Maintaining Retrieval Knowledge in a Case-Based Reasoning System

Susan Craw & Jacek Jarmulak
School of Computer & Mathematical Sciences
The Robert Gordon University, Aberdeen

Ray Rowe
AstraZeneca

Abstract

The knowledge stored in a case-base is central to the problem-solving of a Case-Based Reasoning (CBR) system. Therefore, case-base maintenance is a key component of maintaining a CBR system. However, other knowledge sources, such as indexing and similarity knowledge for improved case retrieval, also play an important role in CBR problem-solving. For many CBR applications the refinement of this retrieval knowledge is a necessary component of CBR maintenance. This paper focuses on the optimisation of the parameters and feature selections/weights for the indexing and nearest-neighbour algorithms used by CBR retrieval. Optimisation is applied after case-base maintenance and refines the CBR retrieval to reflect changes that have occurred to cases in the case-base. The optimisation process is generic and automatic, using knowledge contained in the cases. In this paper we demonstrate its effectiveness on a real tablet formulation application in two maintenance scenarios. One scenario, a growing case-base, is provided by two snap-shots of a formulation database. A change in the company's formulation policy results in a second, more fundamental, requirement for CBR maintenance. We show that, after case-base maintenance, the CBR system did indeed benefit from also refining the retrieval knowledge. We believe that existing CBR shells would benefit from including an option to automatically optimise the retrieval process.

Keywords: Case-Based Reasoning, Maintenance, Retrieval Optimisation, Indexing Knowledge, Similarity Knowledge

1 Introduction

Case-Based Reasoning (CBR) is a popular reasoning methodology for decision support systems because its reasoning is based largely on case knowledge that may already be available in a database. When a new problem is presented to a

CBR system, it first retrieves cases with similar problem descriptions from the case-base. The solutions in these retrieved cases are used to propose a solution for the new problem. It may be necessary to adapt the proposed solution to take account of differences between the new problem and the retrieved problems. In addition to returning the proposed solution as the answer to the new problem, it is common to review the new problem and its solution, and perhaps to retain this problem-solution pair as a new case in the case-base. The CBR process is illustrated in Figure 1.

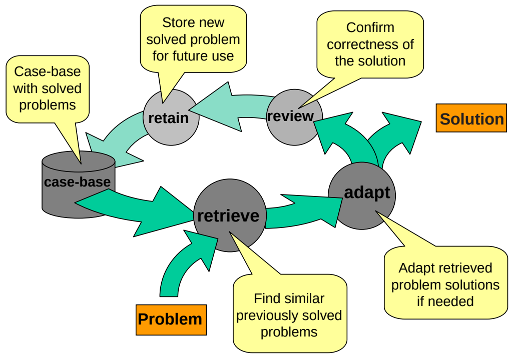

Store new solved problem for future use
Confirm correctness of the solution
Case-base with solved problems
retain
review
Solution
case-base
adapt
retrieve
Adapt retrieved problem solutions if needed
Find similar previously solved problems
Problem

Figure 1: The Case-Based Reasoning Cycle

The case-base is central to CBR, and so one form of CBR maintenance is an effective regime for *case-base* maintenance (Smyth 1998). Case-base maintenance typically involves the addition, removal or revision of cases, but can also include changes to case indexing (Leake & Wilson 1998). However the case knowledge is not the only knowledge that is used in CBR systems and that has to be maintained. CBR systems typically apply additional knowledge distributed over a number of "knowledge containers" (Richter 1998, Wilke, Vollrath, Althoff & Bergmann 1997): the vocabulary knowledge (used to describe the cases and problem domain), the retrieval knowledge (including indexing and similarity knowledge), and the adaptation knowledge, see Figure 2. Given a new problem (Q), indexing knowledge is used to select a subset of relevant cases from the case-base, on which similarity matching focuses. Similarity knowledge influences the calculation of the similarity measure that determines which cases (P) are deemed to be most similar. Adaptation knowledge is applied to alter the proposed solutions (S) in reaction to differences between the new problem and the retrieved cases. If the case-base changes then the retrieval and adaptation knowledge may need to be refined. This is especially the case when the original retrieval and adaptation knowledge was obtained based on the case data, for example, when automated techniques were employed when building the CBR system.

During the lifetime of a CBR system, new cases can be added to a case-base for different reasons. Typically, the retain stage of the CBR cycle gradually

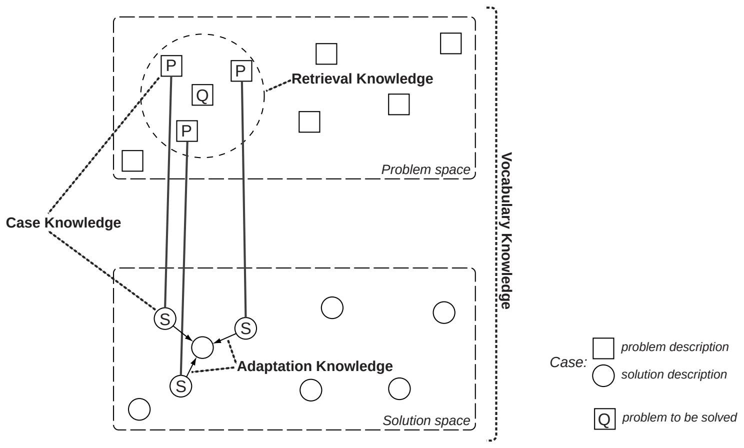


Figure 2: Four types of knowledge used in the CBR process (Wilke *et al.* 1997).

adds cases as new problems are solved. Apart form that, the knowledge engineer may select additional new cases to provide better case coverage. If the problem-solving policy changes over time, then the case-base must also change to reflect the new solutions. The old cases may have to be removed or revised in order to avoid conflicts with new cases. Sometimes the case description may have to be altered (extra features) to better differentiate between cases.

In this paper we concentrate on two maintenance scenarios: an expanding case-base as new cases are added, or an altered case-base where different cases and/or solutions reflect a change in the problem-solving policy. We have met the two scenarios (separately) in our tablet formulation application (Craw, Wiratunga & Rowe 1998) that we are developing with AstraZeneca, a major international pharmaceutical company. New drugs were developed, and so the case-base grew from a collection of formulations for 12 drugs (48 tablets) to one containing formulations for 36 drugs (144 tablets). Secondly, AstraZeneca changed its policy for tablet formulation. For the 12 drug case-base, this resulted in 37 of the 48 formulations changing (i.e., at least one of the formulation components changed). When we consider the two scenarios, we see that the extra cases, available when the case-base has grown, contain extra knowledge about the problem solving and provide an *opportunity* to improve retrieval. In the second scenario, revising the retrieval is *necessary* for good problem solving.

It is well known that constructing CBR systems for some types of problem requires significant knowledge acquisition effort (Cunningham & Bonzano 1999). Maintenance of CBR systems may be just as demanding. Therefore, both knowledge acquisition and maintenance would benefit from availability of tools which help reduce effort needed to perform these tasks. Such tools would be especially useful if integrated with CBR shells. We believe that refining the retrieval process is an important component of CBR maintenance and so we explore automatic refinements to the case-base index and similarity measure that may be prompted by changes to the case-base. The retrieval optimisation we have developed applies a Genetic Algorithm (GA) to select relevant features and choose feature weights to suit the cases in the case-base (Jarmulak, Craw & Rowe 2000a). Although this was initially developed as a knowledge acquisition tool, it can equally well be used to reduce knowledge maintenance effort.

This paper concentrates on maintaining retrieval knowledge to correspond to changes that result from case-base maintenance. First we discuss how CBR retrieval uses indexing and similarity knowledge and how it can be refined (Section 2). Our GA optimisation and the way it adapts the retrieval to case-base changes is described in Section 3. Section 4 introduces our tablet formulation domain and the two maintenance scenarios it presented. Section 5 contains retrieval optimisation results for the original case-base and for the two maintenance scenarios in tablet formulation. Finally, we look at related work in Section 6, before drawing some conclusions about optimising retrieval as a component of CBR maintenance in Section 7.

2 CBR Retrieval

We adopt a standard retrieval model where a decision-tree index selects potentially relevant cases and a nearest-neighbour algorithm applies a similarity measure to select most similar cases, see the left part of Figure 3. This retrieval is commonly used in commercial CBR tools; e.g., ReCall (ISoft), Kate (AcknoSoft), and The Easy Reasoner (The Haley Enterprise). As Figure 3 implies, this retrieval can be optimised by:

- acquiring knowledge about which *features* should be used to induce the decision-tree index, and which *weights* reflect the importance of features for the similarity measure;
- finding the best *parameters* for the induction algorithm that builds the decision tree index, and for the nearest-neighbour algorithm that determines closest matches.

ReCall is the CBR tool we use. In ReCall, C4.5 (Quinlan 1993) induces the decision-tree index and k-NN retrieves the k closest neighbours. Therefore, we are specifically interested in optimising the following:

- Binary *Index Weights* W_I – select index features;
- *Similarity Weights* W_S – the similarity measure is a weighted Euclidean distance;
- *Index Parameters* P_I – influencing minimum leaf size and tree pruning, here `-m` and `-c` for C4.5; and
- *Similarity Parameters* P_S – here k in k-NN.

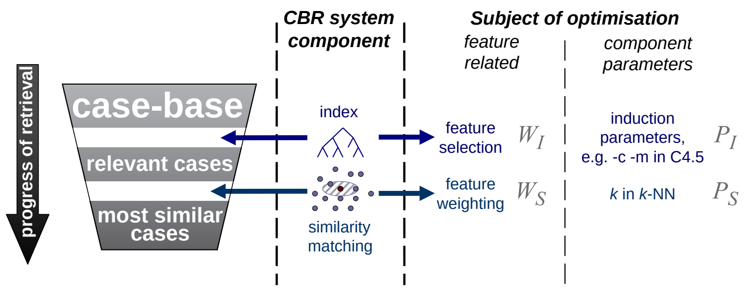


Figure 3: CBR Retrieval: Knowledge and Optimisation

While optimising the retrieval we provide feedback from the *complete* CBR retrieval process. Thus, optimising the index is done when it is part of the retrieval, so that the index chooses good cases for subsequent similarity matching. Similarly, the weights needed by the similarity measure are optimised so that they give best results when similarity measure is applied to only the cases selected by the index, and not to all cases. We have previously shown that simultaneous optimisation is different from separate optimisations (Jarmulak et al. 2000a). Selection and weightings from separately optimised index and similarity measure give inferior retrievals to those optimised simultaneously. Also, our past research concentrated on optimisation of retrieval as part of CBR knowledge acquisition. Here, we explore optimising and re-optimising CBR retrieval after the case-base expands or its cases are altered.

3 Optimising CBR Retrieval

We have chosen a Genetic Algorithm (GA) as the method to optimise CBR retrieval. GAs are well suited to the high dimensionality of the search space and the combination of their crossover and mutation operators with selection is good at preserving successful weights and parameters whilst introducing and experimenting with new different ones. Furthermore, the fitness can reflect our goal of optimising the complete CBR process. Finally, for CBR maintenance, the initial population of weights and parameters can be biased towards the existing retrieval parameters and weights.

3.1 GA Search

The GA that we use represents parameters (P_I, P_S) and feature weights (W_I, W_S) as real-valued genes in the GA chromosome, Figure 4. In each iteration of the GA, the population undergoes a process of mutation, followed by reproduction, and finally selection. Real-valued genes are mutated by adding a Gaussian-distributed offset, a standard crossover operator swaps blocks of genes between parents, and rank selection with elitism retains the best-fit chromosomes at the

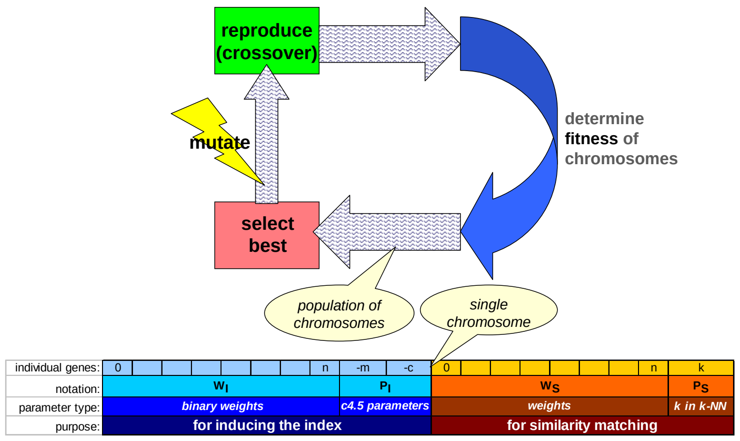


Figure 4: GA for Optimising Retrieval (see Figure 5 for fitness calculation)

end of each cycle. In our experiments, we used a population size of 100 chromosomes and ran the GA through a number of mutation/crossover cycles. In each mutation phase 50 "mutants" were added to the population and in each crossover phase 50 "children" were added. The selection phase at the end of each cycle reduced the population back to 100. The GA ran until no further improvement was observed over a number of cycles. We consider the time needed for retrieval optimisation as not critical because optimisation will be done off-line and is only applied rarely. Therefore, we made no attempt to optimise the GA itself (we use our own implementation in C++), and shorter processing times are possible. The average time required for an optimisation run is approximately 30 minutes on a 400MHz UltraSparc station.

We next define a fitness function that *estimates* the retrieval quality from the case-base using a decision-tree index induced using feature weights W_I and parameters P_I and applying a k-NN algorithm with the feature weights W_S and parameter P_S. By supplying fitness feedback from the complete CBR retrieval process we achieve simultaneous optimisation of both indexing and similarity matching.

3.2 GA Fitness Function

To evaluate the quality of retrieval using the parameters and feature weights in the chromosome, we must solve new problems not already in the case-base. Therefore, we partition the original case-base into a (smaller) case-base and test data for the retrieval experiments. An n-fold cross-validation (the "x-val loop" in Figure 5) partitions the original case-base into n equally-sized disjoint subsets. Repeated cross-validation experiments take one of the folds as the x-val test set and create x-val case-bases containing the cases from the remaining $n-1$ folds. A new decision-tree index is induced for each x-val case-base using the

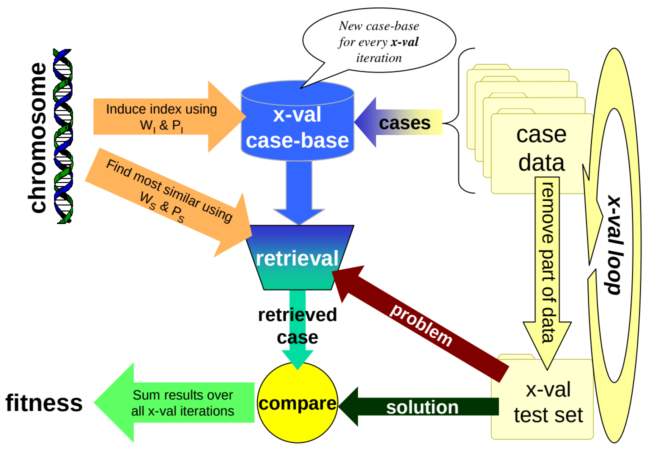

New case-base for every **x-val** iteration
chromosome
Induce index using W_I & P_I
x-val case-base
cases
case data
Find most similar using W_S & P_S
retrieval
remove part of data
x-val loop
retrieved case
problem
fitness
Sum results over all x-val iterations
compare
solution
x-val test set

Figure 5: Calculating GA Fitness

feature weights W_I and parameters P_I from the chromosome. The similarity matching applies the weights W_S and the parameter P_S. This x-val CBR system is evaluated on the corresponding x-val test set, and the solutions predicted by the x-val CBR system are compared with the actual solutions from the original case-base. Finally, the fitness of the chromosome is the average predictive accuracy from the full set of cross-validation experiments.

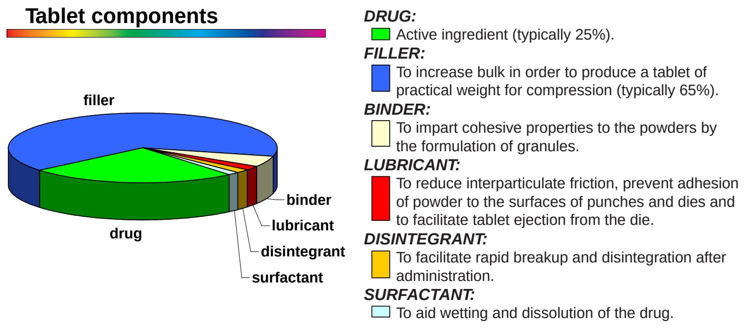
3.3 The Gain of Re-optimising

The above-described approach was initially developed to optimise a standard index-and-nearest-neighbour CBR retrieval during knowledge *acquisition*. However, it is equally applicable to CBR maintenance, where we refine the CBR retrieval to reflect changes in the case-base. Thus, when calculating fitness, the updated case-base is used as the case-data for the x-val loop. Furthermore, if appropriate, we can apply the approach as a re-optimisation where instead of randomly seeding the initial GA population, a portion of the population is created by adding a Gaussian-distributed offset to the weights and parameters already used in the current CBR system. Thus, the re-optimisation benefits from the previous optimisation, but as an optimisation it also reacts to any changes in the case-base.

If the contents of the case-base change as a result of the maintenance then the retrieval may need to change in reaction. If the case-base has grown significantly then the index may have to be constructed anew, using new induction parameters, and also a different k for k-NN similarity matching may be needed. More fundamentally, a larger case-base offers an opportunity to improve feature selection and feature weighting. A change in feature selection and feature weighting may be necessary if novel problems are solved *differently* e.g., as a result of a change in the problem-solving policy itself. The feature weights will need to reflect the new feature relevances and importances for the changed problem-solving.

4 Problem Domain

We demonstrate use of our optimisation approach, which is generic, on a particular application domain, tablet formulation (Rowe 1993). Given a drug and the desired dosage, a tablet formulation consists of a number of compounds (so-called excipients) that are mixed with the drug to make a viable tablet as shown in Figure 6. These provide the tablet with the desired mechanical properties, long-term stability, proper drug release when swallowed, effective manufacture, etc. A tablet formulation identifies the filler, binder, disintegrant, lubricant and surfactant to be added to the drug, together with the quantities of each. There are 8 possible choices for filler, 5 for binder, 2 for lubricant, 5 for disintegrant, and the surfactant can either be present or not.



DRUG:
🟩 Active ingredient (typically 25%).
FILLER:
🟦 To increase bulk in order to produce a tablet of practical weight for compression (typically 65%).
BINDER:
🟨 To impart cohesive properties to the powders by the formulation of granules.
LUBRICANT:
🟥 To reduce interparticulate friction, prevent adhesion of powder to the surfaces of punches and dies and to facilitate tablet ejection from the die.
DISINTEGRANT:
🟧 To facilitate rapid breakup and disintegration after administration.
SURFACTANT:
⬜ To aid wetting and dissolution of the drug.

Figure 6: Ingredients of a Tablet

An important aspect of tablet formulation is that several solutions may be acceptable; e.g., several excipients may be suitable fillers for a tablet of a given drug and dose. Therefore, when evaluating a tablet formulation system, an *exact accuracy*, based on exactly matching the actual filler, is too restrictive as a measure of predictive accuracy. Instead, we constructed an approximate similarity matrix for the fillers based on the weighted Euclidean distance between the filler properties. Our formulation expert provided the weights and also declared that pairs of fillers with similarity greater than 0.9 should be considered as interchangeable, and those with similarity less than 0.5 were incompatible. The 0.9 similarity threshold defines the required degree of match for a weaker *correct accuracy*. A similar approach was used for the other excipients in the formulation. These weights and thresholds were the only feedback on solution quality required from the expert.

AstraZeneca has provided us with historical data on tablet formulation. We have formulations for two sets of drugs each formulated according to the *same* tablet formulation policy, where one set of drugs pre-dates the other. *Form-12* contains formulations for four doses of an early set of 12 drugs. *Form-36* consists of the Form-12 formulations together with formulations for four doses of a later set of 24 drugs. These datasets fit our first maintenance scenario of an expanding case-base. CBR systems built from Form-12 formulations represent earlier formulation experience than those created with Form-36 data.

At a later time, AstraZeneca also changed its formulation policy. Under the new policy, excipients were grouped into four classes; class I excipients were preferred in formulations to those in class II, etc. This change in policy affected many of the 48 formulations in the Form-12 dataset: 7 fillers, 33 binders, 12 disintegrants and 8 surfactants changed. The result of introducing excipient preferences to the formulation policy is visible in the increase in the default accuracy for all four excipient types. These new formulations give us a third dataset: *NewForm-12* contains formulations for the same drugs and doses as Form-12 but now the formulations correspond to the revised formulation policy. The Form-12 and NewForm-12 datasets fit our second maintenance scenario where the case-base changes as a result of a revised problem-solving policy. CBR systems built from the Form-12 formulations capture the original policy and those from NewForm-12 reflect the new policy.

We also have a dataset for the complete set of 39 drugs formulated according to the original policy. The drug doses, and hence the formulations, in this dataset are not necessarily the same as those in the other datasets. *Form-39* consists of formulations for four realistic doses of all 39 drugs and allows additional optimisation experimentation on a larger dataset.

5 Experiments and Results

Our experiments are designed to show whether optimising retrieval achieves improved formulations. In particular, we are interested in the need to refine the CBR retrieval when the case-base changes during maintenance. We also wish to explore whether there is a gain in re-optimising from the existing parameters and weights over simply optimising from a random start point.

For CBR, the tablet formulation problem is represented by a feature vector containing the dose, 5 physical properties of the drug, and 20 chemical properties of the drug with the excipients. Its solution identifies the filler, binder, disintegrant, lubricant and surfactant, together with the quantities of each, Figure 7. The maintenance experiments reported in this paper focus on a subproblem of tablet formulation, choosing which excipients to use. (Because all the tablets in our dataset contained the same lubricant we omit results for lubricant prediction.) It is possible to determine the whole formulation (all 5 excipients) simultaneously in a *single* retrieval, but our formulation expert suggested it was better to choose the excipients individually, in the order filler, binder, lubricant, disintegrant and surfactant, making use of previously chosen excipients and their amounts. This gives better results as well as simplifies the subsequent adaptation task.

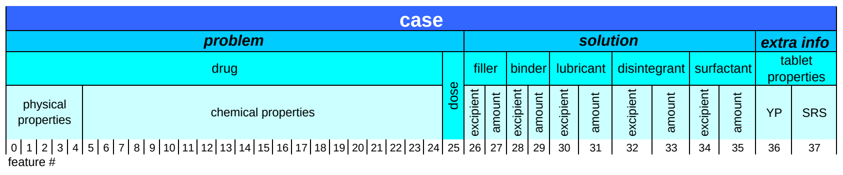


Figure 7: Contents of a Case

The smallness of the 12-drug datasets posed serious over-fitting problems compared to earlier experiments with the 39 drug dataset (Jarmulak, Craw & Rowe 2000b). In an attempt to counter this, we incorporated jitter (Koistinen & Holmstrom 1992) in the CBR retrieval by adding constrained Gaussian noise to the individual features while calculating distances in the similarity measure. This had the desired effect of reducing over-fitting and has been used in all the experiments involving 12-drug datasets. In the earlier experiments we found that a small number of possible weights reduces the danger of over-fitting, and so here we use only binary weights. Interestingly, when our formulation expert was asked to specify feature importance, he used only two possible weights. Nevertheless, we plan to use more, possible weights for those parts of formulation prediction that are not affected much by over-fitting.

For our evaluation experiments, we apply a repeated 6-fold cross-validation. The formulations are partitioned according to drug into 6 disjoint folds. Each single cross-validation experiment uses the formulations from a different fold as the evaluation set and the formulations in the remaining 5 folds are used for the case-base in a CBR system whose retrieval stage we optimise. To obtain statistically meaningful results, these cross-validation experiments are repeated 8 times by creating new 6-fold partitions. The average predictive (correct) accuracy for these 48 sets of retrievals is reported. The results show 95% confidence intervals for the *averages*. We estimate significance of the improvement, relative to non-optimised retrieval, using repeated-measures-design ANOVA (Howell 1997); the bar-charts report the p-value.

Comparing the non-optimised retrievals in Figure 8 illustrates the competence of the various case-bases. Filler and binder prediction under the new formulation policy is more accurate, because the preferred-excipient scheme results in a smaller variety of used excipients. For disintegrant on the other hand, the new-policy formulations have one more excipient, which seems to make the prediction more difficult. Actual numbers of possible excipients used for old & new policies are respectively: 6 & 5 for filler, 5 & 4 for binder and 3 & 4 for disintegrant. It is worth noting that for the disintegrant the change in the formulation policy is due to an emergence of a new excipient on the market which, not only was not used in Form-12 formulations, but has now become the *preferred* excipient in NewForm-12. It is interesting to note that for NewForm-12 the accuracy of non-optimised retrieval is not very much better than default accuracy, this might be due to the use of preferred excipients. Binder, disinte-

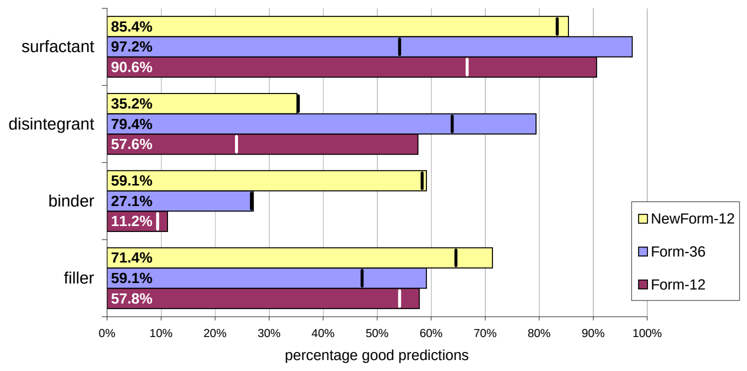


Figure 8: Non-optimised retrieval accuracy for the three case-bases. Default accuracy is marked by thick lines across the bars.

grant and surfactant predictions are all made easier when a larger data set is available; e.g., Form-36 dataset.

5.1 Initially Optimising a CBR System

In this first experiment we compare the predictive accuracy of the non-optimised and optimised CBR systems constructed from the Form-12 dataset. Figure 9 illustrates the difficulty with this small dataset. Only binder results are improved significantly by optimisation. Filler prediction shows no improvement since filler prediction is the hardest task. This is due to the largest possible excipient choice (6) and the prevalence of a particular filler excipient (about 54% of all tablets), which leaves very few tablets with the remaining excipients. Optimisation also brings no improvement for disintegrant, however, for both filler and disintegrant, the optimisation seems to capture some of the domain knowledge, see Section 5.3. Although surfactant prediction itself is a relatively easy task, the optimisation of surfactant prediction is difficult with the small datasets because only a few of these formulations actually have a surfactant (a third for Form-12). We have analysed this problem more closely and it turns out that, in spite of bad average results, optimisation is successful in extracting correct knowledge for surfactant prediction except for these x-val folds where no positive examples are present; there the optimisation fails completely.

For comparison, Figure 10 shows the corresponding results for the Form-39 dataset containing all the drugs (no p-values were calculated because the evaluation sets were not matched and only the overlap of the confidence intervals for the averages is used to determine significance). For completeness, this diagram also includes the predictions of excipient amounts. (Surfactant amount has 100% accuracy with non-optimised retrieval and so these results are omitted.) Form-39 is a larger dataset and so we did not use jitter. The results from Figure 10, can be divided into 3 groups: (1) results without optimisation

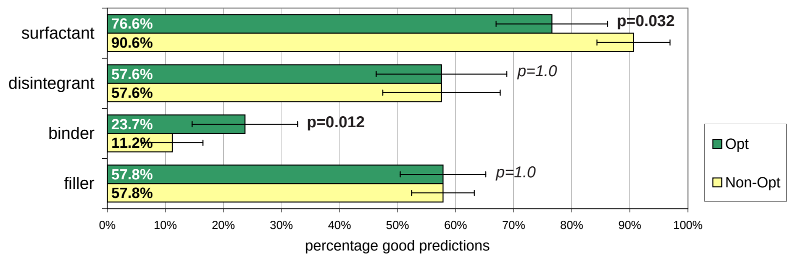


Figure 9: Optimising a CBR Tablet Formulation System – Form-12 Data Set

are already so good that optimisation brings no significant improvement, e.g., surfactant and binder amount; (2) optimisation brings a statistically significant improvement, e.g., disintegrant and all the excipient amounts; (3) optimisation brings hardly any improvement due largely to over-fitting, e.g., for filler and binder. Here adding jitter might help. We should note that even when optimisation brings no improvement in accuracy, like for surfactant, it was found useful in identifying relevant features and thus discovering useful knowledge.

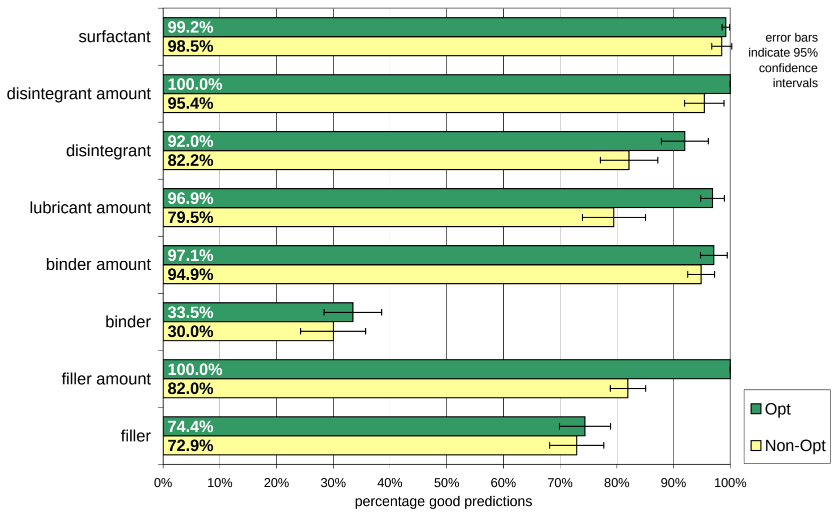


Figure 10: Optimising a CBR Tablet Formulation System – Form-39 Data Set

5.2 Re-Optimising Experiments

In the remaining experiments we treat the Form-12 formulations as original cases, and the Form-36 or NewForm-12 formulations as the revised cases after case-base maintenance. We are interested in the need to refine the retrieval process in reaction to the case-base maintenance, and whether re-optimisation provides a gain. We compare the predictive accuracy of the non-optimised retrieval (*Non-Opt*) with three optimisations:

- *Old-Opt* uses the feature weights and parameters from the corresponding optimised Form-12 CBR system;
- *Re-Opt* is optimised from an initial GA population partially seeded with the feature weights and parameters from the corresponding optimised Form-12 CBR system; and
- *New-Opt* is optimised using a random initial GA population.

Thus, Non-Opt indicates the need to optimise, and Old-Opt shows the need to re-optimise the previously optimised retrieval. Re-Opt and New-Opt indicate the gains of optimisation generally and whether re-optimisation is preferred.

5.2.1 Re-Optimising an Expanding Case-Base

We view the additional Form-36 formulations as solutions for new problems that are added to the case-base. We continue to use a 6-fold cross-validation, but re-specify it to include the new cases present in Form-36. The folding of the Form-12 dataset and the formulations for the 24 new drugs are performed separately, and the folds paired to give 6 folds of Form-36. Therefore each Form-36 CBR system is evaluated on 6 (2 early and 4 recent) drugs and its case-base contains formulations for 30 (10 early and 20 recent) drugs.

The bars in Figure 11 demonstrate the effectiveness of optimisation in this maintenance scenario. Firstly, the previous optimisation (Old-Opt) performs about the same as Non-Opt retrieval, except for surfactant where the accuracy is significantly lower. However, for surfactant we would choose the default retrieval anyway in view of bad Form-12 results. The results for the optimised retrievals indicate the gain of (re-)optimising when the case-base grows. The gain is significant for binder and disintegrant. The non-optimised retrieval for surfactant is already so good that further improvement is difficult. As always, improving the filler prediction is difficult, and more generally we feel that filler prediction requires an adaptation step. Unsurprisingly, the predictions of New-Opt and Re-Opt are similar showing that the GA has achieved stability. Surprisingly, we have found however that Re-Opt requires slightly more cycles (on average 51 as compared to 46) than New-Opt to reach its optimum. The extra cycles might be needed to move away from the local optimum provided by Form-12 optimisation in order to find a better one – we have found that there are many possible feature weight choices leading to the same retrieval results.

5.2.2 Re-optimising for Different Problem-Solving

The NewForm-12 formulations are changed by the revised formulation policy and so form a new case-base. Since the formulation *problems* are unchanged

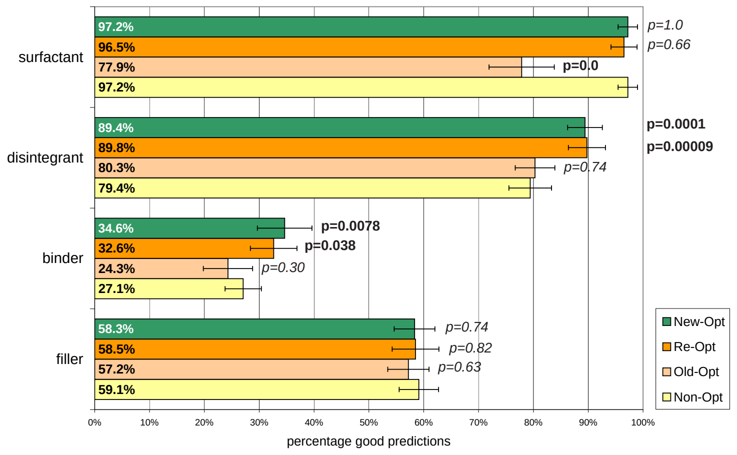


Figure 11: Comparing Optimisations for an Expanding Case-Base – Form-36 Data Set

(only the formulations are revised) we are able to replicate the 6-fold partitionings used in Section 5.1. The results are summarised in Figure 12. We see that for disintegrant the optimised weights from Form-12 give also good results for NewForm-12. However, using Form-12 weights in NewForm-12 leads to significantly worse results for the binder. It seems that the new formulation policy for the binder is quite different from the old one; in Non-Opt the accuracy is much higher than for Form-12, however, it also is more difficult to improve compared to default retrieval. The reason for this is that while in the old formulation policy the 5 possible binder excipients were reasonably equally represented, with a slight preference for one of them, the new policy, which uses 4 possible binders, gives a very strong preference to two (different) binder excipients, resulting in few examples of the use of the remaining two. Similarly as for Form-12, we have the problem with surfactant as only 1/6th of the formulations actually have a surfactant. The average number of GA cycles required for both New-Opt and Re-Opt is about 50.

5.3 Correspondence to Domain Knowledge

We wish to compare the optimised feature selections and weights with those suggested by our formulation expert. We have already noted that our feature selections and weightings are chosen to optimise the complete retrieval process: the index features select optimal cases for subsequent matching and the feature weighting optimises the similarity matching with respect to pre-selected cases. In contrast our expert identified relevant features for the problem-solving in general. Therefore, there will not be a close correspondence between the expert's choice and the optimisation. Another problem is the transitional type

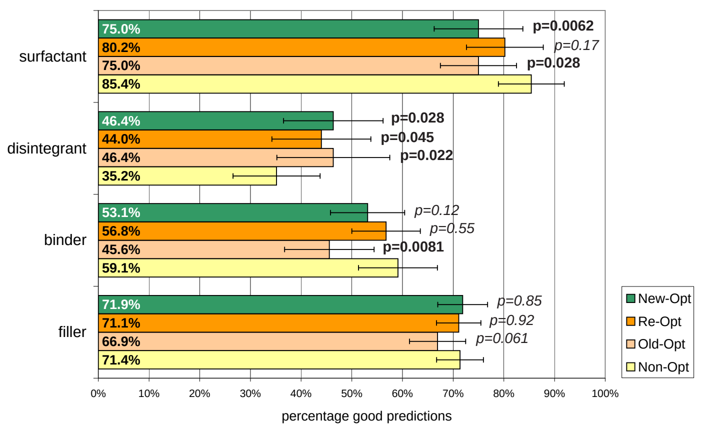


Figure 12: Comparing Optimisations for Different Problem-Solving – NewForm-12 Data Set

of dependence of solutions on relevant features, i.e., if binder depends on filler, but filler depends on physical drug properties, then the optimal weights found for binder prediction may reflect any of these dependencies making the analysis of results difficult. Nevertheless, features known to be relatively important are more likely to be assigned higher weights, and relevant features should be more frequently selected for the index.

Table 1 shows how the optimisation results match the domain knowledge. In each column the results are presented as `Form-36 (Form-12) NewForm-12`. On average, the percentage selected by optimisation for the decision-tree is higher for features declared to be relevant by the expert, compared to the other features. The average similarity weighting for relevant features is also higher than for irrelevant features. Ideally we want no irrelevant features selected and the weights for irrelevant features to be zero. However, with dependencies among features we do not necessarily need all relevant features to be selected. Furthermore, relevant features need not attract a weight of 1, instead non-zero weights can indicate their relative importance.

Table 1: Match of optimisation results and domain knowledge.

	percentage selected features		average feature weight	
task	relevant	irrelevant	relevant	irrelevant
filler	**40%**(23%)**19%**	32%(23%)18%	**0.34**(**0.27**)**0.10**	0.31(0.16)0.06
binder	**40%**(14%)**36%**	24%(**18%**)31%	**0.43**(0.35)**0.42**	0.30(0.27)0.39
disintegrant	**18%**(**38%**)**38%**	5%(14%)13%	0.05(0.40)**0.15**	**0.07**(0.39)0.12
surfactant	**44%**(**38%**)**13%**	1%(1%)6%	**0.08**(**0.21**)**0.26**	0.04(0.03)0.05

6 Related Work

Most of the literature on CBR maintenance is devoted to the problem of case-base maintenance, with only a few publications addressing the overall CBR system maintenance (Heister & Wilke 1998). Many researchers have investigated various case addition, removal and revision strategies with the indicated goals of limiting the case-base size (Surma & Tyburcy 1998), reducing redundancy and inconsistency (Racine & Yang 1997), increasing efficiency while maintaining competence as evaluated using an explicit model (Smyth & McKenna 1999), and conforming to case-authoring guidelines while maintaining precision and efficiency (Aha & Breslow 1997). A framework describing these case-base maintenance methods is proposed in (Leake & Wilson 1998).

Another aspect of case-base maintenance is maintaining the case indexing, whether the indexing takes the form of an index external to the cases themselves or is a part of case representation. Because indexing serves to retrieve relevant cases efficiently, we consider maintenance of the index as part of retrieval knowledge maintenance. Fox & Leake (1995) describe a system capable of on-line learning, where the indexing knowledge of which features should be used to retrieve/distinguish cases is learned introspectively from retrieval failures. Once the useful features are identified, the case-base is re-indexed. The already mentioned case refinement strategy from Aha & Breslow effectively results in changing the case indexing which in turn influences the retrieval. In addition to indexing knowledge, retrieval knowledge also includes similarity knowledge. In CBR systems the learning/maintenance of similarity knowledge has concentrated mainly on learning the weights used in the similarity measure (Munoz-Avila & Huellen 1996, Wilke & Bergmann 1996).

In our approach we learn the retrieval knowledge by *combining* learning which features should be used to construct the decision-tree index, with learning what weights should be used in the similarity measure. We built on the extensive research that has been done within the Machine Learning community on optimising decision-tree classifiers or k-NN classifiers by feature selection or weighting. It is known that, although inductive algorithms themselves perform feature selection, classification problems with many irrelevant features may result in sub-optimal decision trees (Almuallim & Dietterich 1992), and it is better to remove irrelevant features before the decision tree is induced (John, Kohavi & Pfleger 1994). Removing irrelevant features also improves nearest-neighbour classifiers (Aha & Bankert 1994, Skalak 1994). For problems where the relative importance of features plays a role, nearest-neighbour classifiers can also benefit from feature weights in the similarity measure (Wettchereck & Aha 1995, Kohavi, Langley & Yun 1997).

Optimisation algorithms form two classes: *filter* methods operate without feedback from the subsequent performance of the machine learning algorithm; *wrapper* methods utilise this feedback. Both approaches have been discussed in the context of feature selection and weighting (John et al. 1994, Blum & Langley 1997, Aha 1998). Despite their larger computational costs, wrappers are often preferred (Aha & Bankert 1994), and GAs are frequently used as wrappers for feature selections or weightings for k-NN algorithms (Kelly & Davis 1991, Wilson & Martinez 1996). Recently GA wrappers have been applied to optimise feature weights in genuine CBR retrieval systems (Oatley, Tait & MacIntyre 1998, Gomes & Bento 2000). In both systems the fitness is the degree of match between the ranking of the retrieved cases and that defined by the domain expert. Oatley found the expert-defined ranking to be expert intensive and it was subsequently abandoned (Oatley 2000). Gomes & Bento avoid this expert effort by instead asking him to re-sort retrievals if necessary. Furthermore, only a few weights are optimised and so a small number of retrievals may have sufficed.

Our realisation that retrieval optimisation applies equally well to the development and maintenance of CBR systems stems from our work on the refinement of rule-based systems. In addition to the standard use of automated refinement to identify and fix faults in new rule-based systems, we discovered that our rule refinement tool (Craw, Boswell & Rowe 1997) successfully coped with the significant rule-base re-engineering necessitated by AstraZeneca's change of policy for tablet formulation, described in Section 4.

7 Conclusions

The optimisation of feature selection, feature weights and retrieval parameters has been presented as a way of refining CBR retrieval when the case-base is changed as a result of case-base maintenance. Two maintenance scenarios were considered: the case-base expanded after new cases were added; and the solutions in the case-base were altered to reflect a changed problem-solving policy. These scenarios occurred in practice with AstraZeneca's tablet formulation application. Experiments with tablet formulation data showed that in both scenarios updating the case-base only partially addressed the maintenance. Refinements to the retrieval knowledge were also beneficial. Thus we have found that changes to the case-base could also affect indexing and similarity knowledge, with the effect that refinement to this knowledge improved the retrievals from the updated case-base. It is important to note that in our case the initial retrieval knowledge was derived from the case-base data. The whole approach of automatic refinement of retrieval would apply less, or would have to be approached differently, for CBR systems with retrieval knowledge coded by an expert.

The GA optimisation we have already used to acquire the retrieval knowledge during development of a CBR system, is also effective to refine this knowledge. Knowledge acquisition was in fact refinement of the default retrieval knowledge; i.e., all features are relevant for indexing and equally important for the similarity measure, and all retrieval parameters have default values. Experiments on retrieval after case-base maintenance compared optimisation of the default retrieval knowledge with re-optimisation of the retrieval knowledge prior to maintenance. In both situations the retrieval results were comparable and also required similar number of GA cycles. We expect that the the number of cycles required for re-optimisation would be lower once the case-base is larger and the knowledge contained in it stabilises.

The small amount of tablet formulation case-data is a problem to us, although it is a realistic, even generous, size in this domain. Adding jitter to the retrieval only partially addressed the over-fitting problems. Also, the error-bars in our results are quite large despite applying repeated cross-validation. A larger dataset would allow a better train/test evaluation. We would also have larger case-bases with the effect that the fitness might be simplified by applying a more straightforward case-data/test-data splitting than the cross-validation currently applied in the x-val loop.

We have proposed that refinement of the retrieval knowledge is applied when the case-base contents change as a result of broad case-base maintenance. However, it should be noted that the optimisation can be applied at any time and requires no human input, since fitness is calculated using only the cases already in the case-base. This offers the possibility of adaptive retrieval: optimisation is applied between explicit case-base maintenance efforts, either regularly when the case-base grows by a specified amount, or triggered when the retrieval accuracy deviates from that achieved previously. We think that suitable measures of the performance should be local, for example measuring changes in the retrieval accuracy in the neighbourhood of newly added cases, as they are more effective in spotting deterioration of the performance. However, the issue of an appropriate trigger is not crucial, as optimisation is such that if retrieval refinement is not required then the optimisation will have little effect and has simply been applied unnecessarily.

We are convinced that existing CBR shells would benefit from including an option to automatically generate an optimised retrieval process, based on just the cases already present in the case-base. Not only would this reduce the development time and cost for new CBR systems, but it could also be useful for re-optimising retrieval during the lifetime of a CBR system, and in particular after case-base maintenance.

Acknowledgments

This work is supported by EPSRC grant (GR/L98015) awarded to Susan Craw. We thank ISoft for its contribution of the ReCall software.

References

Aha, D. (1998). Feature weighting for lazy learning algorithms, *in* H. Liu & H. Motoda (eds), *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Norwell MA: Kluwer.

Aha, D. W. & Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison, *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning*, AAAI Press, Seattle, pp. 106–112.

Aha, D. W. & Breslow, L. A. (1997). Refining conversational case libraries, *in* D. B. Leake & E. Plaza (eds), *Case-Based Reasoning Research and Development: Proceedings of the 2nd International Conference on CBR, ICCBR-97*, LNAI 1266, Springer Verlag, pp. 267–278.

Almuallim, H. & Dietterich, T. G. (1992). Efficient algorithms for identifying relevant features, *Proceedings of the Ninth Conference on Artificial Intelligence*, Morgan Kaufman, Vancouver, pp. 38–45.

Blum, A. L. & Langley, P. (1997). Selection of relevant features and examples in machine learning, *Artificial Intelligence* **97**(1-2): 245–271.

Craw, S., Boswell, R. & Rowe, R. (1997). Knowledge refinement to debug and maintain a tablet formulation system, *Proceedings of the 9TH IEEE International Conference on Tools with Artificial Intelligence (TAI'97)*, IEEE Press, Newport Beach, CA, pp. 446–453.

Craw, S., Wiratunga, N. & Rowe, R. (1998). Case-based design for tablet formulation, *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, Springer, Dublin, Eire, pp. 358–369.

Cunningham, P. & Bonzano, A. (1999). Knowledge engineering issues in developing a case-based reasoning application, *Knowledge-Based Systems* **12**: 371–379.

Fox, S. & Leake, D. B. (1995). Learning to refine indexing by introspective reasoning, *in* M. Veloso & A. Aamodt (eds), *Proceedings of the 1st International Conference on CBR (ICCBR-95)*, LNAI 1010, Springer Verlag, pp. 431–440.

Gomes, P. & Bento, C. (2000). Learning user preferences in case-based software reuse, *in* E. Blanzieri & L. Portinale (eds), *Proceedings of the 5th European Workshop on CBR – EWCBR-2K*, LNAI 1998, Springer Verlag, pp. 112–123.

Heister, F. & Wilke, W. (1998). An architecture for maintaining case-based reasoning systems, *in* B. Smyth & P. Cunningham (eds), *Proceedings of the 4th European Workshop on CBR – EWCBR-98*, LNAI 1488, Springer Verlag, pp. 221–232.

Howell, D. C. (1997). *Statistical Methods for Psychology*, Duxbury Press, Pacific Grove, CA.

Jarmulak, J., Craw, S. & Rowe, R. (2000a). Genetic algorithms to optimise CBR retrieval, *in* E. Blanzieri & L. Portinale (eds), *Advances in Case-Based Reasoning: Proceedings of EWCBR-2K*, Springer-Verlag, Berlin, Trento, Italy, pp. 137–149.

Jarmulak, J., Craw, S. & Rowe, R. (2000b). Self-optimising CBR retrieval, *Proceedings of the 12th IEEE International Conference On Tools with Artificial Intelligence (ICTAI-2000)*, IEEE Press, Vancouver, Canada.

John, G., Kohavi, R. & Pfleger, K. (1994). Irrelevant features and the subset selection problem, *in* W. W. Cohen & H. Hirsh (eds), *Machine Learning: Proceedings of the 11th International Conference*, Morgan Kaufmann, pp. 121–129.

Kelly, J. D. & Davis, L. (1991). A hybrid genetic algorithm for classification, *Proceedings of the 12th IJCAI*, Morgan Kaufmann, San Mateo, CA, Sidney, Australia, pp. 645–650.

Kohavi, R., Langley, P. & Yun, Y. (1997). The utility of feature weighting in nearest-neighbor algorithms, *Proceedings of the European Conference on Machine Learning (ECML-97)*, Springer-Verlag, Berlin.

Koistinen, P. & Holmstrom, L. (1992). Kernel regression and backpropagation training with noise, *in* J. E. Moody, S. J. Hanson & R. P. Lippman (eds), *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 1033–1039.

Leake, D. B. & Wilson, D. C. (1998). Categorizing case-base maintenance: Dimensions and directions, *in* B. Smyth & P. Cunningham (eds), *Proceedings of the 4th European Workshop on CBR – EWCBR-98*, LNAI 1488, Springer Verlag, pp. 196–207.

Munoz-Avila, H. & Huellen, J. (1996). Feature weighting by explaining case-based problem solving episodes, *in* I. Smith & B. Faltings (eds), *Proceedings of the 3rd European Workshop on CBR – EWCBR-96*, LNAI 1168, Springer Verlag, pp. 280–294.

Oatley, G. (2000). *An Investigation of Case-Based Reasoning for Decision Support of Diagnosis in a Large-Scale Ill-Structured Domain*, PhD thesis, University of Sunderland.

Oatley, G., Tait, J. & MacIntyre, J. (1998). A case-based reasoning tool for vibration analysis, *in* R. Milne, A. Macintosh & M. Bramer (eds), *Applications and Innovations in Expert Systems VI: Proceedings of the BCS Expert Systems '98 Conference*, Springer-Verlag, Cambridge, December 1998, pp. 132–146.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

Racine, K. & Yang, Q. (1997). Maintaining unstructured case-bases, *in* D. B. Leake & E. Plaza (eds), *Case-Based Reasoning Research and Development: Proceedings of the 2nd International Conference on CBR, ICCBR-97*, LNAI 1266, Springer Verlag, pp. 553–564.

Richter, M. M. (1998). Introduction, *in* M. Lenz, B. Bartsch-Spoerl, H.-D. Burkhard & S. Wess (eds), *Case-Based Reasoning Technology: From Foundations to Applications*, Lecture Notes in Artificial Intelligence 1400, Springer Verlag.

Rowe, R. (1993). An expert system for the formulation of pharmaceutical tablets, *Manufacturing Intelligence* **14**: 13–15.

Skalak, D. B. (1994). Prototype and feature selection by sampling and random mutation hill-climbing algorithms, *Proceedings of the Eleventh International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA, New Brunswick, New Jersey, pp. 293–301.

Smyth, B. (1998). Case-base maintenance, *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Springer-Verlag, Berlin, pp. 507–516.

Smyth, B. & McKenna, E. (1999). Building compact competent case-bases, *in* K.-D. Althoff, R. Bergmann & L. K. Branting (eds), *Case-Based Reasoning Research and Development: Proceedings of the 3rd International Conference on CBR, ICCBR-99*, LNAI 1650, Springer Verlag, pp. 329–342.

Surma, J. & Tyburcy, J. (1998). A study on competence-preserving case replacing strategies in case-based reasoning, *in* B. Smyth & P. Cunningham (eds), *Proceedings of the 4th European Workshop on CBR – EWCBR-98*, LNAI 1488, Springer Verlag, pp. 233–238.

Wettchereck, D. & Aha, D. W. (1995). Weighting features, *Proceedings of the 1st International Conference on CBR (ICCBR-95)*, Springer-Verlag, Berlin, pp. 347–358.

Wilke, W. & Bergmann, R. (1996). Considering decision cost during learning of feature weights, *in* I. Smith & B. Faltings (eds), *Proceedings of the 3rd European Workshop on CBR – EWCBR-96*, LNAI 1168, Springer Verlag, pp. 460–472.

Wilke, W., Vollrath, I., Althoff, K.-D. & Bergmann, R. (1997). A framework for learning adaptation knowledge based on knowledge light approaches, *5th German Workshop on Case-Based Reasoning (GWCBR'97).*

Wilson, D. R. & Martinez, T. R. (1996). Instance-based learning with genetically derived attribute weights, *Proceedings of the International Conference on Artificial Intelligence, Expert Systems, and Neural Networks (AIE'96)*, Fukuoka, Japan, pp. 11–14.