

Using Case-Base Data to Learn Adaptation Knowledge for Design*

Jacek Jarmulak and Susan Crow

The Robert Gordon University
Aberdeen, AB25 1HG, Scotland, UK
s.craw@scms.rgu.ac.uk

Ray Rowe

AstraZeneca, Silk Road Business Park
Macclesfield SK10 2NA, UK

Abstract

One advantage of Case-Based Reasoning (CBR) is the relative ease of constructing and maintaining CBR systems, especially as a number of commercial CBR tools are available. However, there are areas of CBR that current tools have not yet addressed. One of these is easing or automating the acquisition of adaptation knowledge. Since tasks like design or planning typically require a significant amount of adaptation, CBR systems for these tasks still do not fully benefit from CBR's promise of reducing the development effort. To address this, we have developed several "knowledge-light" methods for learning adaptation knowledge from the cases in the case-base. These methods perform substitutional adaptation, for both nominal and numerical values, and are suitable for decomposable design problems, in particular formulation and configuration. Tests performed on a tablet formulation domain show promising results. The automatic adaptation methods we present can easily be incorporated in general-purpose CBR tools, thus further contributing to reducing the cost of CBR systems.

1 Introduction

The popularity of Case-Based Reasoning (CBR) can be attributed to a number of factors. One of these is CBR's ability to cope with ill-defined problem domains by basing solutions to new problems on solutions that worked in the past. Given a new problem, a CBR system retrieves a set of similar problems from a case-base of previously solved problems. The retrieved problems provide ballpark solutions to the new problem; if needed, they can be adapted to fit the new problem better. This final solution can be reviewed and stored in the case-base to *improve* future performance if appropriate. This ability to learn is an important advantage of CBR.

A further advantage of CBR is the relative ease of constructing and maintaining CBR systems, especially as a number of commercial CBR tools are available. However, this advantage applies mainly to CBR systems *relying* on retrieval for finding solutions, e.g., many systems for classification and

estimation tasks. This is because current CBR tools concentrate only on certain aspects of building a CBR system: automating construction of the case-base, defining retrieval, and implementing the user interface. Although the tools usually also provide some basic facilities for coding adaptation rules, they do not provide means of automating the task of acquiring adaptation knowledge. This means that CBR systems for more complex tasks, like planning or design, can be developed relatively easily only if their need for automatic adaptation can be eliminated (or reduced) by either having a large number of cases or by using the system in a decision-supporting role where adaptation is performed by an expert.

Recent developments in CBR tools seem to have concentrated on increasing their possible field of application: improving the way they interface with data-base systems, facilitating their deployment in web environments, increasing portability, and improving case representation. However, so far, commercial systems provide few facilities for automating the process of constructing competent case-bases, optimising retrieval with the goal of retrieving most useful cases, and, in particular, automating the acquisition of adaptation knowledge. These issues are now being addressed in recent research: Smyth and McKenna [1999] aim to build compact competent case-bases, Jarmulak *et al.* [2000] introduce ways of automatically optimising retrieval, and Wilke *et al.* [1997] consider knowledge-light approaches for learning adaptation suitable for incorporation into CBR tools.

In this paper we present a selection of generic methods suitable for automating the acquisition of adaptation knowledge for configuration and formulation tasks, a subclass of design tasks. We begin by placing our work in the context of related research in CBR. Then, we describe the methods that we have implemented for automated acquisition of adaptation knowledge from cases in the case-base, and show how the acquired knowledge is used in adaptation. After that, we present our test domain and the results of our experiments, before drawing conclusions and discussing further research.

2 Context & Related Work

We are interested in automating the acquisition of adaptation knowledge for case-based design, in particular for configuration or formulation tasks, where the final solution consists of a more-or-less constant number of components with relatively limited interactions between them. The solution is

*This work is supported by EPSRC grant GR/L98015.

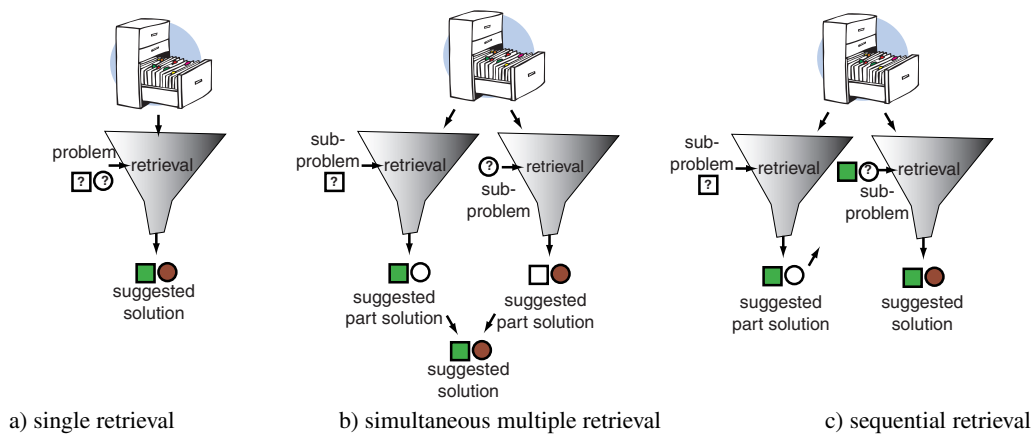


Figure 1: Solving a decomposable configuration problem.

usually determined by finding the correct “fillers” for components, as well as deciding the values of “filler” parameters. Examples of this type of task are: tablet formulation [Craw *et al.*, 1998], formulation of rubber compounds for tires [Bandini and Manzoni, 2000], and PC configuration [Stahl and Bergmann, 2000].

In general, finding a solution for a design problem is made more tractable if the task can be decomposed. Figure 1 contrasts a) a single step retrieval with two multi-step retrievals where, either b) multiple retrievals are merged, or c) the solution retrieved in one cycle informs the retrieval in the next cycle. Decomposition has special advantages for CBR design, because it allows the use of multiple cases. This results in better problem space coverage, and is important for the sparse case-bases typical of CBR design systems. This is demonstrated in the CBR systems *Déjà Vu* [Smyth and Cunningham, 1992] and *EADOCS* [Netten and Vingerhoeds, 1996]. Configuration and formulation tasks are in particular suitable for solving by decomposition; e.g., our *CBRTFS* uses decomposition for tablet formulation [Craw *et al.*, 1998]. *CBRTFS* makes use of the fact that the interaction between solution components is not strong, and that there is a difference in the degree to which the choice of the different components is subject to constraints. Thus it performs a sequential retrieval/adaptation (Figure 1c), proceeding from the most constrained to the least constrained components. Using the solutions to “earlier” subproblems, when searching for solutions to “later” subproblems, achieves good results, while keeping the complexity manageable.

The difficulty of acquiring adaptation knowledge was identified early in CBR research [Kolodner, 1991]. However, overall, little research has been done on automating acquisition or learning adaptation knowledge. Hanney and Keane [1997] have implemented a CBR system that estimates property values, and learns adaptation rules from cases stored in its case-base. *DIAL* [Leake *et al.*, 1996] also learns adaptation knowledge, in this case for a CBR planning system. It stores previous successful adaptation strategies that combine domain independent abstract adaptation rules with search procedures for domain specific information.

There are simple adaptation methods suitable for adjusting a single numeric solution. These methods make use of lazy learning to acquire their adaptation knowledge, i.e., at the point of retrieval they look at neighbouring cases to determine the amount of adaptation required. The simplest such method is weighted voting. Another example of simple adaptation is the difference heuristic of McSherry [1998]. This heuristic compensates for differences in feature values between the problem and the best matching case by searching for case pairs with the same difference in corresponding features. The major restriction of this method is the assumption that the estimated value is approximated well by an additive function. This severely limits its application area.

Wilke *et al.* [1997] present a framework for learning adaptation knowledge using *knowledge-light* approaches. Simple algorithms, like optimisation or inductive learning, can make use of the knowledge contained in the case-base, the similarity measure, and possibly already existing adaptation knowledge, to learn or improve adaptation knowledge. Wilke *et al.* suggest that one can consider optimisation of the k parameter in weighted voting as an example of learning adaptation knowledge; we view this as learning retrieval knowledge [Jarmulak *et al.*, 2000]. Such knowledge-light methods can not only be used for estimation or other simple CBR tasks, but can also be usefully applied for design tasks, especially if the design can be decomposed into subtasks, as is often the case with configuration or formulation.

Because we are interested in methods of learning adaptation knowledge that are suitable for implementation in general-purpose CBR tools, simpler, knowledge-light methods, like those described by Hanney and Keane and Wilke *et al.* seem more suitable, as opposed to the more powerful, but also more complex, methods used in *DIAL*. In the following, we use several knowledge-light approaches to acquire adaptation knowledge for a formulation task.

3 Adaptation

Adaptation methods in CBR are typically classified in the following 3 categories: substitutional adaptation substitutes or modifies a parameter in the solution; transformational adap-

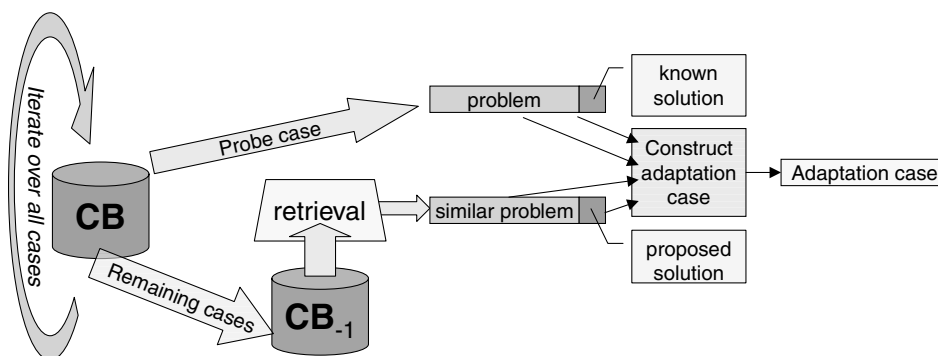


Figure 2: Constructing an adaptation case.

tation involves structural changes to the solution; and generative adaptation redoes the reasoning process that led to a solution [Smyth and Cunningham, 1993]. In this paper we are interested in *substitutional* adaptation, which lends itself best for automatic knowledge acquisition. The other two methods, often used in design tasks, present more difficulties as far as knowledge acquisition is concerned. However, a decomposable design task allows the adaptation to be decomposed into simpler adaptation subtasks, often suitable for substitutional types of adaptation.

3.1 Learning Adaptation Knowledge

The adaptation learning methods we present extract adaptation knowledge from the case data. We propose methods suitable for predicting values for nominal as well as numeric features; we call these targets. Both are needed in configuration or formulation tasks where often the nominal-valued types of components are determined first, and then their numeric parameters are estimated.

Constructing Adaptation Cases

One method we use to extract knowledge that is then used for adaptation, performs leave-one-out retrieval experiments and constructs “adaptation cases”. Figure 2 illustrates how each of the cases is used as a probe case. After retrieving the most-similar case(s) from the reduced case-base, the probe and similar cases are analysed to construct an adaptation case that will be used to adapt the *proposed solution* from the similar case into the correct *known solution* for the probe case.

The way that adaptation cases for numeric targets are constructed is illustrated in Figure 3. We store differences between the problem features of the probe and retrieved cases, and the difference between their solutions. This is sufficient if we expect a uniform dependency between the solution difference and the problem differences over the whole problem space. If, however, the dependency differs in various parts of the problem space, then it is advisable to include also the problem features of the probe and the proposed solution from the retrieved case, as shown in Figure 3.

For nominal targets our adaptation cases are used to predict whether the solution to a probe case proposed by a retrieved case is a good solution. The construction of an adaptation case for nominal targets is illustrated in Figure 4. Firstly, the

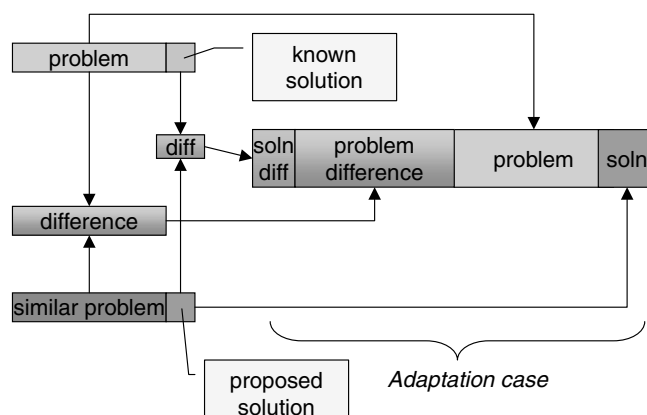


Figure 3: Adaptation cases for numeric targets.

adaptation case must contain data describing the problem features of the probe and the solution proposed by the retrieved case. Because we are dealing with a design task, it may also be possible to include two other types of data in the adaptation case. One type of information is an estimate (*est*) of important properties of the final design that make use of the proposed solution. We estimate these properties using *CBR_E*, a CBR system with the same case-data but with a different target and a more instantiated probe. The final type of information that can be included in the adaptation case is simpler to obtain. Given the predicted solution we determine feature(s) from the problem description (*rel*) which are related to that particular solution. An example from the tablet formulation domain would be chemical stability of a drug (part of problem description) with the chosen component (proposed solution).

Constructing Preference-Profiles

In the method of constructing adaptation cases shown in Figure 4, we estimated values of some design properties (*est*) and found some features (*rel*) related to the proposed solution. When placed in the adaptation case they are used to determine the correctness of the solution. But we can also use these values without constructing adaptation cases. We assume that the values of *est* & *rel* actually occurring in cases correspond to an acceptable range of values for designs,

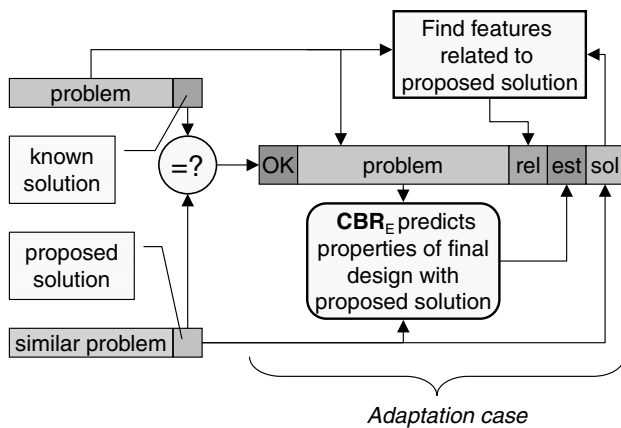


Figure 4: Adaptation cases for nominal targets.

and construct “preference profiles” for these features which reflect the distribution of *est* & *rel* in the case data. Figure 5 shows an example of a profile for the estimated feature *tYP*: a “frequency distribution” of the values of *tYP* in the case-data. Such profiles can be also interpreted as membership functions for a fuzzy concept “preference”. Profiles, once constructed, are then used to estimate the “quality” of suggested solutions. The assumption is that values occurring often in the actual case-data are preferable. Looking at the profile in Figure 5, suppose two nearest neighbours, whose similarity to the problem case are virtually equal, have different solutions, one resulting in an estimated *tYP* of 130, and the other with equal to 530. We would choose the first solution with the higher preference *tYP*.

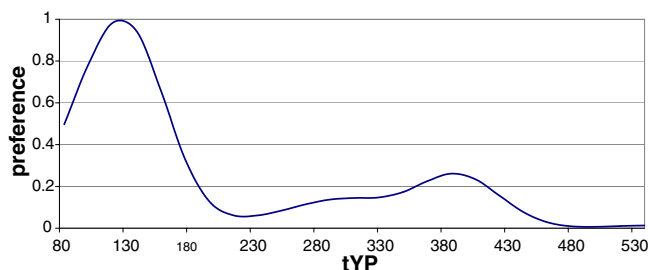


Figure 5: Preference profile.

3.2 Applying Adaptation Knowledge

Now we describe how adaptation cases or preference profiles are used in adaptation. Adaptation cases are applied in a CBR system to determine the adaptation actions. Another possibility would be to induce decision trees to decide the correctness of a nominal solution or to suggest an approximate adjustment for a numeric solution. We could also use other methods, e.g., induce adaptation rules [Hanney and Keane, 1997].

Numeric Targets

Adapting values of numeric targets, is conceptually simple, Figure 6. Given the differences between the problem and the

retrieved case(s) we use CBR_A , a CBR system with adaptation cases like Figure 3, to predict adjustments to the proposed solutions. We can use just a single suggested correction, or k updates from the nearest neighbours or several CBR_A systems and combine the adjustments.

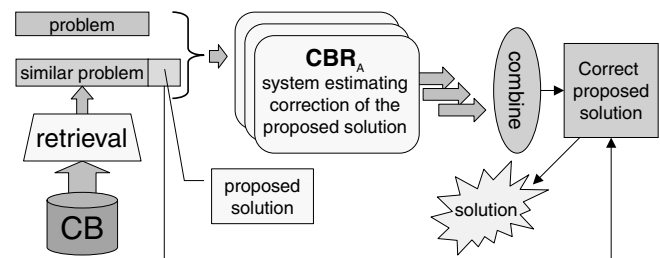


Figure 6: Problem differences to adapt numeric solutions.

Nominal Targets

Figure 7 shows how we can use information about the estimated correctness/quality of the solution in adaptation. The working cycle of the presented system is as follows: (1) given a new problem we retrieve cases from the case-base, these cases give us suggested solutions; (2) we estimate the correctness of the suggested solutions applied to our problem; (3) if the estimate of correctness suggest a high possibility of the solution being correct we return this solution; (4) otherwise, we store this solution with its correctness estimate, and (5) repeat retrieval, but this time ignoring all cases with the solution values which have already been tried out; (6) if all possible solutions have been tried without finding a really good one we will return the one that had the highest estimated correctness, giving preference to the solutions that were retrieved first (the nearest neighbours of the problem case). This approach might look like a search, however, the aspect distinguishing it is the order in which the solutions are tried – this order is determined by case similarity as defined by retrieval.

In this “iterative-adaptation” method the quality of a solution can be estimated using preference profiles; PPROF in Figure 7. To use the profiles, first, we have to obtain values of *est* & *rel* as we did to construct nominal adaptation cases in Figure 4. If several profiles are used then their values can be combined using a weighted sum. If we consider “preference profiles” as fuzzy membership functions then their values can be combined using a fuzzy AND operator, e.g., a minimum or a product.

Figure 7 also shows solution correctness being estimated using a CBR_A system containing adaptation cases like those in Figure 4, instead of PPROF. In this method, we use a combination of the case similarity and the accuracy of the leaves of the case-base index to give us a numeric estimate of the correctness of the solution.

4 Experiments and Results

We have tested these adaptation methods on a simple design task – tablet formulation [Craw *et al.*, 1998]. The design of a new tablet involves identifying inert substances called *excipients* to balance the properties of the drug so that the

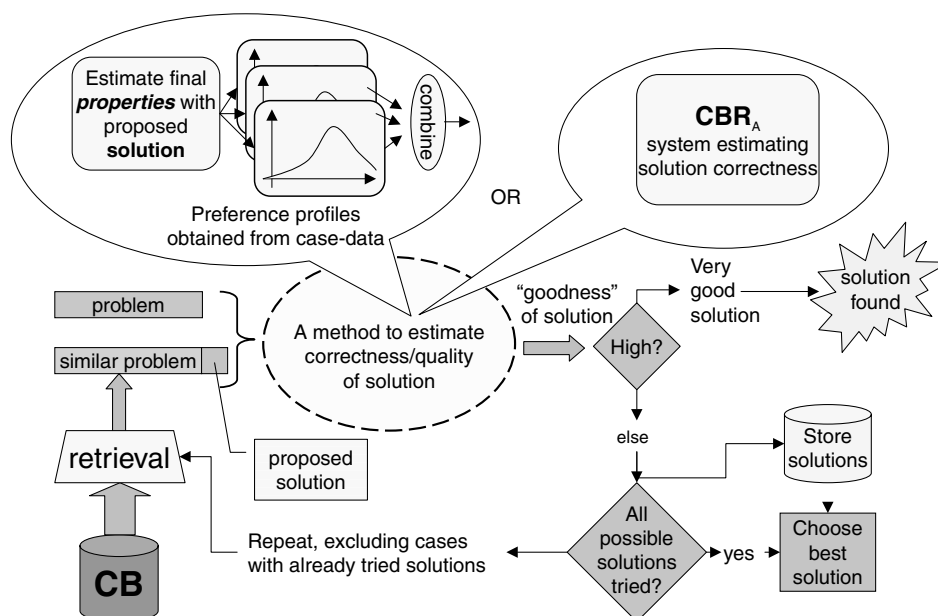


Figure 7: Adaptation of nominal values by iterative “search” for solution.

tablet is manufactured in a robust form, and the desired dose of drug is delivered and absorbed by the patient. Excipients play the role of fillers, binders, lubricants, disintegrants and surfactants in the tablet. Furthermore, a formulation specifies the quantity of each of the added excipients. The difficulty of the formulation task arises from the need to select a set of excipients compatible with the drug, whilst at the same time satisfying a variety of soft and hard constraints. Current tablet formulation practice suggests that this design task is decomposable into subtasks consisting of the choice of excipient component plus determining the excipient amount. It is possible to find a sequence in which the components will be determined, from the most constrained, filler, to the least constrained, surfactant, so that most tablets can be formulated without the need to backtrack in the formulation process.

We conducted adaptation experiments on a case-base containing 156 formulations. Table 1 shows the results for all tablet formulation components, except those where retrieval with *no adaptation* already achieves 100% accuracy. Entries of the form “—” indicate there was no gain from adaptation. The table shows the accuracy as measured by the percentage of acceptable solutions for default CBR retrieval and our automatically optimised retrieval [Jarmulak *et al.*, 2000]. It also shows the gains from using adaptation cases ($CBRA$). For nominal targets, the gains from preference profile adaptation (PPROF) are shown in brackets. The accuracies and gains are averaged over repeated cross-validation experiments that separate the available data into case-data and test problems.

We can see that for several tablet formulation subtasks learned adaptation did indeed bring improvements. We were especially interested in improving the results for the three tasks for which optimised retrieval with no adaptation had accuracies below 90%. Therefore, we are pleased with the improvements in the tabletSRS and binder prediction. For

$CBRA$ / (PPROF)	retrieval only (%)	adaptation gain (%)	optimised only (%)	adaptation gain (%)
filler	68.7	— (—)	80.8	— (—)
binder	30.1	5.4 (11.1)	39.2	4.9 (9.9)
disinteg.	93.5	1.9 (2.4)	96.7	0.7 (—)
filler	85.8	9.8	99.9	0.1
binder	95.9	—	99.4	—
lubricant	88.0	0.8	97.4	0.6
tYPfiller	79.7	1.1	96.2	0.3
tSRS	70.3	—	76.4	3.8

Table 1: Percentage Accuracy and Adaptation Gain

the latter task, we notice that the PPROF method gives much better results than $CBRA$. Preference profiles capture the notion of optimality directly, representing the distribution of feature values for a given prediction; $CBRA$ relies on more local feedback. Moreover, preference profiles are less susceptible to data over-fitting since their knowledge is based on larger training sets.

Because no gains were achieved for the filler excipient task, we analysed this task more closely by looking at the index trees of the $CBRA$ system, see Figure. 8. Our expert confirmed that parts of the index reflected correct domain knowledge. All *est. tSRS* nodes correctly capture the preference for tablets with low SRS. However, other nodes were incorrect, mainly due to the lack of sufficient training examples (adaptation cases). For example, the second level nodes under CCA and MAN are obviously incorrect, and the nodes labelled *filler_related* represent the opposite of the domain knowledge. It is encouraging that the knowledge contained in the case-base index was so easy to interpret. It is also easy

to make corrections to the index. This suggests that learning adaptation knowledge should be followed by a review and refinement of that knowledge by a domain expert. This would still require considerably less effort than acquiring all the adaptation knowledge from the expert.

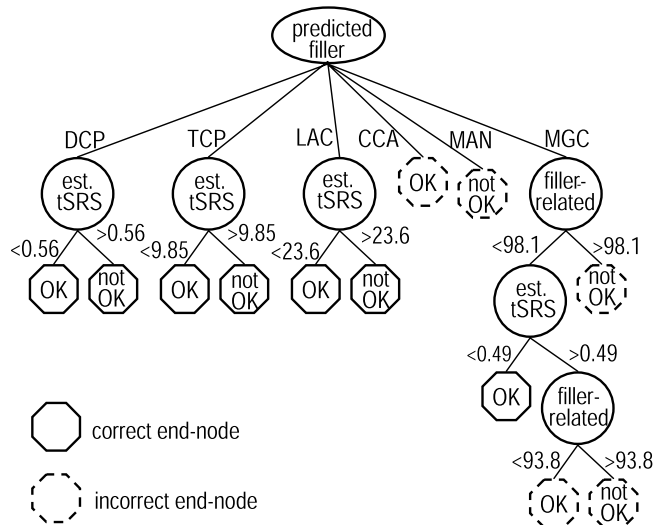


Figure 8: Example case-base index for CBR_A.

5 Conclusions & Future Work

Our research has identified several “knowledge-light” methods of learned adaptation suitable for implementation in general purpose CBR tools. The tool’s adaptation phase must allow the preference profile or CBR_A systems to be invoked and repeated retrieval to be undertaken. Since both adaptation methods are based around cases, the CBR tool itself could be the adaptation platform.

The methods described are suitable for decomposable design tasks as illustrated by our tablet formulation problem. For these adaptation subproblems we have developed effective methods for learning adaptation knowledge. These methods are generic and particularly suited to component-based design because they use constraints and optimality of the components. Tasks other than design may also benefit; features describing the suitability of a proposed solution would be used as the estimated feature.

In our experiments, we found that, depending on the task, different methods, like PPROF or CBR_A, may bring better results. This is in part due to the inherent differences in the design tasks: some may be guided mainly by a search for optimal components while others may use constraints to reject unsuitable components. We therefore think that CBR tools should be equipped with a choice of learned adaptation methods, from which the user selects those giving better results.

An important advantage of the methods presented here is that the acquired knowledge is relatively easy to interpret, e.g. the adaptation cases, the CBR_A systems that use them, and the adaptation profiles. Our further research will concentrate on refining acquired knowledge. We intend to study methods of

presenting the learned adaptation knowledge to an expert, in order to obtain, and then process, expert feedback with the goal of further improving the CBR adaptation stage.

References

- [Bandini and Manzoni, 2000] Stefania Bandini and Sara Manzoni. A support system based on CBR for the design of rubber compounds in motor racing. In *Proc. 5th European Workshop on CBR*, pages 348–357, 2000. Springer.
- [Craw et al., 1998] Susan Craw, Nirmalie Wiratunga, and Ray Rowe. Case-based design for tablet formulation. In *Proc. 4th European Workshop on CBR*, pages 358–369, 1998. Springer.
- [Hanney and Keane, 1997] Kathleen Hanney and Mark T. Keane. The adaptation knowledge bottleneck: How to ease it by learning from cases. In *Proc. 2nd International Conference on CBR*, pages 359–370, 1997. Springer.
- [Jarmulak et al., 2000] Jacek Jarmulak, Susan Craw, and Ray Rowe. Genetic algorithms to optimise CBR retrieval. In *Proc. 5th European Workshop on CBR*, pages 136–147, 2000. Springer.
- [Kolodner, 1991] Janet Kolodner. Improving human decision making through case-based decision aiding. *AI Magazine*, 12(2):52–68, 1991.
- [Leake et al., 1996] David B. Leake, Andrew Kinley, and David Wilson. Acquiring case adaptation knowledge: A hybrid approach. In *Proc. 13th AAAI Conference*, 1996.
- [McSherry, 1998] David McSherry. An adaptation heuristic for case-based estimation. In *Proc. 4th European Workshop on CBR*, pages 184–195, 1998. Springer.
- [Netten and Vingerhoeds, 1996] B. D. Netten and R. A. Vingerhoeds. Incremental adaptation for conceptual design in EADOCS. In *ECAI Workshop on Adaptation in CBR*, Budapest, Hungary, 1996.
- [Smyth and Cunningham, 1992] Barry Smyth and Padraig Cunningham. Déjà Vu: A hierarchical case-based reasoning system for software design. In *Proc. ECAI92 Conference*, pages 587–589, 1992. Wiley.
- [Smyth and Cunningham, 1993] Barry Smyth and Padraig Cunningham. Complexity of adaptation in real-world case-based reasoning systems. In *Proc. 6th Irish Conference on Artificial Intelligence & Cognitive Science*, 1993.
- [Smyth and McKenna, 1999] Barry Smyth and Elizabeth McKenna. Building compact competent case-bases. In *Proc. 3rd International Conference on CBR*, pages 329–342. Springer, 1999.
- [Stahl and Bergmann, 2000] Armin Stahl and Ralph Bergmann. Applying recursive CBR to the customisation of structured products in an electronic shop. In *Proc. 5th European Workshop on CBR*, pages 297–308, 2000. Springer.
- [Wilke et al., 1997] Wolfgang Wilke, Ivo Vollrath, Klaus-Dieter Althoff, and Ralph Bergmann. A framework for learning adaptation knowledge based on knowledge light approaches. In *5th German Workshop on CBR*, 1997.