Cross-Validation folds make up 10 training sets and test sets. Generally, five or ten folds are suitable to perform an evaluation [5, 25].

Working with one particular evaluation method rather than the other depends on the type of case-base in question. Hold-out is suitable when the training and test sets are large enough for them to be representative of the original case-base and the problem domain. However, in situations where this is not possible, perhaps because the case-base is quite small, Cross-Validation is the more appropriate evaluation approach.

The question remains as to how these evaluation methods would be applied. The first idea is to apply the CBM algorithm to the original case-base, and then to evaluate the maintained case-base with either Hold-Out or Cross-Validation. However, this approach has an important drawback: the resulting maintained case-base would probably be smaller than the original case-base due to the removal of redundant and noisy cases; consequently, the test and training sets generated are not equivalent to the original case-base. In this case, statistical measures are not reliable because they are not representative of reality.

For this reason, both Cross-Validation and Hold-Out have been widely used in the CBR literature with some modifications. Instead of first applying the CBM algorithm, the evaluation method is used on the original case-base to generate the training and test sets. Then, the CBM is applied to the training sets, which are equivalent to the original case-base. Below, we shall comment on the most common evaluation method on the CBR field.

In [4, 24] the accuracy of the maintained CBR system is studied using Hold-Out, and a k-nearest neighbour (K-NN) is used as a classifier. Following the same evaluation method, in [29] compare the retrieval time and the size reduction rate.

Another evaluation that uses Hold-Out and a K-NN classifier is presented in [34] and [11].In [34] each case-base is divided randomly into two partitions: an 80% training set and a 20% test set. In [11] the evaluation divides the case-base at random into three splits: 60% for the training set, 20% for the test set and the remaining is ignored. In both papers, CBM algorithm is applied to each training set in order to reduce their size. Later, the maintained training sets are validated with the test set to obtain the accuracy. The process is repeated from 10 to 30 times. At the end, the authors average the accuracies

to obtain the final accuracy estimation before comparing it with the accuracy given by the original case-base.

[12] propose a more complex variation of Cross-Validation. The authors suggest splitting the case-base into three: 60% for the training set, 20% for the test set and 20% for cross-validation. These sets are used in a two-step evaluation process. Given a set of CBM algorithms to evaluate, in the first step all the CBM algorithms are applied to the training set and the resulting case-bases are validated using the cross-validation set. In the second step, the case-base with best results is chosen and the test set is used to obtain the final evaluation results. The observed measures are the percentage of cases deleted and the accuracy given using a K-NN classifier.

In [33], a 10-fold Cross-Validation is used. The CBM algorithms are applied to each training set and the resulting maintained case-base size is recorded. Test set accuracy is measured for the original case-base and for each of the maintained case-bases created by the CBM algorithms. The evaluation is performed up to 10 times, and finally, the accuracy results are averaged and compared with those given by the original case-base.

Some authors propose new measures such as Competence to evaluate a case-base [38]. This measure highlights the proportional improvement in terms of accuracy of the maintained case-base against the original. The authors use Hold-Out as evaluation method, using 30% of the cases as test set and the remaining as training set. The CBM algorithm is applied to the training set and validated on the test set. The observed measures are the reduction size ratio and their proposed Competence measure. The process is repeated up to 20 times, and the results are averaged to compare them with the original measures. The classifier 1-NN is chosen to retrieve the most similar cases to an input problem.

3. $\alpha\beta$ Evaluation

In this work, we propose the $\alpha\beta$ evaluation method, which introduces two new parameters: α and β. The α parameter is aimed at enhancing the reliability of the results for the evaluation when the CBM algorithm under observation is non-deterministic, while the β parameter is intended to repeat the entire Cross-Validation process β times. Furthermore, $\alpha\beta$ evaluation performs a pre-processing of the original case-base to decrease the time needed to run the evaluation.

The $\alpha\beta$ evaluation produces two sets of decision scores for a given CBR system using a particular case-base. The first set is composed of the *Performance Decision Scores*, while the second set is composed of the *Quality Decision Scores*. Whereas the *Performance Decision Scores* comprise measures to quantify the reduction of the case-base and the CBM execution runtime, the *Quality Decision Scores* conform basic statistical measures that helps to determine the suitability of the CBR system to solve the problem domain from the given case-base. For instance, some of these statistical measures are accuracy, recall and specificity, whose definitions are given by the expressions 1, 4 and 5.

In order to compute the decision scores, the evaluation begins with a prior pre-processing of the data in the case-base. First, the most relevant attributes of the problem description are selected. The purpose of this is to decrease the number of problem features of a case in order to reduce the similarity complexity between cases and to speed up the evaluation process. Attribute selection is made with a genetic algorithm, whereby each individual is a binary string with element values in the domain $\{true, false\}$. Each individual represents features selected from the problem description. If the i-th position is set to *true* then the i-th feature is selected. That is, an individual with n positions represents a case-base that has n features to describe a problem, and the positions with the value set to *true* are the selected features of the problem. The population size and the number of generations are set to 100. The one point cross-over probability is 0.9 and the mutation is 0.033. The fitness value of each individual is the accuracy given by a 10 fold Cross-Validation using the equivalent case-base to the individual. Once feature selection is completed, duplicate cases are deleted and where missing values are present, they are replaced with the mean value of the corresponding feature. Finally, all the case features are normalized in the interval $[0, 1]$.

3.1. $\alpha\beta$ Evaluation Process

Once the data pre-processing finishes, the evaluation process starts. Algorithm 1 depicts in detail the steps needed to perform the $\alpha\beta$ evaluation:

Initialization of performance and quality decision scores: The first step is to initialise the Performance and Quality Decision Scores. Since quality decision scores rely on statistical measures, confusion matrices are

used to compute the scores. For each given CBM algorithm, the decision scores are initialised (see algorithm 1 in lines 1 to 4). Then, confusion matrices are created, which have as many dimensions as the number of different solutions that exist in the case-base (see algorithm 1 from line 6 to 8). CM_σ refers to the confusion matrix of the σ algorithm, and $CM[x, y]$ denotes the number of times that the actual solution of the query case is x and the solution provided by the CBR system is y, while both x and y might be the same solution or not. Each given CBM algorithm σ has its own quality and performance decision score, represented as ϕ_σ and ρ_σ, respectively.

β **loop:** In each iteration a new training set and test set are generated. This process is repeated β times (see algorithm 1, lines 9 to 26).

Training and test sets creation: The original case-base M is divided into 10 sub-case-bases with a similar number of cases. These sub-case-bases are denoted by M_i. Stratification is used to improve the representativeness of each solution to build the sub-case-bases. Furthermore, every case in M is present in at least one of the sub-case-bases M_i. The training sets, denoted by $\overline{M_i}$, comprise those cases in M that are not in M_i. The function *complementary* create the training sets (see algorithm 1, lines 10 to 12).

Training and test sets iteration: Since CBM algorithms are executed on each $\overline{M}_i$, and the resulting case-base is evaluated using the case-bases in M_i as queries (see algorithm 1), the loop in line 11 iterates over each i-th element of both sets.

α **loop:** Because CBM algorithms may generate different maintained case-bases with the same input case-base, the loop generates a set of maintained case-bases for each algorithm σ in order to generate a high number of results, and, consequently, to better estimate the real values associated with the quality decision scores (see algorithm 1, lines 13 to 24).

CBM execution and confusion matrix updating: all the CBM algorithms, Σ, are performed on each $\overline{M}_i$ to generate a new maintained case-base $\overline{M}_i^\sigma$. For each case in M_i with solution x, the CBR system using the $\overline{M}_i^\sigma$ returns a solution y. With both solu-

tion x and y, the confusion matrix CM_σ is updated (see algorithm 1, lines 14 to 23).

Calculation of decision scores: As each CBM algorithm σ is executed $\alpha \times \beta \times 10$, the performance decision score ρ_σ contains the average execution time and the reduction rate. The quality decision scores ϕ_σ are calculated from the confusion matrix CM_σ (see algorithm 1, lines 27 to 30.

Algorithm 1 $\alpha\beta$ Evaluation

Require: A CBR system with a case-base M, and a set Σ of CBM algorithms.
Ensure: A set of decision scores for each CBM algorithm.

for all $\sigma \in \Sigma$ **do**
  $\phi_\sigma \leftarrow (\phi_\sigma^{accuracy}, \phi_\sigma^{specificity}, \ldots)$ /* quality decision scores, initialised to 0 */
  $\rho_\sigma \leftarrow \left(\rho_\sigma^{time}, \rho_\sigma^{reduction}\right)$ /* performance decision scores, initialised to 0 */
end for
$S \leftarrow$ number of different solutions in M
for all $\sigma \in \Sigma$ **do**
  $CM_\sigma \leftarrow \begin{bmatrix} 0_{1,1} & \cdots & 0_{1,S} \\ \vdots & \ddots & \vdots \\ 0_{S,1} & \cdots & 0_{S,S} \end{bmatrix}$ /* Confusion matrix of $S \times S$ dimensions */ .
end for
for 1 **to** β **do**
  test $\leftarrow \{M_i : M_i \subset M \wedge M_i \text{ is stratified} | i \in [1, 10] \wedge \forall c \in M, \exists c \in M_i\}$.
  for all $M_i \in$ test **do**
    $\overline{M}_i \leftarrow complementary(M_i)$
    for 1 **to** α **do**
      for all $\sigma \in \Sigma$ **do**
        $t \leftarrow current\_time$
        $\overline{M}_i^\sigma \leftarrow \sigma(\overline{M}_i)$ /* Perform the CBM algorithm σ */
        $\rho_\sigma \leftarrow (\rho_\sigma^{time} + (t - current\_time), \rho_\sigma^{reduction} + \frac{|\overline{M}|_i^\sigma}{|M|})$
        for all $p \in M_i$ **do** /* loop over the cases in the test set */
          $x \leftarrow solution(p)$ /* Get the real solution */
          $y \leftarrow CBR(\overline{M}_i^\sigma, p)$ /* Get the solution given by the CBR system */
          $CM_\sigma[x, y] \leftarrow CM_\sigma[x, y]+1$ /* Update the confusion matrix */
        end for
      end for
    end for
  end for
end for
for all $\sigma \in \Sigma$ **do**
  $\rho_\sigma \leftarrow \left(\frac{\rho_\sigma^{time}}{\alpha \times \beta \times 10}, \frac{\rho_\sigma^{reduction}}{\alpha \times \beta \times 10}\right)$
  $\phi_\sigma \leftarrow$ statistical_measures(CM_σ) /* Calculate measures such as accuracy, specificity, ... */
end for
return ϕ, ρ

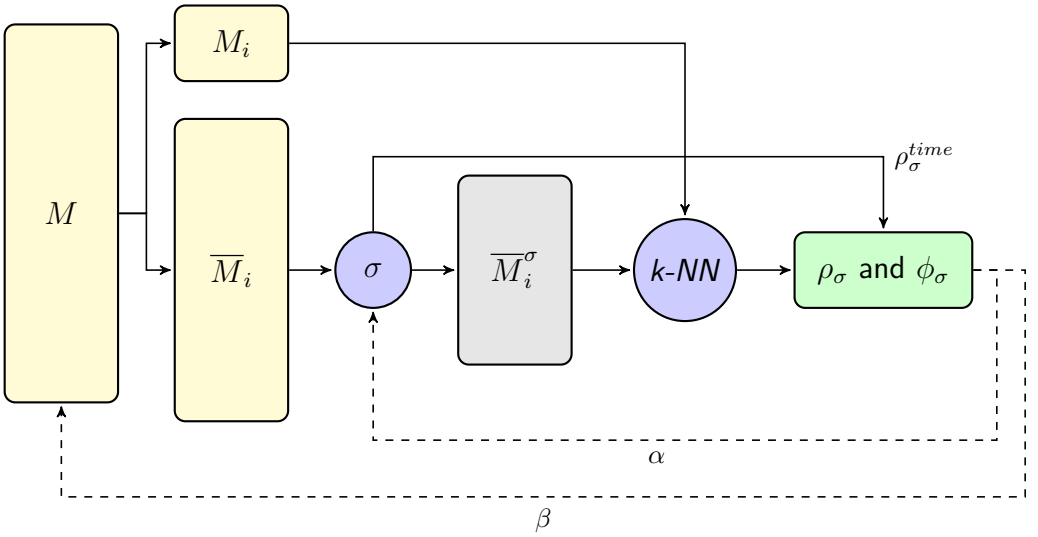

Figure 1: Data flow with the $\alpha\beta$ evaluation. The original case-base M is partitioned into 10 M_i (test sets)and $\overline{M}_i$ (training sets). Later, the CBM method σ is used on the training sets to create up to α maintained case-bases $\overline{M}_i^{\sigma}$, which are used to compute the the Quality and Performance Decision Scores ρ_σ and ϕ_σ. The entire process is repeated up to β times, and finally, the average is computed for the Decision Scores.

3.2. Values of α and β

Setting the β parameter will configure the evaluation process to repeat the Cross-Validation evaluation β times. With $\beta = 1$ and $\alpha = 1$, this evaluation is a Cross-Validation, whereby the CBM algorithms are applied to each training set. With higher values of β the Cross-Validation is repeated. This is the case of the evaluation methods adopted in [13] and [33], which repeat Cross-Validation 10 and 20 times, respectively.

On the other hand, changing the α parameter will lead to each CBM algorithm being repeated up to α times with the same training set. This parameter is useful when the CBM algorithm is non-deterministic. That is, multiple executions of a CBM with the same training set and order of cases return different case-base outputs.

4. A library for Case-base Maintenance: CELSEA

In this work, we present CELSEA[1] (CasE Library SElection Api), a framework for experimenting and evaluating case selection methods. This framework has a triple objective. First, CELSEA provides a simple infrastructure for developing CBM algorithms. Second, CELSEA includes a catalogue of implemented methods to experiment with case-bases and databases. Finally, CELSEA implements the $\alpha\beta$ evaluation proposed in Section 3.

4.1. Main Modules

CELSEA is a Java library composed of five main modules (see figure 2): *distance-classifier, cbr, case selection, evaluation* and *test*. The *distance-classifier module* implements a K-NN classifier and its catalogue of distance functions. The *CBR module* includes the case structure, the case-base and its efficient memory representation (*light memory*). This module also includes a functionality for experiment reporting in spreadsheet format (*report*). The *case selection module* is a catalogue of the following well known CBM algorithms: CNN, RNN, ENN, RENN, All-KNN, Shrink, IB2, IB3, ICF, DROP1, DROP2, DROP3,COV_RNN, RFC_CNN, RC_CNN, COV_CG, RFC_CG and RC_CG (see section 2). The *evaluation module* implements the Cross-Validation engine, α and β loops, etc. Finally, the *test module* includes different evaluation implementations, such as the evaluation $\alpha\beta$.

4.2. Usage

CELSEA can be used for development and experimental purposes. Experiments using CELSEA can be made using command lines providing the case-base file (in *CSV* format), a CBM file (in *XML* format) and the configuration of their parameters (in *XML* format). The CBM file consists of a list of the CBM algorithms to be analysed and their parameters configuration. Finally, the configuration file includes the implementation path to the CBM (σ), the database meta-information, and the values of the parameters described in the evaluation methodology proposed, that is: α , β, the number of partitions (or $folders$), and the classifier configuration.

[1]CELSEA project http://perseo.inf.um.es/∼aike/celsea

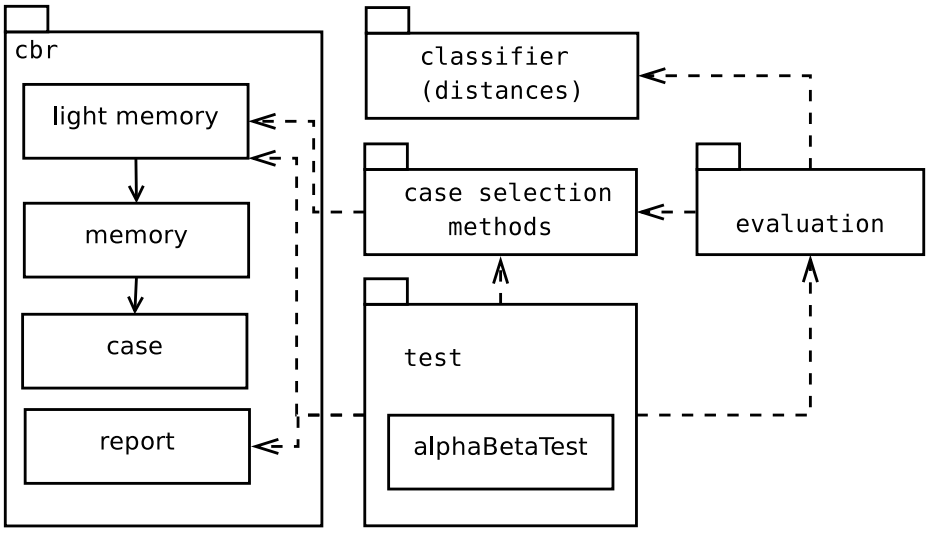

Figure 2: CELSEA framework: main package structure.

5. Experimentation

The following experiments were aimed at deepening our understanding of the $\alpha\beta$ evaluation, and, at showing how both α and β parameters can affect the decision score results. In particular experiments were made using deterministic and non-deterministic CBM algorithms with different values for both parameters. We have chosen CNN as the deterministic CBM algorithm since it is one of the simplest and most widely used CBM algorithms, while the NSGA-II was chosen as the non-deterministic algorithm, which uses the fitness function proposed in [30], in order to perform CBM. This fitness function has three objectives aimed to be minimised: 1. the number of redundant cases, 2. an estimation of the error, and 3. the difference between the current number of cases in the solution and the estimated number of non-redundant cases, which is computed at the beginning of the maintenance algorithm. At the end of the NSGA-II algorithm, there are a set of solutions, which represent case-bases, that converge to case-bases solving similarly the problem domain. Therefore, it is necessary to choose a final case-base for the maintenance process. In this manner, the solution representing the case-base with higher accuracy is chosen as output of the maintenance algorithm.

The organization of the experimental section to obtain the quality decision scores is as follows: first, we study different configurations of α and β

separately. Secondly, we compare the best results of $\alpha\beta$ with the accuracy results given by other evaluation methodologies, such as Cross-Validation, Hold-Out, and another evaluation method commonly used in the literature, which we call Pan evaluation [38]. This evaluation basically consists of repeating 20 times the Hold-Out evaluation. On the other hand, the performance decision score that we evaluate is the reduction rate of the CBM algorithm. Finally, we study the complexity and runtime of each evaluation method since this will help decide which evaluation method is the best.

5.1. Experimental Setup for all the Experiments

All the experiments share the same CBR system configuration, that is, only the case-base is different for each experiment. The CBR system retrieves the most similar cases using a K-NN approach with $k = 3$. Because different solutions may be found within the retrieved set of nearest cases, a majority voting system is used. In this system the most common solution is returned by the CBR system. Furthermore, if all the solutions are different, then the closer to the input problem is returned.

First, a pre-processing of the data in the case-base is performed including selection of the most relevant attributes, deletion of duplicate cases and replacing missing values with the mean of the attribute values. Finally, all the features values in each case-base are normalised in the interval $[0, 1]$. Attribute selection is made with a genetic algorithm, where each individual is a binary string with element values in the domain $\{true, false\}$. Each individual represents the features which are selected from the problem description. If the element is $true$ then the feature is selected. That is, an individual with n elements represents a case-base with n features describing a problem, and the elements with the value set to $true$ are the selected features of the problem. The population size and the number of generation are set to 100. The one point cross-over probability is 0.9 and the mutation is 0.033. The fitness value of each individual is the accuracy given by a 10 fold Cross-Validation using the equivalent case-base to the individual.

The experiments are made with the following public datasets: *iris, liver-bupa, bridges-version1* and *zoo* from UCI repository [14]. To build the case-base, each instance in a dataset is transformed into a case, where the problem description takes all the instance attributes except for the class, which is the solution description. All the experiments are performed 10 times in order to provide enough experimental results. Finally, the results are averaged to observe the differences between all the evaluation results.

5.2. Adjusting α and β Parameters

In this section, first we analyse different values of α in order to observe how it influences the evaluation results. We also study how β affects to the evaluation results. Finally, we compare the $\alpha\beta$ evaluation with other evaluation methodologies, using the α and β values that provided the best evaluation results in the two previous set of experiments.

The values of α and β under consideration are $\alpha = \{1, 3, 5, 7\}$, and $\beta = \{1, 5, 10, 15\}$. These values are selected in order to analyse how incremental values of α and β affect the results of the evaluation.

5.2.1. The effects of α Values for Deterministic and Non-Deterministic CBM algorithms

Figure 3 depicts the accuracy results of the CNN algorithm with the four different case-bases. In this experiments the β is constant ($\beta = 1$) and α varies its value ($\alpha = \{1, 3, 5, 7\}$).

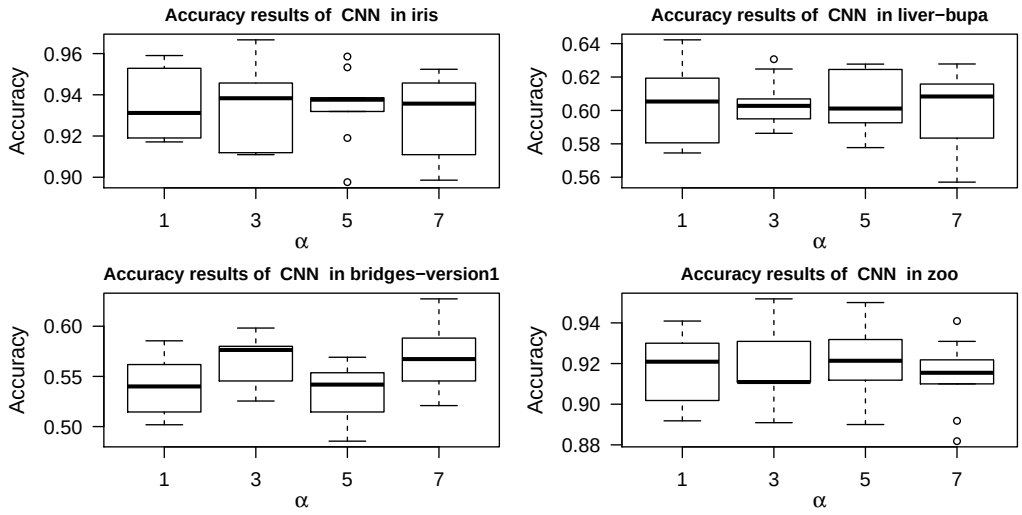

Figure 3: Experiments varying the α parameter for CNN ($\alpha = \{1, 3, 5, 7\}$, and $\beta = 1$).

Figure 4 shows the accuracy results for NSGA-II in each case-base. The results obtained with $\alpha = 1$ show a larger variation, while $\alpha \geq 3$ provides more regular results.

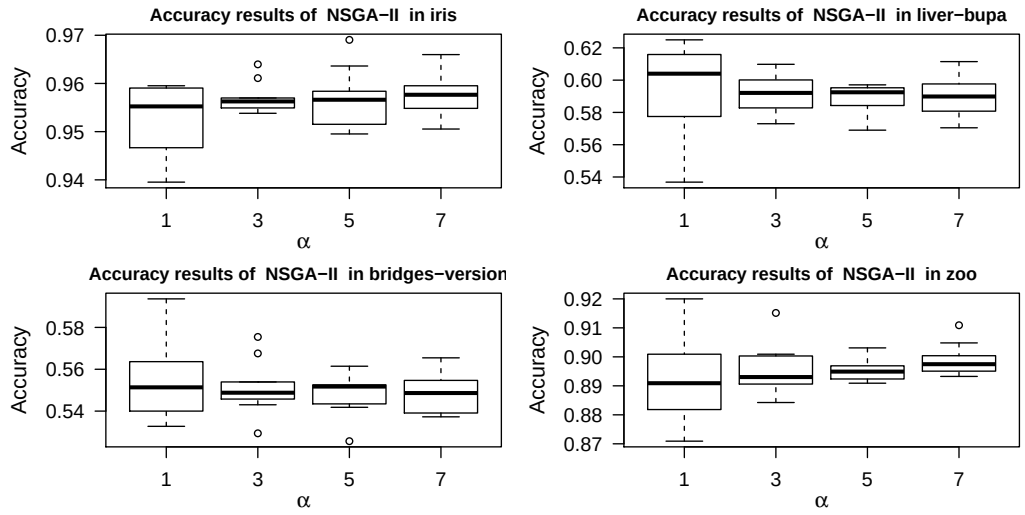

Figure 4: Experiments varying the α parameter for NSGA-II ($\alpha = \{1, 3, 5, 7\}$, and $\beta = 1$).

5.2.2. β values for deterministic and non-deterministic CBM algorithms

Figure 5 depicts the results of the recorded accuracies for each case-base with the CNN algorithm. The value of α is equal to 1, and only β varies its value ($\beta = \{1, 5, 10, 15\}$).

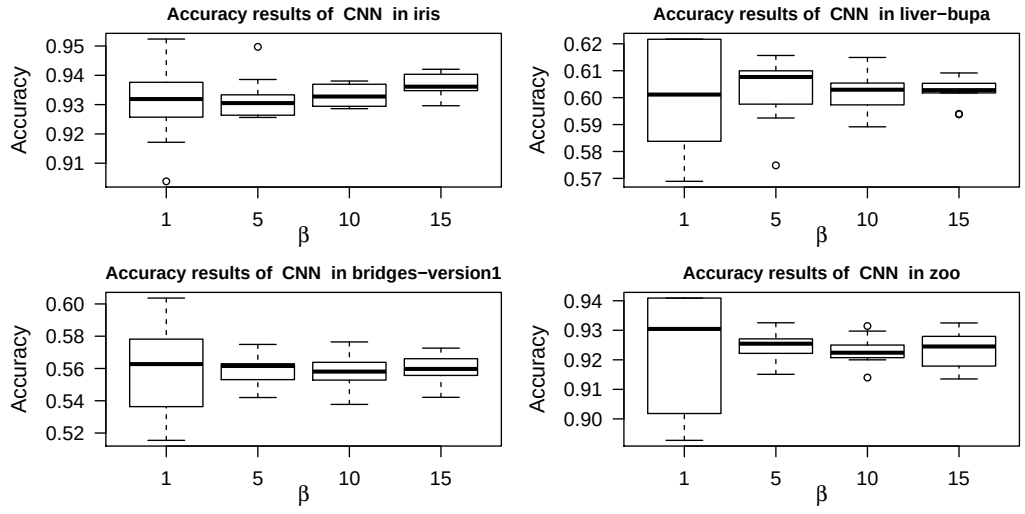

Figure 5: Experiments varying the β parameter for CNN ($\alpha = 1, \beta = \{1, 5, 10, 15\}$).

Figure 6 depicts the accuracy results for each case-base using NSGA-II

as the applied CBM algorithm. The value of α is 1.

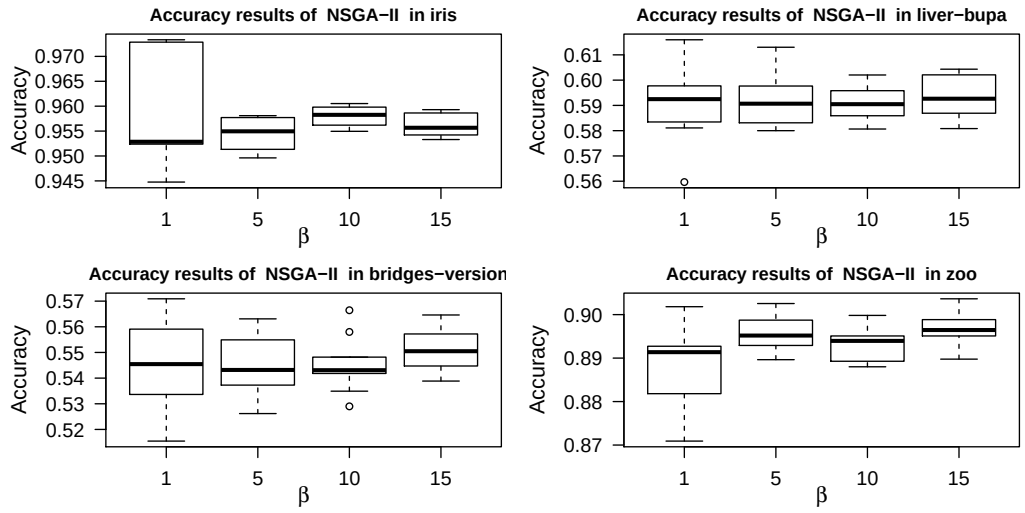

Figure 6: Experiments varying the β parameter for NSGA-II ($\alpha = 1, \beta = \{1, 5, 10, 15\}$).

5.2.3. Comparing α and β Parameters for non-deterministic CBM

The question that remains is whether α and β can be used simultaneously to enhance the reliability of the evaluation results. The following experiment is performed using our evaluation with the values $\alpha = 3$ and $\beta = 5$, and NSGA-II as the CBM algorithm under observation, using the fitness function given in [30]. The results are compared with those given by the experiments made with $\beta = 5, \alpha = 1$ and $\alpha = 3, \beta = 1$.

Figure 7 depicts the accuracy results for the case-bases *iris, liver-bupa, bridges-version1* and *zoo*.

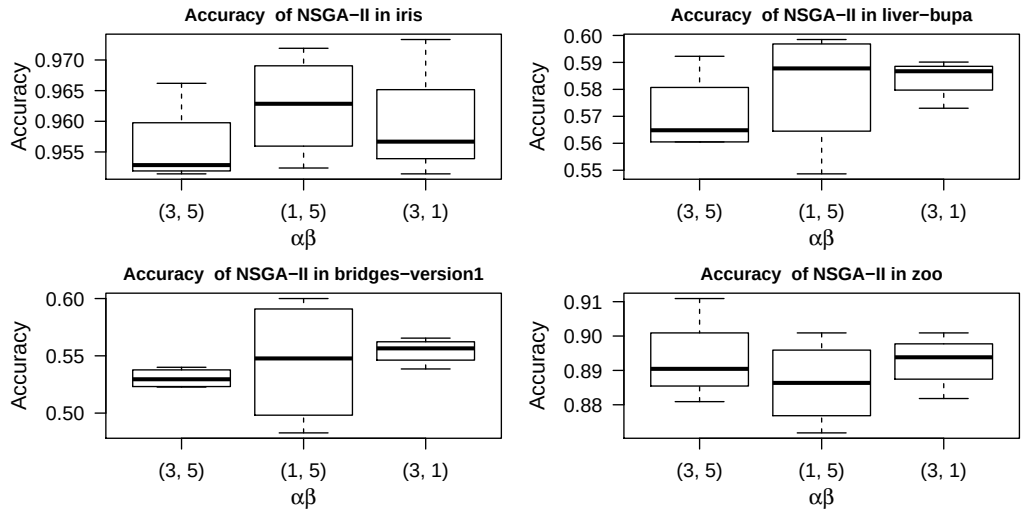

Figure 7: Results of NSGA-II for three different configurations of $\alpha\beta$ evaluation: ($\alpha = 3, \beta = 5$), ($\alpha = 1, \beta = 5$), ($\alpha = 3, \beta = 1$).

5.3. Comparing $\alpha\beta$ with other Evaluation Methods

To analyse whether $\alpha\beta$ evaluation is more suitable than other evaluation methods, we compare the results with those provided by Cross-Validation, Hold-Out and Pan [38] (Pan). Note that both Cross-Validation and Hold-Out take as input the maintained case-base returned by the CBM algorithm under study, whereas, both $\alpha\beta$ and Pan evaluations execute the CBM algorithm on each training set.

First, the experiments will focus on how the evaluation affects the results provided by CNN, a deterministic CBM algorithm. The values for α and β in this experiments are set to 1 and 10, respectively. values chosen because $\alpha$is only useful with non-deterministic CBM algorithms, and the experiments made in subsection 5.2 show that the most suitable value for β is 10 or more.

Secondly, the experiments will deal with a non-deterministic algorithms, the muti-objective evolutionary algorithm NSGA-II, using a fitness function to perform CBM [30]. On this occasion, the parameters values of α and β are set to 3 and 10, respectively, values that have shown good results as figure 7 depicts.

Additionally, each case-base is first evaluated without any case selection. For the sake of clarity, we call *None* the CBM algorithm that does not perform any case selection.

Figure 8 portrays the accuracy results for each evaluation method and with the None algorithm, and figure 9 depicts the results of the CNN algorithm. Finally, figure 10 shows the results with the algorithm NSGA-II. In addition, tables 3, 5 and 7 show the ANOVA (Analysis of Variance) results for the accuracy results given by each evaluation with none CBM, CNN and NSGA-ii, respectively. With this test, the hypothesis is that the accuracy averages are equal between the results given by each evaluation method. The most important output values by ANOVA are the *F value* and *p-value*, which give the difference between the accuracy results by each evaluation method, and the probability to the accuracy averages are equal, respectively. That is, lower the *p-value*, higher the evidence that the accuracy averages are different. Table 2 contains the codes used by the ANOVA tables.

Label	**Name**	**Meaning**
Df	Degrees of freedom	It is related to the number of experiments done
Sum Sq	Sum of squares	The difference between the accuracy averages of each evaluation method
Mean Sq	Mean Squares	The dispersion of the results
F value	$\frac{\text{Mean square Between}}{\text{Mean square Within}}$	A large value indicates relatively more difference between the accuracy results of each evaluation method than the accuracy within groups
Pr(>F)	*p-value*	Probability that the null hypothesis is right. Lower the *p-value*, higher the evidence against the null hypothesis.

Table 2: Codes used in ANOVA tables 3, 5 and 7

Furthermore, in order to compare in detail every pair of evaluation method, tables 4, 6 and 8 show the Tukey Honest Significant Differences [50], which give us the *p-value* from the results of two evaluation methods. Lower the *p-value* between two evaluations, and then lower the probability that the results of each evaluation method have the same results variance.

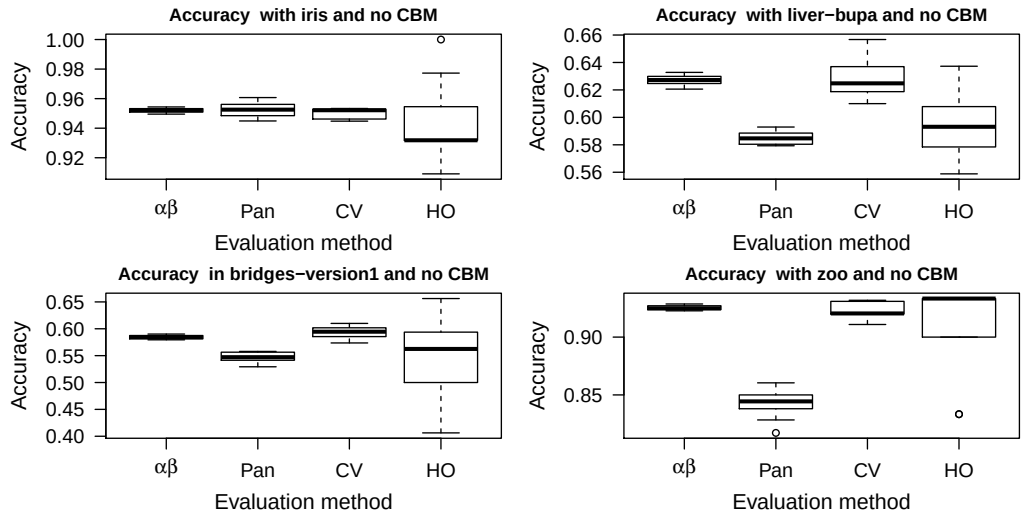

Figure 8: Comparative study of the evaluation methods $\alpha\beta$, Pan, Cross-Validation (CV) and Hold-Out (HO), when None CBM algorithm is executed.

		Df	Sum Sq	Mean Sq	F value	Pr(>F)
iris	Between	3	0.00	0.00	1.34	0.2775
	Within	36	0.01	0.00		
liver-bupa	Between	3	0.01	0.00	23.68	0.0000
	Within	36	0.01	0.00		
bridges-version1	Between	3	0.02	0.01	4.25	0.0114
	Within	36	0.05	0.00		
zoo	Between	3	0.04	0.01	31.73	0.0000
	Within	36	0.02	0.00		

Table 3: ANOVA for the accuracy results given by each evaluation with none CBM.

	iris	liver-bupa	bridges-version1	zoo
	p-value	*p-value*	*p-value*	*p-value*
CV-$\alpha\beta$	0.99	1.00	0.94	0.98
HO-$\alpha\beta$	0.34	0.00	0.17	0.24
Pan-$\alpha\beta$	1.00	0.00	0.12	0.00
HO-CV	0.48	0.00	0.05	0.41
Pan-CV	0.99	0.00	0.03	0.00
Pan-HO	0.33	0.25	1.00	0.00

Table 4: Tukey HSD for the accuracy results given by each evaluation with none CBM algorithm.

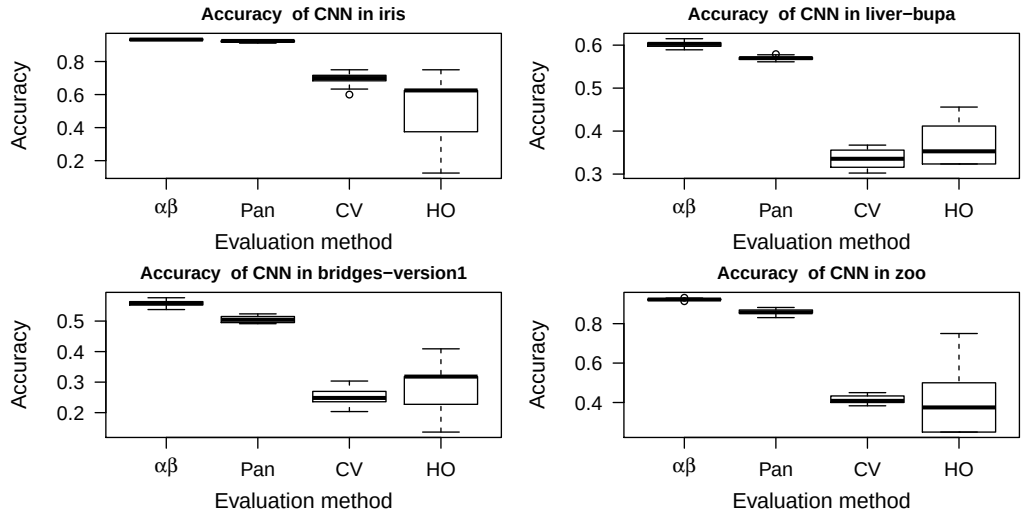

Figure 9: Comparative study of the evaluation methods: $\alpha\beta$, Pan, Cross-Validation (CV) and Hold-Out (HO) evaluations, when CNN algorithm is executed.

		Df	Sum Sq	Mean Sq	F value	Pr(>F)
iris	Between	3	1.10	0.37	36.20	0.0000
	Within	36	0.36	0.01		
liver-bupa	Between	3	0.56	0.19	248.84	0.0000
	Within	36	0.03	0.00		
bridves-version1	Between	3	0.68	0.23	114.26	0.0000
	Within	36	0.07	0.00		
zoo	Between	3	2.35	0.78	112.43	0.0000
	Within	36	0.25	0.01		

Table 5: ANOVA for the accuracy results given by each evaluation with CNN.

	iris	liver-bupa	bridges-version1	zoo
	p-value	p-value	p-value	p-value
CV-$\alpha\beta$	0.00	0.00	0.00	0.00
HO-$\alpha\beta$	0.00	0.00	0.00	0.00
Pan-$\alpha\beta$	1.00	0.07	0.06	0.33
HO-CV	0.01	0.06	0.16	0.97
Pan-CV	0.00	0.00	0.00	0.00
Pan-HO	0.00	0.00	0.00	0.00

Table 6: Tukey HSD for the accuracy results given by each evaluation with CNN.

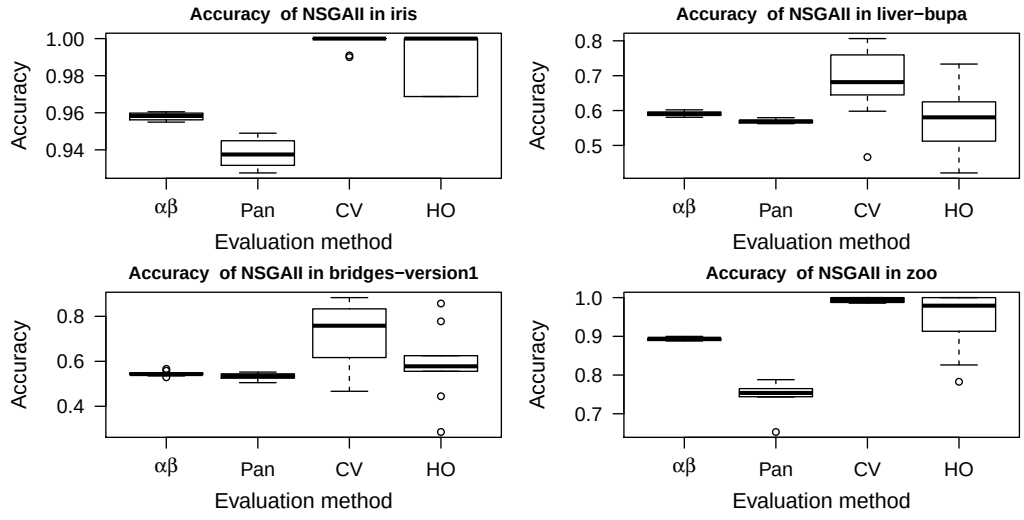

Figure 10: Comparative study of the evaluation methods: $\alpha\beta$, Pan, Cross-Validation (CV) and Hold-Out (HO), when NSGA-II algorithm is executed.

		Df	Sum Sq	Mean Sq	F value	Pr(>F)
iris	Between	3	0.02	0.01	103.81	0.0000
	Within	36	0.00	0.00		
liver-bupa	Between	3	0.08	0.03	5.65	0.0028
	Within	36	0.18	0.00		
bridges-version1	Between	3	0.24	0.08	7.36	0.0006
	Within	36	0.39	0.01		
zoo	Between	3	0.33	0.11	57.31	0.0000
	Within	36	0.07	0.00		

Table 7: ANOVA for the accuracy results given by each evaluation with NSGA-II.

	iris	liver-bupa	bridges-version1	zoo
	p-value	p-value	p-value	p-value
CV-$\alpha\beta$	0.00	0.03	0.00	0.00
HO-$\alpha\beta$	0.00	0.96	0.79	0.07
Pan-$\alpha\beta$	0.00	0.90	0.99	0.00
HO-CV	0.25	0.01	0.02	0.07
Pan-CV	0.00	0.01	0.00	0.00
Pan-HO	0.00	1.00	0.63	0.00

Table 8: Tukey HSD for the accuracy results given by each evaluation with NSGA-II.

Lastly, other important issue is the difference between the accuracy results given by the different evaluation processes when either they do not use a CBM algorithm or some deterministic/non-deterministic CBM algorithm. In order to study these differences, an ANOVA analysis is performed between the accuracy results given with no CBM, CNN and NSGA-II, respectively, for each evaluation process. Therefore, table 9 shows the ANOVA for the $\alpha\beta$ evaluation, table 10 shows the ANOVA for the Pan evaluation, and tables 11 and 12 belongs to the ANOVA results for CV and HO evaluations, respectively .

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Between	2	0.01	0.01	0.22	0.8041
Within	117	3.66	0.03		

Table 9: ANOVA between the accuracy results given by $\alpha\beta$ evaluation process with None, CNN and NSGA-II.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Between	2	0.02	0.01	0.39	0.6758
Within	117	3.51	0.03		

Table 10: ANOVA between the accuracy results given by Pan evaluation process with None, CNN and NSGA-II.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Between	2	4.12	2.06	72.65	0.0000
Within	117	3.31	0.03		

Table 11: ANOVA between the accuracy results given by CV evaluation process with None, CNN and NSGA-II.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Between	2	3.50	1.75	49.11	0.0000
Within	117	4.16	0.04		

Table 12: ANOVA between the accuracy results given by HO evaluation process with None, CNN and NSGA-II.

5.4. Performance Decision Support Experiment

Despite the importance of accuracy, the improvement achieved by a CBM algorithm in terms of performance is also fundamental. In this subsection we focus on the reduction rates and the execution time of each CBM algorithm.

5.4.1. Reduction rate results

Reduction rates are acquired from $\alpha\beta$ and Pan evaluation to observe whether a particular evaluation method may have any impact on this measure. Figures 11 and 12 depict the results for $\alpha\beta$ and Pan evaluation respectively. Each column is the average of all the reduction rates given by all the experiment performed using the corresponding CBM algorithm.

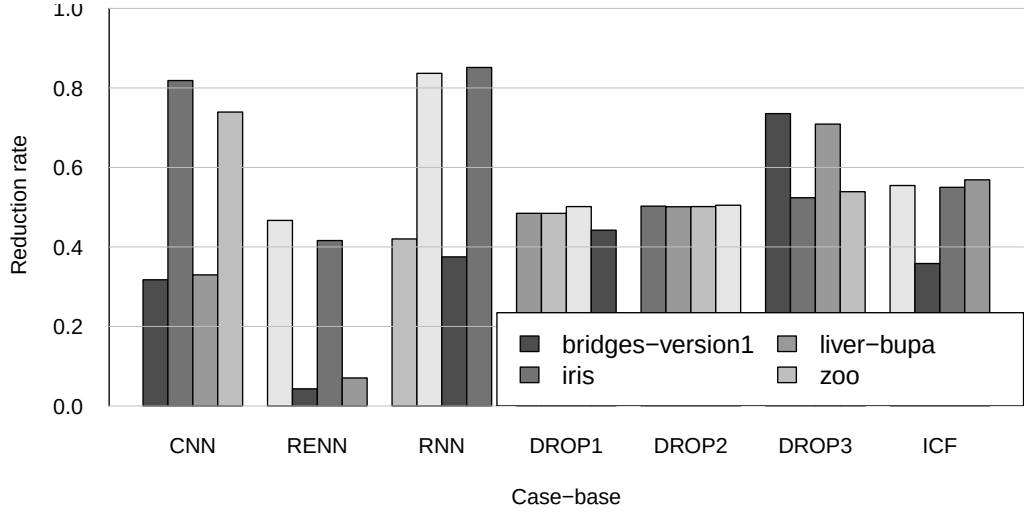

Figure 11: Reduction rate for $\alpha\beta$ evaluation method.

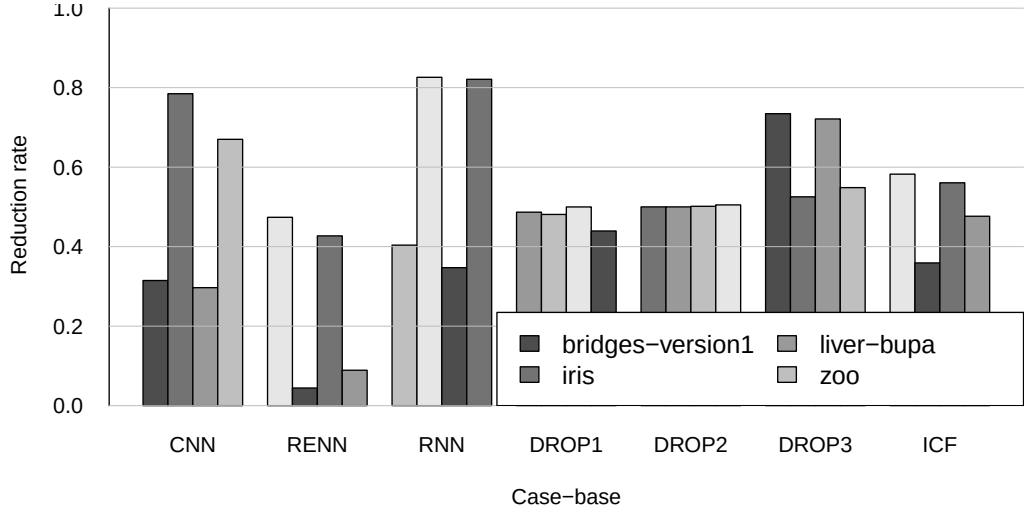

Figure 12: Reduction rate for Pan evaluation method.

5.5. Execution Time of each Evaluation Method

Prior to performing any empirical experimentation to record the runtime of each evaluation method, we define the computational complexity of each one, to ascertain which will take longer execution time. Formally, let n and m

be the respective number of iterations needed to perform the CBM algorithm in the training sets of Cross-Validation and Hold-Out, with $n \geq m$. Let f be the number of folds in a Cross-Validation. The complexity orders of each evaluation are the following:

- $\alpha\beta$ evaluation: $\mathcal{O}(\alpha \times \beta \times f \times n)$.
- Pan $\mathcal{O}(20 \times m)$.
- Cross-Validation: $\mathcal{O}(10 \times n)$.
- Hold-Out $\mathcal{O}(m)$.

According to the complexity, the $\alpha\beta$ will take $\alpha \times \beta$ more time to finish the evaluation than Cross-Validation. Pan evaluation will take 20 times more than Hold-Out. Because $n \geq m$, the evaluations based on Cross-Validation have longer runtime than those based in Hold-Out. Figure 13 depicts the average runtime results for the experiment performed previously, with $\beta = 10$, $\alpha = 1$ and $f = 10$. As we expected, $\alpha\beta$ has the longest runtime, followed by Pan evaluation.

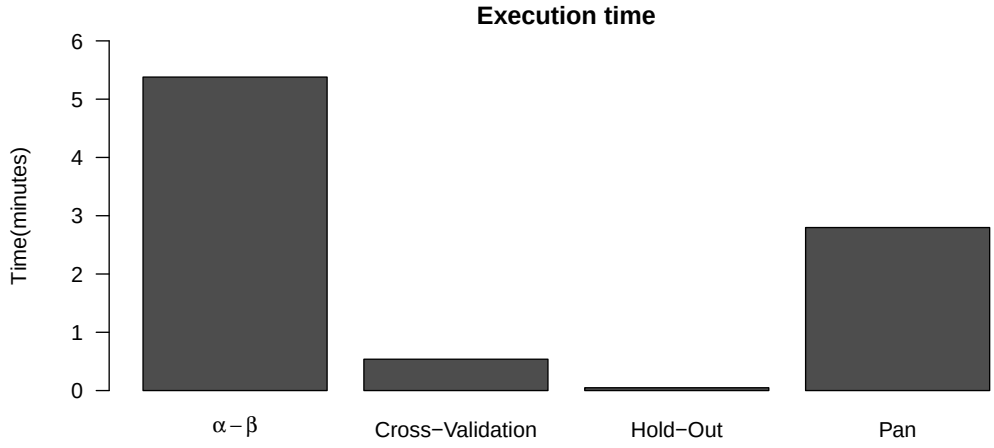

Figure 13: Execution time of the different evaluation methods.

6. Discussion

As regards to the study of the α and β parameters, the following aspects should be mentioned:

1. Higher values of β provide accurate results with lower variance according to figures 5 and 6, which are related with the study of parameter β for deterministic and non-deterministic CBM algorithms. According to the results shown in figures 5 and 6, and ANOVA results from tables 3 and 4, whereas values of $\beta = 1$ show higher variance of accuracy results than those results with $\beta = 5, 10$ and 15, values of $\beta = 5$ and $\beta = 10$ seem to converge to similar accuracy averages. Therefore, these values seem enough to perform the evaluation for deterministic and non-deterministic algorithms.

2. Whereas the α parameter has no effect on the results when the CBM algorithm is deterministic, higher values of α may provide reliable accuracy results for non-deterministic CBM algorithms. According to figure 4 , values of α equal to 3 or 5 seem to be sufficient to obtain reliable accuracy results.

3. When using α and β parameters simultaneously with values greater than one, the method shows reliable accuracy results. However, the variance in the results is only slightly higher than when α parameter alone is used.

After the comparative study of evaluation methods ($\alpha\beta$, Hold-Out, Cross-Validation and Pan evaluations), the following should be pointed out:

1. Whereas the methods based on folds, such as $\alpha\beta$ and Cross-Validation, achieve similar accuracy averages for every case-base with very little variance, those based on Hold-Out provide uneven accuracy results with greater variation. This is probably due to the low variety in the training and test sets.

2. Hold-Out shows the greatest variance in the accuracy results.

3. When a deterministic CBM algorithm is being evaluated, neither Cross-Validation and nor Hold-Out provide a good estimation of the accuracy of the CBR system. Furthermore, $\alpha\beta$ and Pan evaluations have similar results, with low variance, although Pan evaluation underestimates the accuracy in every case-base.

4. For non-deterministic CBM algorithm, the Pan, Cross-Validation and Hold-Out evaluations do not compute reliable accuracy results in some

case-bases. The variation in the results confirms this point. On the other hand, $\alpha\beta$ provides accuracy results with the lowest variation. Furthermore, the results are the closest to the original accuracy results.

A glance at figures 11 and 12 shows that every reduction for every case-base is quite similar. The Pearson correlation between the data from both figures is shown in figure 6, with correlation values close to one.

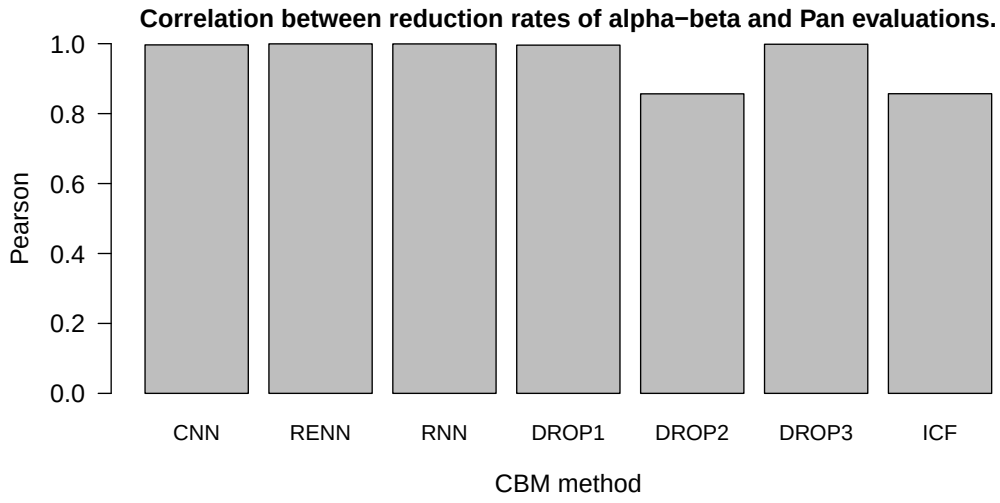

Figure 14: Pearson correlation between $\alpha\beta$ and Pan evaluations.

In light of to the results of Cross-Validation and Hold-Out, one execution is not sufficient to obtain reliable measures, whether for deterministic or non-deterministic CBM algorithms.

7. Conclusions

We introduce the $\alpha\beta$ method to evaluate different CBM algorithms. The evaluation depends on two parameters: α to obtain reliable results when the CBM algorithm under observation is non-deterministic, and β to repeat the entire evaluation β times and obtain an average of the results. We have released a Java library that implements many CBM algorithms. Finally, exhaustive experiments have been performed in order to compare different evaluation methods.

Using public datasets, we have compared our results with those obtained using other evaluation methods in the CBR literature: Cross-Validation, Hold-Out and Pan evaluation [38]. On the one hand, the results obtained show that only one execution of Cross-Validation and Hold-Out is not sufficient to provide reliable results, whether for deterministic or non-deterministic CBM algorithms. On the other hand, according to our experiments, $\alpha\beta$ and Pan evaluation are suitable methods to evaluate deterministic CBM algorithms. Although Pan evaluation needs a lower runtime, $\alpha\beta$ is the only evaluation method able to provide reliable evaluation measurements when dealing with non-deterministic CBM algorithms. In such a scenario, the $\alpha\beta$ evaluation shows itself to be a good option for use as an evaluation method.

In certain domains, such as Medical Diagnosis or Personal Activity Monitoring at home, the concept of time is an important feature for performing reasoning operations. For this reason, in future studies, the $\alpha\beta$ evaluation, as well as the CBM algorithms, will be extended to domains where the case-base contains temporal cases. Additionally, because the $\alpha\beta$ evaluation generates the training and test sets randomly, and there are many CBM algorithms that are order sensitive, we will extend the $\alpha\beta$ evaluation to multiple executions of the CBM algorithm with different training set orders. Finally, emphasis will be placed on improving the efficiency of the $\alpha\beta$ evaluation, in an attempt to reduce the time consumed during its execution.

8. Acknowledgement

This work was partially funded by the Ministry of Economy and Competitiveness, by the Seneca Research Foundation of the Region of Murcia under project 15277/PI/10, and by the Spanish Ministry of Science and Innovation+European FEDER+PlanE funds under the project TIN2009-14372-C03-01 and under the grant BES-2010-030294.

References

[1] Aamodt, A., Plaza, E., 1994. Case-based reasoning: Foundational issues , methodological variations, and system approaches. AI Communications 7, 39–59.

[2] Aha, D., 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal Of Man-Machine Studies 36, 267–287.

[3] Aha, D., Kibler, D., Albert, M., 1991. Instance-based learning algorithms. Machine Learning 6, 37–66.

[4] Ahn, H., Kim, K.j., Han, I., 2007. A case-based reasoning system with the two-dimensional reduction technique for customer classification. Expert Systems with Applications 32, 1011–1019.

[5] Breiman, L., Spector, P., 1992. Submodel selection and evaluation in regression - the x-random case. International Statistical Review 60, 291–319.

[6] Brighton, H., Mellish, C., 1999. On the consistency of information filters for lazy learning algorithms, in: Principles of Data Mining and Knowledge Discovery, pp. 283–288.

[7] Bunke, H., Irniger, C., Neuhaus, M., 2005. Graph matching - challenges and potential solutions, in: Proceedings of the 13th international conference on Image Analysis and Processing, pp. 1–10.

[8] Coello, C.C., Lamont, G., van Veldhuizen, D., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and Evolutionary Computation, Springer.

[9] Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. Information Theory, IEEE Transactions on 13, 21 –27.

[10] Craw, S., Jarmulak, J., Rowe, R., 2001. Maintaining retrieval knowledge in a case-based reasoning system. Computational Intelligence 17, 346–363.

[11] Cummins, L., Bridge, D., 2009. Maintenance by a committee of experts: The MACE approach to case-base maintenance, in: Case-Based Reasoning Research And Development, Proceedings, pp. 120–134.

[12] Cummins, L., Bridge, D.G., 2011. Choosing a case base maintenance algorithm using a meta-case base, in: SGAI Conf., pp. 167–180.

[13] Delany, S., Cunningham, P., 2004. An analysis of case-base editing in a spam filtering system, in: Advances In Case-Based Reasoning, Proceedings, pp. 128–141.

[14] Frank, A., Asuncion, A., 2010. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

[15] Gates, G., 1972. Reduced nearest neighbor rule. IEEE Transactions On Information Theory 18, 431+.

[16] Gil, Y., 2012. Reproducibility and efficiency of scientific data analysis: Scientific workflows and case-based reasoning, in: Case-Based Reasoning Research And Development. volume 7466 of *LNCS*, pp. 2–2.

[17] Hart, P., 1968. Condensed nearest neighbor rule. IEEE Transactions On Information Theory 14, 515+.

[18] Hastie, T., Tibshirani, R., Friedman, J., 2001. The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc.

[19] Houeland, T.G., Aamodt, A., 2010. The utility problem for lazy learners - towards a non-eager approach, in: Bichindaritz, I and Montani, S (Ed.), Case-Based Reasoning Research And Development, 18Th International Conference On Case-Based Reasoning, ICCBR 2010, pp. 141–155.

[20] Ishibuchi, H., Nakashima, T., Nii, M., 2001. Genetic-algorithm-based instance and feature selection, in: Liu, H., Motoda, H. (Eds.), Instance Selection and Construction for Data Mining. Springer US. volume 608 of *The Springer International Series in Engineering and Computer Science*, pp. 95–112. URL: http://dx.doi.org/10.1007/978-1-4757-3359-4_6, doi:10.1007/978-1-4757-3359-4_6.

[21] Juarez, J.M., Guil, F., Palma, J., Marin, R., 2009. Temporal similarity by measuring possibilistic uncertainty in CBR. Fuzzy Sets And Systems 160, 214–230.

[22] Juárez, J.M., Lupiani, E., Palma, J., 2011. Case selection evaluation methodology, in: Industrial Conference on Data Mining - Workshops, pp. 17–29.

[23] Kibler, D., Aha, D., 1987. Learning representative exemplars of concepts: An initial case study, in: Proceedings of the Fourth International Workshop on Machine Learning, pp. 24–30.

[24] Kim, K., Han, I., 2001. Maintaining case-based reasoning systems using a genetic algorithms approach. Expert Systems with Applications 21, 139–145.

[25] Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection, in: IJCAI'95, pp. 1137–1145.

[26] Kuncheva, L., Jain, L., 1999. Nearest neighbor classifier: Simultaneous editing and feature selection. Pattern Recognition Letters 20, 1149–1156.

[27] Leake, D., Wilson, D., 1998. Categorizing case-base maintenance: Dimensions and directions, in: Advances In Case-Based Reasoning, pp. 196–207.

[28] Leake, D., Wilson, M., 2011. How many cases do you need? assessing and predicting case-base coverage, in: Proceedings Of The 19Th International Conference On Case-Based Reasoning Research And Development, pp. 92–106.

[29] Li, Y., Shiu, S., Pal, S., 2006. Combining feature reduction and case selection in building cbr classifiers. IEEE Transactions On Knowledge And Data Engineering 18, 415–429.

[30] Lupiani, E., Craw, S., Massie, S., Juárez, J.M., Palma, J.T., 2013. A multi-objective evolutionary algorithm fitness function for case-base maintenance, in: ICCBR, pp. 218–232.

[31] Markovitch, S., Scott, P., 1988. The role of forgetting in learning, in: Proceedings of The Fifth International Conference on Machine Learning, pp. 459–465. URL: `http://www.cs.technion.ac.il/~shaulm/papers/pdf/Markovitch-Scott-icml1988.pdf`.

[32] Massie, S., Craw, S., Wiratunga, N., 2005. Complexity-guided case discovery for case based reasoning, in: Proceedings of the 20th national conference on Artificial Intelligence - Volume 1, pp. 216–221.

[33] Massie, S., Craw, S., Wiratunga, N., 2006. Complexity profiling for informed case-base editing, in: Advances In Case-Based Reasoning, Proceedings, pp. 325–339.

[34] McKenna, E., Smyth, B., 2001. Competence-guided case-base editing techniques, in: Advances In Case-Based Reasoning, Proceedings, pp. 186–197.

[35] Minton, S., 1990. Quantitative results concerning the utility of explanation-based learning. Artificial Intelligence 42, 363–391.

[36] Montani, S., Portinale, L., Leonardi, G., Bellazzi, R., Bellazzi, R., 2006. Case-based retrieval to support the treatment of end stage renal failure patients. Artificial Intelligence in Medicine 37(1), 31–42.

[37] Olsson, E., Funk, P., Xiong, N., 2004. Fault diagnosis in industry using sensor readings and case-based reasoning. Journal of Intelligent and Fuzzy Systems 15(1), 41–46.

[38] Pan, R., Yang, Q., Pan, S.J., 2007. Mining competent case bases for case-based reasoning. Artificial Intelligence 171, 1039–1068.

[39] Rissland, E.L., 2009. Black swans, gray cygnets and other rare birds, in: Proceedings of the 8th International Conference on Case-Based Reasoning: Case-Based Reasoning Research And Development, pp. 6–13.

[40] Smiti, A., Elouedi, Z., 2011. Article: Overview of maintenance for case based reasoning systems. International Journal of Computer Applications 32, 49–56.

[41] Smiti, A., Elouedi, Z., 2014. WCOID-DG: An approach for case base maintenance based on weighting, clustering, outliers, internal detection and dbsan-gmeans. Journal of Computer and System Sciences 80, 27 – 38. doi:http://dx.doi.org/10.1016/j.jcss.2013.03.006.

[42] Smyth, B., 1998. Case-base maintenance, in: IEA/AIE (Vol. 2), pp. 507–516.

[43] Smyth, B., Keane, M., 1995. Remembering to forget - a competence-preserving case deletion policy for case-based reasoning systems, in: IJCAI-95 - Proceedings Of The Fourteenth International Joint Conference On Artificial Intelligence, Vols 1 And 2, pp. 377–382.

[44] Smyth, B., Mckenna, E., 1999. Building compact competent case-bases, in: Case-Based Reasoning Research And Development, pp. 329–342.

[45] Smyth, B., McKenna, E., 2001. Competence guided incremental footprint-based retrieval. Knowledge-Based Systems 14, 155–161.

[46] Tomek, I., 1976. Experiment with edited nearest-neighbor rule. IEEE Transactions On Systems Man And Cybernetics 6, 448–452.

[47] Tsang, E., Wang, X., 2005. An approach to case-based maintenance: Selecting representative cases. International Journal Of Pattern Recognition And Artificial Intelligence 19, 79–89.

[48] Wilson, D., 1972. Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions On Systems Man And Cybernetics SMC2, 408–&.

[49] Wilson, D., Martinez, T., 2000. Reduction techniques for instance-based learning algorithms. Machine Learning 38, 257–286.

[50] Yandell, B., 1997. Practical Data Analysis for Designed Experiments. Chapman & Hall/CRC Texts in Statistical Science, Taylor & Francis.