Learning to Improve Efficiency for Adaptation Paths

David Leake and Xiaomeng Ye

Luddy School of Informatics, Computing, and Engineering
Indiana University, Bloomington IN 47408, USA
leake@iu.edu, xiaye@iu.edu

Abstract. The ability of case-based reasoning systems to deal with new problems depends on the effectiveness of their case adaptation. One approach to increasing flexibility for novel problems is to perform adaptations by using adaptation paths—chains of adaptations—to address differences beyond those addressable by applying single adaptation rules. A recent approach to adaptation path generation, ROAD, proposes building adaptation paths using heuristic search guided by similarity, with a "reset" mechanism for recovering when similarity fails to predict adaptability. The ROAD approach is beneficial when similarity and adaptability are well aligned, but can make poor choices when similarity and adaptability diverge, increasing adaptation cost. This paper presents methods for increasing adaptation efficiency by maintenance exploiting information from adaptation path generation. The methods improve the similarity measure to better reflect adaptability and condense the adaptation rule set. Experimental evaluation supports the benefits for improving adaptation efficiency while preserving accuracy.

Key words: Adaptation paths, Adaptation rule maintenance, Case adaptation, Case-based reasoning, Machine learning, Similarity maintenance

1 Introduction

Case-based Reasoning (CBR) solves new problems by adapting previous solutions to fit new circumstances (e.g., [16]). The case adaptation process is critical to the flexibility of CBR systems, enabling stored cases to cover a range of new problems. Case adaptation is often rule-guided, based on a set of adaptation rules designed to cover each possible class of difference between old and new problems in a single step (e.g., [6]). However, relying on one-step adaptations may require a large set of adaptation rules, and it may be hard to anticipate which rules will be needed. The knowledge acquisition problem for case adaptation is a classic problem for CBR (e.g., [7]). Covering problems with one-step adaptation may be especially problematic for sparse case bases and domains with highly novel problems. This has motivated research on path-based case adaptation using sequences of adaptation rules [2, 4, 14].

Making path-based adaptation effective depends on addressing two issues. The first is selecting the sequence of adaptation rules to apply, given that adaptation rules have varying reliability and that longer paths may be prone to quality degradation [13]. The second is controlling the computational cost of searching through sequences of adaptation rules. The RObust ADaptation (ROAD) [14] approach proposes addressing these problems by using heuristics to guide a greedy search process exploring alternative adaptation paths. As adaptation rules are applied, ROAD generates intermediate *ghost cases* [12], and extends the path from ghost cases closest to the target, based on similarity distance. This process aims to control adaptation path generation cost by finding short paths rapidly. To reduce the risk of solution quality degradation, ROAD uses a reset mechanism that is triggered when the expected reliability of a path falls below a threshold or when two paths are found to be developing divergent solutions. Previous experiments showed that ROAD can increase accuracy compared to relying on single-step adaptations. This paper presents methods aimed at increasing efficiency of the ROAD process.

The efficiency of similarity-based search with resetting depends on the similarity measure being a good proxy for adaptability. However, the correspondence between similarity and adaptability is not guaranteed [19]. When similarity distances diverge from true adaptation distances, a similarity-based adaptation path may proceed through ghost cases that are not easily adaptable, resulting in longer paths. This paper presents two maintainance methods that use information from the ROAD adaptation process to improve future performance. The first refines the system's similarity measure, bringing it closer to reflecting true adaptability. We call this *Reset-induced Similarity Adjustment* (RISA). RISA uses knowledge of the final path to determine which prior cases should have been retrieved to minimize adaptation cost, and adjusts similarity criteria accordingly to improve future retrievals. This can be seen as ongoing CBR system maintenance [21] of similarity criteria, based on failures revealed by resetting. The RISA approach can be applied to any similarity measure that supports adjusting distances between pairs of cases (e.g., based on a ranking loss function).

The second maintenance method compacts the set of adaptation rules. Especially with the use of large-scale automatic rule generation methods, large sets of adaptation rules may be generated, making rule filtering potentially important to CBR system performance [9]. Also, in path-based adaptation using heuristic search, decreasing the number of rules to consider decreases the branching factor—and consequently, the computational cost—of the search. To compact the adaptation rule set, we propose *Compatibility-based Adaptation Rule Selection* (CARS), which prioritizes adaptation rules for retention based on analysis of pairwise compatibility of adaptation rules in adaptation paths.

Experimental results in this study support that RISA, in conjunction with a local weighting scheme, can improve the similarity measure to produce shorter adaptation paths requiring fewer resets. They also show that trimming the adaptation rule set with CARS can decrease resets and result in shorter adaptation paths while maintaining comparable error rates.

The paper begins by reviewing the ROAD adaptation approach. It then presents the RISA algorithm and evaluation, followed by CARS and its evaluation. It closes by summarizing related work and discussing future directions.

2 The ROAD Adaptation Approach

Using adaptation paths is a promising way to increase adaptation flexibility, but depends on effective methods to guide path generation. ROAD [14] generates paths by similarity-guided greedy search and improves accuracy with a retrieval-based method for resetting the starting points of problematic paths.

2.1 Generating Adaptation Paths by Similarity-Guided Search

ROAD solves problems by retrieving the case most similar to the current problem and applying an adaptation path. It builds the path by greedy search, guided by similarity distance to the target, with the goal of generating short paths. When an adaptation rule is applied to a case, ROAD adapts both its solution and problem description to generate a hypothetical case, called a "ghost case" [12]. The next adaptation rule is selected in the context of that ghost case. After applications of adaptation rules to a case, the resulting ghost cases are compared to the target case, and the adaptation rule leading to the ghost case most similar to the target is used as the next adaptation step in the path. ROAD can pursue multiple paths simultaneously. Paths are prioritized based on their length where the shortest path (unless terminated) is developed first.

An adaptation path is a list containing the retrieved case, a sequence of ghost cases generated by adaptation, the adaptation rules applied, and the target case. The ROAD algorithm is described in detail in Leake and Ye [14].

To illustrate a use of path-based adaptation, we consider the problem of generating a recipe for making buttermilk pancakes from available ingredients, starting from a recipe for regular pancakes, when the agent has no buttermilk available. A first adaptation would be to substitute buttermilk for regular milk in the recipe, thus generating the ghost case of a recipe for buttermilk pancakes. That ghost case is more similar to the target, but still differs, because it does not satisfy the constraint to use available ingredients. However, it is possible to make buttermilk by mixing milk and vinegar. Consequently, a second adaptation could be applied to the ghost case, substituting milk and vinegar for the buttermilk. This two-step adaptation path would result in a recipe matching the target.

2.2 Improving Accuracy by Path Resetting

For a similarity measure that perfectly captures adaptation distance—*i.e*, that enables perfect adaptation-guided retrieval [19]—the initially retrieved source case would always have the smallest adaptation distance to the target of all cases in the case base. However, if the similarity measure does not perfectly capture adaptability, another case might be easier to adapt than the retrieved

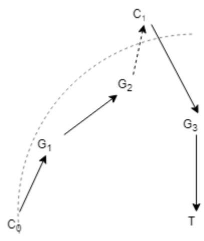

Fig. 1. Illustration of Path Resetting, from Leake and Ye [14]

case. As the adaptation path generation process generates new ghost cases, the ghost cases may be near existing cases in the case base which are closer to the target than the retrieved case in terms of adaptation distance. To recover, ROAD "resets" the path to start from the nearby case by moving the head of the path to its nearby case. The rationale for such resets is to increase accuracy, by starting from a solution known to be correct, rather than relying on the solution of a ghost case generated from a sequence of adaptations. The accuracy benefit has been supported experimentally [14].

The resetting process is illustrated in Figure 1. C_0 and C_1 are two cases in the case base. Given a query Q, for which the solution case would be T, the source case C_0 is retrieved. C_0 is closer to T than C_1 according to the similarity measure (indicated by the dashed arc indicating a radius of equal similarity values). As adaptation rules are applied successively, ghost cases G_1 and G_2 are generated. The ghost case G_2 is found to be more similar to C_1 than C_0. In this situation, ROAD resets the path to C_1. This is expected to increase reliability, because C_1's reliability is guaranteed (as it is a real case), while G_2 is a ghost case produced after adapting C_0 twice. At this point, the path continues from C_1 and yields the ghost case G_3, which is then adapted to T.

3 Reset-induced Similarity Adjustment

The quality of a CBR system's retrieval plays a critical role in system performance. Retrieval is generally based on similarity, which is used as a proxy for adaptability: the goal of retrieval is to retrieve the most adaptable cases [19]. With a perfect similarity measure, ROAD would never need to reset a path. Consequently, when resets are needed, it reveals deficiencies in the similarity measure. These are opportunities for similarity learning.

RISA uses generated adaptation paths to guide similarity learning. The goal of learning is to adjust the similarity measure so that the case to which the path was reset will become the initial retrieval in the future, enabling adaptation to be performed with fewer steps and decreasing the processing cost of future adaptations. The example in Figure 1 illustrates the potential benefit of ROAD for adaptation path length. In the figure, ROAD would generate the same solution regardless of whether it retrieves C_1 or C_0 in its initial retrieval. However, if the system started by retrieving C_1, it would avoid the effort of building the path from C_0 to C_1.

3.1 The Reset-induced Similarity Adjustment Algorithm

The RISA algorithm is shown in Algorithm 1. RISA takes as input (1) information recorded about resets during adaptation, and (2) a procedure to adjust feature weightings for similarity based on the stored information.

Information recorded about resets: To support RISA, the ROAD implementation was augmented with instrumentation to record its reset behaviors. Each time the system resets a path, it also stores a path segment record of the form $(C_{start}, C_{reset}, T)$ where C_{start} is the case most recently retrieved prior to the reset, C_{reset} is the case retrieved by adapting and resetting from C_{start}, and T is the target case. For the first reset record for a path, C_{start} is the case the CBR system retrieved for the original problem. In each subsequent path segment record that is generated during resetting, C_{start} is the case retrieved for the previous reset—from which the path is continuing—and C_{reset} is the case to which it is reset.

There are two situations in which a path segment record may not include a reset: When the path from the initially retrieved case can be pursued to the target without resetting, and when the path from a reset case can be pursued to the target without further resetting. In those cases the record uses *null* for C_{reset}. The presence of C_{reset} indicates a potential defect in the similarity measure. One strategy for addressing similarity defects, pursued in this paper and elsewhere [3], is to adjust feature weights. Other issues such as insufficient vocabulary knowledge and noisy cases might also lead to resets, but are beyond the scope of this paper.

Adjusting feature weights: For a record $(C_{start}, C_{reset}, T)$ produced by a path reset, RISA adjusts similarity criteria to increase the similarity of C_{reset} and T (pulling them closer), and to decrease the similarity of C_{start} and T (pushing them away from each other). As a result, the case retrieval process is more likely to retrieve C_{reset} directly for future problems similar to T.

A potential issue is that this adjustment may have ramifications for other retrievals, possibly affecting situations in which prior retrievals were correct. Consequently, for a $(C_{start}, null, T)$ record produced by a path not involving reset, RISA pulls C_{start} and T closer, to help preserve the current correct retrieval. The goal is to preserve the ability to generate high quality adaptation paths for similar starting and ending points in the updated similarity measure.

In general, there are many ways in which the push/pull effect could be achieved. For example, a feature weight updating policy can adjust the similarity distance between two cases. In our testbed RISA system for evaluation, we follow the approach of Bonzana, Cunningham and Smyth's ISAC [3]. To pull two cases closer, ISAC increases weightings of their matching features and decreases weightings of differing features. Similarity scores between the two cases

Algorithm 1 Reset-induced Similarity Adjustment

```
Input:
Paths: records of paths ($(C_{start}, C_{reset}, T)$). If the path is never reset $C_{reset} = null$
SM: similarity measure
Pull(SM,A,B): Updating procedure for SM that pulls A and B closer
Push(SM,A,B): Updating procedure for SM that pushes A and B away
Output:
SM: the modified similarity measure

for all $(C_{start}, C_{reset}, T)$ in $Paths$ do
    if $C_{reset} = null$ then
        $Pull(SM, C_{start}, T)$
    else
        $Push(SM, C_{start}, T)$
        $Pull(SM, C_{reset}, T)$
return $SM$
```

are thus increased. Similarly, to push two cases away from each other, ISAC decreases weightings of matching features and increases weightings of unmatching features. ISAC adjusts feature weightings using the update formula:

$$w_i(t+1) = w_i(t) \pm \delta * \frac{F_c}{K_c}, \tag{1}$$

where $w_i(t)$ is the i-th feature weighting at time step t. δ is a fixed value. F_c is the number of times the case has been "falsely retrieved"—retrieved when another case would have been more suitable—and K_c is the number of times the case has been successfully retrieved. In the following evaluation, we apply this update procedure in ROAD, with failed retrievals corresponding to retrievals prompting a reset, and successful retrievals those for which no reset was needed.

3.2 Evaluation of RISA

The evaluation of RISA tested the effect of its similarity adjustment on adaptation efficiency and solution accuracy. Adaptation efficiency was measured by the ability to generate shorter adaptation paths and to decrease the number of resets required during adaptation. Solution accuracy was measured by relative error for a numerical prediction task.

The criteria for adaptation efficiency directly measure the ability of the system to retrieve adaptable cases; we expected RISA to increase the system's ability to do so. We did not expect a strong effect on accuracy, but sought to observe whether the decreased path length brought accuracy benefits. The experiments tested RISA for two alternative similarity schemes: Global weighting and local weighting. Because RISA's feature weight adjustments could have unexpected side-effects on retrievals of distant cases when the adjusted weights are global, we expected better performance for local weightings.

Experimental Design

Task Domain: The evaluation task was automobile price prediction, using the Kaggle automobile dataset [10]. The first two features were removed because they relate to insurance risk. Cases with missing features were removed as well, leaving 193 cases, each with 13 numeric and 10 nominal features in addition to price. Because the need for adaptation paths and difficulty of adaptation are affected by case-base sparsity, the experiments simulated varying levels of sparsity by removing the closest N cases to the target case before each trial, for varying N. For additional discussion of that process, see Leake and Ye [14].

Similarity measure: The similarity between two cases is a weighted sum of feature similarity, with each feature weighted by either global or local weighting. Similarity of nominal features is 1 if they are identical and 0 otherwise; similarity between numerical features is their absolute difference normalized into $[0, 1]$. All feature weights were initialized to the same value.

Global vs. local similarity: We test the effect of RISA for two feature weighting methods: (1) Global weighting relies on a single set of feature weights applied for all similarity comparisons; (2) Instance-specific weighting allows each case to have its own set of feature weightings, used for comparisons to that case [1, 3, 5].

Adaptation Rules: Adaptation rules were generated automatically from the case base using the case difference heuristic (CDH) approach [7]. This approach compares pairs of cases and generates rules that apply when a retrieved case and target case have similar problem differences, and adjusts the solution of the retrieved case according to the solution difference in the case pair from which the rule was learned. The process used here follows the algorithm in Leake and Schack [12]. The rule set generated depends on following parameters:

1. **Rule Count**: The number of rules to generate.
2. **Rule Specificity**: Rule specificity is determined by the number of feature differences to record in the rules. For example, $rspec = 0.1$ if 10% of all feature differences between two cases are included in the rule. Smaller $rspec$ values result in rules that are more generally applicable but less accurate, because their antecedents take into account fewer features.
3. **Rule Generating Distance**: The distance between pairs of cases generating rules. For example, if $ruleGenDist = 0.1$, rules are generated from cases whose difference is less than 10% of the maximum possible difference. A small $ruleGenDist$ value leads to rules covering only small inter-case differences.

A set of 300 rules is generated from pairs of random cases ($ruleGenDist = 1.0$), using half of the feature differences ($rspec = 0.5$). These configuration parameters were chosen based on a simple preliminary experiment to identify rule characteristics for which path lengths were high and increased efficiency would be most useful. In every run of the preliminary experiments, the closest 150

#	Rule Specificity	Rule Gen Dist	Avg Path Len	Avg Path Len after Last Reset
1	0.5	1.0	6.523	2.208
2	1.0	1.0	2.476	1.079
3	1.0	0.2	1.992	0.830
4	0.8	0.2	2.111	0.974

Table 1. Rule Set Configurations and Corresponding Path Lengths

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Path Length	6.975	6.687	6.204	6.101	6.064	5.431
	Sd of Path Length	1.670	1.682	1.666	1.683	1.688	1.663
After RISA	Average Path Length	**6.581**	6.572	**5.850**	**5.548**	**5.651**	5.195
	Sd of Path Length	1.671	1.687	1.620	1.643	1.648	1.666
P Value		**.021**	.505	**.035**	**.001**	**.015**	.17

Table 2. Effect of RISA on Path Length under Global Weighting

stored cases were removed around the query case to increase the need for longer paths. The average length of full paths and the average length of the paths after their last resets are recorded. The difference between the two measures shows the potential saving in efficiency: If the last reset case were retrieved directly, the system would avoid building the path from the original source case to the last reset case. The preliminary experiment used four rule sets of 300 rules. As shown in Table 1, rule set #1 has the longest average length and the biggest proportional benefit, so was chosen as the testbed rule set.

ROAD configuration for experiments: All experiments are based on adaptation path generation by the ROAD system, described in Leake and Ye [14]. The performance of ROAD depends on multiple parameters. The test version of ROAD had the following configurations: (1) Multiple adaptation paths are generated simultaneously (at most 5 paths); (2) Maximum path length is 10; (3) Paths are reset when reliability decays below a threshold or when two paths disagree on the solutions.

Experimental Results

Effect of RISA with Global Weighting Using every case in the case base as a target, by 10-fold cross validation, Table 2 and Table 4 show that the effect of RISA with global weighting consistently decreases average path length and the number of resets. As shown, most differences are significant ($p < 0.05$).

Table 3 shows the effect of RISA with global weights on the average relative error. After RISA, the effect on error is mixed. The rates tend to become worse, but the differences are not significant. We hypothesize that this is due to the coarse-grained nature of updating weights for global weighting. Updating global

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Error	0.481	0.461	0.388	0.410	0.282	0.275
	Sd of Error	0.816	0.814	0.724	0.890	0.544	0.527
After RISA	Average Error	0.465	0.452	0.406	0.419	0.319	0.345
	Sd of Error	0.734	0.799	0.848	0.868	0.672	0.747
P Value		.83	.92	.82	.92	.55	.29

Table 3. Effect of RISA on Error under Global Weighting

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Total Number of Resets	221	261	224	225	248	160
After RISA	Total Number of Resets	177	248	202	182	196	145

Table 4. Effect of RISA on Resets under Global Weighting

weighting influences the similarities between all cases, which can have adverse effects on similarities for cases other than those prompting the updates. Thus with global weighting, RISA decreases path lengths as desired, but with possible degradation of accuracy.

Effect of RISA with Instance-specific Weighting To address the issue of side-effects for adaptation, we tested RISA for local weighting. With local weighting, updating the feature weighting for a case only influences the similarities between this case and other cases, but not the similarities among other cases.

Table 6 and Table 7 show the effects of RISA on efficiency in the instance-specific weighting configuration. In all runs, the number of resets and average path length consistently drop. Most path length results are significant, while the change in error is statistically insignificant, as shown in Table 5.

4 Compatibility-based Adaptation Rule Selection

Commonly, adaptation rules are assumed to be independent, and selected without regard for interactions with other rules. However, it is well known that rules

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Error	0.438	0.381	0.430	0.410	0.340	0.293
	Sd of Error	0.691	0.617	0.835	0.832	0.683	0.529
After RISA	Average Error	0.451	0.393	0.372	0.410	0.321	0.265
	Sd of Error	0.709	0.631	0.673	0.828	0.700	0.501
P Value		.85	.85	.45	.99	.78	.60

Table 5. Effect of RISA on the Error under Instance-Specific Weighting

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Total Number of Resets	296	261	254	232	233	181
After RISA	Total Number of Resets	261	238	237	201	219	160

Table 6. Effect of RISA on Resets under Instance-Specific Weighting

	Number of Cases Removed	150	125	100	75	50	25
Before RISA	Average Path Length	7.154	6.712	6.502	6.218	6.003	5.794
	Sd of Path Length	1.706	1.649	1.654	1.698	1.689	1.718
After RISA	Average Path Length	**6.652**	**6.278**	6.205	**5.737**	**5.672**	5.494
	Sd of Path Length	1.645	1.585	1.618	1.631	1.607	1.673
P Value		**.004**	**.009**	.08	**.005**	**.05**	.08

Table 7. Effect of RISA on Path Length under Instance-Specific Weighting

may not capture all aspects of a situation, resulting in uncertain outcomes, potentially resulting in degradation of adaptation results [13]. An adaptation path might further compound the cumulative error by applying multiple rules. In response, we propose a method for using information about the interactions of adaptation rules to learn which adaptation rules to favor.

Given a case base and an adaptation rule set, it is possible to estimate reliability of adaptation rules by building paths with resetting disabled, and then comparing the final solution from the path with the actual solution of the nearest neighbor. The path's error can then be used to represent the reliability of the rules involved. This is the approach of CARS.

4.1 Compatibility-based Adaptation Rule Selection Algorithm

Algorithm 2 shows the process that CARS uses to assess rule compatibility. It computes a rule compatibility matrix representing the compatibility between every pair of rules, calculated from a set of adaptation paths of length 2, generated based on test adaptations of a selected set of cases. This set of cases could be the entire case base or a subset (for efficiency).

Given *rcount* rules, the compatibility matrix has dimension $rcount \times rcount$. The entry (i, j) in the matrix records that $rule_i$ and $rule_j$ are compatible if $rule_i$ and $rule_j$ can be applied in sequence to a case. If adaptation rules are commutative (e.g., if each rule is multiplying a numerical solution value by a feature difference in a single dimension), entry (i, j) is equal to entry (j, i) and the matrix is symmetric. However, in many domains rules depend on each other and must be applied in a particular order (e.g., recipe generation).

CARS estimates compatibility between two rules based on an estimated error value after the two rules are successively applied to a case in a 2-step adaptation path. Error is estimated by retrieving the stored case closest to the ghost case generated by the adaptation path, and comparing the stored case and ghost case

Algorithm 2 Assessing Rule Compatibility

```
Input:
CaseSet: Cases for testing
rules: List of adaptation rules
Output: Rule compatibility matrix

$R = size(rules)$, $N = size(CB)$
Initialize matrix $M$ of size $R \times R$
for $i \leftarrow 1$ to $R$ do
    for $j \leftarrow 1$ to $R$ do
        $M[i][j] \leftarrow undefined$
for $i \leftarrow 1$ to $R$ do
    for $j \leftarrow 1$ to $R$ do
        $totalError = 0$
        $errorCount = 0$
        for all $case$ in $CaseSet$ do
            if $rule[i].isApplicableTo(case)$ then
                $ghost1 = rule[i].applies(case)$
                if $rule[j].isApplicableTo(ghost1)$ then
                    $ghost2 = rule[j].applies(ghost1)$
                    $target = CaseSet.nearest(ghost2)$
                    $totalError+ = errorInSolution(ghost2, target)$
                    $totalCount+ = 1$
                else
                    continue
            else
                continue
        if $totalError \neq 0.0$ then
            $M[i][j] = totalError/totalCount$
return $M$
```

solutions. After computing the compatibility matrix, CARS uses the average value of each row as a proxy for the overall reliability of the corresponding rule.

CARS compresses the adaptation rule set by retaining only the most reliable rules. To do so, it sorts all rules based on reliability and trims the rule set by retaining only those falling above a selected percentile of the original rule set.

4.2 Evaluation of CARS

The evaluation of CARS addresses two questions:

- How does CARS rule deletion affect efficiency of adaptation path generation?
- How does CARS rule deletion affect solution accuracy?

Efficiency is measured in three ways: number of resets prompted by path reliability decay, number of resets prompted by disagreement between alternative paths being explored, and average adaptation path length. Tests assess the effect of CARS both on ROAD and on a baseline ablated version of ROAD that performs single-rule adaptation instead of using adaptation paths.

Experimental Design

As in the previous experiment, tests used the Kaggle automobile data set [10]. In all runs, after building the compatibility matrix based on the entire case base, 75 cases around the target query are removed to simulate situations where multiple adaptations are needed but the adaptation paths have moderate average length. The experiment uses the previous rule set configuration ($Rcount = 300$, $RuleGenDist = 1.0$, $Rspec = 0.5$), again with 10-fold cross validation. In the experiments, CARS is applied to assess compatibility and retain the top 80%, 60%, 40%, and 20% of rules; these conditions are compared to a baseline of 100% retention. We note that because of the case removals to simulate a sparse case base, as well as the rule retention and path building mechanisms in ROAD, the case bases and rules used to solve the test problems are different from those used when building the rule compatibility matrix.

Experimental Results

Table 8 shows the experimental results. We observe:

- The number of resets consistently decreases when fewer rules are used. With fewer rules the path searching algorithm explores fewer options, decreasing the number of prior cases near the path. In addition, we hypothesize that because the retained rules are more reliable, paths tend to agree with each other more often, decreasing resets due to path disagreement.
- The average path lengths become smaller as fewer rules are retained, because of fewer resets.
- The error improves markedly as the worst 20% of the rules are deleted (80% retention). Observed error is better than the initial rule set for 60% and 40% retention, with the benefit dropping compared to 80% retention, but the differences below 80% are not statistically significant.

Thus the results support the efficiency benefits of rule set compression, as well as improved accuracy at low compression rates. Effects for higher compression rates are an interesting question for future study. We expect a tradeoff between higher average rule quality and lower coverage with the compressed rule sets, resulting in decreasing accuracy when the rule set is too sparse. However, refining the reliability estimate process might help to delay serious accuracy loss.

In summary, using CARS increases the efficiency of both the baseline and ROAD by reducing the number of resets and reducing path lengths, with a stronger effect on ROAD. Also, deletion of an initial set of the worst rules benefits accuracy for both. Beyond that deletion level there is weak or no significant effect on the accuracy until accuracy falls for very high deletion levels.

Percent of Rules Retained	100%	80%	60%	40%	20%
Using Single-Rule Reliability (baseline)					
Resets due to Reliability Decay	227	152	99	37	14
Resets due to Path Disagreement	137	81	72	42	22
Average Error	0.372	0.268	0.323	0.324	0.427
SD of Error	0.407	0.229	0.321	0.281	0.365
P Value (Comparing to no Trimming)	N/A	.002	0.19	0.18	0.16
Average Path Length	6.126	4.689	3.841	2.662	1.932
SD of Path Length	3.011	2.575	2.366	1.630	1.200
P Value (Comparing to no Trimming)	N/A	<.001	<.001	<.001	<.001
Using Reliability from Compatibility Matrix (CARS)					
Resets due to Reliability Decay	227	49	17	1	0
Resets due to Path Disagreement	137	83	41	10	1
Average Error	0.372	0.303	0.305	0.356	0.420
SD of Error	0.407	0.267	0.244	0.281	0.316
P Value (Comparing to no Trimming)	N/A	.049	.05	.65	.19
Average Path Length	6.126	3.322	2.346	1.548	1.231
SD of Path Length	3.011	2.053	1.366	0.719	0.468
P Value (Comparing to no Trimming)	N/A	<.001	<.001	<.001	<.001

Table 8. Performance by Rule Retention Level

5 Related Work

RISA is a learning method for refining similarity criteria; CARS is a method for prioritizing adaptation rules for retention.

Learning to Refine Similarity Criteria: An extensive body of CBR research has addressed similarity learning (see, for example, Wettschereck et al. [20], and, for a recent overview, Mathisen et al. [18]). RISA differs in aiming to refine existing similarity criteria on the fly, rather than generating a similarity measure from scratch. RISA's failure-driven method for learning to refine similarity criteria is most closely related to Bonzano, Cunningham and Smyth's ISAC [3], which adjusts similarity weights in response to failed and successful retrievals, and whose updating strategy is applied by RISA. This work is also in the spirit of work on building similarity measures by Xiong and Funk [22], which adjusts local similarity to reflect the utility of pairs of cases.

Learning to Prioritize Adaptation Rules: Hanney and Keane's initial proposal for generating adaptation rules by the case difference heuristic also proposed selective retention, based on the frequency with which particular rules were generated from cases [8]. Adaptation rule maintenance to remove duplicates and resolve conflicts was proposed by Li et al. [15]. Additional work focuses on how to prioritize rule selection, without deleting rules from the rule set [11], and on combining systematic accuracy testing with retention of top-ranked rules [9]. The current work differs in addressing adaptation paths, rather than only individual rules, and in focusing on minimizing adaptation path length.

6 Conclusion and Future Directions

This paper presents an initial investigation of applying information from adaptation paths to improving efficiency of path-based adaptation. It proposes two knowledge-light approaches, focusing on similarity measures and adaptation rules. By learning from path generation failures, as shown by path resets in ROAD, RISA helps align the similarity measure with adaptability. Experimental results support its value for retrieving more adaptable cases. By favoring pairs of rules that have participated in successful adaptation paths, CARS compresses the adaptation rule set while retaining useful rules. Experimental results support its value for increasing adaptation efficiency and that deletion of least reliable rules can improve accuracy, subject to an efficiency/coverage tradeoff.

A next step is to test the methods for additional domains and to gather information on the domains for which the methods are most appropriate. For similarity maintenance, we also plan to compare the alternative strategy of learning only from the final reset rather than all intermediate resets. Learning from the final reset would focus on a more definitive retrieval, but would also reduce the amount of data available to the learning algorithm. For adaptation rule set compression, an interesting question is whether the criteria for assessing rule usefulness could be refined, for example, by considering compatibility and result accuracy separately, to prioritize rule retention based on a composite criterion. Another interesting question is the possible benefit of retaining rules with limited compatibility but adding constraints to avoid their use in combination.

A future opportunity is to apply information from adaptation paths for maintaining the case-base. Mathew and Chakraborti [17] show the value of taking potential adaptation chains into account when guiding case retention. In ROAD, parts of the problem space for which ghost cases are generated suggest gaps in the case base. Based on the CBR premise that similar problems are likely to recur in the future, those gaps are good candidates for case acquisition to increase efficiency by shortening commonly-used adaptation paths.

References

1. Aha, D.W., Goldstone, R.L.: Concept learning and flexible weighting. In: In Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society. pp. 534–539. Erlbaum (1992)
2. Badra, F., Cordier, A., Lieber, J.: Opportunistic adaptation knowledge discovery. In: Case-Based Reasoning Research and Development, ICCBR 2009. pp. 60–74. Springer, Berlin (2009)
3. Bonzano, A., Cunningham, P., Smyth, B.: Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In: Proceedings of the Second International Conference on Case-Based Reasoning (ICCBR-97). pp. 291–302. Springer, Berlin (1997)

4. D'Aquin, M., Lieber, J., Napoli, A.: Adaptation knowledge acquisition: a case study for case-based decision support in oncology. Computational Intelligence 22(3/4), 161–176 (2006)
5. Friedman, J.H.: Flexible metric nearest neighbor classification. Tech. rep., Stanford University (1994)
6. Hammond, K.: Case-Based Planning: Viewing Planning as a Memory Task. Academic Press, San Diego (1989)
7. Hanney, K., Keane, M.: Learning adaptation rules from a case-base. In: Proceedings of the Third European Workshop on Case-Based Reasoning. pp. 179–192. Springer, Berlin (1996)
8. Hanney, K., Keane, M., Smyth, B., Cunningham, P.: What kind of adaptation do CBR systems need? a review of current practice. In: Proceedings of the Fall Symposium on Adaptation of Knowledge for Reuse. AAAI (1995)
9. Jalali, V., Leake, D.: On retention of adaptation rules. In: Case-Based Reasoning Research and Development, ICCBR 2014. Springer, Berlin (2014)
10. Kaggle: Automobile dataset (2017), data retrieved from Kaggle, `https://www.kaggle.com/toramky/automobile-dataset`
11. Leake, D., Dial, S.: Using case provenance to propagate feedback to cases and adaptations. In: Proceedings of the Ninth European Conference on Case-Based Reasoning. pp. 255–268. Springer (2008)
12. Leake, D., Schack, B.: Exploration vs. exploitation in case-base maintenance: Leveraging competence-based deletion with ghost cases. In: Case-Based Reasoning Research and Development, ICCBR 2018. pp. 202–218. Springer, Berlin (2018)
13. 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. pp. 194–208. Springer-Verlag, Berlin (2007)
14. Leake, D., Ye, X.: On combining case adaptation rules. In: Case-Based Reasoning Research and Development, ICCBR 2019. pp. 204–218. Springer (2019)
15. Li, H., Hu, D., Hao, T., Wenyin, L., Chen, X.: Adaptation rule learning for case-based reasoning. In: Semantics, Knowledge and Grid, Third International Conference on. pp. 44–49 (2007)
16. López de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision, and retention in CBR. Knowledge Engineering Review 20(3) (2005)
17. Mathew, D., Chakraborti, S.: Competence guided model for casebase maintenance. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. pp. 4904–4908. IJCAI (2017)
18. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. Progress in Artificial Intelligence (10 2019)
19. Smyth, B., Keane, M.: Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. Artificial Intelligence 102(2), 249–293 (1998)
20. Wettschereck, D., Aha, D., Mohri, T.: A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review 11(1-5), 273–314 (February 1997)
21. Wilson, D., Leake, D.: Maintaining case-based reasoners: Dimensions and directions. Computational Intelligence 17(2), 196–213 (2001)
22. Xiong, N., Funk, P.: Building similarity metrics reflecting utility in case-based reasoning. Journal of Intelligent and Fuzzy Systems 17(4), 407–416 (2006)