Learning Adaptation Knowledge to Improve Case-Based Reasoning

Susan Craw [a,*] Nirmalie Wiratunga [a] Ray C. Rowe [b]

[a] *School of Computing, The Robert Gordon University, Aberdeen AB25 1HG, UK*

[b] *Institute of Pharmaceutical Innovation, University of Bradford, Bradford BD7 1DP, UK*

Abstract

Case-Based Reasoning systems retrieve and reuse solutions for previously solved problems that have been encountered and remembered as cases. In some domains, particularly where the problem solving is a classification task, the retrieved solution can be reused directly. But for design tasks it is common for the retrieved solution to be regarded as an initial solution that should be refined to reflect the differences between the new and retrieved problems. The acquisition of adaptation knowledge to achieve this refinement can be demanding, despite the fact that the knowledge source of stored cases captures a substantial part of the problem-solving expertise. This paper describes an introspective learning approach where the case knowledge itself provides a source from which training data for the adaptation task can be assembled. Different learning algorithms are explored and the effect of the learned adaptations is demonstrated for a demanding component-based pharmaceutical design task, tablet formulation. The evaluation highlights the incremental nature of adaptation as a further reasoning step after nearest-neighbour retrieval. A new property-based classification to adapt symbolic values is proposed, and an ensemble of these property-based adaptation classifiers has been particularly successful for the most difficult of the symbolic adaptation tasks in tablet formulation.

Key words: case-based reasoning, adaptation knowledge, knowledge acquisition, machine learning, introspective learning.

* Corresponding author

Email addresses: S.Craw@comp.rgu.ac.uk (Susan Craw), nw@comp.rgu.ac.uk (Nirmalie Wiratunga), R.C.Rowe@bradford.ac.uk (Ray C. Rowe).

URLs: www.comp.rgu.ac.uk/docs/info/staff/smc.php (Susan Craw), www.comp.rgu.ac.uk/docs/info/staff/nw.php (Nirmalie Wiratunga), www.brad.ac.uk/acad/ipi/ (Ray C. Rowe).

1 Introduction

Case-Based Reasoning (CBR) solves new problems by retrieving previously solved problems and their solutions from a knowledge source of cases, the case-base. The solutions in the retrieved cases are reused as a proposed solution to the new problem. CBR's retrieval mechanism identifies similar problems from the past, in the expectation that these solutions will be useful for the new problem, and will be similar to its solution. CBR commonly uses a k-Nearest Neighbour (k-NN) algorithm to retrieve similar past cases from the case-base. CBR has some key differences to k-NN.

- *Case Knowledge* The case-base consists of carefully selected cases whose knowledge represents both prototypical situations and exceptional circumstances. Estimating the competence of a case-base and identifying missing or redundant cases are important areas of CBR research [1].
- *Similarity* The k-NN algorithm is designed to be used with numeric data that allows a natural distance function to define similarity. CBR systems often also have symbolic attributes for which similarity knowledge is needed.
- *Reuse* Aamodt & Plaza's cycle [2] contains *Retrieve* as the first stage and corresponds to k-NN. The *Reuse* and *Revise* stages follow, where the retrieved solution may be refined to reflect the effect of differences between the new and retrieved problems, or feedback on the solution's suitability. Finally, there is a *Retain* stage where the newly solved problem may be added as new case knowledge. Reuse can be more knowledge intensive than k-NN and Revise and Retain are additional stages peculiar to CBR [3].

Classification tasks (e.g. diagnosis) may reuse the retrieved solution directly. It would be normal to expect at least one case representing each class, but a case-base with insufficient coverage may not have a case which is similar to a particular new set of circumstances, and thus adaptation may still be necessary. For constructive tasks like design or planning, it is impossible for the case-base to contain all possible solutions (i.e. all designs, plans, etc). Therefore it is common for the retrieved solution to be refined to take account of differences in the requirements between the new and retrieved problems [8].

Case knowledge is a fundamental knowledge source for a CBR system and no problem-solving would be achieved without it. However, Richter identified two other knowledge containers whose knowledge improves the reasoning of a CBR system [4]. *Similarity knowledge* captures the similarities between the values of each attribute and is particularly important for symbolic attributes, and the relative importances of attributes is similar to feature selection and weighting in machine learning research; e.g. [5–7]. *Adaptation knowledge* is peculiar to CBR, since it is used in the Reuse stage to refine the initial solution proposed by the k-NN Retrieve stage.

Learning adaptation knowledge with which to refine CBR reasoning is the subject of this paper. We develop an approach that assembles adaptation training data from the case-base itself, and learns adaptation knowledge that refines initial CBR solutions retrieved from the case-base. Section 2 reviews existing research on adaptation in CBR. Our approach to acquiring adaptation training examples from the case-base is described in Section 3. Section 4 introduces pharmaceutical tablet formulation as a component-based design task and motivates its suitability as our test domain for adaptation. Section 5 presents results for a simple retrieve-only case-based approach and a CBR system with adaptation using the adaptation training examples for tablet formulation. Section 6 develops our adaptation approach by applying rule-learning algorithms to the adaptation training examples. Section 7 introduces a new property-based adaptation for symbolic solutions which is used to learn adaptation classifiers for these properties. An ensemble of the individual property-based classifiers is evaluated for the symbolic components in tablet formulation. Our conclusions on the introspective approach for adaptation, the challenges of adapting symbolic solutions and the suitability of the various learning algorithms for different adaptation tasks are summarised in Section 8.

2 Related Work

Given the open-ended nature of constructive tasks, it is not surprising that much of the CBR research on adaptation is devoted to design and planning systems; e.g. [9–12]. Another popular, simpler domain that demands refinement of the retrieved solution are domains involving numeric predictions; e.g. house price predictions [13]. The difficulty of acquiring adaptation knowledge was identified in early CBR research [14]. However, the dominance of classification in early CBR systems has meant that research into the adaptation needs of CBR has been quite limited [15]. More recent understanding of CBR for non-classification tasks [16] has highlighted the importance of adaptation, and the subsequent increased knowledge engineering demands of case-based design and planning systems. This has provided the motivation to devote effort to understanding adaptation knowledge, and exploring automated tools to assist with the acquisition of adaptation knowledge for the reuse stage.

Leake et al.'s DIAL [12] incorporates case-based adaptation for which the knowledge sources are humans, a rule-based repair theory, and existing adaptation cases. During problem-solving, adaptation is achieved using CBR retrieval from the adaptation case-base if possible, or otherwise applying a theory-generated or human-based adaptation. Any successful new adaptation is retained in the adaptation case-base allowing it to grow as new problems are solved and new adaptations are applied.

Hanney & Keane create adaptation rules by considering numeric differences between pairs of cases in a house price case-base [13]. Rule generalisation replaces each subset of rules with the same antecedent, with a single rule whose conclusion is the average of the price differences in the subset. This simple form of learning works well for their domain because of its small set of problem differences. McSherry's CREST system utilises a similar numeric difference approach, also for house prices [17]. When a new problem is to be solved, the CBR system retrieves three cases from the case-base: the new problem's nearest neighbour; and two cases whose differences are the same as those of the new and retrieved cases. The retrieved solution is updated with the solution difference between the pair of same-difference cases. Again this case-based approach works well when the case-base has a good coverage of the problem space and the differences that occur. CREST employs no form of generalisation of the differences; e.g. by retrieving pairs with similar differences.

The existing knowledge in the case-base is a natural source of information. Case-base maintenance tools routinely assume that the case-base contains knowledge that is representative of the problem-solving domain, and case-base competence modelling has relied on leave-one-out-testing from the case-base to predict CBR performance [1]. Wilke et al. [18] proposed a framework for learning adaptation knowledge from knowledge already in the CBR system. They apply their framework to create a knowledge-light learning system to optimise the value of k for k-nearest neighbour retrieved solutions. They also fit Hanney & Keane's learning of house price adaptations to their framework. Wilke et al.'s framework is the inspiration for our work but we learn more sophisticated adaptations and incorporate different learning algorithms to achieve more robust learning. Our earlier work acquiring similarity knowledge for the retrieve stage has also used introspective learning to exploit the knowledge already in the case-base [19]. This work is a natural extension, but developed for the more demanding task of acquiring adaptation knowledge.

3 Adaptation Task

A CBR system retrieves cases corresponding to similar problems from its case-base. The adaptation step must recognise differences between the new and retrieved problems, and refine the retrieved solution to reflect these differences, as appropriate. For example, in a house pricing CBR system, a retrieved house with fewer bedrooms should have the retrieved price increased to reflect the additional bedrooms in the new house. Kolodner [20] identifies three types of adaptation:

- *Substitution* replaces values in the retrieved solution with new values appropriate for the new problem (e.g. changing a house price);

- *Transformation* alters the retrieved solution by adding, deleting or replacing parts of the retrieved solution to suit the new problem (e.g. altering steps in a plan); and
- *Special* methods apply specialised heuristic knowledge to repair the retrieved solution, or replay the method used to derive the retrieved solution for the new problem.

For adaptation, the task is to recognise when an adaptation should be applied because the new and retrieved problems are sufficiently different in some relevant way, and to perform some change(s) to the retrieved solution. An adaptation can be considered as a situation/action pair. The *situation* contains the differences between the new and retrieved problems. The *action* captures the update for the retrieved solution: new values for the reused solution (substitution); solution components to be added, deleted or changed (transformation); or more specialised knowledge to achieve the update (special).

This paper discusses adaptation by substitution, but similar techniques are likely to be appropriate for the more complex transformation methods where the structure of the solution is altered in some way. Special methods are the most knowledge intensive adaptation and may be more difficult to capture, particularly for knowledge-light methods, since the update to be performed will be more complex and must be represented somehow for the action part.

3.1 Training Data for Learning Adaptations

The case-base is a useful source of knowledge about the problem-solving domain. It contains a set of cases each comprising a problem, together with its solution. By its nature, the case-base is deemed to be a representative sample of problem-solving in the domain, and so adaptation training data acquired from this source is likely to generate representative adaptation knowledge.

We adopt a leave-one-out approach to generate adaptation examples as illustrated in the top half of Figure 1. One case, consisting of a problem part and solution, is removed from the CBR case-base, and the remaining cases are used to form a smaller, cut-down, version of the case-base. As a result, the extracted case and the most similar cases retrieved from the cut-down case-base, are available to provide data to assemble the adaptation training example, as shown in the lower half of Figure 1. Each adaptation example captures the adaptation situation that the extracted problem and the retrieved case demonstrate and the adaptation action that should be applied. The adaptation situation consists of the attribute values in the problem part of the extracted case, the differences (dissimilarities) between the extracted and retrieved cases for each of these problem part attributes, and the retrieved solution. The adap-

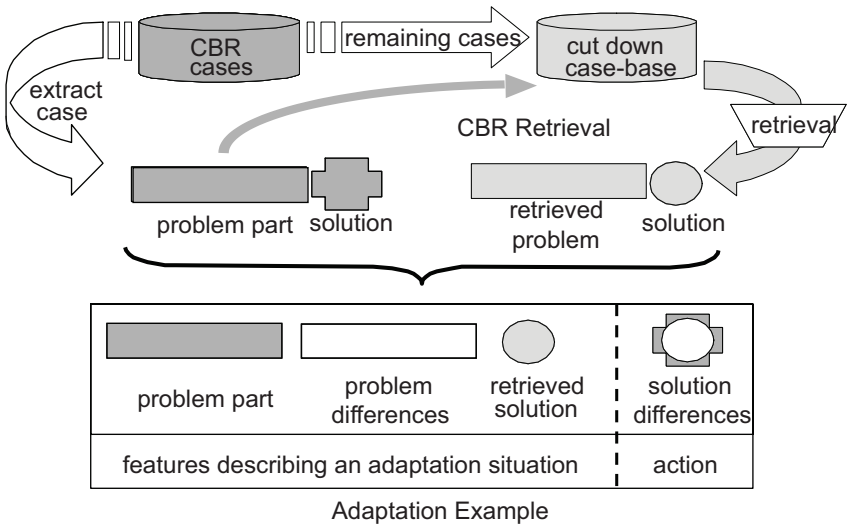

Fig. 1. Assembling Adaptation Training Data from the Case-base

tation action captures the update that is necessary to change the retrieved solution into the solution of the extracted case.

It is important to note that adaptation examples are training data in which actions are recorded together with situations. The adaptations are generalised later by whatever learning algorithm is applied. Since each of the k nearest neighbours retrieved for an extracted case from its cut-down case-base can be the basis of an adaptation example, the set of adaptation examples may be several times larger than the original case-base.

3.2 Adaptation Actions

The adaptation action captures the update that transforms the retrieved solution into the target solution for the extracted case. For numeric solutions, the action part of the adaptation example is a calculation, perhaps to add the numeric difference between the target and retrieved solutions. Adaptation of symbolic solutions is more challenging for several reasons.

- Numeric attributes have a natural way to calculate dissimilarity and this can be used to generate the update to a numeric solution. Symbolic attributes in the solution may not have a similarity defined for them since CBR uses only the similarity of attributes in the problem part.
- The numeric dissimilarity can be used to generate the refined solution. In contrast there may not be a symbolic solution that has the correct dissimilarity to the retrieved solution as determined by the adaptation action.
- There is a natural calculation based on the numeric dissimilarity that can be applied to the retrieved numeric solution to calculate a refined solution, but there is no equivalent for symbolic dissimilarities.

Nevertheless, adaptation data for numeric solutions provides insight into what is desirable for the adaptation of symbolic solutions. Figure 2 gives examples of the initial adaptation actions we propose in this section. For a numeric solution like age, when the target and retrieved solutions are both 40, then the action noted is to add zero, but when the values are 40 and 21 respectively then -19 is noted as the correction to the retrieved solution. The first method developed here for symbolic solutions uses the standard binary similarity (different or same) to inform a coarse adaptation (whether adaptation is needed or not). For a symbolic solution like colour, when the target is 'red', and 'red' is retrieved, then the value 'OK' indicates that no adaptation is needed. When they are different, say a 'red' target and 'yellow' is retrieved, the value '¬OK' indicates that a different solution should be used; e.g. from repeated CBR retrieval. In the second approach, the '¬OK' adaptation is replaced by a naïve multi-class adaptation which nominates the target solution as the refined solution. For the colour example, a retrieved yellow creates an adaptation example where the adaptation action indicates that a new red solution should be used.

Numeric	Target	Retrieved	Adaptation Action
Age	40	40	0
	21	40	-19

Symbolic	Target	Retrieved	Adaptation Action	
Colour	red	red	OK	
	red	yellow	¬OK	*(binary)*
	red	yellow	red	*(multi-class)*

Fig. 2. Translating Solution Differences into Adaptation Actions

4 Tablet Formulation Domain

We evaluate our methods to automatically acquire adaptation knowledge on a CBR system for a demanding design task, chemical formulation for a tablet of a particular drug and dose. CBR has been used successfully for other challenging chemical formulation tasks: Cheetham's FormTool for selecting pigments to produce a given colour [21]; formulation of rubber compounds for tyres [22]. Formulation domains provide good opportunities to improve the problem-solving of retrieve-only CBR using adaptation.

Tablet formulation and the characteristics that make it suitable for this evaluation are introduced in this section. Pharmaceutical companies create a formulation when they develop a tablet to deliver a given dose of a new drug. A

formulation is a recipe for the tablet in which inert compounds, called excipients, are added to the drug to deliver certain physical and chemical properties of the tablet as a whole. A tablet contains five components in addition to the drug: the *filler* provides bulk for compression; the *binder* makes the tablet ingredients hold together; the *disintegrant* allows rapid breakdown after swallowing; the *lubricant* enables the tablet to be ejected from the manufacturing die; and the *surfactant* aids mixing of the ingredients and disintegration. The formulation is the choice for each of filler, binder, disintegrant, lubricant and surfactant, selected from a list of possible excipients, and the quantity of each. The tablet formulation problem is represented by the dose of the drug, together with five physical and 22 chemical properties of the drug. Figure 3 shows the contents of a Tablet Formulation case.

Problem Part	Physical Properties of the Drug					
	Dose	Solubility	Contact Angle	Yield Pressure	YP Fast	Strain-Rate Sensitivity
Type	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric
Range	0–720	0–300	0–110	0–160	0–140	0–130

	Chemical Properties of the Drug				
	Stabilities of Drug with Excipients				
Type	Numeric	Numeric	Numeric	.....	Numeric
Range	0–130	0–130	0–130	.....	0–130

Solution Part	Components				
	Filler	Binder	Disintegrant	Lubricant	Surfactant
Type	Symbolic	Symbolic	Symbolic	Symbolic	Symbolic
#Values	5	5	3	1 [a]	2

	Quantities				
	FillerQ	BinderQ	DisintegrantQ	LubricantQ	SurfactantQ
Type	Numeric	Numeric	Numeric	Numeric	Numeric
Range	50–190	2–12	2–20	1–4	0–1.2

	Tablet Properties	
	tYP	tSRS
Type	Numeric	Numeric
Range	80–550	0–25

[a] Only one lubricant is used so the task reduces to predicting its quantity.

Fig. 3. Contents of a Tablet Formulation Case

Components are chosen to balance the physical properties of the drug; e.g. solubility, hardness as measured by yield pressure, etc. Components must also be chemically stable with the drug. The overall accuracy of tablet formulation

is dominated by the important choice of filler and binder. Fillers provide bulk and so one can increase or decrease the quantity to compensate for physical properties. A different filler may be acceptable in a formulation and so formulation accuracy applies our expert defined similarity threshold for fillers to indicate when formulations should be considered correct. The binder prediction task is particularly demanding because there are six dissimilar binders and small quantities of binder have to provide several physical properties.

Tablet formulation can be described as a component-based design problem because there are a small number of components (five) and only a few interactions among them. The design is decomposable because it can be solved by choosing the more important components first: filler and its quantity, then binder and its quantity, leaving the less demanding disintegrant, lubricant, and surfactant until later. It allows incremental multi-task learning where we use multiple retrieval with increasingly instantiated problems; e.g. the binder is chosen assuming the original physical and chemical properties of the drug plus the filler name and quantity. Further details are available in an early paper on case-based tablet formulation [23].

Adaptation can be decomposed in a similar component-based fashion and this means that substitution adaptation is often suitable. The relative simplicity of this adaptation means it is a good target with which to explore automated acquisition of adaptation knowledge. Formulation is an ideal test-bed for our research because the formulations (solutions) contain both symbolic components and numeric quantities. The organisation as component-based design allows CBR to concentrate on the problem-solving rather than the case representation, and enables the adaptation methods to be structured in a similar component-based fashion. Figure 4 shows the content of an adaptation case.

5 Results from Case-Based Adaptation

Our initial experiments use the adaptation training examples directly in a nearest-neighbour case-based retrieval system, to achieve adaptation within the CBR cycle. Case-based adaptation is illustrated in Figure 5; it is named CBR_A to distinguish it from the main problem-solving CBR system. The top part of the diagram shows the standard CBR retrieve stage suggesting an initial solution for a new problem. The adaptation problem is then assembled from the problem part of the original problem, the differences between the original problem and the retrieved (CBR) problem, and the retrieved (CBR) solution. In the middle of the diagram, the CBR_A system retrieves an adaptation case consisting of a CBR_A problem and solution. At the base of the diagram, the retrieved CBR_A solution (an adaptation action) is applied to refine the retrieved (CBR) solution.

Problem Part of an Adaptation Case

Formulation Problem

	Physical Properties				Chemical Properties		
	Dose	Sol	...	SRS	Stabilities with Excipients		
Type	Numeric	Numeric	...	Numeric	Numeric	...	Numeric
Range	0–720	0–300	...	0–130	0–130	...	0–130

Problem Differences

	Physical Differences				Chemical Differences		
	ΔDose	ΔSol	...	ΔSRS	Stability Differences		
Type	Numeric	Numeric	...	Numeric	Numeric	...	Numeric
Range	0–720	0–300	...	0–130	0–130	...	0–130

Retrieved Formulation

	Components			Quantities		
	Filler	...	Surfactant	FillerQ	...	SurfactantQ
Type	Symbolic	...	Symbolic	Numeric	...	Numeric
Range	5	...	2	50–190	...	0–1.2

Tablet Properties

	Similarity	tYP	tSRS
Type	Numeric	Numeric	Numeric
Range	0–1	80–550	0–25

Solution Part of an Adaptation Case

One of these Adaptation Actions	Component Adaptation Binary	Component Adaptation Multi-Class	Quantity Adaptation
Attribute Type	Symbolic	Symbolic	Numeric
#Values/Range	2	#Components	0–130

Fig. 4. Contents of Adaptation Cases for Tablet Formulation

5.1 Experimental Design

The performance of case-based adaptation is evaluated by comparing the competence of CBR systems with and without adaptation.

- Retrieve-Only has no adaptation and its Reuse simply copies the retrieved solution.
- NN_A includes a simple CBR_A system where a non-optimised retrieval selects the nearest adaptation example to apply.
- CB_A includes a more knowledge-rich CBR_A system where the adaptation cases incorporate some feature selection. Sets of adaptation examples are

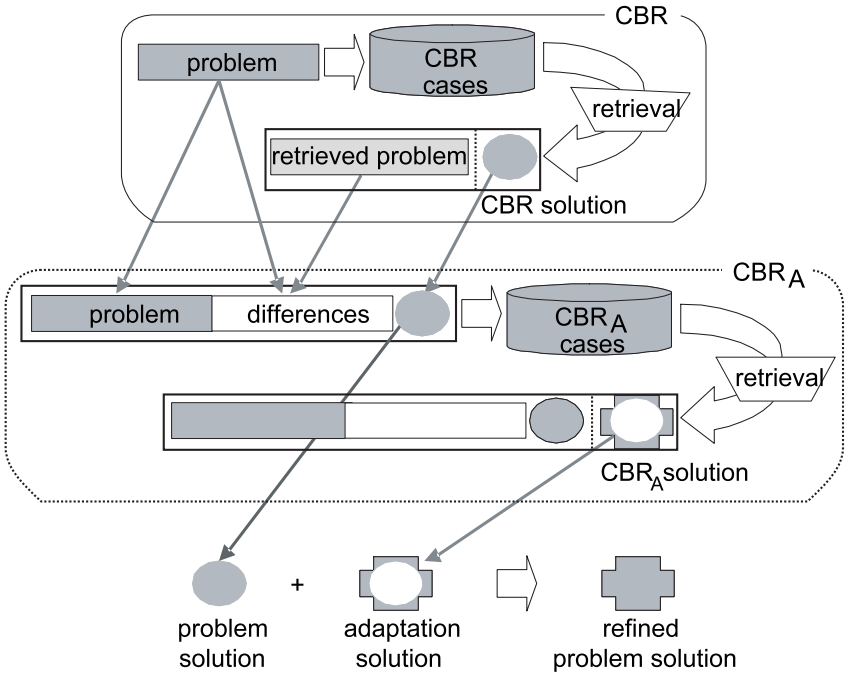

Fig. 5. Case-Based Adaptation

created in which different adaptation problem features are selected. C4.5 [24] is applied to each set of adaptation examples and the sets whose decision trees have the highest predicted accuracy are used to form the case-base for CB_A. Thus CB_A has access only to adaptation cases that have problem feature selections that discriminate the adaptation actions.

The CBR cases and retrieval knowledge for the formulation CBR systems are fixed in the experiments, but the adaptation enabled CBR systems have an additional stage which uses either NN_A or CB_A with the adaptation examples.

Our complete dataset contains 156 tablet formulations: formulations for 39 drugs and 4 doses of each. It is common for tablets with different doses of the same drug to have similar formulations, and so we apply a 'leave N *drug* out' testing, rather than 'leave N out'. Figure 6 shows the 'leave 4 drug out' experiment we used. The formulations for four randomly selected drugs (16 formulations) are extracted to be used as the unseen test set. The 140 formulations corresponding to the remaining 35 drugs become the case-base. The 35-drug case base generates its own adaptation examples using 10-NN retrieval from the 34-drug cut-down case-base in a further 'leave 1 drug out' method as illustrated in Figure 1. The accuracy of the 35-drug case-base, together with its 1400 adaptation examples, is measured for the 16 withheld formulations. This experiment is repeated 25 times with randomly selected test sets consisting of four 4-drug formulations.

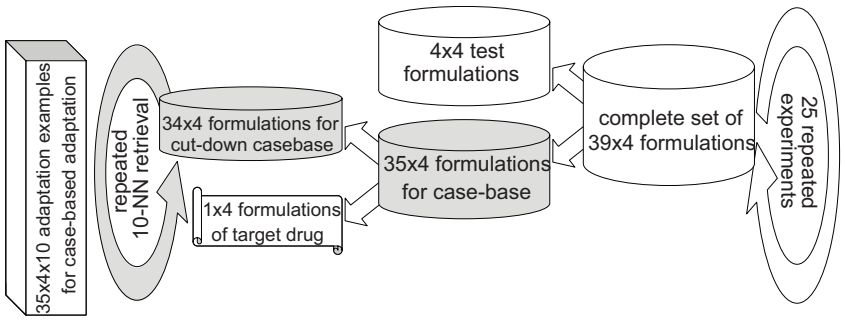

Fig. 6. Dataset Split into Test Formulations and Case-Base

Performance for numeric quantities is evaluated by calculating the root mean-squared error for the test set. Percentage accuracy is used for symbolic components, where a predicted filler/binder is deemed to be correct if it is at least 90% similar to the actual filler/binder. Our domain expert has provided these excipient similarities. This accuracy allows fillers to be correct without matching exactly, but binders are sufficiently dissimilar that this happens to be identical to an exact match.

5.2 Results

Table 1 contains the adaptation results for NN_A and CB_A for the symbolic filler and binder and their numeric quantities. The other tablet components and their quantities are less demanding, and are often solved well using Retrieve-Only CBR [25]. The straw-man Retrieve-Only results are shown in italics and the best results are highlighted in bold. These simple case-based adaptations have not made significant improvements. Only CB_A adaptation for filler quantities has made a modest gain.

Table 1
Average Accuracies/Errors for NN_A and CB_A Adaptation

	Retrieve-Only	NN_A	CB_A	Metric
Filler	***81.1%***	80.2%	80.0%	% Accuracy
Binder	***33.7%***	22.7%	31.8%	
Filler Quantity	*0.0496*	0.0638	**0.0447**	Error
Binder Quantity	*0.0020*	**0.0017**	0.0020	

Adaptation is a refinement stage after retrieval. Therefore the less accurate the Retrieve-Only solutions, then the more opportunity there is to improve the solution. Table 1 shows that, overall, binder retrieval offers much more scope for adaptation, than does filler, and conversely the binder quantity retrieval already has an extremely low error. This effect is also demonstrated across the

25 experiments. For all our evaluations, we also present a scatter plot showing the relative performances of the Retrieve-Only and with-adaptation methods. In each plot Retrieve-Only is presented as a grey line and the experiments have been ordered so that the lowest problem-solving performance Retrieve-Only experiments are to the left of the graphs. The adaptation methods are presented as a scatter plot around this line.

Figure 7 is an error plot for filler and binder quantities. The Retrieve-Only graph drops from left to right because the error decreases as problem-solving performance increases, and effective adaptation occurs when the error after adaptation lies *below* the Retrieve-Only error. These plots demonstrate a repeated pattern: at the left-hand side, where Retrieve-Only is relatively poor, adaptation is more likely to improve the predictions. Both NN_A ($\bigcirc$) and CB_A ($\triangle$) have made improvements when the Retrieve-Only errors are quite high. The optimised CB_A has continued to gain even when Retrieve-Only is performing well, whereas NN_A has made some good improvements but also dramatic deteriorations. CB_A's case and feature selection was designed to reduce over-fitting and this is demonstrated in these initial results. Retrieve-Only binder quantities are already very accurate; the highest binder error is 5 times smaller than the lowest filler error. Therefore our adaptation knowledge needs to be better at predicting when the retrieved quantity is OK.

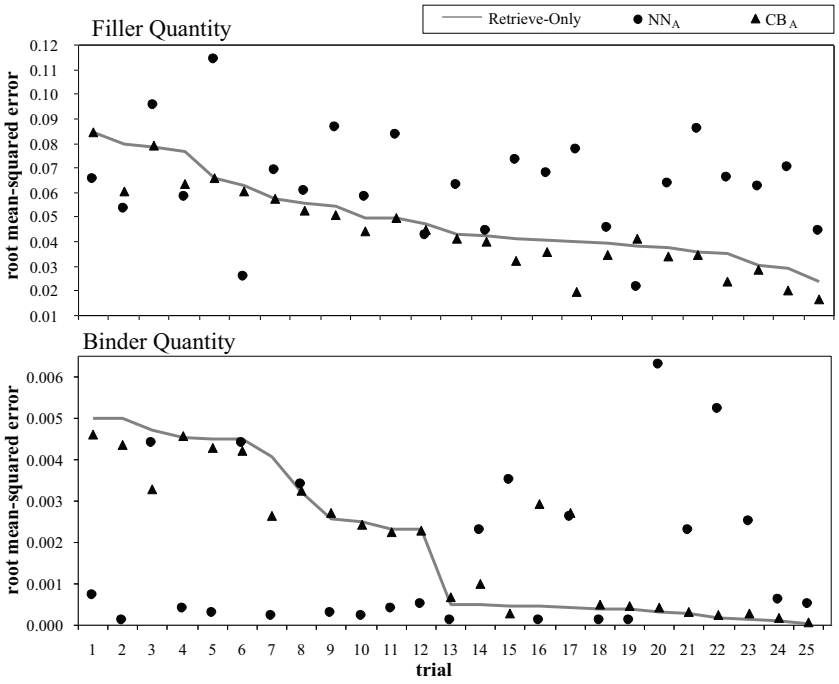

Fig. 7. Quantity Predictions with NN_A and CB_A Knowledge

Figure 8 shows the corresponding NN_A, CB_A and Retrieve-Only accuracy plots for filler and binder predictions. In accuracy plots, the Retrieve-Only graph rises from left to right because the accuracy rises as the problem-solving performance increases, and adaptations *above* the Retrieve-Only accuracy are sought. For these experiments CB_A predicts when repeated retrieval is necessary according to the binary OK/¬OK decision [25], and NN_A predicts a new excipient according to the multi-class adaptation action. For both filler and binder, CB_A adaptation has had little effect over Retrieve-Only accuracy and the few gains are counter-balanced by losses. Nearest neighbour retrieval has not been effective for filler and binder adaptations and more effective adaptation mechanisms for these symbolic predictions are explored in the rest of this paper.

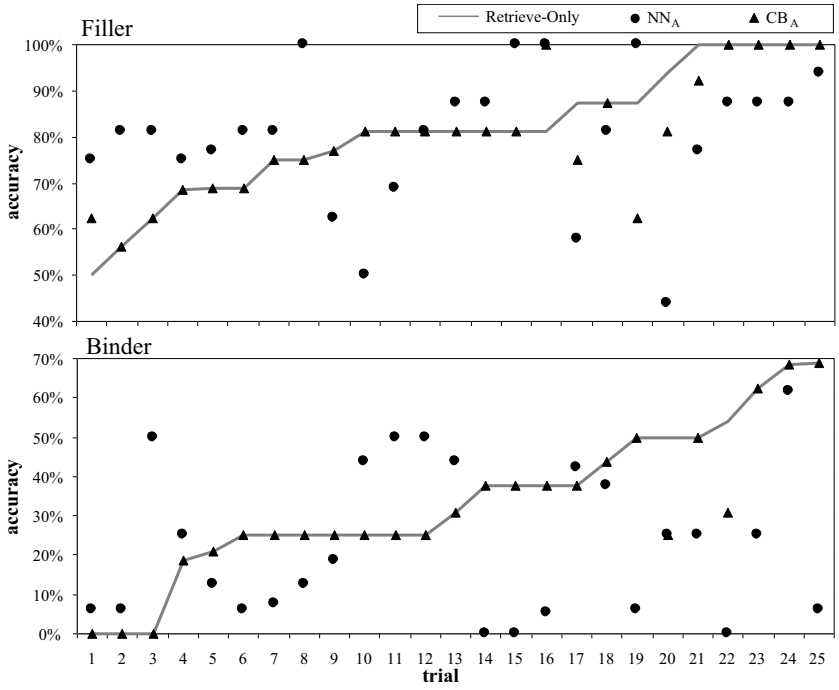

Fig. 8. Filler and Binder Predictions with NN_A and CB_A Knowledge

6 Learning Generalised Adaptation Knowledge

Nearest neighbour adaptation in NN_A and CB_A uses adaptation training examples directly as the knowledge source, and generalisation to unseen adaptation problems is achieved using k-NN retrieval rather than explicit generalised knowledge. Having extracted adaptation training examples from the case-base, we now explore other machine learning algorithms with which to learn explicit adaptation knowledge. The CBR_A system in Figure 5 is replaced with the knowledge learned from the same adaptation examples, and the same adaptation problem is solved by applying this learned adaptation knowledge

instead of CBR_A. We explored a model tree learning algorithm for adapting numeric solutions, and a decision tree and a rule learning algorithm for adapting symbolic solutions, each to be applied to the adaptation training data that has been assembled from the original case-base.

6.1 Decision Tree Learning

C4.5 [24] was applied to learn a decision tree to adapt symbolic solutions. The adaptation examples with multi-class adaptations were used as training data. The decision nodes are chosen from the attributes describing the adaptation situation; e.g. the formulation problem attributes, the problem differences, and the retrieved formulation, in the top part of Figure 4. The decision tree nodes are those that have the highest information gain when predicting the adaptation actions; e.g. the new filler. The leaf nodes predict the updated formulation component using a majority vote of the class attribute in its adaptation examples. Figure 9 shows a fragment of a sample C4.5 decision tree for filler adaptation. The uppercase internal nodes are stabilities of the drug with the named excipient or a stability difference. Other internal nodes are physical properties, dose, retrieved solution, similarity of retrieved problem etc. The leaf nodes name the new filler.

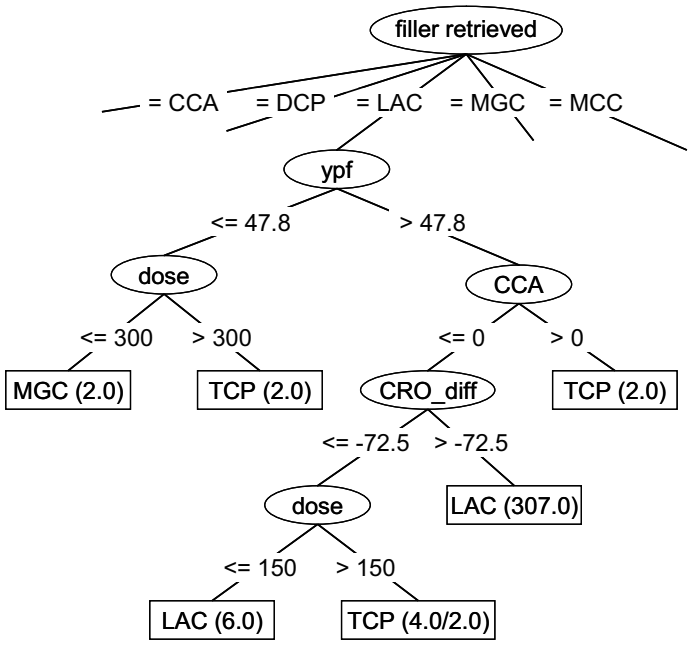

Fig. 9. Fragment of C4.5 Decision Tree for Filler Adaptation

The algorithm moves in a *general-to-specific* direction in which the nodes lower in the tree contain more specific partitions of adaptation examples. Boosting is known to improve the performance of learning algorithms particularly with tasks that exhibit varying degrees of difficulty [26]. Since this is typical of de-

sign tasks, and the adaptation of initial designs, we used a boosted version of C4.5 [27]. Boosting iteratively generates rule-set classifiers, with each learner biased by the incorrectly adapted examples in the previous iteration. Training examples are initially weighted equally, but when a training example is adapted correctly by the classifier from one iteration, its weight is reduced by an error-based factor. Thus incorrectly adapted examples have increased importance and the learner tries harder to correctly adapt these difficult training examples in the next classifier. When learning is completed, the set of rule-sets apply a weighted vote to adapt unseen examples.

We used a decision tree learning approach similar to C4.5 for learning adaptations for numeric solutions. The M5′ algorithm induces a model tree where the leaf nodes contain a linear expression to predict the numeric solution [28]. Figure 10 shows a fragment of a sample M5′ model tree for filler quantity adaptation. The linear equation calculating the quantity difference to be applied for one of the leaf nodes LM1 is also given. Adapting quantities in tablet formulation is not as variable as adapting the symbolic components and so boosting was not explored.

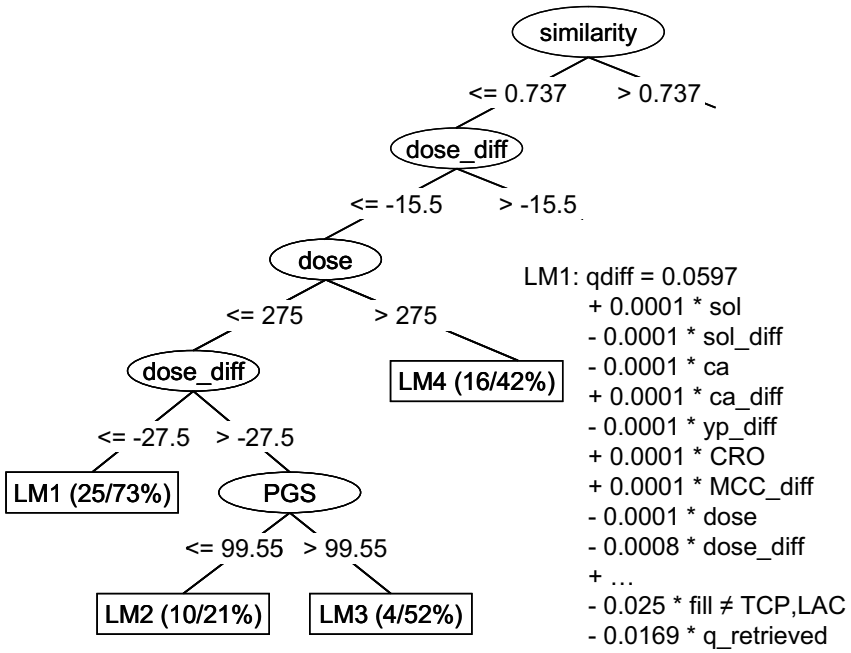

Fig. 10. Fragment of M5′ Model Tree for Filler Quantity Adaptation

6.2 Bottom-Up Learning

We also explored a hybrid classification algorithm that combines nearest neighbour retrieval with rule induction. Domingos' Rise algorithm grows rules around clusters of examples with the same class [29]. Rise's rules are generalised by broadening numeric intervals and dropping conditions. Therefore

the conditions in a RISE rule use those attributes that may appear in the internal nodes of the decision tree, and the conclusion chooses the new component. RISE's *specific-to-general* direction of search provides a contrasting alternative to C4.5's search, and generalises the local search of 1-NN. We developed and applied a boosted version of RISE, the details of which can be found in [30].

6.3 Adaptation Results for Quantities with Model Tree Learning

Table 2 summarises the average root mean-squared error for filler and binder quantities for Retrieve-Only, the two nearest neighbour adaptations NN_A and CB_A in section 5.2, and the M5′ adaptation knowledge. M5′ has produced significant adaptation gains for filler quantities over the other systems. However improving the binder quantities remains a serious problem with M5′ making substantially worse predictions than Retrieve-Only!

Table 2
Average Errors for Generalised Numeric Adaptation Knowledge

Quantity Error	Retrieve-Only	NN_A	CB_A	M5′
Filler	*0.0496*	0.0638	0.0447	**0.0250**
Binder	*0.0020*	**0.0017**	0.0020	0.0112

Figure 11 compares the adaptation gains from M5′ (□) with those from CB_A (△). The gain for filler quantities is clear. For binder quantities, a single very extreme M5′ error (0.2315 not shown) in trial 4 spoils an otherwise improved performance compared to CB_A for higher retrieve errors. There is nothing unusual about the tablets in trial 4, and it is hard to explain why the M5′ calculations here are so wrong. None of our adaptations have made an impact on the right-hand side of the binder plots. Our conclusion is that the retrieval of binder quantities is already effective, and that adaptation should concentrate on recognising situations when no adaptation should be performed.

6.4 Results for Rule-based Adaptation of Symbolic Components

Table 3 summarises the average accuracies for filler and binder prediction using the rule-based adaptation learned by boosted versions of C4.5 and RISE, applied to adaptation examples that capture the multi-class adaptation. The table also contains the earlier CBR_A adaptations and Retrieve-Only predictions. The generalised adaptation rules have provided significant improvements over Retrieve-Only predictions. However successful filler adaptation is achieved only by C4.5 knowledge, and binder prediction is improved only by RISE. The adaptation space for each type of tablet component seems to be different: filler

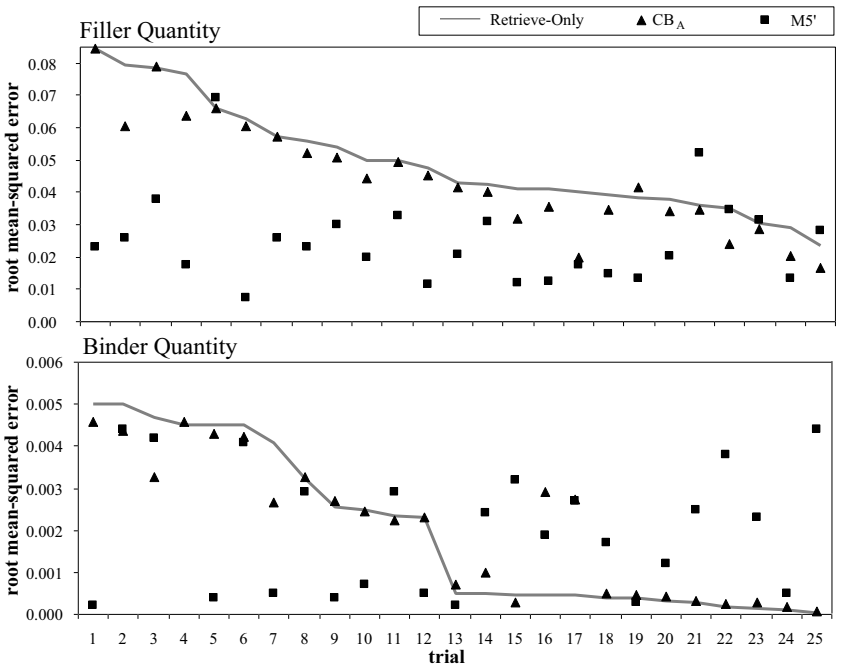

Fig. 11. Quantity Predictions with M5′ Knowledge

adaptation suits hyper-plane partitioning; whereas binder adaptation fits a generalised nearest neighbour. This may be explained by binders having to achieve a difficult balancing act with physical properties of the drug, whereas the increased volume of filler allows a trade-off, but it is more important to satisfy strict stability thresholds for fillers.

Table 3
Average Accuracies/Errors for Generalised Adaptation Knowledge

Accuracy	Retrieve-Only	NN_A	CB_A	C4.5	Rise
Filler	*81.1%*	80.2%	80.0%	**83.9%**	79.6%
Binder	*33.7%*	22.7%	31.8%	26.6%	**44.9%**

Figure 12 contains the accuracy plots for C4.5 (△) and Rise (□) knowledge. The nearest neighbour adaptations are not superimposed because their accuracies were very close to Retrieve-Only. For filler, both C4.5 and Rise tend to improve Retrieve-Only predictions to a similar level until Retrieve-Only accuracy is above 80%. The average filler accuracy for Rise is lowered by several very poor results particularly for drug selections with a high retrieval accuracy. Binder Retrieve-Only accuracy is low in all trials and Rise is effective in almost all trials. The improvement with Rise adaptation is quite substantial but it is important to note that the average binder accuracy is still less than 45%!

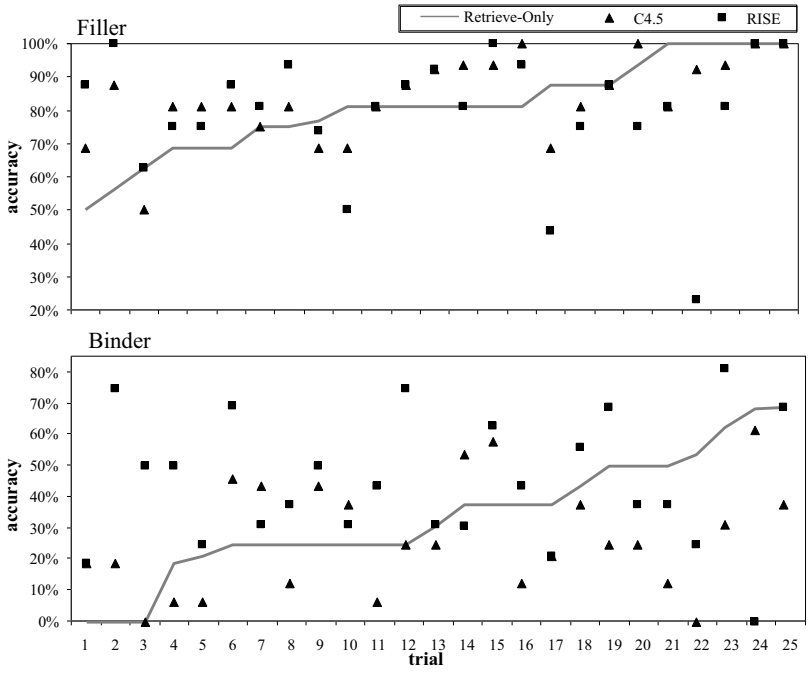

Fig. 12. Filler and Binder Predictions with C4.5 and RISE Knowledge

7 Adaptation Ensembles for Symbolic Solutions

The adaptation of symbolic components in tablet formulation remains a challenge. However, the adaptations considered so far do not capture the spirit of CBR adaptation. The OK/¬OK prediction with repeated retrieval is not really adaptation, but repeated problem-solving from scratch. On the other hand, adaptation that predicts a different filler or binder is a new choice, rather than a refinement of the retrieved solution. Instead, adaptation of a symbolic solution should change its value in a way that achieves some desired effect.

7.1 Property-based Adaptations

This motivation led us to consider *properties* related to a symbolic solution. In tablet formulation the solutions (e.g. fillers) have physical properties; e.g. solubility, yield pressure, etc. These properties will be used to drive the adaptation of symbolic solutions. In tablet formulation there happens to be corresponding physical properties of the drug (problem), the components (solution), and the resulting tablet. However, property-based adaptation in general does not depend on such a correspondence, but instead relies on *some* properties of the solution that will guide adaptation.

Table 4 shows filler examples of the binary OK/¬OK and multi-class adaptations from section 3.2, together with a new property-based adaptation, based

on the yield pressure of the filler. Notice that in the last two lines of the table the numeric difference in yield pressure is not used to suggest fillers whose yield pressures are exactly 76 higher (or 14 lower) than that of the retrieved filler; such a filler is unlikely to exist. Instead, the value INC (or DEC) indicates that the new filler should have an increased (or decreased) yield pressure.

Table 4
Property-Based Adaptation Actions

Property-of	Attribute	Target	Retrieved	Adaptation Action	Type
	Filler	MGC	MGC	OK	
		MGC	DCP	¬OK	Binary
		MGC	DCP	MGC	Multi-class
Yield Pressure	Filler	166	166	OK	
		166	90	INC	Property-based
		166	180	DEC	Property-based

To illustrate the use of property-based adaptation, suppose situations involving solubility appear in yield pressure adaptation examples with DEC as the adaptation action, then a rule that might be learned could be:

"*If the Drug Solubility > 10 and the Retrieved Filler Solubility > 8 then select other fillers with* DEC*reased Yield Pressure*"

There are many possible property-based adaptations. To restrict these refinements to informative properties, we use an approach similar to McSherry's [31] in which we select the n most discriminatory properties from a C4.5 decision tree that predicts the filler name from all the filler properties. A set of adaptation examples is collected for each of the selected properties and each set is used to build a classifier for that property-based adaptation. We use only these properties as the basis for refinements.

7.2 Ensemble Results for Filler and Binder

We have experimented with ensembles of 1-NN, C4.5 and RISE property-based classifiers. Figure 13 shows an ensemble of experts where each expert$_i$ is devoted to adapting the component with respect to a particular property. The knowledge in expert$_i$ is learned from the adaptation examples corresponding to that property. Each expert$_i$ predicts OK, INC or DEC for its property and hence nominates new components from this property-based view. A weighted vote selects the new solution from the ensemble-based adaptation.

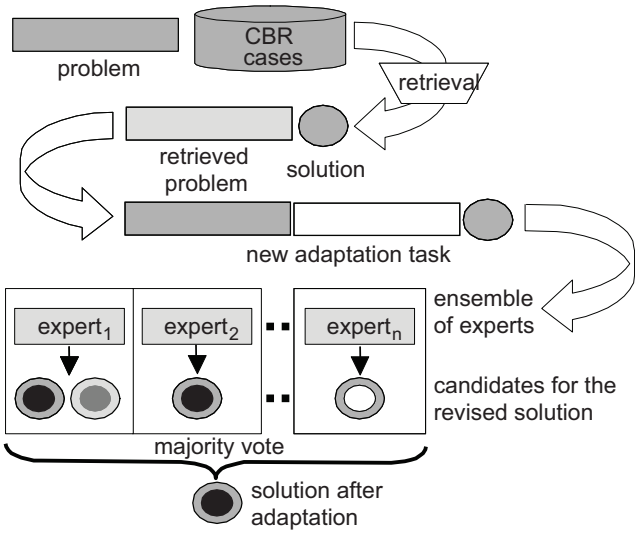

Fig. 13. Ensemble-Based Adaptation

Table 5 shows the average accuracy for filler and binder of an ensemble of property-based experts each learned from a set of property-based adaptation training examples. The results for ensembles of 1-NN, C4.5 and RISE property-based experts are reported. Those for Retrieve-Only, the earlier CBR_A adaptations (binary CB_A and multi-class NN_A), and the most recent multi-class rule-based adaptations (C4.5 and RISE) are also included.

Table 5
Average Accuracies for Adaptation Ensembles

Accuracy	Retrieve-Only		1-NN	CB_A	C4.5	RISE
Filler	*81.1%*	Single	80.2%	80.0%	83.9%	79.6%
		Ensemble	77.2%		**85.7%**	78.0%
Binder	*33.7%*	Single	22.7%	31.8%	26.6%	44.9%
		Ensemble	**74.1%**		64.7%	72.7%

For filler, the C4.5 ensemble method has made a further improvement on the gain already made with the individual C4.5 rule-based method. The instance-driven ensemble methods, like their individual counterparts, make no improvement over Retrieve-Only for filler; in fact the property-based ensembles are worse than NN_A and RISE. Fillers are used to balance physical properties of the drug but being the largest component of a tablet it is very important that the stability is high enough. Thus adaptation may be driven by this stability requirement and this would suit C4.5's hyper-plane approach.

Binder prediction is the big success. Its continued low accuracy until now was one of the motivating factors to try a property-based ensemble. The improvement has been spectacular, with accuracy more than doubled compared to Retrieve-Only. We believe that this success is due to the nature of the binder

prediction task. There are several independent constraints on binders, and balancing them is what an ensemble of independent property-based adaptations can achieve through negotiation and consensus. We regard these results as a major triumph because binder prediction is a very important subtask of tablet formulation. Our results for binder up until now had been very disappointing despite retrieval optimisation and adaptation efforts in several directions.

Figure 14 shows the scatter plots for ensemble-based adaptation. The results for earlier adaptation methods are not superimposed to improve clarity because the ensemble gains are much bigger than for previous adaptations. Looking at individual filler experiments, the success of ensemble based methods is still variable. However, there is some agreement that when one adaptation ensemble does well then the others do also, and vice versa. The C4.5 ensemble may be best overall but it is nevertheless worse than the retrieved filler for nine trials. But in each of these, the other ensembles are also poor. There is still a tendency not to recognise when the retrieved filler is good enough and adaptation is unnecessary.

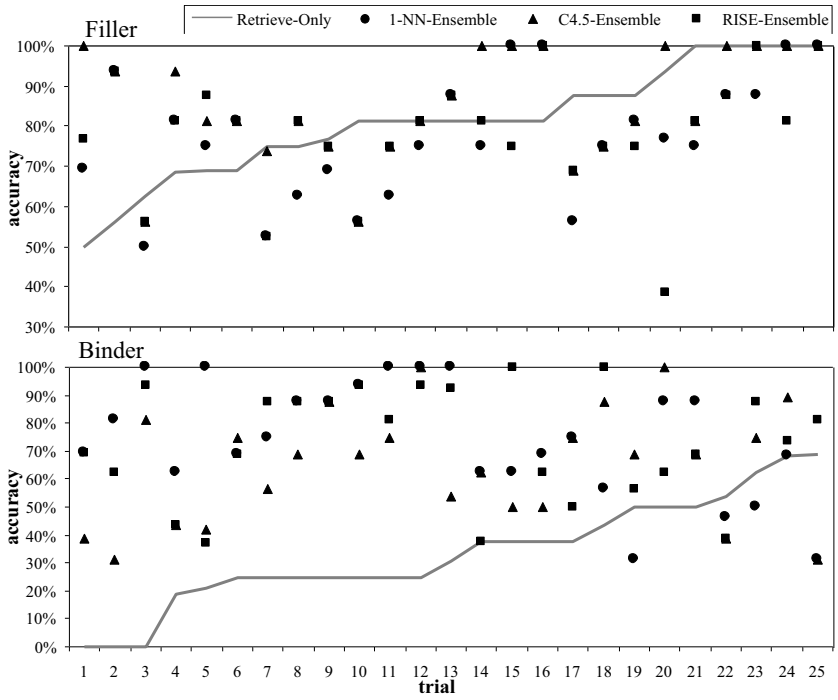

Fig. 14. Filler and Binder Predictions with Ensemble-based Adaptation

The binder graphs show the general dramatic improvement in all experiments for all the ensemble-based adaptations. Although the 1-NN ensemble has an average accuracy above that of the RISE and C4.5 ensembles, it has the highest number of trials where the adapted accuracy is worse than Retrieve-Only (trials 19, 22, 23, 25). The C4.5 ensemble is worse for only two of these (22 and 25), and trial 22 is the only poorer accuracy for the RISE ensemble. The retrieved binders in trial 22 must be quite hard to adapt because all the other,

non-ensemble, adaptations also make its accuracy worse. The 1-NN, C4.5 and RISE property-based binder ensembles each give improvement over the corresponding individual classifier in all but one, two and four trials, respectively. It is noticeable from the high trial numbers that the better retrieval is, the more difficult it is for adaptation to improve.

8 Conclusions

Retrieve-Only CBR has been improved by learning adaptation knowledge in the form of cases or rules. The learning has been introspective because adaptation knowledge is acquired by exploiting the original knowledge in the problem-solving case-base. The adaptation knowledge has been applied to the retrieved solution as part of the reuse stage of CBR. Predicting filler, binder, and their quantities in tablet formulation are challenging tasks with different influences, and so they provide varying demands on adaptation. Nevertheless, adaptation has improved the prediction of Filler and Binder for tablet formulation and has reduced the error of the predicted quantities of these components. Not *all* adaptation methods have been successful for *all* tablet formulation subtasks but at least one method has improved each of the tasks. This demonstrates the usefulness of the knowledge captured in the adaptation examples and indicates the importance of choosing the right learning algorithm to gain the appropriate generalisation from this training data.

The naturalness of numeric differences to update retrieved solutions is hard to replicate directly for symbolic CBR solutions. Our novel, property-based, adaptation chooses a new component based on the need to increase or decrease some property of the retrieved solution. Thus it captures the purpose behind the refinement of the solution. The approach is generally applicable, unless there are no properties of the solution to relate different solutions; in this case there can be no reason to choose the new value for the solution. Adaptation can be based on the requirements of a single property. However, in our domain, there are several competing properties of the filler or binder, and so property-based ensembles have been effective in exploiting the trade-offs. The effect of adaptation was particularly dramatic for binder, where each ensemble method roughly doubled the accuracy of previous adaptation methods.

Several adaptation methods have adversely affected the basic Retrieve-Only CBR system. Binder quantity retrievals are already very accurate and so reducing the error further is difficult; but also unnecessary since the error is already well within the manufacturing tolerances. However, this problem has highlighted that adaptation needs to predict no change more readily. Our adaptation examples for binder quantities never capture pairs of exactly matched formulation and retrieved problems and their resulting null adapta-

tions, and so knowledge that predicts no adaptation for close problem matches is hard to learn. One approach would represent a small quantity change as zero in an adaptation example and thus avoid over-fitting small changes during learning. A similar problem happens for multi-class adaptation for components, where the new component is named explicitly. If we were to add the class "SAME" to the existing classes then we would explicitly capture circumstances when no change was needed. Capturing "no change" more effectively in adaptation examples for both numeric and symbolic solutions would enable the generalisation of situations when no adaptation should be used.

The dependency between the adaptation and problem-solving cases raises the issue that any bias in the problem solving cases will be replicated in the adaptation cases. Although the complete dataset contains 156 formulations, many are very similar because they differ only in drug dose. Viewed instead as a dataset of 39 drugs, the sparseness of formulation data could mean that the adaptation examples are limited in variety. Also, the 10-NN retrieval to generate adaptation data could suffer from adaptation examples based on distant formulations. The 4-dose groupings of formulations for each drug will also create clusters of similar 4-dose adaptations. Therefore the lack of tablet formulation data, together with the case and adaptation clusterings, is likely to have made adaptation for this dataset particularly difficult.

We have developed and evaluated the different approaches to learning adaptation examples on a single problem solving domain, tablet formulation. However, the techniques are applicable in general to CBR systems. Formulation solutions contain both symbolic and numeric parts and so adaptation of these different types is demonstrated. The distinctive problem-solving traits of the different formulation tasks put different demands on adaptation. It is important to realise that the choice of learning method depends on features of both the problem-solving and adaptation tasks. We have seen that fillers are best adapted using C4.5 because this recognises the stability threshold, but that binders require the trade-off in binder properties offered by an ensemble.

Further work is needed to understand the features of the adaptation space for different domains. Firstly, an understanding of adaptation is needed so that adaptation tasks can be matched with the appropriate learning algorithm; e.g. local search or hyper-plane partitioning. Secondly, an appropriate representation of possible adaptation problems would be helpful, particularly for those problems where small differences result in no adaptation. Finally a greater understanding of learning suitable adaptations would allow multi-algorithm ensembles to be created where the individual algorithm for the ensemble would be specifically chosen for the particular property it refined. Understanding the learning environment of adaptation better is an important precursor to improving the automated acquisition of adaptation knowledge.

Acknowledgments

The authors thank Dietrich Wettschereck and Stewart Massie for feedback that helped to improve and correct this paper. Jacek Jarmulak also contributed to the earlier retrieval knowledge phase of this research, and its underpinning of the project's subsequent approach to learning adaptation knowledge.

The first two authors would like to acknowledge EPSRC for its support of this project (GR/L98015) and the two industrial collaborators on this research, AstraZeneca [1] and ISoft [2], for their contributions of data, software and expertise.

References

[1] B. Smyth, E. McKenna, Competence models and the maintenance problem, Computational Intelligence 17 (2) (2001) 235–249.

[2] A. Aamodt, E. Plaza, Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, AICOM 7 (1) (1994) 39–59.

[3] R. López de Mántaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt, I. Watson, Retrieval, reuse, revision, and retention in case-based reasoning, Knowledge Engineering Review 20 (3) (2005) 215–240.

[4] M. M. Richter, Introduction, in: M. Lenz, B. Bartsch-Spoerl, H.-D. Burkhard, S. Wess (Eds.), Case-Based Reasoning Technology: From Foundations to Applications, LNAI 1400, Springer-Verlag, 1998, pp. 1–16.

[5] D. Wettschereck, D. W. Aha, T. Mohri, A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, Artificial Intelligence Review 11 (1997) 273–314.

[6] R. Kohavi, P. Langley, Y. Yun, The utility of feature weighting in nearest-neighbor algorithms, Poster Papers: 9th European Conference on Machine Learning (1997).
URL `http://robotics.stanford.edu/~ronnyk/diet.ps.Z`

[7] D. B. Skalak, Prototype and feature selection by sampling and random mutation hill-climbing algorithms, in: S. Wrobel (Ed.), Proceedings of the 11th International Conference on Machine Learning, Morgan Kaufmann, San Mateo, CA, 1994, pp. 293–301.

[1] www.astrazeneca.com

[2] www.alice-soft.com/html/about.htm

[8] A. K. Goel, S. Craw, Design, innovation and case-based reasoning, Knowledge Engineering Review 20 (3) (2005) 271–276.

[9] A. Gómez de Silva Garza, M. L. Maher, A Process Model for Evolutionary Design Case Adaptation, in: J. S. Gero (Ed.), Proceedings of the Artificial Intelligence in Design Conference, Kluwer Academic Publishers, 2000, pp. 393–412.

[10] K. J. Hammond, Explaining and repairing plans that fail, Artificial Intelligence 45 (1–2) (1990) 173–228.

[11] M. Veloso, J. Carbonell, Case-based reasoning in Prodigy, in: R. Michalski, G. Tecuci (Eds.), Machine Learning: A Multistrategy Approach Volume IV, Morgan Kaufmann, San Francisco, CA, 1994, pp. 523–548.

[12] D. B. Leake, A. Kinley, D. Wilson, Acquiring case adaptation knowledge: A hybrid approach, in: W. J. Clancey, D. Weld (Eds.), Proceedings of the 13th National Conference on Artificial Intelligence, AAAI Press, 1996, pp. 684–689.

[13] K. Hanney, M. T. Keane, The adaptation knowledge bottleneck: How to ease it by learning from cases, in: D. B. Leake, E. Plaza (Eds.), Proceedings of the 2nd International Conference on Case-Based Reasoning, LNAI 1226, Springer-Verlag, 1997, pp. 359–370.

[14] J. Kolodner, Improving human decision making through case-based decision aiding, AI Magazine 12 (2) (1991) 52–68.

[15] B. Smyth, P. Cunningham, Complexity of adaptation in real-world case-based reasoning systems, in: Proceedings of 6th Irish Conference on Artificial Intelligence & Cognitive Science, 1993, pp. 228–240.

[16] D. Wettschereck, D. W. Aha (Eds.), Proceedings of the ECML-97 Workshop on Case-Based Learning: Beyond Classification of Feature Vectors, Prague, Czech Republic, 1997.

[17] D. McSherry, An adaptation heuristic for case-based estimation, in: B. Smyth, P. Cunningham (Eds.), Proceedings of the 4th European Workshop on Case-Based Reasoning, LNCS 1488, Springer-Verlag, 1998, pp. 184–195.

[18] W. Wilke, I. Vollrath, K.-D. Althoff, R. Bergmann, A framework for learning adaptation knowledge based on knowledge light approaches, in: Proceedings of the 5th German Workshop on Case-Based Reasoning, 1997, pp. 235–242.

[19] S. Craw, J. Jarmulak, R. Rowe, Maintaining retrieval knowledge in a case-based reasoning system, Computational Intelligence 17 (2) (2001) 346–363.

[20] J. L. Kolodner, Case-Based Reasoning, Morgan Kaufmann, San Mateo, CA, 1993.

[21] W. Cheetham, Tenth anniversary of the plastics color formulation tool, in: Proceedings of the 16th Conference on Innovative Applications of Artificial Intelligence, AAAI Press, 2004, pp. 770–776.

[22] S. Bandini, S. Manzoni, A support system based on CBR for the design of rubber compounds in motor racing, in: E. Blanzieri, L. Portinale (Eds.), Proceedings of the 5th European Workshop on Case-Based Reasoning, LNAI 1898, Springer-Verlag, 2000, pp. 348–357.

[23] S. Craw, N. Wiratunga, R. Rowe, Case-based design for tablet formulation, in: B. Smyth and P. Cunningham (Eds.), Proceedings of the 4th European Workshop on Case-Based Reasoning, LNAI 1488, Springer-Verlag, 1998, pp. 358–369.

[24] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.

[25] J. Jarmulak, S. Craw, R. Rowe, Using case-base data to learn adaptation knowledge for design, in: B. Nebel (Ed.), Proceedings of the 17th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 2001, pp. 1011–1016.

[26] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: L. Saitta (Ed.), Machine Learning: Proceedings of the 13th International Conference, Morgan Kaufmann, 1996, pp. 148–156.

[27] J. R. Quinlan, Bagging, boosting, and C4.5, in: W. J. Clancey, D. Weld (Eds.), Proceedings of the 13th National Conference on Artificial Intelligence, AAAI Press, 1996, pp. 725–730.

[28] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, San Francisco, 2000.

[29] P. Domingos, Unifying instance-based and rule-based induction, Machine Learning 24 (1996) 141–168.

[30] N. Wiratunga, S. Craw, R. Rowe, Learning to adapt for case-based design, in: S. Craw, A. Preece (Eds.), Proceedings of the 6th European Conference on Case-Based Reasoning, LNCS 2416, Springer-Verlag, 2002, pp. 423–437.

[31] D. McSherry, Minimising dialog length in interactive case-based reasoning, in: B. Nebel (Ed.), Proceedings of the 17th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 2001, pp. 993–998.