Choosing a Case Base Maintenance Algorithm using a Meta-Case Base

Lisa Cummins and Derek Bridge

Abstract In Case-Based Reasoning (CBR), case base maintenance algorithms remove noisy or redundant cases from case bases. The best maintenance algorithm to use on a particular case base at a particular stage in a CBR system's lifetime will vary. In this paper, we propose a meta-case-based classifier for selecting the best maintenance algorithm. The classifier takes in a description of a case base that is to undergo maintenance, and uses meta-cases—descriptions of case bases that have undergone maintenance—to predict the best maintenance algorithm. For describing case bases, we use measures of dataset complexity. We present the results of experiments that show the classifier can come close to selecting the best possible maintenance algorithms.

1 Introduction

In Case-Based Reasoning (CBR), *case base editing* is a form of *case base maintenance* in which algorithms use heuristics to select cases to delete from case bases. In the context of using CBR for classification, *noise reduction algorithms* seek to delete noisy cases, with the goal of increasing classification accuracy; *redundancy reduction algorithms* seek to delete redundant cases, with the goal of increasing retrieval efficiency while, as much as possible, not harming classification accuracy.

As is apparent from the previous paragraph, case base maintenance is a multi-objective optimization problem. An algorithm cannot, in general, maximize both the number of cases it deletes and the accuracy of the resulting case base. To see this in extreme form, imagine an algorithm that proposes deletion of all but one

Lisa Cummins
Department of Computer Science, University College Cork, Ireland, e-mail: lec1@cs.ucc.ie

Derek Bridge
Department of Computer Science, University College Cork, Ireland, e-mail: d.bridge@cs.ucc.ie

case: deletion has been maximized but a case-based classifier will now incorrectly predict the class of all target problems whose true class is different from the class of the case that remains in the case base.

In our previous work, we proposed two strategies for investigating the trade-off between these two objectives [5]. One was to compute the *Pareto front*, i.e. those algorithms not dominated by any other algorithms. The other was to combine the percentage of cases deleted and the percentage accuracy of the case base after maintenance into a single number. We proposed to use their *harmonic mean*. The harmonic mean penalizes large differences between the two values so that a high mean is only produced if both individual values are high. In this way, we identify algorithms which have a high value for both accuracy and deletion. The disadvantage of the more conventional *arithmetic mean* is that it can produce similar values for, e.g., an algorithm with very high accuracy but very low deletion and an algorithm with medium deletion and accuracy. We found empirically that algorithms with high harmonic mean tended to correspond to the subset of those on the Pareto front which struck a good balance between deletion and accuracy.

Most case base maintenance algorithms are composites which combine a noise reduction phase and a subsequent redundancy reduction phase. The three most common are:

Brighton & Mellish's Iterative Case Filtering algorithm (ICF) [3]: ICF uses the Repeated Edited Nearest Neighbour algorithm (RENN) [26] to remove noisy cases. RENN regards a case as noisy if it has a different class from the majority of its k nearest neighbours. After running RENN, ICF then removes redundant cases. It regards 'interior' cases, i.e. ones within clusters of same-class cases, as redundant ones, and aims to retain cases on the boundaries between classes because these cases are important for classification accuracy. Specifically, the ICF redundancy reduction phase removes cases that are solved by more cases than they themselves solve.

McKenna & Smyth's algorithm (RC) [18]: RC, like ICF, uses RENN to remove noisy cases. But its redundancy reduction phase is quite different. It aims to retain a case if it is surrounded by many cases of the same class, while treating as redundant, and deleting, those that surround it.

Delany & Cunningham's Case Base Editing algorithm (CBE) [6]: CBE's first phase uses Blame-Based Noise Reduction (BBNR), which regards a case as noisy if it causes other cases to be mis-classified. CBE's second phase uses Conservative Redundancy Reduction (CRR), which, like ICF, regards 'interior' cases as redundant. But CRR has a different heuristic from ICF: CRR removes cases that solve other cases.

There are many other possible algorithms—indeed, in our own earlier work we defined a constructive framework called MACE (Maintenance by a Committee of Experts), using a grammar of maintenance algorithms, which defines an infinite set of case base maintenance algorithms [5].

Their different heuristics give case base maintenance algorithms different biases. It is well-known that the algorithms perform differently on different case bases (see,

e.g., [3, 5]): one size does not fit all in case base maintenance. Additionally, it may be that even the same case base may need to undergo maintenance by different algorithms at different points in its lifetime. For example, a case base developed from a legacy database may require noise removal in its early stages to make it cleaner, but redundancy removal as it matures and collects more experience. The research problem that we tackle in this paper is: how to choose a good maintenance algorithm for a given case base at a given time.

Imagine that we collect experience across the CBR community to say which algorithm or algorithms worked well for a case base at a particular stage in its lifetime. With this experience, we could train a decision-support system and then use this system to assist the human user in selecting maintenance algorithms in the future. In this paper we set out to build such a system and use it to help us to solve the problem of choosing which maintenance algorithm to use for a given case base.

In developing such a decision-support system we have a number of decisions to make. One decision is whether to formulate the problem as one of regression or of classification. We could run different maintenance algorithms on a sample of case bases and measure values such as the percentage of deletion and the difference between the original case base classification accuracy and the accuracy of the case base after maintenance. From this training data, we could then build a system that uses *regression* to predict deletion and accuracy values for a given maintenance algorithm and case base. Alternatively, if we could make a judgement about which algorithm works best for a given case base, we could run different algorithms on sample case bases and pick the best algorithm for each. From this training data, we could then build a system that uses *classification* to predict which algorithm is best to use for a given case base.

Another decision is what type of regression or classification system to use to make the predictions. We could use, for example, a neural net, a rule-based system, a decision tree or a CBR system.

We must decide also how we will describe case bases—both those in the training set from which the decision-support system is built, and those that are to undergo maintenance by an algorithm recommended by the decision-support system.

There is an obvious alternative to training a decision-support system using previous case base maintenance experience. Whenever it is deemed that a case base requires maintenance, we could simply run all maintenance algorithms on copies of the case base and choose whichever performs best, e.g. whichever has the highest harmonic mean of the percentage of cases deleted and post-maintenance accuracy. But this is only feasible if the case base is small, if the case base maintenance algorithms do not take long to run and, most especially, if there is only a small, finite number of case base maintenance algorithms. In general, however, there is a very large number of algorithms. As we said earlier, our MACE framework defined an infinite number. But even if we ignore the most complex algorithms that MACE defines, it is important to realise that we are still left with a large number—more than just the three that we use in the experiments that illustrate the feasibility of our approach in this paper.

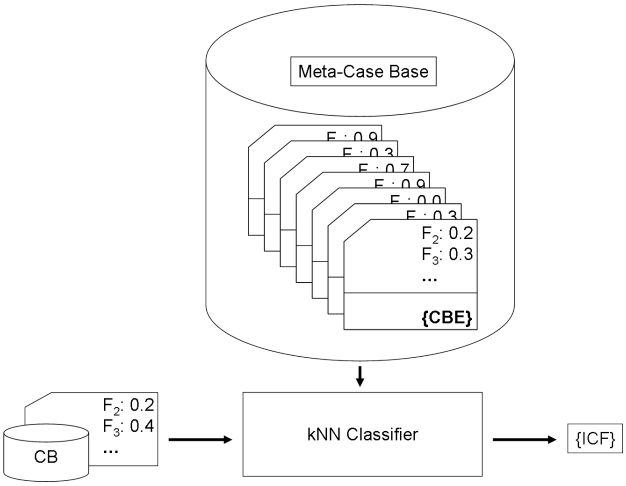

Fig. 1 Maintenance algorithm selection as meta-case-based classification

Our goals in this work are to find a way to describe case bases and then to establish the feasibility of training a decision-support system for choosing a maintenance algorithm for a given case base. Comparing different decision-support systems (regression versus classification, neural nets versus rules and so on), and using more than just three candidate maintenance algorithms, is left to future work. We focus on the feasibility of a decision-support system that is a classifier and we use case-based classification. We call this *meta-case-based classification*, since we are performing case-based classification but doing so for case base maintenance.

In Section 2, we describe our meta-case-based classifier for case base maintenance. Sections 3 and 4 describe two sets of experiments. Then Sections 5 and 6 describe related work and ideas for future work.

2 Maintenance Algorithm Selection as Meta-Case-Based Classification

To perform meta-case-based classification, we need to assemble a case base to use for predicting maintenance algorithms. This is a case base whose cases describe other case bases (Figure 1). We call such a case base a *meta-case base*, and we refer to the cases it contains as *meta-cases*. Each meta-case has a *meta-problem description* and a *meta-solution*. The meta-problem description is made up of the values for *meta-attributes*. A meta-case's meta-solution contains the name or names of the best maintenance algorithms (of those tested) for the case base that this meta-case describes.

2.1 Meta-Problem Description

Each meta-case describe a case base. Its meta-problem description comprises values for a set of meta-attributes. The meta-attributes must characterize the case base that the meta-case describes in a way that is likely to be predictive of the best case base maintenance algorithm. There is a body of work that describes dataset complexity measures, each of which characterizes a dataset in a different way. In this paper, we explore the feasibility of using these complexity measures as the meta-attributes in our meta-case based classifier.

Ho & Basu survey many of the geometrical dataset complexity measures [12, 13]. They divide the measures which they survey into three categories depending on what they focus on:

Measures of overlap of attribute values: These measures look at how effective the attributes are at discriminating between the classes.

Measures of separability of classes: These measures look at how separable the classes are by estimating the length and linearity of the class boundaries.

Measures of geometry, topology and density of manifolds: These measures characterize the position, shape, overlap and density of the manifolds that contain the instances of each class.

Subsequently, Orriols-Puig et al. have made available DCoL, the Data Complexity Library, which is an open-source C++ implementation of thirteen measures based on those in the Ho & Basu survey [19].[1] We have augmented these with four more, drawing from the CBR literature: we define two from Massie et al.'s complexity profile measure [17]; then there is Fornells et al.'s separability emphasis measure [8]; and we define another from Smyth & McKenna's notion of case base competence [24]. Table 1 summarizes these seventeen measures. Our four new measures are C_1, C_2, N_5' and T_3. Their definitions can be found in [4]. Additionally, we found it necessary to make changes to the definitions of five of the measures to make them more fit for purpose. These are the ones with primes in their names (e.g. F_2') in Table 1. The revised definitions can be found in [4].

Table 1 also indicates which measures can be computed for multi-class classification datasets (where there are more than two classes) and which can be computed on datasets whose attributes are symbolic-valued and not just numeric- or Boolean-valued. In this paper, we use just the twelve measures that can handle multi-class classification datasets and symbolic-valued attributes, since these are common in CBR. However, it is possible that not all of the complexity measures may be predictive of the best maintenance algorithm. Equally, two or more measures may compute very similar properties of the case base, rendering one or more measures redundant; to use several similar measures would give extra weight to that property of the case base. We use *feature selection* as a way to choose a good subset of the measures.

There has been much research done in the area of feature selection [1, 11, 16]. There are three types of feature selection algorithms: wrapper, filter and embedded

[1] `http://dcol.sourceforge.net/`

Table 1 The dataset complexity measures that we use as candidate meta-attributes

Measure	Description	Multi-class	Symbolic
F_1	Maximum Fisher's Discriminant Ratio	*Not DCoL*	No
F_2'	Volume of Overlap Region	Yes	Yes
F_3'	Maximum Attribute Efficiency	Yes	Yes
F_4'	Collective Attribute Efficiency	Yes	Yes
N_1'	Fraction of Instances on a Boundary	Yes	Yes
N_2	Ratio of Average Intra/Inter Class Distance	Yes	Yes
N_3	Error Rate of a 1NN Classifier	Yes	Yes
L_1	Minimized Sum of Error Distance of a Linear Classifier	*Not DCoL*	*Not DCoL*
L_2	Training Error of a Linear Classifier	*Not DCoL*	*Not DCoL*
C_1	Complexity Profile	Yes	Yes
C_2	Similarity-Weighted Complexity Profile	Yes	Yes
N_5'	Separability Emphasis Measure	Yes	Yes
L_3	Nonlinearity of a Linear Classifier	*Not DCoL*	No
N_4	Nonlinearity of a 1NN Classifier	Yes	No
T_1'	Fraction of Maximum Covering Spheres	Yes	Yes
T_2	Number of Instances per Attribute	Yes	Yes
T_3	Dataset Competence	Yes	Yes

[11]. Wrappers use a model to evaluate the predictive power of subsets of the attributes and choose the best attribute subset according to this model. Filters instead rely on general characteristics of the data to choose and evaluate a subset. Examples of these characteristics include correlation between attributes and mutual information. Embedded attribute selection methods are similar to wrappers in that they use a model to evaluate a set of attributes. However, with embedded methods, the feature selection is built into the learning process itself, while wrappers perform feature selection as a preprocessing step before learning takes place.

There are also three types of search strategies used by feature selection algorithms: complete, sequential and random. Complete search guarantees to find the optimal attribute subset. Sequential search gives up completeness and therefore may miss optimal subsets, but is faster than a complete search. Random search starts with a random subset and either continues with sequential search by adding or removing attributes, or generates another random subset and retains the better subset.

Since the predictive accuracy of a k-NN classifier is a suitable criterion to use to evaluate attribute subsets, we use a wrapper algorithm. We use a best-first search algorithm (hence, sequential search), which picks the most promising complexity measure at each point. It begins with an empty set and tests the accuracy of the classifier using each measure as a single meta-attribute separately. The accuracy of the classifier for each meta-attribute is measured by leave-one-out cross-validation. On the basis of the accuracies obtained by using each as a meta-attribute, the feature selection algorithm picks the best measure, and adds it to the empty set of selected meta-attributes. It repeats in order to choose further measures that become meta-attributes, stopping when there is no measure that can be added to the set of meta-attributes to improve accuracy.

2.2 Meta-Solution

The best maintenance algorithm or algorithms for a case base are taken to be the meta-solution in the corresponding meta-case. In this section we look at how to choose which maintenance algorithms are considered to be the best for a case base.

We have already acknowledged that there is an infinite number of maintenance algorithms which can be used to maintain a case base. However, since it is not possible to run an infinite number of maintenance algorithms on a case base to see which one works best, we need to choose a subset of candidate algorithms on which we base our predictions.

The set of candidate algorithms depends on what the person maintaining the case base hopes to achieve. If the focus is on deleting noise, the candidates might only contain noise reduction algorithms such as RENN and BBNR; if the focus is on aggressive deletion, the candidates may be composites like RC; for more careful and conservative deletion, candidates may be composites like CBE.

Once a set of candidate algorithms is chosen, each of the candidate algorithms is run on a sample of existing case bases. For each case base, we then need to decide which of the algorithms are 'winners' so that they can become the meta-solution. We have already discussed the problem of deciding on the best algorithm: an algorithm that does well in maintaining or improving accuracy may not perform much deletion, and an algorithm that deletes a large percentage of the case base may not maintain good accuracy. We have argued that the harmonic mean of the two provides a good balance [5]. Therefore a good choice for the meta-solution would be a set that contains the names of the maintenance algorithms whose resulting case bases give the highest harmonic mean value. Alternatively, a weighted harmonic mean can be used to bias the choice towards accuracy or deletion if either one was more of a priority for the person maintaining the case base.

2.3 Meta-Case-Based Classification

Assume that we have taken a sample of existing case bases. We have turned each one into a meta-case. Specifically, for each case base in the sample, we have computed the values of the dataset complexity measures, which become the values of candidate meta-attributes; we have run a set of candidate algorithms and the meta-solution is the set that contains the case base maintenance algorithm or algorithms with the highest harmonic mean. Then, on the meta-case base, we have run a feature selection algorithm to decide which of the candidate meta-attributes should be used by this classifier. Now we are ready to use this meta-case base for decision-support.

When we are presented with a case base that requires maintenance, we turn this case base into a query for our meta-case-based classifier. First, we compute the values of the dataset complexity measures on the case base. Obviously, we only need to compute the values of those measures that were selected by feature selection. This gives us the values of the meta-attributes of the query. We then use our meta-case

base to classify the query using k-NN: k similar cases are retrieved from the meta-case base and the predicted algorithm is the outcome of a similarity-weighted vote among the algorithms in the meta-solutions of these k cases. The result of the vote will be the maintenance algorithm or algorithms which the system considers to be the best one to use for that case base.

Since all the meta-attributes are numeric-valued, we measure similarity as the complement with respect to 1.0 of the Manhattan distance computed over the meta-attributes, range-normalising their values to make them comparable.

3 Holdout Experiments

We took 25 classification datasets: 19 from the UCI repository [10]; plus the Breathalyser dataset [7]; and five email datasets [6]. We divided each dataset randomly into three splits: a 60% training set, a 20% cross-validation set, and a 20% test set. We created 10 different splits of the data and we report all results as averages over the 10 splits.

Each of the 25 training sets becomes a meta-case in a meta-case base. On each training set, we compute the 12 complexity measures, and these are the values of the candidate meta-attributes. We run each of the candidate maintenance algorithms—which, in these experiments, are ICF, RC and CBE—on the training set. We record the percentage of cases deleted. We use the cross-validation set to compute the accuracy of the training set after maintenance by each of the algorithms. In other words, we take each case from the cross-validation set and treat it as a query. We present it to a k-NN classifier that uses the training set as its case base. We compute the percentage of the cross-validation set that the k-NN classifier correctly predicts. For each of the maintenance algorithms, we can now compute the harmonic mean of the percentage of cases deleted and the post-maintenance classifier accuracy on the cross-validation set. The algorithm or algorithms with highest harmonic mean are the best algorithms to use on this training set, and therefore a set containing these algorithms is the meta-solution for this meta-case. We do this for each of the 25 training sets to give us 25 meta-cases in our meta-case base.

At this stage, each meta-case has 12 candidate meta-attributes, one for each complexity measure. We run the feature selection algorithm on the meta-case base to decide which of the candidate meta-attributes to use in the rest of the experiment.

We apply a similar process to each of the 25 test sets. The goal here is to create queries that we can present to the meta-case-based classifier. It is important to keep in mind that the queries to this classifier are not individual cases; they are case bases that are to undergo maintenance (see Figure 1). For each test set, we compute the values of the complexity measures. The only measures we need are those that were selected by the feature selection algorithm previously. These become the values of the meta-attributes of the query.

We need to know the best maintenance algorithm to use on each test set—to act as the true class. So we run each of the candidate maintenance algorithms on the

Table 2 Repeated holdout experiment

Algorithm	Error (%)	Harmonic Mean	Accuracy (%)	Deletion (%)
Best Possible	0	76.02	71.68	85.09
Meta-CBR	43.2	69.79	69.53	78.21
RC	24.4	74.49	68.87	87.50
ICF	79.2	62.29	69.63	64.54
CBE	88.4	60.41	72.64	57.11

test set. We note the best algorithm or algorithms, i.e. those that have the highest harmonic mean of the percentage of cases deleted and post-maintenance accuracy on the cross-validation set.

At this point, we have a meta-case base of 25 meta-cases, and a meta-test set of 25 case bases that are to undergo maintenance. We take each case base in the meta-test set in turn. We present it to the meta-case-based classifier (henceforth referred to as the meta-CBR system), which uses 3-NN to predict an algorithm in the way described in Section 2.3.

In the experiment, we measure two things. We record the percentage of the meta-test set that the meta-CBR system incorrectly predicts—the error rate. A correct prediction is one in which the algorithm that the meta-CBR system predicts is the best algorithm—or, in general, it is one in which one of the algorithms that the meta-CBR system predicts is one of the best algorithms.

However, prediction error is not the whole story. Even if the meta-CBR system fails to choose the best algorithm (or one of the joint best algorithms), it may choose one that, when run, has a performance not too far below that of the best algorithm. So for each of the 25 case bases in the meta-test set, we run the meta-CBR system's predicted algorithm and record the percentage of cases deleted, the post-maintenance accuracy on the cross-validation set and their harmonic mean.

The results are shown in Table 2. The Table compares the meta-CBR system to four other options. One option is to always run the ICF algorithm, irrespective of the characteristics of the query case base. Similarly, we can instead always run RC or CBE. The other option (Best Possible) is a benchmark figure, which supposes we have an oracle available enabling us to always choose the best algorithm.

We see that RC is the best algorithm—or one of the best algorithms—in 75.6% of meta-cases and so always choosing RC gives an error rate of 24.4%. ICF and CBE are only rarely among the best algorithms and so always choosing them gives quite high error. Unfortunately, in terms of prediction error, in this experiment meta-CBR is performing poorly: it gets 43.2% of predictions wrong. But, as we said, this is not the whole story.

The results in Table 2 show that the highest possible harmonic mean value if we chose the best algorithm every time is 76.02. The algorithm that comes closest to this is RC, with a harmonic mean value of 74.49. Our meta-CBR classification system results in a harmonic mean value of 69.79, which is the next highest value after RC. What the mean hides is that our meta-CBR system results in case bases which, on average, exhibit post-maintenance classification accuracy that is slightly higher than

RC's, 69.53% compared to 68.87%. But, RC deletes over 9% more cases from the case bases on average, and this is why it ends up with the higher harmonic mean.

Always choosing CBE results in the highest classification accuracy (72.64%), higher even than choosing the algorithm with the best possible harmonic mean, but this is unsurprising given how conservative CBE is: it deletes by far the lowest proposition of cases. Always choosing ICF also results in a conservative maintenance strategy (although less conservative than CBE), hence its accuracy is higher than meta-CBR's too. But, because they delete a lower proportion of the case base, ICF and CBE have lower harmonic means than meta-CBR.

Although our meta-CBR system does not have a higher harmonic mean than RC, it has a higher harmonic mean than ICF and CBE. Additionally, it maintains a good level of accuracy at just 2% lower than the best possible on average. This is a promising result for our meta-CBR system.

There are reasons to believe that these results may not be wholly representative. Some of the case bases that we present as queries are very small. They comprise just 20% of their full datasets, which may themselves be small. Some of them may even end up containing cases all of the same class. A first problem is that they may not contain enough cases, or enough different-class cases, for the dataset complexity measures to be informative. For example, if all cases are of the same class, then all complexity measures should produce their lowest values. A second problem is that, if most or all of the cases are of the same class, then this will favour RC's aggressive redundancy reduction, and we are seeing this in the results Table 2. Accordingly, we decided to complement the holdout experiments with leave-one-out experiments, in which the case bases that we present as queries are larger and much less likely to contain cases all of the same class.

4 Leave-One-Out Experiments

In our leave-one-out experiments, we use the same splits of the 25 datasets. We use exactly the same meta-case bases as before—constructed from the 60% training splits and having meta-solutions chosen with the assistance of the 20% cross-validation splits. We do not need a separate meta-test set, so we can discard the 20% test splits. (We could instead have re-split the datasets into just two parts, e.g. 70% training and 30% cross-validation, or 60% and 40%, but then we would have incurred the substantial cost of building new meta-case bases.)

In a fashion akin to classic leave-one-out cross-validation, we remove one meta-case from the meta-case base. We present this meta-case as a query to the meta-CBR system, which uses the remaining 24 meta-cases. We then return it to the case base. We repeat, taking each meta-case in turn.

The advantage is that each query case base is now larger, since it was built from 60% of the original dataset, and it is less likely that it contains only cases of the same class.

The results are shown in Table 3. In these experiments, RC is less often one of

Table 3 Leave-one-out experiment

Algorithm	Error (%)	Harmonic Mean	Accuracy (%)	Deletion (%)
Best Possible	0	82.41	79.39	87.65
Meta-CBR	26.8	80.57	76.81	88.54
RC	35.2	80.30	76.24	88.71
ICF	70.0	74.41	74.69	78.38
CBE	94.4	62.56	77.96	54.88

the best algorithms and so always choosing it is not such a good strategy, resulting in 35.2% error. By contrast, the performance of the meta-CBR system is much improved. Its error, which was 43.2% in the holdout experiment, has fallen to 26.8%.

In these experiments, the highest possible harmonic mean value is 82.41. Our meta-CBR system comes closest to this with a harmonic mean value of 80.57, followed closely by RC with a value of 80.30. We also perform better than ICF and CBE, which have harmonic mean values of 74.41 and 62.56 respectively. Meta-CBR has post-maintenance accuracy that is just 2.6% off that achieved by using an oracle and second only to highly conservative CBE. It deletes a slightly greater proportion of the case base than the oracle achieves, and second only to very aggressive RC. This is a big improvement over our holdout experiments. This shows us that our complexity measures are more reliable and more predictive of the best maintenance algorithm to use when they are computed for case bases which have good coverage of the domain.

5 Related Work

The idea of meta-reasoning—reasoning about reasoning—has been explored in a number of different areas. For example, there is an amount of research into meta-planning [25, 27]. Within CBR, Fox and Leake [9] build a system that reasons about CBR: it uses introspection to improve indexing in a case-based planning system. They use a model to evaluate the performance of a case-based planner in comparison to its expected ideal performance, and they use this model to suggest a repair when the planner does not perform as it should.

Within machine learning and data mining, there has been an amount of research into meta-learning, in particular training meta-classifiers that predict the best classifier to use on a dataset. For example, Lindner & Studer take an approach similar to our own: they use a number of dataset complexity measures and other statistical measures as their meta-attributes [15]. But their system selects classifiers, not maintenance algorithms. Peng et al. also use a k-NN meta-classifier to choose between a number of classification algorithms (rather than maintenance algorithms) [20]. For their meta-attributes, they build decision trees on datasets and compute characteristics of the decision trees, such as the length of the longest branch. Bensusan et al., by contrast, use a representation of the decision tree itself as the description of the

dataset [2]. Pfahringer et al. take an approach that they call landmarking, whereby the meta-attributes come from measuring the performance of simple classifiers on the datasets [21]. Of course, some of our dataset complexity measures also have this character, e.g. N_3, which is the error rate of a 1NN classifier.

Within CBR also, Recio-García et al. use CBR to choose a template from a case base of templates to aid the design and implementation of CBR systems [22] . In particular, they describe a case-based recommender that recommends recommender system templates from a case base of twelve templates. Van Setten [23] compares decision-support systems built using CBR and built using decision trees that help the user to design a hybrid recommender system. Leake et al [14] use CBR to learn adaptation knowledge from a case base. This adaptation knowledge is stored as adaptation cases, which are then in turn used to help the CBR system in the future.

6 Conclusions and Future Work

We have presented a meta-case-based classifier that can form the basis of a decision-support system that assists case base administrators. It can choose a case base maintenance algorithm that is suited to the characteristics of a given case base at a given point in its lifetime. A key design decision was how to characterize case bases—both those that are described by meta-cases in our meta-case base, and incoming query case bases that are to undergo maintenance. We have used a number of measures of dataset complexity, which attempt to characterize the geometry of the case bases. The actual meta-attributes that we use are chosen by a feature selection algorithm from a large set of complexity measures.

The results of our first experiment, which used a holdout method, were promising. The meta-CBR system picked maintenance algorithms that produced case bases that achieved slightly higher classification accuracy than always picking the RC algorithm. However, always choosing RC resulted in a much larger proportion of cases being deleted, and hence a higher harmonic mean. Similarly, compared with selecting the best possible maintenance algorithm using an oracle, the meta-CBR system picked algorithms that produced case bases with comparable classification accuracy, but again was too conservative in the proportion of cases deleted.

Our second set of experiments used a leave-one-out method, enabling us to make better use of the available data. In particular, query case bases were three times larger than in the first experiment, and were much less likely to contain cases most or all of which were of the same class. In this experiment, the meta-CBR system outperformed the approach of always picking the same algorithm. For example, the algorithms chosen by meta-CBR deleted a slightly larger proportion of cases, and resulted in case bases with slightly higher classification accuracy, than always using RC. This makes the point that for meta-CBR to work, the case bases need to be large enough, and representative enough, that the dataset complexity measures are informative. In practice, of course, this is likely to be so, otherwise the case base administrator would not have deemed case base maintenance to be worthwhile.

An obvious line of future research is to perform more experiments. In particular, these could use a wider range of candidate maintenance algorithms. The meta-case base, and the incoming query case bases, could be based on a wider variety of datasets too. In this regard, it would be particularly interesting to create meta-cases from case bases at different points in their lifetimes, and similarly to use query case bases also at different points in their lifetimes. Now that the feasibility of using decision-support technology to choose a maintenance algorithm is established, a comparison with other decision-support technologies, such as neural nets or decision trees, would be of value too.

One concern is that current maintenance algorithms delete whole subsets of the cases in a case base. But as soon as so much as one case is deleted, it no longer follows that the same algorithm is the best to decide on the next case to delete. We are working on incremental versions of the case base maintenance algorithms, which recommend the deletion of only one case at a time. We are also working on incremental versions of some of the dataset complexity measures, whose values can be rapidly re-calculated for a case base following deletion of a case. This will allow us to build an incremental meta-CBR system, which will repeatedly select the best incremental case base maintenance algorithm, which will delete one case, and then return to the meta-CBR system to select an algorithm to delete the next case, and so on until the system predicts there to be no advantage in deleting a further case.

Acknowledgements This paper is based upon work partially supported by the Science Foundation Ireland under Grant Number 05/RFP/CMS0019.

References

1. Aha, D.W., Bankert, R.L.: A comparative evaluation of sequential feature selection algorithms. In: Procs. of the Fifth International Workshop on Artificial Intelligence and Statistics, pp. 1–7 (1994)
2. Bensusan, H., Giraud-Carrier, C., Kennedy, C.: A higher-order approach to meta-learning. In: Procs. of the Workshop on Meta-Learning at the European Conference on Machine Learning, pp. 109–117 (2000)
3. Brighton, H., Mellish, C.: On the consistency of information filters for lazy learning algorithms. In: J. Rauch, J. Zytkow (eds.) Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery, pp. 283–288. Springer-Verlag (1999)
4. Cummins, L.: Combining and choosing case base maintenance algorithms. Ph.D. thesis, Department of Computer Science, University College Cork, Ireland (2011 (forthcoming))
5. Cummins, L., Bridge, D.: Maintenance by a committee of experts: The MACE approach to case-base maintenance. In: L. McGinty, D.C. Wilson (eds.) Procs. of the 8th Intl. Conference on Case-Based Reasoning, pp. 120–134 (2009)
6. Delany, S., Cunningham, P.: An analysis of case-based editing in a spam filtering system. In: P. Funk, P. González-Calero (eds.) Procs. of the Seventh European Conference on Case-Based Reasoning, vol. LNAI 3155, pp. 128–141. Springer (2004)
7. Doyle, D., Cunningham, P., Bridge, D., Rahman, Y.: Explanation oriented retrieval. In: P. Funk, P. González-Calero (eds.) Procs. of the Seventh European Conference on Case-Based Reasoning, *Lecture Notes in Artificial Intelligence*, vol. 3155, pp. 157–168. Springer (2004)

8. Fornells, A., Recio-García, J., Díaz-Agudo, B., Golobardes, E., Fornells, E.: Integration of a methodology for cluster-based retrieval in jcolibri. In: L. McGinty, D.C. Wilson (eds.) Procs. of the Eighth International Conference on Case-Based Reasoning, vol. LNCS 5650, pp. 418–433. Springer (2009)
9. Fox, S., Leake, D.: Using introspective reasoning to refine indexing. In: C. Mellish (ed.) Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pp. 391–397. IJCAI, Morgan Kaufmann, San Francisco, California (1995)
10. Frank, A., Asuncion, A.: UCI machine learning repository (2010). URL `http://archive.ics.uci.edu/ml`
11. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3**, 1157–1182 (2003)
12. Ho, T.K., Basu, M.: Measuring the complexity of classification problems. In: Proceedings of the Fifteenth International Conference on Pattern Recognition, pp. 43–47 (2000)
13. Ho, T.K., Basu, M.: Complexity measures of supervised classification problems. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(3), 289–300 (2002)
14. Leake, D.B., Kinley, A., Wilson, D.C.: Learning to improve case adaption by introspective reasoning and CBR. In: M.M. Veloso, A. Aamodt (eds.) Proceedings of the First International Conference on Case-Based Reasoning, *Lecture Notes in Computer Science*, vol. 1010. ICCBR, Springer, Heidelberg, Germany (1995)
15. Lindner, G., Studer, R.: AST: Support for algorithm selection with a CBR approach. In: Procs. of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 418–423 (1999)
16. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering **17**, 491–502 (2005)
17. Massie, S., Craw, S., Wiratunga, N.: Complexity profiling for informed case-base editing. In: T. Roth-Berghofer, M. Göker, H.A. Güvenir (eds.) Procs. of the Eighth European Conference on Case-Based Reasoning, pp. 325–339. Springer-Verlag (2006)
18. McKenna, E., Smyth, B.: Competence-guided case-base editing techniques. In: E. Blanzieri, L. Portinale (eds.) Proceedings of the Fifth European Workshop on Case-Based Reasoning, vol. LNCS 1898, pp. 186–197. Springer-Verlag (2000)
19. Orriols-Puig, A., Macià, N., Bernadó-Mansilla, E., Ho, T.K.: Documentation for the data complexity library in C++. Tech. Rep. GRSI Report No. 2009001, Universitat Ramon Llull (2009)
20. Peng, Y., Flach, P., Soares, C., Brazdil, P.: Improved data set characterisation for meta-learning. In: Procs. of the 5th International Conference on Discovery Science, pp. 141–152 (2002)
21. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: Procs. of the Seventeenth International Conference on Machine Learning, pp. 743–750 (2000)
22. Recio-García, J.A., Bridge, D., Díaz-Agudo, B., González-Calero, P.A.: CBR for CBR: A case-based template recommender system for building case-based systems. In: K.D. Althoff, R. Bergmann, M. Minor, A. Hanft (eds.) Procs. of the Ninth European Conference on Case-Based Reasoning, vol. LNCS 5239, pp. 459–473. Springer-Verlag (2008)
23. van Setten, M.: Supporting people in finding information: Hybrid recommender systems and goal-based structuring. Ph.D. thesis, University of Twente (2005)
24. Smyth, B., McKenna, E.: Modelling the competence of case-bases. In: B. Smyth, P. Cunningham (eds.) Procs. of the Fourth European Workshop on Advances in Case-Based Reasoning, vol. LNCS 1488, pp. 208–220. Springer (1998)
25. Stefik, M.: Planning and meta-planning (MOLGEN: Part 2). Artificial Intelligence **16**(2), 141–170 (1981)
26. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics **6**(6), 448–452 (1976)
27. Wilensky, R.: Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. Cognitive Science **5**(3), 197–233 (1981)