Using Case Provenance to Propagate Feedback to Cases and Adaptations*

David Leake and Scott A. Dial

Computer Science Department, Lindley Hall 215
Indiana University
Bloomington, IN 47405, U.S.A.
{leake, scodial}@cs.indiana.edu

Abstract. Case provenance concerns how cases came into being in a case-based reasoning system. Case provenance information has been proposed as a resource to exploit for tasks such as guiding case-based maintenance and estimating case confidence [1]. The paper presents a new bidirectional provenance-based method for propagating case confidence, examines when provenance-based maintenance is likely to be useful, and expands the application of provenance-based methods to a new task: assessing the quality of adaptation rules. The paper demonstrates the application of the resulting quality estimates to rule maintenance and prediction of solution quality.

1 Introduction

Case provenance concerns tracking how the cases in a case-based reasoning system came into being, whether from external sources or from internal reasoning processes [1]. Just as humans consider a case's sources when determining its trustworthiness [2], it may benefit a case-based reasoning system to consider the origins of externally-provided cases to estimate cases' applicability or reliability, and some systems have considered case sources in their reasoning [3, 4]. More generally, internal provenance information provides a basis for CBR systems to refine their own processing through introspective reasoning (for an overview of introspective reasoning, see [5]). Leake and Whitehead [1] hypothesized that information about internal case provenance—how a CBR system derived a new case from other cases—can be exploited for many purposes in CBR system maintenance such as assessing case confidence, explaining system conclusions, and improving the ability of case-base maintenance to respond to delayed feedback (as might arise CBR tasks such as design or loan decisions) or case obsolescence (as might arise when predicting prices for a real estate domain). In principle, provenance-based methods could also help focus maintenance effort on knowledge containers beyond the case base, such as similarity information or adaptation knowledge.

* This material is based on work supported in part by the National Science Foundation under Grant No. OCI-0721674.

Leake and Whitehead provided empirical illustrations of the value of provenance information to guide maintenance in the case of delayed feedback, and demonstrated that provenance information about adaptation history could help to estimate case quality. The focus of these approaches is to use provenance to identify low-confidence cases and how those potentially problematic cases arose, in order to anticipate possible problems before the case is applied and, after feedback is available, to focus maintenance activities on cases or adaptation rules which may have contributed to the problems.

This paper builds on that work, focusing on how provenance considerations can enable more effective use of feedback at any time. It advances provenance-based maintenance in three ways. First, it proposes and tests a new bidirectional strategy for propagating case confidence, and provides a finer-grained examination of the use of provenance information to estimate case quality. Second, it examines how initial case-base quality affects the benefit of provenance-based feedback propagation. Third, it presents and evaluates a first study of the use of provenance information to guide maintenance of case adaptation rules, a novel area for CBR system maintenance. Experimental studies support the promise of these new directions for exploiting case provenance information.

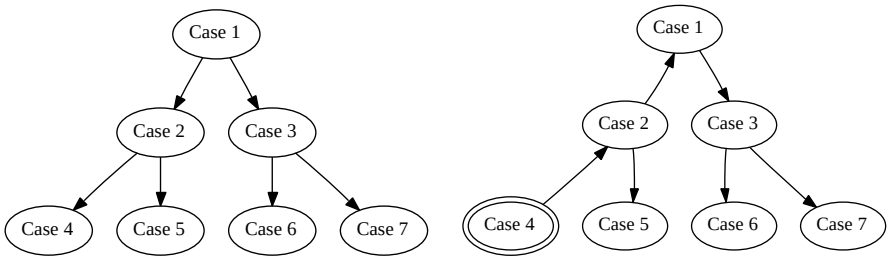
2 Bidirectional Feedback Propagation

When a CBR system derives new cases from the cases in its case library, their provenance trace includes the cases from which they were derived and the adaptations used to derive them. Leake and Whitehead's work suggested that propagating feedback to related cases (as determined by adaptation history) provides a computationally practical and effective way of exploiting feedback concerning flawed conclusions. Their studies considered the effects of propagating feedback either to parents of a case—the cases from which the case was derived—or to the case's children—cases which had been derived from it prior to the feedback being received. Both methods were shown to improve performance, but downward propagation (to descendants) performed better in their tests [1].

To determine whether a bidirectional method could improve on both, we developed the algorithm shown in Figure 2. When the system receives feedback on a case, it propagates the feedback to the case's ancestors and repeats any adaptations to descendants (we will refer to this as *repairing* the case base). An example of a case base with adaptation provenance is shown in Fig. 1(a); Fig. 1(b) then gives an example of the propagation of feedback if the feedback was given for "Case 4." We note that adapting children to find solutions for the problems of their parent cases is not always possible. However, in practice the ability to adapt cases is often symmetric, and the algorithm assumes the ability to perform such adaptation.

Two factors complicate the propagation process:

1. Repeated ancestors: A single case may appear more than once in the ancestry trace.



(a) Graph of provenance (b) Path of feedback propagation

Fig. 1. Sample provenance and feedback propagation paths, beginning with "Case 4."

2. Repeated descendants: A single case may appear underneath more than one parent (e.g., for k-NN with $k > 1$).

Consequently, bidirectional propagation must address the risk of cycles and multiple paths.

To address the problem of repeated ancestors, the algorithm only traverses the graph upwards to parents which have not yet been visited. Because the search is breadth-first, this ensures that the parent receives feedback along the shortest possible chain. By the heuristic of using chain length as a proxy for amount of knowledge degradation during propagation (which has been shown to give reasonable performance in some tests [1]), in the absence of finer-grained information we expect this to be the most reliable feedback.

To address the problem of repeated descendants, the algorithm simply recalculates the effect of each adaptation path in the provenance trace. When the same adaptation that was previously used still applies (e.g., for numerical averaging methods such as used by k-NN), this correctly reflects the change in each case's contribution to the solution. In general, if changes to the cases are small, we might assume that the same adaptations would apply, by the basic CBR assumption that similar problems (in this case, adaptation problems) should have similar solutions (in this case, adaptations). In domains for which updates to a case may invalidate the adaptation previously applied to it, how to handle propagation is an open question.

3 Estimating Confidence in Adaptation Rules

Because adaptation rules may be expected to provide somewhat approximate results, some loss of solution quality might be expected over long adaptation chains. Leake and Whitehead explored a very simple method for estimating case confidence based on the provenance trace: to predict a degradation of case quality proportional to the number of adaptations applied. Their experiments showed that in the absence of other feedback on case quality, this criterion can be a useful heuristic for choosing cases to maintain.

```
0: GiveBidirectedFeedback(C_f, C_t)
1:    /* Let C_f be the feedback case and C_t be the target of feedback. */
2:
3:    Replace(C_t, C_f)
4:
5:    work ← ∅ /* A queue of {target, source, direction} tuples */
6:    parents ← ∅ /* The set of parent cases that have been seen */
7:
8:    /* Propagate feedback to the parents and children of C_t. */
9:    for all p ∈ Parents(C_t) do
10:      work.push({C_t, p, UP})
11:   end for
12:   for all c ∈ Children(C_t) do
13:      work.push({C_t, c, DOWN})
14:   end for
15:
16:   while work ≠ ∅ do
17:      {f, t, d} ← work.pop()
18:
19:      if d = UP ∧ t ∉ parents ∧ ¬IsReferenceCase(t) then
20:         Replace(t, Adapt(f, Problem(t)))
21:         parents ← parents ∪ {t}
22:
23:         /* Propagate feedback to the parents and children of t. */
24:         for all p ∈ Parents(C_t) do
25:            work.push({C_t, p, UP})
26:         end for
27:         for all c ∈ Children(C_t) do
28:            work.push({C_t, c, DOWN})
29:         end for
30:      else if d = DOWN then
31:         Replace(t, Adapt(Parents(t), Problem(t)))
32:
33:         /* Propagate feedback to the children of t. */
34:         for all c ∈ Children(C_t) do
35:            work.push({C_t, c, DOWN})
36:         end for
37:      end if
38:   end while
```

Fig. 2. Algorithm for bidirectional feedback propagation in a case-base, guided by provenance information.

However, provenance information about adaptations may be used in another way, to guide maintenance of the adaptation rules themselves. If a solution is flawed, the flaw may result from flaws in the retrieval process (selecting the wrong case(s) as starting point), flaws in the case(s) from which the solution was derived (e.g., due to obsolescence), flaws in the rules used to adapt those cases to the solution, or from a combination. If we assume that cases in the case base are approximately correct and retrieval is generally reliable, erroneous solutions can be attributed to problems in adaptation rules.

To explore the use of provenance to guide rule maintenance, we have developed a method to rank the performance of a system's adaptation rules, assuming that the cases to which they are applied are correct. Problem rules may then be flagged for expert assessment and maintenance if necessary. In what follows, we assume that a numerical error value can be assigned to any suboptimal solution.

Propagation approach: The rule ranking algorithm exploits a provenance trace, which for each case records all of the rules invoked for a given adaptation. When the system receives feedback about the performance of a solution in the case base, it recursively assigns blame to rules. The propagation process follows the same upward path as shown in Figure 1. However, feedback is not propagated downwards to children; feedback only has bearing on the adaptations that directly led to the creation of the case through the case's parents.

Blame assignment: The blame assignment process is inspired by back-propagation in neural networks [6]. The feedback on an erroneous case is treated as a training sample for a network, and each rule used in adaptation is treated as a weighted edge. The weight is modified in response to the error determined from feedback. The algorithm divides the local error evenly among all of the rules (a possible future refinement would be to estimate the relative influence of each rule). The algorithm then proceeds recursively through the ancestry (backwards) as in backpropagation.

Despite the natural relationship to backpropagation, the differing tasks result in a few differences:

1. Because the weights have no direct effect on the error of the system, local errors do not converge towards zero as propagation proceeds. Consequently, error weights tend to accumulate.
2. Unlike backpropagation, the algorithm does not visit all edges (rules) an equal number of times.
3. Because a new case may arise from adaptation rules in complicated ways, rather than from simple application of, e.g., backpropagation's sigmoid function, blame assignment could require sophisticated reasoning.

For our purposes, difference (1) is unimportant: We are concerned only in ranking rules by error levels, rather than in any specific error values. Difference (2) can be addressed by normalizing the weights by the number of times that they have been updated. The accumulation of error by a rule decreases confidence in that rule. The lower the confidence, the worse the average performance of the rule. This

confidence information enables modifying or removing rules that are adversely affecting the performance of the system.

Difference (3), concerning the transfer of error, is more difficult to address. Because their is no canonical way to project backwards through the adaptation to assign blame to the inputs, we have chosen the simple approach of assigning a fixed proportion of the output's error to each rule. The fractional coefficient, or *decay rate*, reduces change to adaptation rule weights more distant from the feedback case. The decay reflects the assumption that less is known about sources of the error after it is passed backwards through an adaptation, and that it consequently should have less effect on more distant weights. The full algorithm is presented in Fig. 3.

```
0: GiveRuleFeedback(C, E)
1:    /* Let case C be the target of feedback, E be the relative error of this case's
2:       solution, and let η be the decay rate. */
3:
4:    work ← ∅ /* A queue of {target, error} pairs */
5:    work.push({C, E})
6:    while work ≠ ∅ do
7:       {c, e} ← work.pop()
8:
9:       /* Adjust the weights of all of the rules invoked. */
10:      for all r ∈ Rules(c) do
11:         r_weight ← (1 − η·e / |Rules(c)|) · r_weight
12:         r_visited ← r_visited + η / |Rules(c)|
13:      end for
14:
15:      /* Add the parents to the work queue. */
16:      for all p ∈ Parents(c) do
17:         work.push(p, η · e)
18:      end for
19:   end while
```

Fig. 3. Algorithm for learning adaptation rule quality from feedback and provenance information. The result is a weighting reflecting each rule's contribution to system error.

4 Experimental Evaluation of Bidirectional Repair

To study the bidirectional feedback method, we performed experiments to address two questions:

1. How does the benefit of bidirectional repair compare to that of repair directed only to either ancestors or descendants?
2. When is provenance-based maintenance most useful?

For the second question, we focused on the effects of initial case-base quality (measured by solution accuracy) on the incremental benefit of provenance-guided feedback.

4.1 Experimental Design

Our system was developed using the Indiana University Case-Based Reasoning Framework (IUCBRF) [7]. We extended IUCBRF to automate the tracking of case provenance by maintaining a directed graph recording adaptation history for cases in the case base and to perform the record-keeping needed for the algorithms presented in this paper.

The first set of experiments tested the system using the Boston Housing dataset and the Abalone dataset from the UCI Machine Learning Repository [8]. The Boston Housing dataset contains 506 cases with attributes capturing the quality of housing in the Boston area. This dataset includes an attribute denoting the median value of owner-occupied homes, and the system's task is to determine home values. The Abalone dataset contains 4177 cases with physical attributes for the Abalone, which are used to predict age.

For both datasets, the system used 3-NN retrieval with the similarity determined by weighted Euclidean distance, for which feature weights were determined by a multiple linear regression on the given cases. The three retrieved cases are adapted to the target problem by the scaling of a distance-weighted mean. The adapted solutions are retained as new cases in the case base. Feedback is given as the relative error of the solution.

In our trials, case bases were randomly populated with 100 cases, and the system then tested on 200 problems randomly selected from the remaining set. Each new solution was placed in the case base, with a case randomly selected and removed from the case base after each iteration to keep case base size constant. To evaluate the average accuracy during a trial, the system was tested by leave-one-out testing with all problems from the original dataset. The absolute error was measured, and the mean of these errors recorded as the mean absolute error (MAE) of the case base.

4.2 Comparing Bidirectional Feedback to Prior Methods

In order to be able to compare results from [1], we recreated the experiment from that paper. In this version of the system, we randomly choose a case to give feedback after each problem is posed. We repeated this experiment for 1000 trials to produce the average performance shown in Fig. 4 and Fig. 5.

The results show that in all cases, the bidirectional propagation has the lowest error compared to the previous best methods. With respect to the Boston Housing dataset, the improvement is not as great as that with the Abalone dataset. However, this is not entirely surprising because Leake and Whitehead noted that propagation to descendants proved more useful than the ancestors for that dataset, suggesting that the addition of propagation to ancestors might have less benefit.

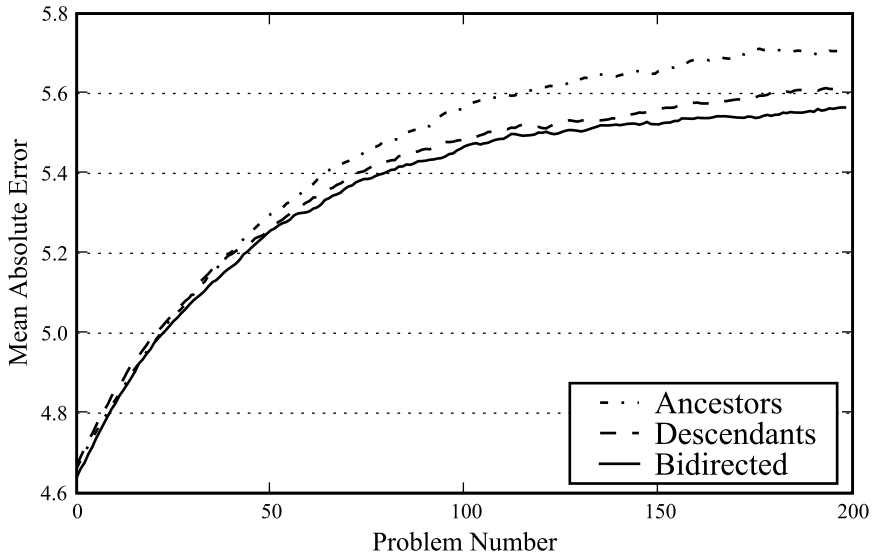


Fig. 4. Mean absolute error of the Boston Housing system for bidirectional propagation, propagation to ancestors, and propagation to descendants.

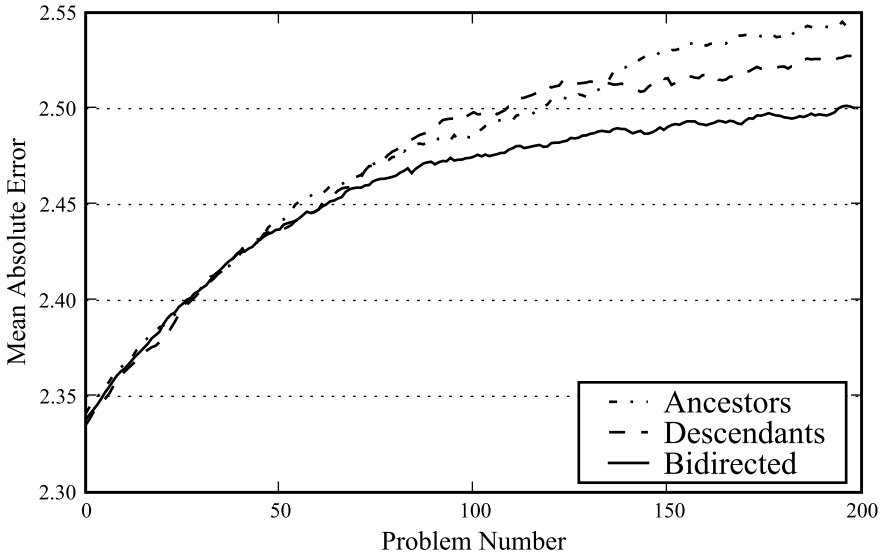


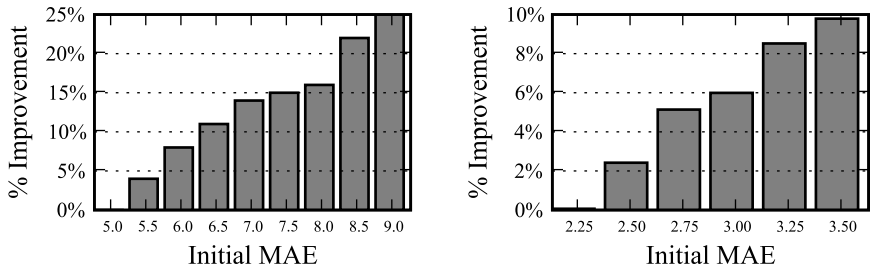
Fig. 5. Mean absolute error of the Boston Housing system for bidirectional propagation, propagation to ancestors, and propagation to descendants.

4.3 How Case Base Quality Affects Benefits of Provenance-Based Propagation

An interesting question for any maintenance strategy is when it is likely to be most useful. This experiment assessed how the benefit of the bidirectional strategy depended on the original quality of the case base.

In this experiment, after selection of the original 100 cases and solution of 250 problems, the case bases were evaluated for quality of coverage. Twenty-five cases were then randomly selected from the case base to have their solutions replaced by correct feedback, simulating expert maintenance, with the case-base repaired by bidirectional propagation. The quality of coverage was then recalculated to determine to what degree the system was improved.

Figure 6 shows the results of this experiment as a histogram broken down by the initial error in the system. This shows a clear trend towards increased percent benefit with higher-error case bases.



(a) Boston Housing (b) Abalone

Fig. 6. Performance of the feedback propagation system for various ranges of initial MAE values.

5 Experimental Evaluation of Adaptation Rule Maintenance

In a second set of experiments, we investigated the ability of the provenance-based algorithm of Figure 3 to identify low-quality rules, in order to guide maintenance. We explored the question "Can the rule ranking algorithm identify rules whose removal will improve system accuracy?"

5.1 Experimental Design

Because this experiment required a domain for which a rich set of adaptation rules was available, for it we selected a domain conducive to the generation of adaptation rules. We extracted cases from the Homefinder.org website [9], which

contains real estate listings for Bloomington, Indiana, U.S.A.. The extracted data contain a number of features useful for predicting the value of a home, as well as the listing price for each home, which was the target value for the system to predict. The collected data was filtered for erroneous values, and those cases were removed. The final dataset—a snapshot of listings on February 22, 2008—contains 333 cases.

To generate a large set of rules, we applied an algorithm based on the automatic adaptation rule acquisition work of Hanney [10], which also used a real estate domain. Our algorithm produced rules that consider only a single feature at a time, to simplify the implementation; more complex adaptations can be achieved by successively applying multiple rules. We generated 272 rules of this form.

As with our first experiment, for each run we populated the case base with 100 random cases with known solutions and tested the system with 200 problems. Finally, 25 cases in the case base were randomly selected for feedback in the form of the known solution. As feedback was applied, rule quality estimates were updated according to the rule confidence algorithm.

We then considered two questions:

1. Does the algorithm properly identify problematic rules?
2. Are the rule confidence values useful for predicting case confidence of adapted cases?

5.2 Identifying Problematic Rules

After each run, the lowest-ranked rules are removed from the system and the trial is repeated with the same initial conditions. If the rule ranking identifies bad rules, we expect that the removal of those rules will improve the system's performance. As a baseline, the same tests were performed removing random rules.

The results of this experiment are shown in Fig. 7. Removing low-ranked rules yields a significant performance improvement for the system. Given the simple approach taken to generate rules, it is reasonable to expect a number of low-quality rules. We observe that benefits are achieved for removal of even large numbers of rules, though with diminishing returns as larger numbers of rules are removed. The fact that random removals often provide benefit is initially surprising. Given that our rule generation procedure produces rules with a wide range of quality, we hypothesize that this may result from occasional serendipitous removal of very low-quality rules, but this and the discrepancy between benefit of initial and later random deletions are subjects for further investigation.

Figure 8 shows the average marginal benefit of removing each rule. We hypothesize that two factors affect the diminishing returns shown by the graph. First, if the algorithm is performing as desired, the worst rules should tend to be removed first; additional removed rules tend to be of higher quality. Second, available feedback is limited, limiting the system's ability to assess rule quality for rules used infrequently. The improvement gained from removing rules based on insufficient feedback is similar to the effect of removing random rules.

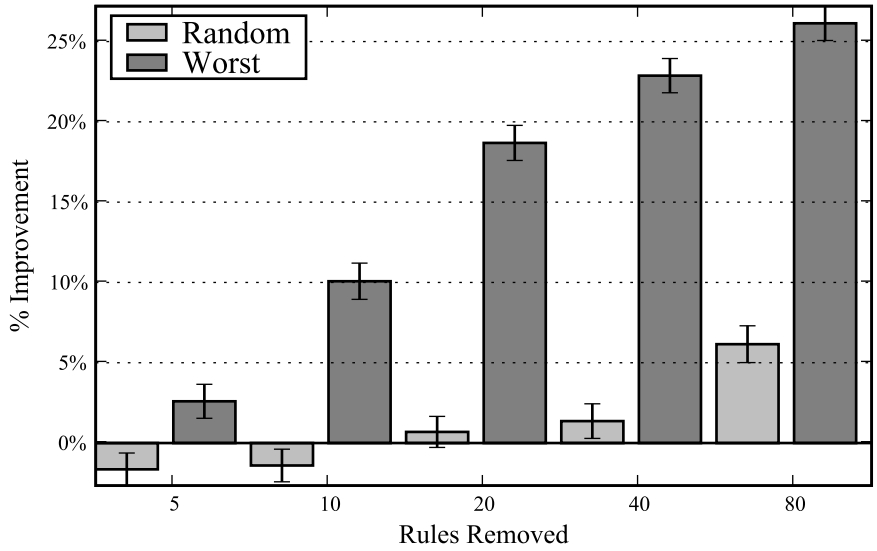


Fig. 7. Percent improvement in relative error after removing rules considered worst according to the rule confidence algorithm, compared to random rule deletion, based on the mean of 1000 runs. The error bars represent 95% confidence intervals.

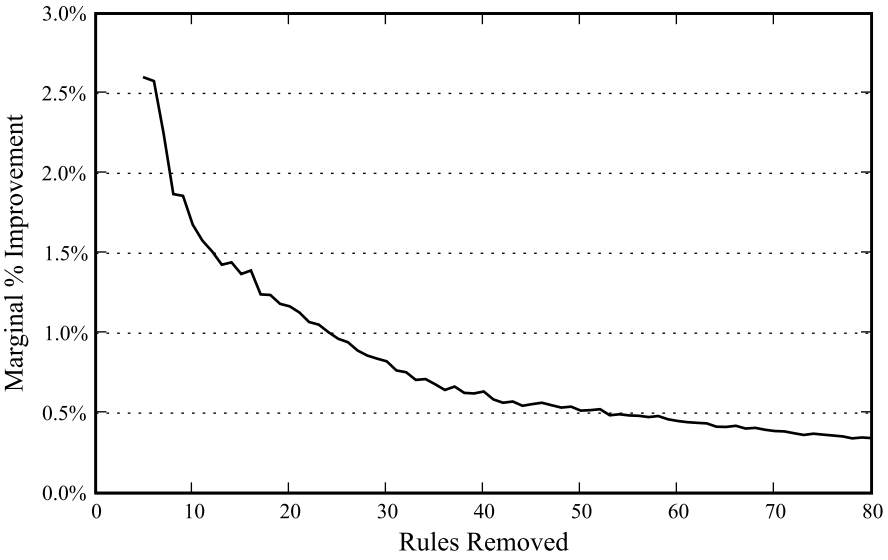


Fig. 8. A plot showing the amount of improvement per rule by removing a given number of worst rules. The plot shows 1000 trials of N rules being removed, where N extends from 5 to 80 rules.

5.3 Using Rule Confidence to Predict Case Confidence

Leake and Whitehead's [1] experimental test of provenance-based confidence prediction treated all adaptations identically. Here we exploit the availability of rule confidence information to explore a finer-grained approach, estimating solution confidence based on the system-generated adaptation rule confidence for the rules used to generate the solutions. After an adaptation, confidence in a solution is adjusted by the mean weight of the rules used to adapt it. We use the following confidence rule, where the parameter α controls how large of an effect the adaptation confidence has on the solution:

$$\text{SConfidence(c)} = \text{SConfidence(Parent(c))} \cdot \left(\sum_{r \in Rules(c)} \frac{\text{RConfidence(r)}}{|\text{Rules(c)}|} \right)^{\alpha}$$

SConfidence(c) denotes the confidence in a case c, RConfidence(r) denotes the confidence in a rule r, and Rules(c) denotes the set of rules invoked to adapt a case c.

In this test, to increase the quality of the rule weights, feedback is provided to the system after every solution. For the experiment, we also modified the system to retrieve five of the nearest neighbors of a case and adapt each one to the target problem separately, returning the solution that has the greatest confidence. We empirically determined an appropriate α – approximately 0.1.

We have recorded the mean absolute error of the solutions over the 100 test cases, for 1000 random trials of the system. We observed an average of a $4\% \pm 1\%$ (95% confidence interval) reduction in the error of the system. We believe this improvement, observed even with very simple methods, suggests the promise of considering adaptation rule confidence when predicting case confidence. Future work will refine the rule confidence estimation procedure.

6 Related Work

The notion of provenance tracking is receiving considerable attention in the e-Science community, for tracking the derivation of scientific data [11]—and even for case mining [12]—as well as in the semantic Web community (e.g., [13]). Tracing the derivation of beliefs has a long history in AI as well, extending to early work on truth maintenance systems [14].

Within CBR research, storage of meta-cases was proposed by Goel and Murdock [15] to capture a CBR system's reasoning for explanation, and reasoning traces are used for introspective failure repair in Fox's ROBBIE system [16].

Case-base maintenance has long been an active CBR area (see [17] for a sampling of some of this work), but there has been little attention to the maintenance of existing case adaptation knowledge. Often, the adaptation component of a CBR system consists of static expert-specified rules that do not change over the course of a CBR system's lifetime. Existing work has focused on augmenting

adaptation knowledge, rather than on identifying problems in adaptation knowledge, as done in this paper. For example, work has explored mining adaptation knowledge from pre-existing cases, as by Hanney and Keane [18], Craw, Jarmulak and Rowe [19], and Patterson, Rooney, and Galushka [20]; other work has focused on capturing increasing adaptation knowledge by acquiring adaptation cases [21, 22]. Wilke et al. [23] propose knowledge-light approaches for refining adaptation knowledge using knowledge already contained in the CBR system, and Patterson and Annad [24] propose methods for mining adaptation rules; McSherry's on-demand adaptation using adaptation triples [25] is in a similar spirit. This work also relates to Aquin et. al's CABAMAKA system, which combines case base mining with expert guidance [26]. Rial et al. [27] introduced a method for revising adaptation rules using belief revision [28].

7 Conclusion

Case provenance provides a promising source for reasoning to guide CBR system maintenance. This paper investigates the use of provenance to guide the propagation of feedback, describing a bidirectional propagation method. It also provides a first assessment of the case base characteristics for which such propagation is likely to be useful, providing support for the hypothesis that the highest percentage improvements arise for lower-quality case bases.

The paper also describes, to our knowledge, the first use of provenance information to guide maintenance of another knowledge container, the system's adaptation knowledge. It introduces an algorithm inspired by backpropagation to assign blame to adaptation rules, identifying low-quality rules for revision or removal. Evaluations suggest the promise of this approach and its potential application to assessing case confidence. In future research we expect to develop more refined methods for the evaluation of case and rule confidence and provenance-based identification of problematic rules.

References

1. Leake, D., Whitehead, M.: Case provenance: The value of remembering case sources. In: Case-Based Reasoning Research and Development: Proceedings of the Seventh International Conference on Case-Based Reasoning, ICCBR-07, Berlin, Springer-Verlag (2007)
2. Evans, M.: Knowledge and Work in Context: A Case of Distributed Troubleshooting Across Ship and Shore. PhD thesis, Indiana University (2004)
3. Göker, M., Roth-Berghofer, T.: Development and utilization of a case-based helpdesk support system in a corporate environment. In Althoff, K.D., Bergmann, R., Branting, L.K., eds.: Proceedings of the Third International Conference on Case-Based Reasoning, Springer Verlag (1999) 132–146
4. Leake, D., Sooriamurthi, R.: Case dispatching versus case-base merging: When MCBR matters. International Journal of Artificial Intelligence Tools **13**(1) (2004) 237–254

5. Cox, M.: Metacognition in computation: A selected research review. Artificial Intelligence **169**(2) (2005) 104–141
6. Werbos, P.: Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University (1974)
7. Bogaerts, S., Leake, D.: IUCBRF: A framework for rapid and modular CBR system development. Technical Report TR 617, Computer Science Department, Indiana University, Bloomington, IN (2005)
8. Asuncion, A., Newman, D.: UCI machine learning repository. Technical report, University of California, Irvine, School of Information and Computer Sciences (2007)
9. Bloomington MLS, Inc.: homefinder.org (2007)
10. Hanney, K.: Learning adaptation rules from cases. Master's thesis, Trinity College, Dublin (1997)
11. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-Science. SIGMOD Record **34**(3) (2005) 31–36
12. Leake, D., Kendall-Morwick, J.: Towards case-based support for e-science workflow generation by mining provenance information. In: Proceedings of the Nineth European Conference on Case-Based Reasoning, Springer (2008) In press.
13. Murdock, J., McGuiness, D., da Silva, P.P., Welty, C., Ferrucci, D.: Explaining conclusions from diverse knowledge sources. In: Proceedings of the Fifth International Semantic Web Conference (ISWC20063), Berlin (2006) 861–872
14. Doyle, J.: A truth maintenance system. Artificial Intelligence **12** (1979) 231–272
15. Goel, A., Murdock, J.: Meta-cases: Explaining case-based reasoning. In: Proceedings of the Third European Workshop on Case-Based Reasoning, Berlin, Springer Verlag (1996) 150–163
16. Fox, S., Leake, D.: Modeling case-based planning for repairing reasoning failures. In: Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms, Menlo Park, CA, AAAI Press (March 1995) 31–38
17. Leake, D., Smyth, B., Wilson, D., Yang, Q., eds.: Maintaining Case-Based Reasoning Systems. Blackwell (2001) Special issue of *Computational Intelligence*, 17(2), 2001.
18. Hanney, K., Keane, M.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In: Proceedings of the Second International Conference on Case-Based Reasoning, Berlin, Springer Verlag (1997)
19. Craw, S., Jarmulak, J., Rowe, R.: Learning and applying case-based adaptation knowledge. In Aha, D., Watson, I., eds.: Proceedings of the Fourth International Conference on Case-Based Reasoning, Berlin, Springer Verlag (2001) 131–145
20. Patterson, D., Rooney, N., Galushka, M.: A regression based adaptation strategy for case-based reasoning. In: Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence, AAAI Press (2002) 87–92
21. Sycara, K.: Using case-based reasoning for plan adaptation and repair. In Kolodner, J., ed.: Proceedings of the DARPA Case-Based Reasoning Workshop, San Mateo, CA, Morgan Kaufmann (1988) 425–434
22. Leake, D., Kinley, A., Wilson, D.: Acquiring case adaptation knowledge: A hybrid approach. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, Menlo Park, CA, AAAI Press (1996) 684–689
23. Wilke, W., Vollrath, I., Althoff, K.D., Bergmann, R.: A framework for learning adaptation knowledge based on knowledge light approaches. In: Proceedings of the Fifth German Workshop on Case-Based Reasoning. (1997) 235–242

24. Patterson, D., Anand, S., Dubitzky, W., Hughes, J.: Towards automated case knowledge discovery in the M^2 case-based reasoning system. In: Knowledge and Information Systems: An International Journal, Springer Verlag (1999) 61–82
25. McSherry, D.: Demand-driven discovery of adaptation knowledge. In: Proceedings of the sixteenth International Joint Conference on Artificial Intelligence (IJCAI-01), San Mateo, Morgan Kaufmann (1999) 222–227
26. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07), San Mateo, Morgan Kaufmann (2007) 750–755
27. Rial, R.P., Fidalgo, R.L., Rodriguez, A.G., Rodriguez, J.C.: Improving the revision stage of a CBR system with belief revision techniques. Computing and Information Systems **8** (2001) 40–45
28. Alchourrón, C., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic **50** (1985) 530–541