Justification-based Selection of Training Examples for Case Base Reduction

Santiago Ontañón and Enric Plaza

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain).
{santi,enric}@iiia.csic.es, http://www.iiia.csic.es

Abstract. Maintaining compact and competent case bases has become a main topic of Case Based Reasoning (CBR) research. The main goal is to obtain a compact case base (with a reduced number of cases) without losing accuracy. In this work we present JUST, a technique to reduce the size of a case base while maintaining the classification accuracy of the CBR system. JUST uses *justifications* in order to select a subset of cases from the original case base that will form the new reduced case base. A justification is an explanation that the CBR system generates to justify the solution found for a given problem. Moreover, we present empirical evaluation in various data sets showing that JUST is an effective case base reduction technique that maintains the classification accuracy of the case base.

Keywords: CBR, Case Base Management, Case Base Reduction.

1 Introduction

Maintaining compact and competent case bases has become a main topic of Case Based Reasoning (CBR) research. The main goal is to obtain a compact case base (with a reduced number of cases) without losing problem solving accuracy. In modern CBR systems, with large case bases, it has been found that adding new cases into the case base is not always beneficial. Smyth and Cunningham [7] analyze this problem and find that although similarity-based methods (such as the ones typically used in CBR) do not usually suffer from overfitting when adding new cases into the case base, the efficiency of the system can degrade. The efficiency of the system can be divided in two factors: the retrieval time and the reuse time. While reuse time diminishes as the case base grows, retrieval time increases. Therefore, by adding new cases into an already saturated cases base, the same problem solving performance is achieved, but with a reduced efficiency.

There has been significant research in case base maintenance recently. Aha et al. [2, 1] propose several algorithms for reducing case bases (CBL2, CBL3 and CBL4) based on the rule "if I can correctly solve a P with a case base C, then it is not interesting to add P to C". Related to CBL2 is also the Condensed Nearest Neighbor algorithms [4, 9, 5].

Another line of case base maintenance policies are those of Smyth and Keane [8]. They define several competence metrics based on finding "competence groups" inside a case base that they later use to define case base maintenance policies by deleting from the case base those cases with minimum competence. Later, Zhu and Yang [10] propose an alternative version of the Smyth and Keane strategy based on case addition instead of case deletion.

Salamó and Golobardes [6] propose two deletion policies for CBR systems based on the rough set theory: *Accuracy-Classification Case Memory* ACCM and *Negative Accuracy-Classification Case Memory* NACCM. Both ACCM and NACCM are able to keep classification accuracy at a very high level (sometimes even improving the accuracy of the complete case base), but the reduction of cases obtained is not as large as algorithms such as CBL2.

In this paper we present a novel approach to case base maintenance based on the concept of *justifications*. A justification is an explanation that the CBR system generates to justify the solution found for a given problem. The analysis of justifications is able to find the weak points (or the "competence holes") of a case base. We will present a technique called JUST (JUstification-based Selection of Training examples) to reduce the size of a case base while maintaining the classification accuracy of the CBR system. Suppose that we have a CBR system with a case base C consisting of n cases. JUST will construct a reduced case base C^r by selecting cases from C. JUST follows the case addition strategy presented by Zhu and Yang [10] (in the sense that it also constructs a new case base by selecting cases from C) but using justifications in order to select which cases to select from C instead of a competence measure.

The structure of the paper is as follows. First, in Section 2, we formally present the idea of justifications. Then, Section 3 presents the JUST technique. After that, we present an empirical evaluation of JUST and the paper closes with the conclusions section.

2 Justifications

Let $C = \{c_1, \ldots, c_n\}$ be the case base of a CBR system, composed of n cases. Each case $c_i = \langle P, S \rangle$ is a tuple containing a problem P and a solution S. We will use the dot notation to refer to the elements inside a tuple, i.e. we will note $c_i.S$ to make reference to the solution in case c_i. Usually, when a classifier solves a problem P, the output is only a solution class (or at most a ranked list of solution classes with an associated probability). However, some machine learning methods can provide more information than just the solution class. Specifically, some methods can output a *justification*.

Definition: A *justification* J built by a CBR method to solve a problem P that has been classified into a solution class S_k is a description containing the relevant information of P for having predicted S_k as the solution for P.

In our work, we use LID [3], a lazy learning method for CBR systems capable of building symbolic justifications. LID uses the feature term formalism to rep-

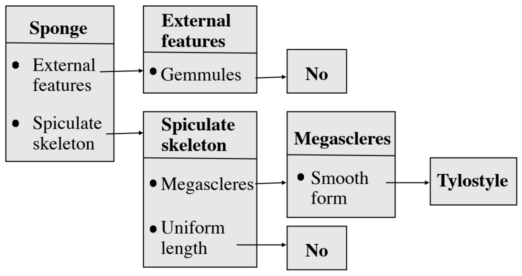

Fig. 1. Simbolic justification returned by LID.

resent cases. *Feature Terms* (or ψ-terms) are a generalization of the first order terms. The main difference is that in first order terms (e.g. $person(x_1, x_2, x_3)$) the parameters of the terms are identified by position, while in a feature term the parameters (called *features*) are identified by name (e.g. $person[name \doteq x_1, father \doteq x_2, mother \doteq x_3]$). Another difference is that feature terms have a *sort*, for instance, the previous example belongs to the sort *person*. These sorts can have subsorts (e.g. *man* and *woman* are subsorts of *person*). Feature terms have an informational order relation ($\sqsubseteq$) among them called subsumption, where $\psi \sqsubseteq \psi'$ means all the information contained in ψ is also contained in ψ'. We say that ψ subsumes ψ' (or that ψ' satisfies ψ); we can also say that ψ is a generalization of ψ'. When a feature term has only a sort and no features, it is called a *leaf*.

Figure 1 shows a symbolic justification J returned by LID, represented as a feature term. Each box in the figure represents a node. On the top of a box the sort of the node is shown, and on the lower part, the features with a known value are shown. The arrows mean that the feature on the left part of the arrow takes the node at the right as value; nodes *No* and *Tylostile* are leaf nodes. LID has returned the justification J for classifying a problem P in a specific solution class S_k such that $J \sqsubseteq P$. In addition, there is a subset of cases $c_1, \ldots . c_r$ retrieved from the case base such that $\forall c_i \in \{c_1, \ldots . c_r\}, J \sqsubseteq c_i.P$. These are the cases that *endorse* S_k as a solution for P, since all (or the majority) of them have solution S_k. Moreover, J is a symbolic similarity description since it contains what is shared between P and $c_1, ..., c_r$ and that is relevant (not all that is shared). Notice that albeit J is a generalization of all the cases $c_1, ..., c_r$ and of the problem P LID is still a lazy learning method. The difference between induction and lazy learning is that induction builds a global approximation of the concept to be learnt, while lazy learning builds local approximations around the problem P to be solved. From this viewpoint, a justification J for a problem P is the local approximation built by LID, and the form of J is a symbolic description that generalizes $c_1, ..., c_r$ and P.

3 Justification-based Selection of Training Examples

This section presents the JUST (Justification-based Selection of Training examples) technique. JUST is a case base reduction technique whose goal is to reduce the number of cases in a given case base without reducing the classification accuracy obtained with that case base.

JUST is an iterative technique that selects cases from case base C and adds them to another (reduced) case base C^r, until certain termination criterion is met. The termination criterion could be any property of the new case base C^r, but we will focus on these two: finding a case base C^r with at most M cases, finding a case base C^r with a certain accuracy level α. Before explaining JUST, we need to introduce some concepts:

Definition: An *exam* $E = \{P_1, ..., P_m\}$ is a set of problems (i.e. unlabeled examples) for which the system knows the solution and that JUST has still not added into C^r, i.e. $E \subseteq \{c_i.P | c_i \in C \wedge c_i \notin C^r\}$

The way JUST builds exams is by maintaining a set C^u of cases that are present in C and that are not in C^r. The problems in any subset of cases from C^u are a valid exam. The idea of the exams is to build a set of problems with which to evaluate the performance of the new case base C^r. Moreoever, when the system solves a problem, a *Justified Endorsing Record* is built:

Definition: A Justified Endorsing Record (JER) $\mathbf{J}^k = \langle P_k, S, J \rangle$ is a tuple containing the problem P_k, the solution class S predicted for the problem P_k, and the justification J for S being a solution for P_k.

We will note $\mathbf{J}_E = \{\mathbf{J}^k | \mathbf{J}^k.P = P_k \in E\}$ the set of JERs build by the system using the reduced case base C^r for each problem $P_k \in E$. Moreover, since we know that the correct solution for P_k is S_k, we can define the set of incorrect JERs $\mathbf{J}_E^- = \{\mathbf{J}^k | \mathbf{J}^k \in \mathbf{J}_E \wedge \mathbf{J}^k.S \neq S_k\}$ (i.e. the JERs for the problems in E for which the system has predicted an incorrect solution using the case base C^r.

A case c_i is a *counterexample* of an incorrect JER ($\mathbf{J}^k \in \mathbf{J}_E^-$) if $c_i.P$ is subsumed by the incorrect justification $\mathbf{J}^k.J$ and $c_i.S$ is a different solution class than the predicted one, i.e. $\mathbf{J}^k.J \sqsubseteq c_i.P$ and $c_1.S \neq \mathbf{J}^k.S$. Moreover, we can define also a *valid counterexample* as a counterexample c_i that predicts the correct solution for P_k, i.e. $c_i.S = S_k$.

Definition: The *refutation set* $R_{\mathbf{J}^k}$ of an incorrect JER $\mathbf{J}^k$ is defined as the set of cases from C^u that are valid counterexamples of $\mathbf{J}^k$. Formally: $R_{\mathbf{J}^k} = \{c_i \in C^u | \mathbf{J}^k.J \sqsubseteq c_i.P \wedge c_i.S = S_k\}$, where S_k is the correct solution class for the problem $\mathbf{J}^k.P = P_k$.

The examples in a refutation set $R_{\mathbf{J}^k}$ are the examples that can potentially prevent the system from making similar errors in the future and therefore they are candidate examples to be added to C^r. The collection of the refutation sets for all the incorrect JERs $\mathbf{J}_E^-$ will be noted as $\mathbf{R} = \{R_{\mathbf{J}^k} | \mathbf{J}^k \in \mathbf{J}_E^-\}$. Finally, we define the *belying set* as follows:

```
Function JUST ($C, \mathcal{T}, m$)
    $t = 0$; $C_0^r = \emptyset$; $C_0^u = C$;
    Do
        $E_t$ = select-exam($C_t^u, m$);
        $\mathbf{J}_{E_t}$ = build-JERs($E_t$);
        $\mathbf{J}_{E_t}^- = \{\mathbf{J}^k | \mathbf{J}^k \in \mathbf{J}_{E_t} \wedge \mathbf{J}^k.S \neq S_k\}$;
        $\mathbf{R}_t$ = build-refutation-sets($\mathbf{J}_{E_t}^-, C_t^u$);
        $B_t$ = build-belying-set($\mathbf{R}_t$);
        $C_{t+1}^r = C_t^r \cup B_t$; $t = t + 1$;
    While(not $\mathcal{T}$);
    Return($C_t^r$);
End-Function
```

Fig. 2. The JUST algorithm, where C is the initial case base, $\mathcal{T}$ is the termination criterion and m is the exam size.

Definition: The *belying set* B is the minimum set of counterexamples that belies all the incorrect justifications built by the CBR system over an exam E using the case base C^r. Formally, the belying set is the collection of cases $B \subseteq C^u$ such that $\forall R_{\mathbf{J}^k} \in \mathbf{R} : B \cap R_{\mathbf{J}^k} \neq \emptyset$ and that $\nexists B' | \forall R_{\mathbf{J}^k} \in \mathbf{R} : B' \cap R_{\mathbf{J}^k} \neq \emptyset \wedge B' \subset B$. Notice that a) the belying set contains at least one counterexample that belies each one of the incorrect justifications and b) the belying set contains the minimum number of counterexamples that belie all incorrect justifications. For instance, if two refutation sets for two JERs $\mathbf{J}^k$ and $\mathbf{J}_j$ share a counterexample c_i, including c_i into B is enough to belie both incorrect justifications.

After introducing these definitions we can now present the JUST method. Figure 2 shows the JUST iterative algorithm. At each iteration t, JUST will select some cases from C to be added to the new reduced case base. We will define three sets of cases at each iteration t: C_t^r is the reduced case base created by JUST (containing all the cases selected in the previous iterations), $C_t^u = C - C_t^r$ are the cases from C that are not in C_t^r and finally B_t is the set of cases selected by JUST from C to be added to C_t^r in the iteration t.

The JUST method works as follows: Initially, $t = 0$, $C_0^r = \emptyset$ and $C_0^u = C$. At each iteration t, JUST builds an exam $E_t \subseteq C_t^u$ of size m. The size m of the exams is a parameter of JUST, in Section 4 we will analyze the effect of varying the parameter m. The CBR system solves all the problems in E_t using the case base C_t^r and builds the set of JERs $\mathbf{J}_{E_t}$. Then, the set $\mathbf{J}_{E_t}^-$ containing all the incorrect JERs in $\mathbf{J}_{E_t}$ is built. Notice that JUST can determine whether a JER $\mathbf{J}^k \in \mathbf{J}_{E_t}$ is correct or not because the solution for each problem in E_t is known (since they are problems extracted from cases of C_t^u). Next, JUST builds a refutation set for each incorrect JER in $\mathbf{J}_{E_t}^-$, obtaining the collection of refutation sets $\mathbf{R}_t$. Finally, the belying set B_t is built; B_t is the set of cases that JUST will add to C_t^r in the iteration t: therefore, $C_{t+1}^r = C_t^r \cup B_t$. If the termination criterion $\mathcal{T}$ is still not met, a new iteration starts.

There is a special situation for JUST when the belying set B_t is empty and the termination criterion is not met: then JUST selects a single random case from C_t^u and adds it to C_t^r. This is done to ensure convergence, avoiding an unbounded number of iterations. This way, the maximum number of iterations of JUST is exactly n (the number of cases in C), since at each iteration, at least one case is added to C_t^r.

When the termination criterion is met (at an iteration t) the case base C_t^r is considered the target case base C^r, and C_t^r is returned. If the set C_t^u is empty, this means that all the cases from C have been selected, and that C^r contains all the cases from C, and the process is also terminated.

3.1 Termination Criterion

In our experiments we have used two different termination criteria $\mathcal{T}$. If the termination criterion is to obtain a case base C^r of a given size M, JUST will finish once C_t^r has reached the size M. In fact, JUST will output C_{t-1}^r when it detects at iteration t that $size(C_t^r) > M$.

When the termination criterion is to obtain a case base C^r with a minimum accuracy level α, JUST uses the answers of the exams as an estimation of the current classification accuracy. However, depending on the size of the exam, this estimation may not be very reliable. If the size of the exam is large, the accuracy obtained on that exam is a good estimation of the classification accuracy of the CBR system; thus, when the accuracy obtained by the system in a large exam is above α, the JUST process can terminate. In our experiments, α takes values around 90%. Moreover, if the size of the exam is small, JUST needs more than one exam to have a good estimate of the accuracy.

The number of exams needed to have a good estimation can be determined assuming that the correctness of an answer can be modeled as a binomial distribution. Using the binomial distribution, for estimating accuracy values around $\alpha = 90\%$, 60 answers are enough to have a 66% certainty of having an error smaller than the 4% in the estimation of the accuracy. For instance, for an exam size $m = 20$, 3 exams are enough for being 66% sure that the accuracy of the CBR system does not differ more than a 4% from the estimated one. Thus, if the average accuracy α' of 3 consecutive exams of size $m = 20$ is higher than α, JUST can terminate with a 66% certainty that the accuracy of the CBR system is in a $\pm 4\%$ margin around α'. For an exam size of $m = 10$, 6 consecutive exams are needed for the same result. Summarizing, the termination policy is the following: if the average accuracy in the last $60/m$ exams is above α, JUST will stop.

3.2 JUST in a nutshell

The idea behind JUST is quite simple: at each iteration, the system tests which are the weak points (or competence holes) of the new reduced case base C^r by solving an exam. The justification J given for an incorrectly answered problem P in an exam is an incorrect local approximation of the neighborhood of P.

Moreover, since a justification J is also a symbolic description, it can be used to find a case (a counter example) that satisfies that description and that proves that J is incorrect (i.e. the case has a solution different from the predicted by J). Adding that case into the reduced case base prevents the incorrect local approximation to be generated again, thus improving problem solving in that area of the problem space.

In fact, JUST selects the minimum set of cases that are counterexamples of all the incorrect justifications (the belying set) in order to minimize the number of cases of the reduced case base. Therefore, JUST iteratively constructs a case base that is more competent at each iteration, and this increase of competence is done trying to minimize the number of cases needed. Of course, all the process strongly depends on the ability of JUST to detect the weak points in the case base C^r. Therefore, we expect that the larger the exam size, the better JUST will work. In the extreme, all the remaining cases in C_t^u can be used as the exam in iteration t to obtain the smallest case base that JUST can obtain.

4 Experimental results

This section presents the experimental results comparing the performance achieved by a CBR system after using the JUST case base reduction technique with the performance of the system without reducing the case base.

The lazy learning method we use is LID [3], a CBR method that is able to generate justifications and that can work both with propositional and relational data. We have used three different datasets to test out approach: *soybean*, a well known propositional dataset, *zoo*, another propositional data set from the UCI machine learning repository, and *marine sponges*, a complex relational data set [3]. Sponges have a complex structure, making them amenable to build complex justifications. The soybean data set consists of 307 examples, each one with 35 attributes (some of them with missing values), and there are 19 possible solution classes. the zoo data set consists of 101 examples, each one with 17 attributes and 7 solution classes. The sponges data set consists of 280 examples, each one with between 10 and 50 attributes (depending on its structure), and there are 3 solution classes.

The presented results are an average of 5 10-fold cross validation runs. Each 10-fold cross validation run involves 10 experimental runs. In an experimental run, a 10% of the cases are separated from the rest and will be used as the test set. The other 90% of the cases is used as the system case base, then the accuracy is measured after applying the case base reduction technique.

We have made experiments comparing the JUST technique with two base strategies: a base CBR system that does not use any case base reduction technique, and a CBR system that uses the CB2 [1] case base reduction technique. The idea of CB2 is simple: as with JUST we have two case bases C and C^r, individual cases are randomly selected from C, if the system can solve them using the cases in C^r, then they are discarded, otherwise they are added to C^r. The process is iterated until all the cases in C have been selected. Therefore, we

	Sponges		*Soybean*		*Zoo*	
	Accuracy	CB size	Accuracy	CB size	Accuracy	CB size
JUST (m=20)	88.12%	32.34%	88.59%	55.00%	95.44%	38.86%
CB2	82.14%	22.71%	81.00%	28.62%	95.24%	18.59%
Complete CB	88.21%	100.00%	88.50%	100.00%	95.45%	100.00%

Table 1. Comparison of the classification accuracy and case base size of JUST agains CB2 and with the complete case base.

will use no termination criteria for CB2 (as the ones used in JUST) but let it execute till the end.

Table 1 shows the results obtained by three CBR systems, one using the JUST case base reduction technique, another using the CB2 case base reduction technique and the third one using the complete case base for the three datasets (sponges, soybean and zoo). We have used JUST with an exam size of $m = 20$, and a termination criterion of reaching an accuracy of about 90% for the sponges and soybean data sets, and of about 96% in the zoo data set (we have chosen those parameters as slightly greater values than the accuracy values of the complete case bases).

The table shows that JUST has been able to reduce the size of the case bases to the 32.34% of the total number of cases in the sponges case base, to the 55.00% in the soybean case base and to the 38.86% in the zoo case base. This reduction is achieved without losing classification accuracy: notice that the accuracy for JUST in the sponges data set is 88.12% while the accuracy without case reduction is 88.21%, the difference being not statistically significant. For the soybean data set, JUST has achieved a classification accuracy of 88.59% while the CBR system without case reduction achieves a 88.50% of classification accuracy; again the difference is not statistically significant. In the zoo data set, the accuracy achieved by JUST is 95.44%, and the accuracy achieved with the complete case base is 95.45%. Moreover, the termination criterion of JUST requested case bases with a 90% of classification accuracy in soybean and sponges and 96% in the zoo data set. Notice that JUST has stopped before reaching that accuracy in all the case bases. The reason is that the termination criterion used in our experiments has a margin of error of $\pm 4\%$ (see Section 3.1). A termination criterion with a lower margin of error could be used if need be.

Comparing JUST with CB2 in Table 1, notice that CB2 obtains reduced case bases that are even smaller than the achieved by JUST: 22.71% in the sponges data set, 28.62% in the soybean data set and 18.59% in the zoo data set versus 32.34%, 55.00% and 38.86% achieved with JUST. However, CB2 reduces the case base without preserving the classification accuracy in two of the three data sets; CB2 has been able to keep the degree of accuracy of the complete case base only in the zoo data set, in the other two data sets, the accuracy achieved by CB2 is appreciably lower than that of the complete case base: 82.14% in the sponges data set and 81.00% in the soybean data set. JUST, however, maintains the accuracy of the complete case base, namely 88.12% and 88.59% respectively.

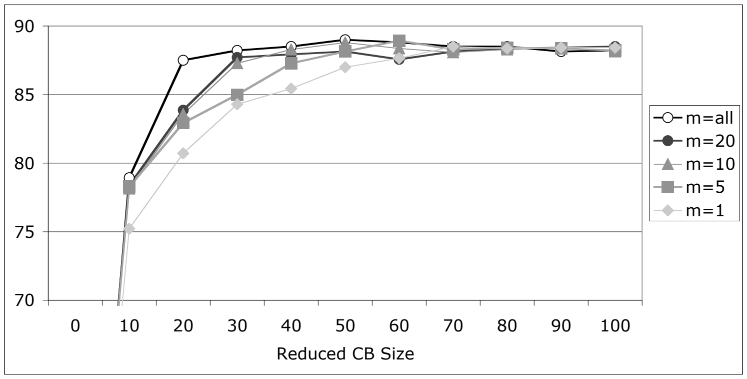

Fig. 3. Comparison of the accuracy evolution in the reduced case bases for several exam sizes in the sponges dataset using JUST.

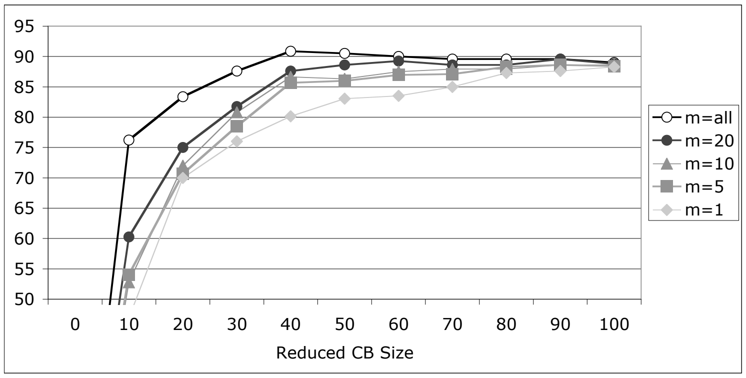

Fig. 4. Comparison of the accuracy evolution in the reduced case bases for several exam sizes in the soybean dataset using JUST.

CB2 has problems in two data sets because cases are discarded in a very eager way. JUST, however, has a broader view of the problem and never discards any case until termination is decided. Thus, JUST is able to discard a considerable number of cases while maintaining the accuracy levels of the complete case base.

In order to test the effects of the size of the exams in JUST we have experimented with several exam sizes: 1, 5, 10, 20 and unlimited (when exam size is unlimited, the whole set of cases C_t^u is used as the exam). Figures 3 and 4 show the accuracy results for JUST in the sponges and soybean data sets for several exam sizes. The unlimited exam size is shown in the figures as $m = all$. This experiments are performed using the case base size termination criterion (see Section 3.1) for sizes 10%, 20%, ..., and up to 100% percentage of the complete case base. Figures 3 and 4 plot the accuracy achieved by JUST varying the desired size of the reduced case base. For each exam size, a different plot is shown.

We have made experiments with the three data sets but, for lack of space, we only present here results concerning marine sponges and soybean data set.

Figure 3 shows that as the exam size increases, JUST is able to reach higher accuracies with smaller case bases. For instance, reaching an accuracy higher than 85% with an exam size $m = 1$, JUST needs a case base of the 40% of the size of the complete case base, while with the exam size is $m = 5$, only a 30% of the original cases are needed. In the extreme, when the exam size is unlimited (i.e. all the cases in C_t^u are used as the exam at each iteration), only a 20% of the cases are needed. This is because when the exam size is larger, JUST can obtain more information of the weak points in the reduced case base C_t^r, and therefore make a more accurate choice of which cases to select to add to the reduced case base C_t^r. Moreover, notice that when the termination criterion is to obtain a case base with more than the 70% of the cases in the complete case base, there is no difference in the classification accuracy by varying the exam size. Notice also that in some experiments JUST has been able to obtain case bases that reach higher accuracy than the complete case base. For instance, when the exam size is unlimited, the accuracy achieved with a case base with the 50% of the cases of the complete sponges case base is 89.00% while the accuracy of the complete case base is 88.21%.

Figure 4 shows the experiments using the soybean data set. Notice that as the exam size increases, as before, the accuracy achieved by JUST also increases. However, the accuracy achieved by JUST in the soybean data set with an unlimited exam size is much higher than the accuracy with smaller exam sizes. For instance, with a case base containing the 40% of the cases in the complete case base, JUST with an unlimited exam size achieves an accuracy of 90.88% while the complete case base accuracy is 88.50%. This means that the exam size needed by JUST in the soybean data set to achieve a good performance is larger than the exam size needed in the sponges data set. The reason seems to be that the soybean data set has 19 solution classes and the sponges data set only 3. The larger the number of classes, the larger the exams should be in order to obtain representative information of the weak points of the reduced case base.

The overall conclusion is that the larger the exam size, the higher the performance of JUST, i.e. as we increase the exam size, we will obtain reduced case bases that are smaller and more accurate. However, as we increase the exam size, we also increase the computational cost of JUST. Let us analyze JUST in computational cost as the number of retrievals performed during the case base reduction process. The cost of JUST can be divided in two costs: the cost of solving the exams, and the cost of building the belying sets. Let T be the number of iterations that JUST has executed, n the number of cases in the complete case base C, and m the exam size. The cost of solving the exams is at most $T \times min(m, n)$ retrievals, and the cost of building the belying set is also at most $T \times min(m, n)$. Therefore, the complexity of JUST is of order $T \times min(m, n)$. As explained in Section 3, the maximum number of iterations is n, the number of cases in the complete case base C. Therefore, the worst case complexity is $n \times min(m, n)$, i.e. $O(n^2)$.

We have also performed an empirical evaluation of the JUST complexity varying the exam size in the soybean data set, as the following table shows:

m	1	5	10	20	all
retrievals	256.8	713.0	458.0	1158.0	1627.7
iterations	256.8	142.6	45.8	57.9	8.2

The termination criterion used to perform those experiments is to reach an accuracy of the 90%. We see that the number of retrievals increases as the exam size increases (as predicted by the theoretical complexity of $n \times min(m, n)$). However, the practical complexity is much lower than the theoretical complexity, specially for large exam sizes, where the number of iterations is much smaller than the theoretical maximum n. This results point out that JUST can be used with large exam sizes without having to pay a high computational cost. Notice that the practical cost for an unlimited exam size, is 1627.7 retrievals in average, while the theoretical bound is $n^2 = 276 \times 276 = 76{,}176$ retrievals, since in the soybean data set the complete case base C has 276 cases (the other 10% is reserved as the test set). We can conclude that if the computational cost is not a problem in our CBR system an unlimited exam size should be used in order to obtain the maximum benefit from JUST. Moreover, although the cost of JUST with large exam sizes is not prohibitive (as we have seen in our experiments), smaller exam sizes may be used in order to reduce the computational cost if need be.

5 Conclusions and Future Work

In this paper we have presented JUST, a case base reduction technique based in the notion of justifications. In our experiments we have used LID as a CBR method that is able to generate justifications, but our work is not restricted to LID. Many other CBR methods can be adapted to generate justifications, for instance CBR systems using case indexing based on decision trees can return the portion of the decision tree used to solve a problem as the justification. We have seen that a justification J can be considered a local approximation of the neighborhood of the problem to solve, i.e. as a generalization of all the retrieved cases to solve a problem. Therefore, as future work, we plan to apply JUST to other CBR methods, such as nearest neighbor, that cannot provide justifications. How to adapt a nearest neighbor classifier is not obvious. However, we can see the set of retrieved cases as a local approximation to solve a problem P. From that local approximation an ad-hoc justification can be built by computing some generalization(s) of all or some of the retrieved cases. These generalizations could then be used by JUST to compute the belying sets. An interesting question here is whether this approach will work for any type of lazy classification method.

We have also seen that JUST is a parametric case base reduction method. By varying the exam size m, we can modify the behavior of JUST: with small exam sizes we can obtain moderate case base reductions at a low cost, and with large exam sizes we can obtain large case base reductions, but at a higher computational cost. This is clearly an advantage with respect to other case base reduction methods that are not parametric, since JUST can be adapted to several CBR systems that have different size and computational time restrictions. Moreover, JUST can accept another parameter: the termination criterion. By changing the termination criterion, we can request JUST to obtain reduced case bases that satisfy any desired conditions.

Finally, we have seen in the experiments section that there are reduced case bases that achieve higher accuracies than the complete case base. For instance, in Figure 4, the optimal point (with an unlimited exam size) is to build a reduced case base with the 40% of the original cases (since this is where the maximum accuracy was reached). Instead than by looking at the plot as we have done now, it remains as future work to automatically find the optimal accuracy point.

Acknowledgements The authors thank Eva Armengol and Josep-Lluís Arcos of the IIIA-CSIC for the development of the LID and of the Noos agent platform respectively. Support for this work came from CIRIT FI/FAP 2001 grant and project SAMAP (MCYT-FEDER) TIC2002-04146-C05-01.

References

[1] David W. Aha. Case-based learning algorithms. In *DARPA Case-Based Reasoning Workshop*, pages 147–158, 1991.

[2] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[3] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In Luc de Raedt and Peter Flach, editors, *EMCL 2001*, number 2167 in Lecture Notes in Artificial Intelligence, pages 13–24. Springer-Verlag, 2001.

[4] P. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1967.

[5] David B. Leake and David C. Wilson. Remembering why to remember: Performance-guided case-base maintenance. In *EWCBR*, pages 161–172, 2000.

[6] Elisabet Golobardes Maria Salamó. Hybrid deletion policies for case base maintenance. In *FLAIRS'2003*, pages 1150–155, 2003.

[7] B. Smyth. The utility problem analysed: A case-based reasoning persepctive. In *Third European Workshop on Case-Based Reasoning EWCBR-96*, Lecture Notes in Artificial Intelligence, pages 234–248. Springer Verlag, 1996.

[8] Barry Smyth and Mark T. Keane. Remenbering to forget: A competence-preserving case delection policy for case-based reasoning systems. In *Proceedings of IJCAI-95*, pages 377–382, 1995.

[9] Barry Smyth and Elizabeth McKenna. Building compact competent case-bases. *Lecture Notes in Computer Science*, 1650:329–342, 1999.

[10] Jun Zhu and Qiang Yang. Remembering to add: Competence-preserving case-addition policies for case base maintenance. In *IJCAI*, pages 234–241, 1999.