# Deleting and Building Sort Out Techniques for Case Base Maintenance

Maria Salamó and Elisabet Golobardes

Enginyeria i Arquitectura La Salle, Universitat Ramon Llull,
Psg. Bonanova 8, 08022 Barcelona, Spain
{mariasal,elisabet}@salleurl.edu

**Abstract.** Early work on case based reasoning reported in the literature shows the importance of case base maintenance for successful practical systems. Different criteria to the maintenance task have been used for more than half a century. In this paper we present different sort out techniques for case base maintenance. All the sort out techniques proposed are based on the same principle: a Rough Sets competence model. First of all, we present sort out reduction techniques based on deletion of cases. Next, we present sort out techniques that build new reduced competent case memories based on the original ones. The main purpose of these methods is to maintain the competence and reduce, as much as possible, its size. Experiments using different domains, most of them from the UCI repository, show that the reduction techniques maintain the competence obtained by the original case memory. The results are analysed with those obtained using well-known reduction techniques.

## 1 Introduction

Case-Based Reasoning (CBR) systems solve problems by reusing the solutions to similar problems stored as cases in a case memory [12] (also known as case-base). However, these systems are sensitive to the cases present in the case memory and often its accuracy rate depends on the significance of the stored cases. Therefore, in CBR systems it is important to reduce the case memory in order to remove noisy cases and also to achieve a good generalisation accuracy.

This paper presents two approaches, based on a Rough Sets competence model, to reduce the case memory while maintaining the competence. The two approaches are: (1) reduction techniques based on deletion of cases; (2) reduction techniques based on the construction of new competent cases. Both approaches have been introduced into our Case-Based Classifier System called BASTIAN[15]. This paper continues the initial Rough Sets approaches presented in our previous work [14], defining a competence model based on Rough Sets and explaining new approximations to improve weak points.

The reduction techniques proposed for the deletion approach are: Sort out Case Memory (SortOutCM) and Sort Out Internal Case Memory (SortOutInternalCM). The sort out reduction techniques obtain the quality of approximation to each case and separate it in the space of classes. Both techniques contain

the same foundations, but the SortOutCM has a more restrictive behaviour on deletion than SortOutInternalCM.

The reduction techniques proposed for the building approach are: SortOutMeanCM and SortOutMeanInternalCM. These techniques follow the initial approach of sort out techniques but their goal is different: the construction of new competent case memories using the information provided by the original ones.

The paper is organised as follows. Section 2 introduces some relevant related work. Next, section 3 describes the foundations of the Rough Sets Theory used in our reduction techniques. Then, section 4 details the proposed Rough Sets reduction techniques based on the deletion of cases and on the construction of new case memories. Section 5 describes the testbed of the experiments and the results obtained. Finally, section 6 presents some conclusions and further work.

## 2 Related Work

Case-Based Reasoning systems solve problems by reusing a corpus of previous solving experience stored as a case memory $T$ of solved cases $t$. A performance goal for any practical CBR system is the maintenance of a case memory $T$ maximizing coverage and minimizing case memory storage requirements.

Many researchers have addressed the problem of case memory reduction [21, 20] and different approaches have been proposed. The first kind of approaches are based on *nearest neighbours editing* rule (CNN,SNN,DEL,ENN,RENN)[21]. The second kind of approaches are related to Instance Based Learning algorithms (IBL) [1]. Another approach to instance pruning systems are those that take into account the order in which instances are removed (DROP1 to DROP5)[21].

Another way to approach this problem is to modify the instances themselves, instead of simply deciding which ones to keep. RISE [3] treats each instance as a rule that can be generalised. EACH [16] based on the *Nested Generalized Exemplars* (NGE) theory, uses hyperrectangles to replace one or more instances, introducing a generalization mechanism over the original training set. Another approach is the one proposed by GALE [7]. The goal is to induce a set of compact instances using evolutionary computation.

Finally, researchers have also focused on increasing the overall competence, *the range of target problems that can be successfully solved* [17], of the case memory through case deletion. Strategies have been developed for controlling case memory growth. Several methods such as competence-preserving deletion [17] and failure-driven deletion [10], as well as for generating compact case memories through competence-based case addition [18, 23, 19]. Leake and Wilson [6] examine the benefits of using fine-grained performance metrics to directly guide case addition or deletion. These methods are specially important for task domains with non-uniform problem distributions. The maintenance integrated with the overall case-based reasoning process was presented in [11]. Finally, a case-base maintenance method that avoids building sophisticated structures around a case-base or complex operations is presented by Yang and Wu [22]. Their method partitions cases into clusters that can be converted to new smaller case-bases.

## 3 Rough Sets theory

Zdzislaw Pawlak introduced Rough Sets theory in 1982 [9]. The idea of Rough Sets relies on the approximation of a set by a pair of sets. These sets are known as the lower and the upper approximation. These approximations are generated by the available data about the elements of the set.

We use Rough Sets theory for extracting the dependencies of knowledge. These dependencies are the basis for computing the relevance of instances into the Case-Based Classifier System. We use two measures of case relevance to decide which cases have to be deleted from the case memory applying different policies. The first measure (***Accuracy Rough Sets***) captures the *degree of completeness* of our knowledge. The second one (***Class Rough Sets***) computes the *quality of approximation* of each case. The following sections introduce some concepts and definitions required to define how to extract these two measures.

### 3.1 Introduction to the Rough Sets Theory

We have a **Universe** ($U$) (finite not null set of cases that describes our problem, i.e. the case memory). We compute from our universe the **concepts** (cases) that form partitions. The union of all the *concepts* make the entire Universe. Using *all the concepts* we can describe all the **equivalence relations** ($R$) over the universe $U$. Let an equivalence relation be a *set of features* that describe a specific concept. $U/R$ is the family of all **equivalence classes** of $R$. The universe and the relations form the **knowledge base** ($K$), defined as $K =< U, \hat{R} >$, where $\hat{R}$ is the family of equivalence relations over $U$. Every relation over the universe is an elementary concept in $K$. All the concepts are formed by a set of equivalence relations that describe them. Thus, the goal is to search for the minimal set of $R$ that defines the same concept as the initial set.

**Definition 1 (Indiscernibility Relations)**
$IND(\hat{P}) = \bigcap \hat{R}$ where $\hat{P} \subseteq \hat{R}$. The indiscernibility relation is an equivalence relation over $U$. Hence, it partitions the concepts (cases) into equivalence classes. These sets of classes are sets of instances indiscernible with respect to the features in $P$. Such a partition is denoted as $U/IND(P)$. In supervised machine learning the sets of cases indiscernible, with respect to the class attribute, contain the cases of each class.

***Approximations of Set.*** Given a condition set that contains all cases present in the case memory and a decision set that presents all the classes that the condition set has to classify. We are searching for a subset of the condition set able to classify the decision set. The following definitions explain this idea.

Let $K =< U, \hat{R} >$ be a knowledge base. For any subset of cases $X \subseteq U$ and an equivalence relation $R \in \hat{R}$, $R \subseteq IND(K)$ we associate two subsets called: Lower $\underline{R}X$; and Upper $\overline{R}X$ approximations. If $\underline{R}X{=}\overline{R}X$ then $X$ is an *exact set* (definable using subset $R$), otherwise $X$ is a ***rough set*** with respect to $R$.

**Definition 2 (Lower approximation)**
The lower approximation, defined as: $\underline{R}X = \bigcup\{Y \in U/R : Y \subseteq X\}$ is the set of all elements of $U$ which can *certainly* be classified as elements of $X$ in knowledge $R$.

**Definition 3 (Upper approximation)**
The upper approximation, $\overline{R}X = \bigcup\{Y \in U/R : X \bigcap Y \neq \emptyset\}$ is the set of elements of $U$ which can *possibly* be classified as elements of $X$, employing knowledge $R$.

***Reduct and Core of knowledge*** This part is related to the concept of reduction of the feature search space that defines the initial knowledge base. Next, this reduced space is used to extract the relevance of each case. Intuitively, a **reduct** of knowledge is its essential part which suffices to define all concepts occurring in the knowledge, whereas the **core** is the most important part.

Let $\hat{R}$ be a family of equivalence relations and $R \in \hat{R}$. We will say that:

- $R$ is *indispensable* if $IND(\hat{R}) \neq IND(\hat{R} - \{R\})$; otherwise it is *dispensable*. $IND(\hat{R} - \{R\})$ is the family of equivalence $\hat{R}$ extracting $R$.
- The family $\hat{R}$ is *independent* if each $R \in \hat{R}$ is *indispensable* in $R$; otherwise it is *dependent*.

**Definition 4 (Reduct)**
$\hat{Q} \in \hat{R}$ is a reduct of $\hat{R}$ if : $\hat{Q}$ is *independent* and $IND(\hat{Q}) = IND(\hat{R})$. Obviously, $\hat{R}$ may have many reducts. Using $\hat{Q}$ it is possible to approximate the same concept as using $\hat{R}$. Each reduct has the property that a feature can not be removed from it without changing the indiscernibility relation.

**Definition 5 (Core)**
The set of all indispensable relations in $\hat{R}$ will be called the *core* of $\hat{R}$, and will be denoted as: $CORE(\hat{R}) = \bigcap RED(\hat{R})$. Where $RED(\hat{R})$ is the family of all reducts of $\hat{R}$. It is the most characteristic part of knowledge and can not be eliminated.

### 3.2 Measures of relevance based on Rough Sets

**Accuracy Rough Sets** and **Class Rough Sets** measures use the information of reducts and the core to compute the relevance of each case.

***Accuracy Rough Sets*** This measure computes the *Accuracy* coefficient (**AccurCoef**) of each case $t$ in the knowledge base (case memory $T$) as:

$$For\ each\ instance\ t\ \in\ T\ it\ computes : AccurCoef(t) = \frac{card\ (\ \underline{P}(t))}{card\ (\ \overline{P}\ (t))} \quad (1)$$

Where $AccurCoef(t)$ is the relevance of the instance $t$; $T$ is the training set; *card* is the cardinality of one set; $P$ is the set that contains the *reducts* and *core* obtained from the original data; and finally $\underline{P}(t)$ and $\overline{P}(t)$ are the presence of $t$ in the lower and upper approximations, respectively.

The accuracy measure expresses the degree of completeness of our knowledge about the set $P$. The accuracy coefficient explains if an instance is on an internal region or on a border line region, thus $AccurCoef(t)$ is a binary value. When the value is 0 it means an internal case, and a value of 1 means an outlier case. Inexactness of a set of cases is due to the existence of a borderline region. The greater a borderline region of a set, the lower the accuracy of the set. The accuracy expresses the percentage of possible correct decisions made when classifying cases employing knowledge $P$.

***Class Rough Sets*** In this measure we use the *quality of classification* coefficient (**ClassCoef)**. It is computed as:

$$For\ each\ instance\ t\ \in\ T\ it\ computes: ClassCoef(t) = \frac{card\ (\ \underline{P}(t))}{card\ (T)} \quad (2)$$

Where $ClassCoef(t)$ is the relevance of the instance $t$; $T$ is the training set; $card$ is the cardinality of a set; $P$ is a set that contains the reducts and core; and finally $\underline{P}(t)$ is the presence of $t$ in the lower approximation.

The $ClassCoef$ coefficient expresses the percentage of cases which can be correctly classified employing the knowledge $t$. This coefficient has a range of values between 0 to 1, where 0 and 1 mean that the instance classifies incorrectly and correctly, respectively, the range of cases that belong to its class. The higher the quality, the nearer to the outlier region.

## 4 Reduction Techniques

In this section, we present the competence model and the two approximations techniques to reduce the case memory proposed in this paper: (1) Deletion techniques; (2) Building techniques. All these reduction techniques are based on the Rough Sets measures described in section 3.2. Each measure is a different point of view of the coverage of each case in the case memory. Once we have the coverage measures, we decide to combine both approaches in order to achieve a better competence and compact case memory using them.

The first part of this section describes the competence model in terms of our environment and our coverage measures. The second part assumes a competence model and defines two deletion techniques and two building techniques using this model. The aim is to verify if the model is feasible in its foundations. Therefore, we define the most simple building case memory technique, because we know in advance that if the model does not assure a minimal competence, the building technique will considerably degrade the competence.

### 4.1 Sort Out Case Memory techniques: Defining the model

First of all, we present the key concepts in categorising the cases in the sort out case memory (see figure 1(a)). The *coverage* and *reachability* concepts are modified, for our *coverage* coefficients and to our problem task, with regard to B. Smyth and M. Keane [17]. However, we maintain as far as possible the essence of the original ones. The *coverage* is computed using the Rough Sets coefficients explained in section 3.2. On the other hand, the *reachability* in this case is adapted to classification tasks.

**Definition 6 (Coverage)**
Let $T = \{t_1, t_2, ..., t_n\}$ be a training set of instances, $\forall\ t_i \in T$:
$Coverage(t_i) = AccurCoef(t_i) \vee ClassCoef(t_i)$

The *coverage* of a case is the accuracy and quality when it is used to solve a target problem. The *coverage* is computed using the $AccurCoef$ if it is 1 else the $ClassCoef$.

**Definition 7 (Reachability)**
Let $T = \{t_1, t_2, ..., t_n\}$ be a training set of instances, $\forall\ t_i \in T$:

$$Reachability(t_i) = \begin{cases} class\ (t_i) & if\ it\ is\ a\ classification\ task \\ adaptable(t_i, T) & if\ it\ is\ not\ a\ classification\ task \end{cases} \quad (3)$$

The original definition is maintained and extended to classification tasks. The *reachability* of a target problem is the set of cases that can be used to provide its solution.

**Definition 8 (Coverage group)**
Let $T = \{t_1, t_2, ..., t_n\}$ be a training set of instances and let $S$ be a subset of instances where $S \in T$. For all instances $i$ and $j$ in $S$:
$CoverageGroup(S) = Coverage(i) = Coverage(j)$

A coverage group (see figure 1(a)) is a set of cases from the case memory where all the cases have the same *coverage* without taking into account the class of each case. The coverage group shows space regions of our knowledge. The bigger a coverage group, the higher outlier the set of cases. The lower the coverage group, the higher an internal set of cases.

**Definition 9 (Reachability group)**
Let $T = \{t_1, t_2, ..., t_n\}$ be a training set of instances and let $S$ be a subset of instances where $S \in T$. For all instances $i$ and $j$ in $S$:
$ReachabilityGroup(S) = Reachability(i) = Reachability(j)$

A reachability group (see figure 1(a)) is the set of instances that can be used to provide a solution for the target. The reachability group produce the sort out of the case memory. However, a reachability group can contain different coverage groups. Every coverage group shows the levels of information (border line regions) in the reachability group.

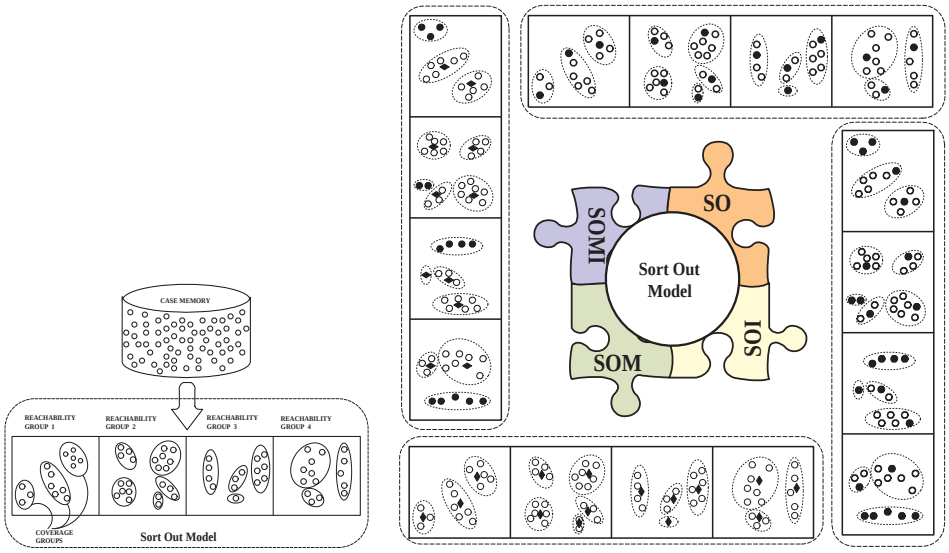
**Definition 10 (Master case)**
Let $S = \{s_1, s_2, ..., s_n\}$ and $T = \{t_1, t_2, ..., t_n\}$ be two sets of instances, where $S \in T$. For each $CoverageGroup(s) \in ReachabilityGroup(S)$ we have a:
$MasterCase(t) =$ A selected case $t$ from $ReachabilityGroup(S) \wedge CoverageGroup(s)$

Each coverage group contains a master case. Thus each reachability group contains as many master cases as coverage groups. The master cases will depend on the selection policies we use in our reduction techniques. These will be explained in the following sections.

### 4.2 Sort Out Deletion Policies

The reader will notice that algorithm 1 SortOutCM treats all cases using the same policy. The aim is to reduce as much as possible the number of cases in the case memory, and to treat not selected cases as $MasterCase$ as if they were redundant or irrelevant cases. There is no difference between the outlier cases and

the internal ones. However, it is known that the outlier cases contribute greatly to the competence of a system. The deletion of outliers reduces the competence of a system.




(a) Sort Out Case Memory (b) Sort Out Techniques

**Fig. 1.** Figure (a) describes the distribution of a case base for a four classes example using the sort out model and figure (b) describes the behaviour of each Sort Out technique for the previous example, where a ∘ represents a deleted case, • a $MasterCase$, and ⧫ a mean $MasterCase$.

This idea promotes a modification of the previous policy. We prefer to maintain or even improve the competence, selecting a fewer number of cases to be deleted from the case memory. The aim is twofold: first, to maintain the competence; second to improve utility of our case memory maintaining its diversity. Thus, an extension of the previous algorithm is algorithm 2 SortOutInternalCM.

**Algorithm 1 SortOutCM**

**SortOutCM** (CaseMemory $T$)
1. Sort out each instance $t \in T$ in its corresponding $ReachabilityGroup(S)$
2. Order decremented each $ReachabilityGroup(S)$ by $CoverageGroup(s) \in ReachabilityGroup(S)$
3. **for** each $ReachabilityGroup(S)$
4. **for** each $CoverageGroup(s) \in ReachabilityGroup(S)$
5. Select the first instance as a $MasterCase(t)$ to maintain in $T$
6. Delete the rest of instances from CaseMemory $T$ in the $CoverageGroup(s)$
7. **end for**
8. **end for**
9. **return** CaseMemory $T$

This algorithm 2 modifies only the internal *CoverageGroups* and maintains all the cases present in an outlier *CoverageGroup*. Therefore, the selection process in this algorithm uses a less restrictive policy. The outlier cases are isolated cases that no other case but itself can solve. Thus, it is important to maintain them because a $MasterCase$ can not be a good representative of the *CoverageGroup*. In this case, each case in a outlier *CoverageGroup* is an isolated space region of each class. It could be possible to find an outlier coverage group whose $MasterCase$ could be a good representative $MasterCase$, but this part involves further work.

**Algorithm 2 SortOutInternalCM**

**SortOutInternalCM** (CaseMemory $T$)
1. Sort out each instance $t \in T$ in its corresponding $ReachabilityGroup(S)$
2. Order decremented each $ReachabilityGroup(S)$ by $CoverageGroup(s) \in ReachabilityGroup(S)$
3. **for** each $ReachabilityGroup(S)$
4. **for** each $CoverageGroup(s) \in ReachabilityGroup(S)$
5. Select the first instance as a $MasterCase(t)$
6. **if** $Coverage(t) \neq 1.0$, Delete the rest of instances from $T$ in the $CoverageGroup(s)$
7. **elseif** $Coverage(t) = 1.0$
   Select the rest of instances as a $MasterCase(t)$ to maintain in $T$
8. **endif**
9. **end for**
10. **end for**
11. **return** CaseMemory $T$

### 4.3 Sort Out Building policies

Deletion techniques prompt a question: What is the reason for selecting the first case as a $MasterCase$? Actually, there is no specific reason but the implementation. The sort out case memory has all the cases ordered by their *coverage* and *reachability*, but when two cases have the same coverage, then the order of the initial case memory is maintained. However, this answer suggests new questions: Could it be possible to build a new case memory based on the original one? Could the sort out case memory be a model that guarantees a minimal competence?

The last question focuses on the assumption that the sort out case memory taken as a model itself enables the CBR system to maintain the competence. Therefore if we apply a building policy, which used without the model will surely decrease the competence, the model has to maintain it.

This section explains the modificacions in the previous algorithms in order to build new case memories using the *coverage*, the *reachability* and on the initial case memory. In order to test the reliability of this option, we use a simple policy. Algorithm 3 called SortOutMeanCM shows the modifications.

The SortOutMeanCM creates a new case for each *ReachabilityGroup* using all the cases that belong to the same *CoverageGroup*, computing the mean value for each attribute of these cases. This policy is based on gravity pointers.

**Algorithm 3 SortOutMeanCM**

**SortOutMeanCM** (CaseMemory $T$)
1. Sort out each instance $t \in T$ in its corresponding $ReachabilityGroup(S)$
2. Order decremented each $ReachabilityGroup(S)$ by $CoverageGroup(s) \in ReachabilityGroup(S)$
3. **for** each $ReachabilityGroup(S)$
4. **for** each $CoverageGroup(s) \in ReachabilityGroup(S)$
5. **for** each instance present in the $CoverageGroup(s)$ computes the mean value of each attribute $a$ as: $$\frac{\sum_{i=1}^{F} a_i}{card\ (\ cases\ in\ CoverageGroup(s)\ )}$$
6. **end for**
7. Delete all the instances from CaseMemory $T$ present in $CoverageGroup(s)$
8. Add computed Mean instance $t$ in $T$ with the same $Coverage(s)$ and $Reachability(S)$
9. **end for**
10. **end for**
11. **return** CaseMemory $T$

Initially the selection of gravity pointers is not a good policy. However, if the model produces a good distribution of cases, the competence will be maintained.

This algorithm inherits the same problem as the initial deletion algorithm: it treats the outlier cases the same as the internal cases. The case memory generated is a consequence of the previous data and the coefficients extracted using Rough Sets, which also uses the original case memory. Therefore, we modify the previous algorithm to select the set of cases that belong to an internal *CoverageGroup*. The algorithm 4 SortOutMeanInternalCM maintains the outlier cases without changing their content.

**Algorithm 4 SortOutMeanInternalCM**

**SortOutMeanInternalCM** (casememory $T$)
1. Sort out each instance $t \in T$ in its corresponding $ReachabilityGroup(S)$
2. Order decremented each $ReachabilityGroup(S)$ by $CoverageGroup(s) \in ReachabilityGroup(S)$
3. **for** each $ReachabilityGroup(S)$
4. **for** each $CoverageGroup(s) \in ReachabilityGroup(S)$
5. Select the first instance as a $MasterCase(t)$
6. **if** $Coverage(t) \neq 1.0$
7. **for** each instance present in the $CoverageGroup(s)$ computes the mean value of each attribute $a$ as: $$\frac{\sum_{i=1}^{F} a_i}{card\ (\ cases\ in\ CoverageGroup(s)\ )}$$
8. Delete $MasterCase(t)$
9. Delete all the instances from CaseMemory $T$ present in $CoverageGroup(s)$
10. Add computed Mean instance $t$ in $T$ with the same $Coverage(s)$ and $Reachability(S)$
11. **elseif** $Coverage(t) = 1.0$ Select the rest of instances of the $CoverageGroup(s)$ as a $MasterCase(t)$
12. **endif**
13. **end for**
14. **end for**
15. **return** CaseMemory $T$

Both methods (algorithms 3 and 4) contain the same number of cases as the deletion techniques (algorithms 1 and 2). The only difference between them are the sources of their instances. Deletion policies use the original case memory. However, building policies modify the original case memory to construct or build a new compact competence case memory generating new $MasterCases$. Figure 1(b) shows the behaviour of each algorithm when applied to a case base.

## 5 Experimental study

This section is structured as follows. First, we describe the testbed used in the empirical study. Then, we discuss the results obtained using the reduction techniques based on Rough Sets. We compare the results obtained to CBR system working with the original case memory. Finally, we also compare the results with some related learning systems.

### 5.1 Testbed

In order to evaluate the performance rate, we use ten datasets. These datasets can be grouped in two ways: *public* and *private*. The datasets and their characteristics are listed in table 1. **Public datasets** are obtained from the UCI repository [8]. They are: *breast cancer Wisconsin (Breast-Wisconsin), Glass, Ionosphere, Iris, Sonar and Vehicle*. **Private datasets** come from our own repository. They deal with *diagnosis* of breast cancer and *synthetic* datasets. Datasets related to diagnosis are *Biopsy* and *Mammogram*. *Biopsy* [4] is the result of digitally processed biopsy images, whereas *Mammogram* consists in detecting breast cancer using the microcalcifications ($\mu$Ca) present in a mammogram [5]. In *Mammogram* each example contains the description of several $\mu$Ca present in the image; in other words, the input information used is a set of real valued matrices. We also use two *synthetic* datasets to tune up the learning algorithms, because we knew their solutions in advance. *MX11* is the eleven input multiplexer. TAO-*grid* is a dataset obtained from sampling the TAO figure using a grid [7].

These datasets were chosen in order to provide a wide variety of application areas, sizes, combinations of feature types, and difficulty as measured by the accuracy achieved on them by current algorithms. The choice was also made with the goal of having enough data points to extract conclusions.

**Table 1.** Datasets and their characteristics used in the empirical study.

| | Dataset | Ref. | Samples | Numeric feat. | Symbolic feat. | Classes | Inconsistent |
|---|---|---|---|---|---|---|---|
| 1 | *Biopsy* | *BI* | 1027 | 24 | - | 2 | Yes |
| 2 | *Breast-Wisconsin* | *BC* | 699 | 9 | - | 2 | Yes |
| 3 | *Glass* | *GL* | 214 | 9 | - | 6 | No |
| 4 | *Ionosphere* | *IO* | 351 | 34 | - | 2 | No |
| 5 | *Iris* | *IR* | 150 | 4 | - | 3 | No |
| 6 | *Mammogram* | *MA* | 216 | 23 | - | 2 | Yes |
| 7 | *MX11* | *MX* | 2048 | - | 11 | 2 | No |
| 8 | *Sonar* | *SO* | 208 | 60 | - | 2 | No |
| 9 | *TAO-Grid* | *TG* | 1888 | 2 | - | 2 | No |
| 10 | *Vehicle* | *VE* | 846 | 18 | - | 4 | No |

The study described in this paper was carried out in the context of BASTIAN, a *case*-**BA***sed* **S***ys***T***em* **I***n cl***A***ssificatio***N**. BASTIAN has been developed in JAVA, for details see [13]. All techniques were run using the same set of parameters for all datasets. The configuration of BASTIAN platform for this paper is set as follows. It uses a 1-Nearest Neighbour Algorithm. The case memory is represented as a list of cases. Each case contains the set of attributes, its class

and the AccurCoef and ClassCoef coefficients. Our goal is to test the reliability and feasibility of the reduction techniques. Therefore, we have not focused on the case representation used by the system. The retain phase does not store any new case in the case memory, so the CBR system only contains the initial case memory. Finally, no weighting method is used in this paper in order to test the reliability of our reduction techniques. Further work will consist of testing the influence of these methods in conjunction with weighting methods.

The percentage of correct classifications has been averaged over stratified ten-fold cross-validation runs. We analyse the significance of the performance using paired *t*-test on these runs.

### 5.2 Experimental analysis of the reduction techniques

The experimental results for each dataset using CBR system and Rough Sets reduction techniques (SortOutCM (SO), SortOutInternalCM (SOI), SortOutMeanCM (SOM) and SortOutMeanInternalCM (SOMI)) are shown in table 2.

**Table 2.** Mean percentage of correct classifications (%PA) and mean storage size (%CM). Two-sided paired t-test (p = 0.1) is performed, where a • and ◦ stand for a significant improvement or degradation of our CBR related to the reduction technique compared. Bold font indicates the best prediction accuracy.

| **Ref.** | **CBR** | | **SO** | | **SOI** | | **SOM** | | **SOMI** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM |
| *BI* | 83.15 | 100.0 | 75.17◦ | 0.94 | **83.75** | 88.74 | 79.96◦ | 0.94 | **83.75** | 88.74 |
| *BC* | **96.28** | 100.0 | 95.59 | 3.17 | 95.85 | 29.42 | 95.41 | 3.17 | 95.99 | 29.42 |
| *GL* | **72.42** | 100.0 | 63.64 | 18.11 | 64.48 | 37.89 | 65.89 | 18.11 | 64.37 | 37.89 |
| *IO* | 90.59 | 100.0 | 83.48◦ | 4.71 | **91.16** | 50.68 | 88.02 | 4.71 | 90.03 | 50.68 |
| *IR* | **96.0** | 100.0 | 91.33◦ | 12.44 | 91.33◦ | 13.18 | 94.0 | 12.44 | 93.33 | 13.18 |
| *MA* | **64.81** | 100.0 | 59.75 | 7.98 | 58.04 | 25.36 | 59.74 | 7.98 | 57.19 | 25.36 |
| *MX* | **78.61** | 100.0 | 66.74◦ | 0.1 | **78.61** | 99.9 | 66.35◦ | 0.1 | **78.61** | 99.9 |
| *SO* | 84.61 | 100.0 | 68.90◦ | 4.45 | 86.42• | 65.15 | 73.38◦ | 4.45 | **87.50•** | 65.15 |
| *TG* | **95.76** | 100.0 | 89.60◦ | 1.37 | 89.66◦ | 1.37 | 91.31◦ | 1.37 | 91.31◦ | 1.37 |
| *VE* | 67.37 | 100.0 | 56.79◦ | 3.68 | 69.70 | 68.33 | 58.46◦ | 3.68 | **69.95** | 68.33 |

The aim of our reduction techniques is to reduce the case memory while maintaining the competence of the system. This priority guides our sort out reduction techniques based on Rough Sets competence model. Following this criterion, the results related to SortOutCM and SortOutMeanCM are not good because, as predicted in their description, the deletion or building of outlier cases produce a competence loss. However, the sort out internal techniques have a different behaviour. For example, the *Sonar* dataset obtains a good competence as well as it reduces the case memory, in both approximations: SortOutInternalCM (SOI) and SortOutMeanInternalCM (SOMI). Thus, we denote that the sort out case memories need to maintain the outlier cases present in the original case memory.

Comparing deletion versus building reduction techniques, we conclude that both techniques obtain similar competence on all datasets. However, building methods obtain on average the best competence. These results influence our further work because the gravity points policy chosen was the most simple in order to test the reliability of these techniques and the feasibility of the sort out case memories.

To sum up, the results obtained using the sort out reduction techniques, deleting or building policies, on average maintain the competence of the system while reducing as much as possible the case memory. There are some datasets that present competence loss whereas the reduction increases. This occurs because some of the existing *CoverageGroups* must be deleted and not selected to build or maintain a *MasterCase*, because the coverage of the group is so near the outlier space regions that its maintenance prevents the case base reasoning system from separating correctly between different classes. This fact can be observed in some datasets, for example *Mammogram* and *Glass*, where the results obtained using whatever method are similar. The solution of this weak point is part of our further work.

**Table 3.** Mean percentage of correct classifications (%PA) and mean storage size (%CM). Two-sided paired t-test (p = 0.1) is performed, where a • and ○ stand for a significant improvement or degradation of our SOMI approach related to the system compared. Bold font indicates the best prediction accuracy.

| Ref. | SOMI | | SOI | | CBR | | IB2 | | IB3 | | IB4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM |
| *BI* | **83.75** | 88.74 | **83.75** | 88.74 | 83.15 | 100.0 | 75.77• | 26.65 | 78.51• | 13.62 | 76.46• | 12.82 |
| *BC* | 95.99 | 29.42 | 95.85 | 29.42 | **96.28** | 100.0 | 91.86• | 8.18 | 94.98 | 2.86 | 94.86 | 2.65 |
| *GL* | 64.37 | 37.89 | 64.48 | 37.89 | **72.42** | 100.0 | 62.53 | 42.99 | 65.56 | 44.34 | 66.40 | 39.40 |
| *IO* | 90.03 | 50.68 | **91.16** | 50.68 | 90.59 | 100.0 | 86.61 | 15.82 | 90.62 | 13.89 | 90.35 | 15.44 |
| *IR* | 93.33 | 13.18 | 91.33 | 13.18 | 96.0 | 100.0 | 93.98 | 9.85 | 91.33 | 11.26 | **96.66** | 12.00 |
| *MA* | 57.19 | 25.36 | 58.04 | 25.36 | 64.81 | 100.0 | **66.19** | 42.28 | 60.16 | 14.30 | 60.03 | 21.55 |
| *MX* | 78.61 | 99.9 | 78.61 | 99.9 | 78.61 | 100.0 | **87.07**○ | 18.99 | 81.59 | 15.76 | 81.34 | 15.84 |
| *SO* | **87.50** | 65.15 | 86.42 | 65.15 | 84.61 | 100.0 | 80.72 | 27.30 | 62.11• | 22.70 | 63.06• | 22.92 |
| *TG* | 91.31 | 1.37 | 89.66• | 1.37 | **95.76**○ | 100.0 | 94.87○ | 7.38 | 95.04○ | 5.63 | 93.96○ | 5.79 |
| *VE* | **69.95** | 68.33 | 69.70 | 68.33 | 67.37 | 100.0 | 65.46• | 40.01 | 63.21• | 33.36 | 63.68• | 31.66 |

Sort out internal techniques (SOI and SOMI) obtain on average a higher generalisation on accuracy than IBL, as shown in table 3. The performance of IBL algorithms decline, in almost all datasets (e.g. *Breast-Wisconsin, Biopsy*), when case memory is reduced. SOMI and SOI obtains on average higher prediction accuracy than IB2, IB3 and IB4. On the other hand, the mean storage size obtained is higher in our reduction techniques than those obtained in IBL schemes.

Finishing the experimental study, we also run several well-known reduction schemes on the previous data sets. The reduction algorithms are: CNN, SNN, DEL, ENN, RENN, DROP1, DROP2, DROP3, DROP4 and DROP5 (a complete explanation of them can be found in [21]). We use the same datasets described above but with different ten-fold cross validation sets. We want to analyse the results obtained using the proposed SortOutMeanInternalCM reduction technique with those obtained by these reduction techniques. Tables 4 and 5 illustrate the mean prediction accuracy and the mean storage size for all systems in all datasets, respectively.

Table 4 shows the behaviour of our SortOutMeanInternalCM reduction technique in comparison with CNN, SNN, DEL, ENN and RENN techniques. SOMI results are on average better than those obtained by the reduction techniques studied. RENN improves the results of SortOutMeanInternalCM (SOMI) in

**Table 4.** Mean percentage of correct classifications (%PA) and mean storage size (%CM). Two-sided paired t-test (p = 0.1) is performed, where a • and ○ stand for a significant improvement or degradation of our SOMI approach related to the system compared. Bold font indicates the best prediction accuracy.

| Ref. | SOMI | | CNN | | SNN | | DEL | | ENN | | RENN | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM |
| *BI* | **83.75** | 88.74 | 79.57• | 17.82 | 78.41• | 14.51 | 82.79• | 0.35 | 77.82• | 16.52 | 81.03• | 84.51 |
| *BC* | 95.99 | 29.42 | 95.57 | 5.87 | 95.42 | 3.72 | 96.57 | 0.32 | 95.28 | 3.61 | **97.00**○ | 96.34 |
| *GL* | 64.37 | 37.89 | 67.64○ | 24.97 | 67.73 | 20.51 | 64.87 | 4.47 | 68.23 | 19.32 | **68.66**○ | 72.90 |
| *IO* | **90.03** | 50.68 | 88.89 | 9.94 | 85.75• | 7.00 | 80.34• | 1.01 | 88.31 | 7.79 | 85.18• | 86.39 |
| *IR* | 93.33 | 13.18 | **96.00** | 14.00 | 94.00 | 9.93 | **96.00** | 2.52 | 91.33 | 8.59 | **96.00** | 94.44 |
| *MA* | 57.19 | 25.36 | 61.04 | 25.06 | 63.42 | 18.05 | 62.53 | 1.03 | 63.85 | 21.66 | **65.32** | 66.92 |
| *MX* | 78.61 | 99.9 | 89.01○ | 37.17 | 89.01○ | 37.15 | 68.99• | 0.55 | 85.05○ | 32.54 | **99.80**○ | 99.89 |
| *SO* | **87.50** | 65.15 | 83.26 | 23.45 | 80.38 | 20.52 | 77.45• | 1.12 | 85.62 | 19.34 | 82.74 | 86.49 |
| *TG* | 91.31 | 1.37 | 94.39○ | 7.15 | 94.76○ | 6.38 | 87.66• | 0.26 | **96.77**○ | 3.75 | 95.18○ | 96.51 |
| *VE* | **69.95** | 68.33 | 69.74 | 23.30 | 69.27 | 19.90 | 62.29• | 2.55 | 66.91• | 20.70 | 68.67 | 74.56 |

some data sets (e.g. *Breast-Wisconsin*) but its reduction on the case memory is lower than SOMI.

**Table 5.** Mean percentage of correct classifications (%PA) and mean storage size (%CM). Two-sided paired t-test (p = 0.1) is performed, where a • and ○ stand for a significant improvement or degradation of our SOMI approach related to the system compared. Bold font indicates best prediction accuracy.

| Ref. | SOMI | | DROP1 | | DROP2 | | DROP3 | | DROP4 | | DROP5 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM | %PA | %CM |
| *BI* | 83.75 | 88.74 | 76.36• | 26.84 | 76.95• | 29.38 | 77.34• | 15.16 | 76.16• | 28.11 | 76.17• | 27.03 |
| *BC* | 95.99 | 29.42 | 93.28• | 8.79 | 92.56• | 8.35 | 96.28 | 2.70 | 95.00 | 4.37 | 93.28• | 8.79 |
| *GL* | 64.37 | 37.89 | 66.39 | 40.86 | 69.57○ | 42.94 | 67.27 | 33.28 | 69.18○ | 43.30 | 65.02 | 40.65 |
| *IO* | 90.03 | 50.68 | 81.20• | 23.04 | 87.73 | 19.21 | 88.89 | 14.24 | 88.02 | 15.83 | 81.20• | 23.04 |
| *IR* | 93.33 | 13.18 | 91.33 | 12.44 | 90.00 | 14.07 | **92.66** | 12.07 | 88.67• | 7.93 | 91.33 | 12.44 |
| *MA* | 57.19 | 25.36 | **61.60** | 42.69 | 58.33 | 51.34 | 58.51 | 12.60 | 58.29 | 50.77 | **61.60** | 42.64 |
| *MX* | 78.61 | 99.9 | 87.94○ | 19.02 | **100.00**○ | 98.37 | 82.37○ | 17.10 | 86.52○ | 25.47 | 86.52○ | 18.89 |
| *SO* | 87.50 | 65.15 | 84.64 | 25.05 | **87.07** | 28.26 | 76.57• | 16.93 | 84.64 | 26.82 | 84.64 | 25.11 |
| *TG* | 91.31 | 1.37 | 94.76○ | 8.03 | **95.23**○ | 8.95 | 94.49○ | 6.76 | 89.41• | 2.18 | 94.76○ | 8.03 |
| *VE* | 69.95 | 68.33 | 64.66• | 38.69 | 67.16 | 43.21 | 66.21• | 29.42 | 68.21 | 43.85 | 64.66• | 38.69 |

The results in table 5 report that SortOutMeanInternalCM obtains a balanced behaviour between competence and size. On the other hand, there are some reduction techniques that obtain best competence for some data sets making a smaller reduction of the case memory size. Sort out technique shows better competence for some data sets (e.g. *Biopsy, Breast-w, Vehicle*), although its results are also worse in others (e.g. *MX11*).

All the experiments (tables 3, 4 and 5) lead to some interesting observations. First, it is worth noting that the individual SortOutInternalCM (SOI) and SortOutMeanInternalCM (SOMI) work correctly in all data sets, obtaining better results using SOMI because the gravity pointer selection of $MasterCases$ is more representative than the first case of each $CoverageGroup$. Therefore, as a second conclusion, we demonstrate the feasibility of the sort out case memories and the competence model. Finally, the results in all tables suggest that all the reduction techniques work well in some, but not all, domains. This has

been termed the *selective superiority problem* [2]. Consequently, future work will consist of improving the selection of $MasterCases$ in order to enlarge the outlier cases from the internal ones to improve the overall competence in all the domains.

## 6 Conclusions and further work

This paper presents a competence model defined using Rough Sets theory. Under this competence model it introduces two different approaches to the reduction of the case memory: the first one presents different deletion techniques; the second one relies on building techniques. The aim of this paper was twofold: (1) to denote that the deletion techniques of sort out case memories are reliable, and (2) to reveal that the construction of case memories will be feasible using a competence model and sort out case memories. Empirical studies show that these reduction techniques produce a higher or equal generalisation accuracy on classification tasks. We can conclude that the deletion policies could be improved in some facets and the building policies are a promising area to study. Our further work will be focused on these observations. Firstly, we can modify the selection of $MasterCases$ in order to enlarge the distance from internal cases to outlier ones and to obtain a higher competence. Secondly, the building policies have to avoid some gravity points. Therefore, it could be interesting to study different methods to build the case memory. Finally, we want to analyse the influence of the weighting methods and similarity functions in these reduction techniques.

## Acknowledgements

This work is supported by the *Ministerio de Sanidad y Consumo, Instituto de Salud Carlos III*, *Fondo de Investigación Sanitaria* of Spain, Grant No. 00/0033-02. We wish to thank *Enginyeria i Arquitectura La Salle* (Ramon Llull University) for their support to our Research Group in Intelligent Systems. We also wish to thank D. Aha for providing the IBL code as well as D. Randall Wilson and Tony R. Martinez who provided the code of the other reduction techniques. Finally, we wish to thank the anonymous reviewers for their useful comments.

## References

1. D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning, Vol. 6*, pages 37–66, 1991.
2. C.E. Brodley. Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Proceedings of the 10th International Conference on Machine Learning*, pages 17–24, 1993.
3. P. Domingos. Context-sensitive feature selection for lazy learners. In *AI Review*, volume 11, pages 227–253, 1997.
4. J.M. Garrell, E. Golobardes, E. Bernadó, and X. Llorà. Automatic diagnosis with Genetic Algorithms and Case-Based Reasoning. 13:367–362, October 1999. Elsevier Science Ltd., ISSN 0954-1810.

5. E. Golobardes, X. Llorà, M. Salamó, and J. Martí. Computer Aided Diagnosis with Case-Based Reasoning and Genetic Algorithms. *Knowledge-Based Systems*, (15):45–52, 2002.
6. D. Leake and D. Wilson. Remembering Why to Remember: Performance-Guided Case-Base Maintenance. In *Proceedings of the Fifth European Workshop on Case-Based Reasoning*, pages 161–172, 2000.
7. X. Llorà and J.M. Garrell. Inducing Partially-Defined Instances with Evolutionary Algorithms. In *Proceedings of the 18th International Conference on Machine Learning (ICML'2001)*, pages 337–344. Morgan Kaufmann Publishers, 2001.
8. C. J. Merz and P. M. Murphy. UCI Repository for Machine Learning Data-Bases [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. *Irvine, CA: University of California, Department of Information and Computer Science*, 1998.
9. Z. Pawlak. Rough Sets. In *International Journal of Information and Computer Science*, volume 11, 1982.
10. L. Portinale, P. Torasso, and P. Tavano. Speed-up, quality and competence in multi-modal reasoning. In *Proceedings of the Third International Conference on Case-Based Reasoning*, pages 303–317, 1999.
11. T. Reinartz and I. Iglezakis. Review and Restore for Case-Base Maintenance. *Computational Intelligence*, 17(2):214–234, 2001.
12. C.K. Riesbeck and R.C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, NJ, US, 1989.
13. M. Salamó and E. Golobardes. BASTIAN: Incorporating the Rough Sets theory into a Case-Based Classifier System. In *Butlletí de l'acia: III Congrés Català d'Intel·ligència Artificial (CCIA'00)*, pages 284–293, Barcelona, Spain, October 2000.
14. M. Salamó and E. Golobardes. Rough sets reduction techniques for case-based reasoning. In *Proceedings 4th. International Conference on Case-Based Reasoning, ICCBR 2001*, pages 467–482, Vancouver, BC, Canada, 2001.
15. M. Salamó, E. Golobardes, D. Vernet, and M. Nieto. Weighting methods for a Case-Based Classifier System. In *LEARNING'00*, Madrid, Spain, October 2000. IEEE.
16. S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309, 1991.
17. B. Smyth and M. Keane. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the Thirteen International Joint Conference on Artificial Intelligence*, pages 377–382, 1995.
18. B. Smyth and E. McKenna. Building compact competent case-bases. In *Proceedings of the Third International Conference on Case-Based Reasoning*, pages 329–342, 1999.
19. B. Smyth and E. McKenna. Competence Models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.
20. D.C. Wilson and D.B. Leake. Maintaining Case-Based Reasoners:Dimensions and Directions. *Computational Intelligence*, 17(2):196–213, 2001.
21. D.R. Wilson and T.R. Martinez. Reduction techniques for Instance-Based Learning Algorithms. *Machine Learning, 38*, pages 257–286, 2000.
22. Q. Yang and J. Wu. Keep it Simple: A Case-Base Maintenance Policy Based on Clustering and Information Theory. In *Proc. of the Canadian AI Conference*, pages 102–114, 2000.
23. J. Zhu and Q. Yang. Remembering to add: Competence-preserving case-addition policies for case base maintenance. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, volume 1, pages 234–239, 1999.