Simulated Assessment of Ripple Round Rules

Ivan Bindoff and Byeong Ho Kang

University of Tasmania, School of Computing and Information Systems
{Ivan.Bindoff,bhkang}@utas.edu.au

Abstract. A new Ripple Down Rules based methodology which allows for the creation of rules that use classifications as conditions has been developed, and is entitled Multiple Classification Ripple Round Rules (MCRRR). Since it is difficult to recruit human experts in domains which are appropriate for testing this kind of method, simulated evaluation has been employed. This paper presents a simulated evaluation approach for assessing two separate aspects of the MCRRR method, which have been identified as potential areas of weakness. Namely, "Is the method useful in practice?" and "Is the method acceptable, computationally?" It was found that the method appears to be of value in some, but not many, "traditional" multi-class domains, and that due to computational concerns with one aspect of the method it is considered unsuitable for domains with a very large number of cases or rules. These issues are discussed and solutions are proposed.

Keywords: ripple, down, rules, multiple, classification, round, configuration, knowledge acquisition, simulated, expert, assessment.

1 Introduction and Previous Work

Knowledge based systems are notoriously difficult to evaluate objectively for several reasons. Particularly, it is difficult to get experts to give their time to the training of the project and it is virtually impossible to make them train the same system multiple times. Furthermore, it is difficult to get a true gauge of how correct or optimal the system really is, as it is difficult to convince the expert or other independent experts to verify the results of the system after the fact. To compound these problems further still, different experts will have different opinions as to what correct is, and are likely to perform somewhat inconsistently from one train of a system to the next. Because of these features, multiple experts should really be assessed and contrasted multiple times each. Since experts are - virtually by definition - scarce, and their time valuable, this kind of assessment is very rare [1]. To overcome this problem, simulated experts have been previously employed for the task of evaluating new expert systems methodologies, particularly in Ripple Down Rules (RDR) related studies, such as this one [2-4].

Having developed a new RDR based method, and having very limited access to human experts with appropriate domains to evaluate, it was considered necessary to again employ simulated experts to evaluate the method. However, the particular

© Springer-Verlag Berlin Heidelberg 2010

characteristics of the method required some modifications to the existing simulated expert evaluation process.

2 Ripple Down Rules

The RDR methodology makes use of a true-false binary tree structure in order to ensure that rules are always added in context [5, 6]. An example of this structure is shown in Figure 1.

Using this same sample knowledge base we can also consider the inference process that is used in the RDR methodology. If we consider a sample case in which we have [X=5, Y=5, Z=10] then we will consider rule 1 initially and find that X>4 is true, so rule 2 will be considered. Y<3 is false, so rule 7 must be considered. The condition Z<9 is also false, but as there is no false branch deriving from rule 7 we have hit a dead end and as such will simply fire the last considered true statement, which results in rule 1 firing [4].

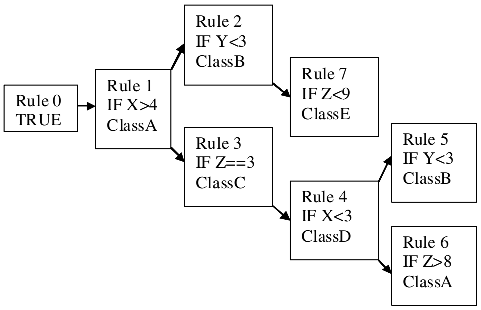

Fig. 1. A sample RDR knowledge base where arrows pointing upwards indicate the TRUE path while arrows heading downwards indicate FALSE paths

RDR, as described thus far, still encounters problems with maintainability. The addition of a new rule may cause previously considered cases to become misclassified, however, this problem was easily solved. When creating a new rule, the system would store the current case against that new rule as a "cornerstone" case, a copy of the case as it existed when the rule was created. Later, when creating further rules, the system will detect if this new rule caused a conflict with the past cornerstone. That is, did the new rule change the cornerstone's classification? If so, then at this stage the expert was required to select a relevant difference between the current case and the cornerstone case [7, 8]; this process is often called the validation and verification stage of RDR.

The RDR method as described is unsuitable for multiple classification tasks, since it would require the use of either multiple knowledge bases or compound classifications, which can cause an undesirable explosion in the amount of knowledge required [1].

To extend this method to multiple classification tasks, Kang altered the underlying knowledge representation structure to that of an n-tree, altered the cornerstone case approach such that multiple cornerstone cases might apply to a single case, and modified the inference strategy such that it would not cease operation when a correct classification was found, but instead would add any deepest-satisfied node to the result set [4]. This method was entitled Multiple Classification Ripple Down Rules (MCRDR).

RDR methods, unlike many proposed expert system methodologies, have proven to be quite valuable in real world tasks, with one very successful commercial system for pathology interpretation [9], another for tesco.com [10], and several commercial applications under development in areas including high volume call centre management [11] and medication review [12].

2.1 Using Classifications as Conditions

One of the compromises that were made when developing the RDR methodology, when compared to some traditional expert systems methodologies, was that the ability to create rules which used classifications as conditions was lost. This ability had previously been integral to success in some domains, and is of particular value in complex configuration and planning tasks where the positioning of each module/event may influence the positioning of other modules/events.

This shortcoming was recognized quite early, with Mulholland developing an RDR based system to solve Ion Chromatography configuration tasks, although this solution was highly specialized and would require extensive redesign in order to be applied in other domains [13]. It was also unreliable, since it was possible to create rules which would cause infinite cycles, which necessitated a program halt, and had to be removed manually after they were observed [13]. After this, Beydoun & Hoffman developed Nested RDR, a single classification RDR approach which allowed the creation of "intermediate classifications", as stepping stones towards the end classification [14]. This method was more generally applicable than Mulholland's offering, being not targeted at a single, specific domain, but was targeted primarily at single classification problem domains, with intermediate classifications being treated separately to true classifications [14]. Later, a proposition was made for a more generalized version of RDR, which included provision for a Repeat Inference MCRDR (RIMCRDR) approach whereby the existing MCRDR method was augmented with the ability to use classifications as conditions [15, 16]. However, this approach included many restrictions as to when and how these types of rules could be used, and about how the knowledge base must be inferred and interpreted. This was done in an attempt to eliminate the potential for cyclic rules – rules which depend on the existence of a classification, yet upon firing, may cause that same classification to be retracted. RIMCRDR asserts that rules must be inferred in strict chronological order, and that no retractions are allowed [16]. In making these restrictions, it is felt that the RIMCRDR method will alienate some experts (since they will feel artificially restricted when the system does not allow them to use a classification as a condition in a rule simply because it is an exception), and make it unsuitable for many complex domains where retractions may be necessary.

Having identified these concerns, an attempt was made to develop a new multiple classification RDR (MCRDR) based method which would preserve all the essential benefits and strategies of the RDR method, while augmenting it with the ability to create rules which can use classifications as conditions. It was determined that it should do this while offering minimal restrictions as to when and where the expert may define these rules.

3 Method

In order to achieve the goals outlined above, the authors revised the existing MCRDR approach substantially. Changes were made to both the knowledge representation approach and the inference strategy. These changes in turn necessitated some revisions to the knowledge acquisition process, including provisions for the detection of cyclic rule definitions and an update to the cornerstone case mechanism. The resultant method has been entitled Multiple Classification Ripple Round Rules (MCRRR).

3.1 Knowledge Representation

The existing MCRDR knowledge representation, an n-tree, was deemed inadequate for the purposes of this new method. It was, however, desirable to maintain all the essential benefits that this structure offers. To this end, the structure was altered, becoming essentially a directed graph, similar in some ways to Gaines' Exception Directed Acyclic Graph [17]. Importantly, the underlying n-tree structure was still present, with each rule being a node with one or more conditions and a classification; however, nodes were given the added ability to store zero or more classifications, which must be satisfied in order for the rule to fire. These classifications could each be defined as classifications which either must be present, or must *not* be present in order for the rule to fire. These additions were termed switches, to reflect their simple mode of operation. Each switch represented a classification, and maintained a counter such that whenever a classification was added to the result set during the inference strategy, every switch concerning that classification was incremented. Conversely, if a classification was removed from the result set, the counter was decremented. It was important that these switches be a counter, rather than a Boolean, since it is entirely possible that a particular classification may be reached by more than one path, making it necessary to know how many instances of that classification are still present in the result set.

In order to know when and where updates were required during an inference process, a store of dependencies was also necessary. By maintaining a list of dependents and dependencies for each classification, it can be easily determined which nodes must be revisited and re-inferred whenever a classification was added or removed from the result set.

3.2 Inference

The typical MCRDR inference strategy is quite simple, being largely similar to that of a depth first search, where the deepest satisfied node is added to the result set, only where the search does not stop until every node at the first level of the tree has been traversed.

The new inference strategy must be substantially more complex, since there is now the possibility of nodes being revisited and of results being removed. There is also the additional need to check that all switches are active, before adding a rule, but this is a trivial step. The new inference algorithm for MCRRR is shown in a simplified pseudo-code form here.

```
infer(Node, Case) :-
{
        clearResult(Node, Case)
        If (Node’s rule is satisfied
        AND all of its children’s rules aren’t)
               If (All Node’s parent rules are satisfied)
                      Add Node to result list
                      Mark Node as having fired
                      Activate all dependents of Node’s class
                             For each node that changed state
                             infer(ActivatedNode, Case)

        If (Node has a non-root parent rule)
               clearResult(Node’s parent)
        For each Child that isn’t marked as avoid
               Clear avoid markers
               infer(Child, Case)
}

clearResult(Node, Case) :-
{
        If (Node is marked as having fired)
               Remove it from the result set
               Clear its fired flag
               Deactivate all dependents of Nodes class
               For each node that changed state
                      infer(DeactivatedNode, Case)
               If (Node has a non-root parent rule
               AND all Node’s parent rules are satisfied
               AND no siblings are satisfied)
                      Mark Node to avoid
                      infer(Node’s parent, Case)
}
```

3.3 Knowledge Acquisition

From the user’s perspective, the knowledge acquisition process remains largely unchanged from MCRDR. They are presented with a case, and the system’s current “belief” regarding which classifications apply to it. If the expert believes a classification is missing, or that a classification is incorrectly provided by the system, they indicate as such and enter the rule creation process much as is done in MCRDR. During this process the expert may select what the classification should be and which valid conditions of the case are relevant to this classification, as normal. The only difference here is that the expert is also able to select any of the classifications the system is currently aware of as conditions of the case. If the classification is currently present on the case, then it can be added to reflect that the classification must be present in order for this rule to fire. If it is not currently present, the condition will be added to reflect that the classification must *not* be present in order for this rule to fire.

Cycles. Where the knowledge acquisition process does change is largely behind the scenes. Whenever the expert creates a rule, the system must check that their new rule does not have any potential to cause a cycle in the knowledge base. An example of a cycle can be seen in Figure 2, where each node is represented by 3 boxes, the top left being the switches which must be on for the rule to fire, the bottom left being the conditions which must be satisfied, and the bottom right being the classification that the rule will add to the result set if it fires. It can be seen that in a case where X, Y and Z are all satisfied, the inference algorithm would add ClassA, which in turn would cause it to consider the next rule, adding ClassB, which in turn would add the ClassC rule. This ClassC rule is an exception to the rule which added ClassA, and thus would remove ClassA from the result set, which would in turn remove ClassB and so forth, with no termination possible.

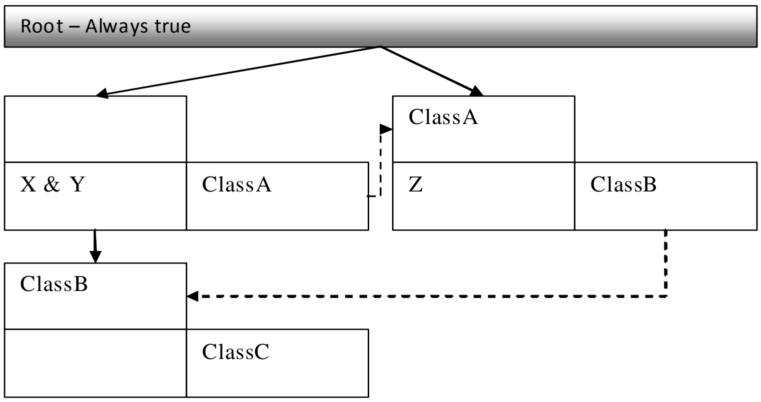

Fig. 2. A simple example of a cyclic rule set

Perhaps the most common method of detecting cycles in a directed graph structure is to perform a topological sort on the graph, as described by Kahn [18]. If a sorted topology cannot be found then there is a cycle. This method is efficient when considering whole graphs, however it was noted that in the context of this method it is only necessary to check the dependencies that are relevant to the new rule. As such, a method was used where the dependencies of all the classifications present in the new rule were examined in turn to see if they lead to a condition where the classification was no longer valid. That is, is there a dependency chain such that the classification might ultimately be dependent on itself being false? If a potential cycle was identified, the expert was informed that their new rule could cause conflicts, and was asked to revise the rule until no further conflicts were found. However, this approach is not expected to perform, in a computational sense, any better than Kahn's topological sort.

Cornerstone cases. It was also necessary to adjust the cornerstone case strategy in light of the alterations made to the method. In MCRDR, a cornerstone case can be robustly defined as any case which has been previously considered and approved by the expert, but which would be altered by the addition of the new rule that the expert is trying to create [4]. From the user's perspective, the cornerstone case process remains essentially unchanged here. When they attempt to add a rule to the knowledge base, the system will prompt them with a list of any past cases which would be altered by this new rule, and they must either select differences which will eliminate the cornerstone cases, or they must confirm that the new rule should in fact alter the past case. However, behind the scenes, there is a loss of efficiency. MCRDR was able to interrogate cornerstone cases very efficiently, as a reference to each cornerstone case could be stored against the nodes they relate to. With MCRRR this approach may become impractical, since it would now be necessary to also store each classification of each case in addition to the attributes. This in itself would not be such a problem, until it is considered that the classifications of the past cases may also be changed by the addition of the new rule to the knowledge base. When the complexities of attempting to maintain this library of cornerstone cases were fully considered, it was deemed simpler to just re-infer all past cases with a temporary version of the knowledge base that included the new rule. Each case which had the new rule in its result set after inferencing would be considered a cornerstone case. This is likely to cause significant efficiency concerns as both the number of past cases and the size of the knowledge base grows, but it was felt that the majority of systems are sufficiently small to be managed with this approach. However, this hypothesis required testing.

3.4 Task

At this stage, the new method required testing and validation in several areas.

Obviously it was unknown whether the issue of cycles might become a limiting factor in the expert's use of the system, in a true configuration task. An evaluation of this was performed using a human expert and a blocks placement configuration task. However, this evaluation will not be described here, due to limited space.

Another unknown quantity was precisely how useful the additional features of MCRRR might be, compared to the existing MCRDR approach.

Additionally, it had been identified that there was a risk of the computational performance of the method becoming inadequate as the knowledge base and number of cases grew, particularly when adding rules, since, in its current form, it necessitates an inference of every past case in the system for each added rule. Each inference takes longer as progressively more rules are added to the system, and more inferences must be run as progressively more cases are assessed by the expert. However, it was unclear as to when these computational concerns might become apparent, or what effect the use of classifications as conditions or exceptions, or combinations thereof might also have.

To test these concerns two forms of simulated assessment were undertaken.

3.4.1 Simulated Experts and Grouping Rules

The first form follows closely with the simulated expert evaluation approach used initially by Compton to evaluate RDR [2], and Kang for MCRDR [4]. It has subsequently been used several times [2, 3]. Under this approach, four sets of simulated experts are trained using InductRDR [19] on an established single classification machine learning dataset, by randomizing the order in which cases are seen. Each simulated expert is then applied to the dataset incrementally on a case-by-case basis and interrogated to determine what rule conditions it used to reach its indicated classification. The expert is simulated at several levels of proficiency by using a certain portion of the total number of conditions which are identified for each classification. For example, a “stupid” expert might use only 25% of the known conditions, thus making very non-specific rules which have a high chance of causing later errors, whilst a “clever” expert might use 75% of them, resulting in a low chance of the rule causing later errors. For this study, each expert had four incarnations, ones that used 25%, 50%, 75%, and 100% of the available rule conditions respectively.

However, there has been significant progress made in the field of multi-label machine learning methods since Kang’s efforts were undertaken [1]. As such, the approach has been updated by employing the binary relevance classifier found in the Mulan [20] extension to Weka [21]. This allows for the creation of a genuine multi-class simulated InductRDR simulated expert, requiring only minor modifications to the technique.

The resultant MCRDR knowledge base could then be compressed into a crude MCRRR knowledge base, by searching for groups of conditions which are used several times over, and replacing them instead with a grouping rule. These “grouping” rules can be assessed for efficacy by considering the overall reduction in the number of conditions they provide. Unfortunately, this reduction is unlikely to be particularly significant. Consider the example of a grouping rule provided in Figure 3. This example indicates a situation where P,Q and X,Y are used to reach an intermediate class “ClassA”, which can then in turn be used to reach a final class “Class B”. The total number of conditions in this knowledge base is 6. If we instead represented this example without using the grouping rule, we would have two rules: “If P,Q,H then ClassB” and “If X,Y,H then ClassB”, which also contains 6 conditions. To actually reduce the number of conditions in the knowledge base, the grouping rule must be used more than once, but even then its improvement on the overall number of conditions is only fractional. Despite this, it is felt that the reduction in conditions metric is a useful measure of how effective the MCRRR additions are for a given domain. It must simply be kept in mind that the MCRRR additions may have other, less measurable benefits, when used by a real human expert on a complex domain.

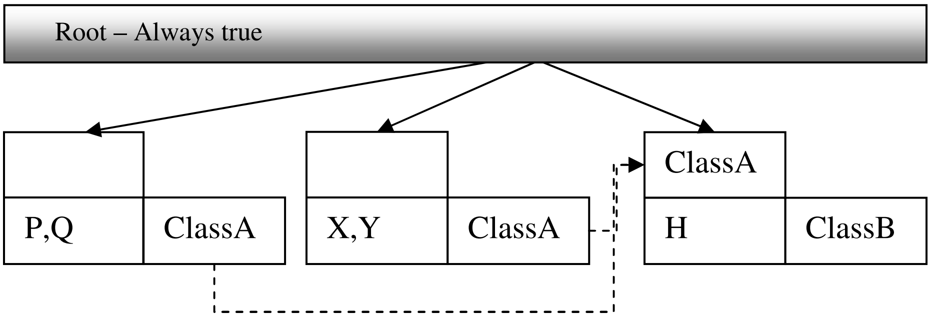

Fig. 3. A simple example of a (bad) grouping rule

The test was run for 7 freely available multi-label datasets, as shown in Table 1.

Table 1. The multi-label datasets used

		Attributes					
Name	**Domain**	**Instances**	**Nominal**	**Numeric**	**Labels**	**Cardinality**	**Density**
Emotions	Music	593	0	72	6	1.869	0.311
Genbase	Biology	662	1186	0	27	1.252	0.046
Scene	Multimedia	2407	0	294	6	1.074	0.179
Yeast	Biology	2417	0	103	14	4.237	0.303
Bibtex	Text	7395	1836	0	159	2.402	0.015
Enron	Text	1702	1001	0	53	3.378	0.064
Medical	Text	978	1449	0	45	1.245	0.028

3.4.2 Stress Test

The second method of simulated assessment undertaken was essentially a stress test, designed to test the computational performance of the method under a range of knowledge base environments. To do this, a rule generator was developed which would incrementally examine cases and create rules at random which satisfy the case. It would continue examining cases, creating one rule per case, until the limit (1000 rules/cases) was reached. Two adjustable parameters were used; the first indicated how likely each randomly generated rule was to be an *exception* to an existing rule (also selected at random), and the second indicated how likely they were to use a classification as a condition (the classification was also picked at random from the list of all classifications. The stress test was run for the enron (nominal) and scene (numeric) datasets, with a range of parameters, seen in Table 2. Each test was run 4 times.

Table 2. The parameters used in the stress test of each dataset

Rules	Cases	Exceptions	Classifications
1000	1000	10%	10%
1000	1000	20%	10%
1000	1000	40%	10%
1000	1000	10%	20%
1000	1000	20%	20%
1000	1000	40%	20%
1000	1000	10%	40%
1000	1000	20%	40%
1000	1000	40%	40%
1000	1000	10%	80%
1000	1000	20%	80%
1000	1000	40%	80%

It is anticipated that a performance in the rough order of $O(n^2)$ will be observed, reflecting an actual value of (cornerstone cases * rules), where cornerstone cases and rules are equivalent. However, it is also expected that the real time will increase in the simulations with higher intermediate rule chances, although not leaving order $O(n^2)$. In light of the fact that equivalent values of cornerstone cases and rules are being used, a 100 case inference was performed 9 times throughout each experiment to confirm that the actual time taken specifically to inference was progressing linearly with the number of rules in the system. The addition of this check allows for confirmation through elimination that the n^2 operation was a result of the combined effect of increasing numbers of cornerstone cases as well as increasing numbers of rules.

4 Results and Discussion

4.1 Simulated Experts and Grouping Rules

For each dataset, a table is shown to represent the average number of conditions in the knowledge base, both before and after conversion to MCRRR through the addition of grouping rules. The average reduction in conditions is also calculated.

Shown in Table 3 are the results found for each of the datasets tested. It can be seen here that the reduction in conditions was fairly minor, only 4.87% in the best case for the enron dataset. The emotions dataset also saw a modest reduction, despite having a limited number of rules to compress. The yeast, scene, bibtex, and medical datasets saw only minor reductions, while the genbase dataset simply did not have enough complex rules to see any reduction at all.

Table 3. The number of conditions in the various knowledge bases, before and after MCRRR conversion

	Avg. Conditions		
bibtex	**Before**	**After**	**Reduction**
25%	644	640.25	0.58%
50%	697.25	687.75	1.36%
75%	1244.25	1214.25	2.41%
100%	1467	1442.5	1.67%
emotions			
25%	56.3	55.5	1.42%
50%	52.7	51.9	1.52%
75%	84.3	81.7	3.08%
100%	160.3	154.2	3.81%

Table 3. (*continued*)

enron			
25%	956.6	918	4.04%
50%	925.7	880.6	4.87%
75%	1108.1	1059.7	4.37%
100%	1948.6	1860.1	4.54%
genbase			
25%	22.9	22.9	0.00%
50%	23.7	23.7	0.00%
75%	26.1	26.1	0.00%
100%	29.8	29.8	0.00%
medical			
25%	110.5	109.5	0.90%
50%	113.4	112	1.23%
75%	151.8	149.9	1.25%
100%	222.6	219.8	1.26%
scene			
25%	123.1	120.9	1.79%
50%	93.9	93.6	0.32%
75%	273.5	268.1	1.97%
100%	775.1	759.6	2.00%
yeast			
25%	122.1	120.6	1.23%
50%	160.5	157.2	2.06%
75%	563.6	550.4	2.34%
100%	1552	1520.4	2.04%

Of the datasets tested here, only the emotions and enron dataset appear to stand out as having any particular use for the grouping rules feature, with the enron dataset achieving respectable compression rates, and the emotions dataset achieving a surprising level of compression despite having limited rules to work with. Referring back to Table 1, unfortunately, no similarities can be detected between the properties of these two datasets. Of those datasets which performed relatively poorly, genbase, medical, scene, and bibitex, there also appears to be no identifying characteristics in Table 1.

As disappointing as these findings may be, they are not surprising. One feature all the datasets tested here share is that they were all designed for machine learning applications, and thus are all traditional machine learning problems. Certainly none of these datasets are configuration or planning tasks, areas in which it is expected that the additional features of MCRRR might become required. Further to this, the metric itself is perhaps flawed, since an overall reduction in the number of conditions required to solve the problem may please experts because they potentially have less work to do to achieve the same result, but it is never expected to be the primary reason why an expert might want the ability to use classifications as conditions in their rule. One can imagine that the expert might want these features simply because expressing the rules in that manner "*makes more sense*" to them. So, although measuring condition reductions such has been done here might be easy, it appears, from these results at least, that it is not useful.

4.2 Stress Test

It was anticipated that a computational performance in the order of $O(n^2)$ would be observed, reflecting O(cornerstone cases * rules), where the number of cornerstone cases and the number of rules were roughly equal in quantity. This outcome was observed, for all parameters on both the enron and scene datasets. The effect of increasing the likelihood of exceptions had little to no impact on the outcome. However, increasing the likelihood of classifications as conditions did cause more sporadic outcomes, although never order of magnitude alterations. Examples of this can be seen in Figures 4 and 5.

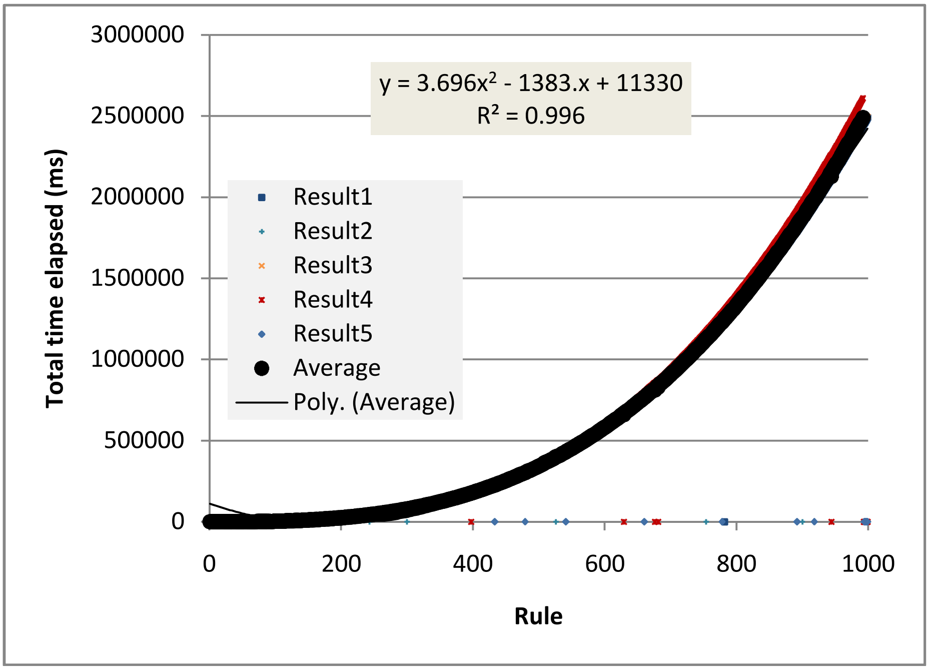

Fig. 4. A simulation with a 10% chance of exceptions and a 10% chance of rules using classifications

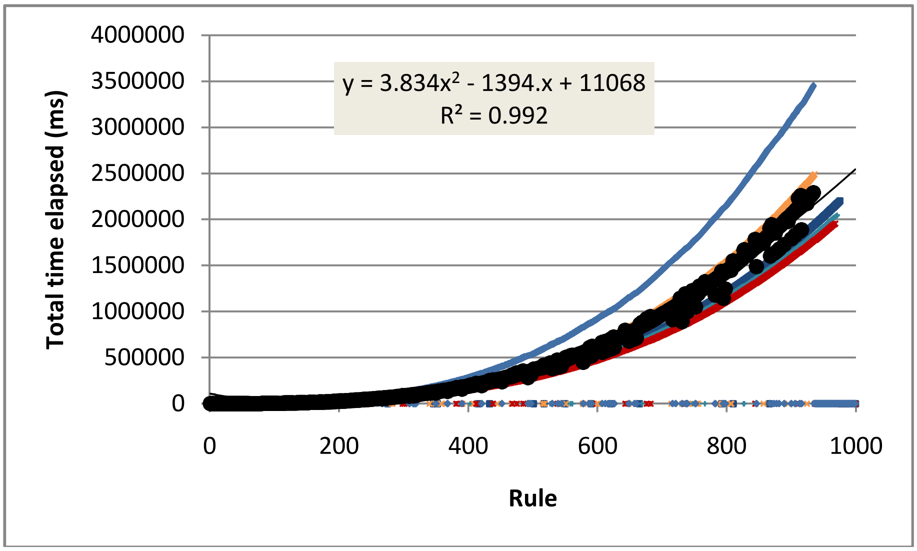

Fig. 5. A simulation with a 10% chance of exceptions and an 80% chance of rules using classifications

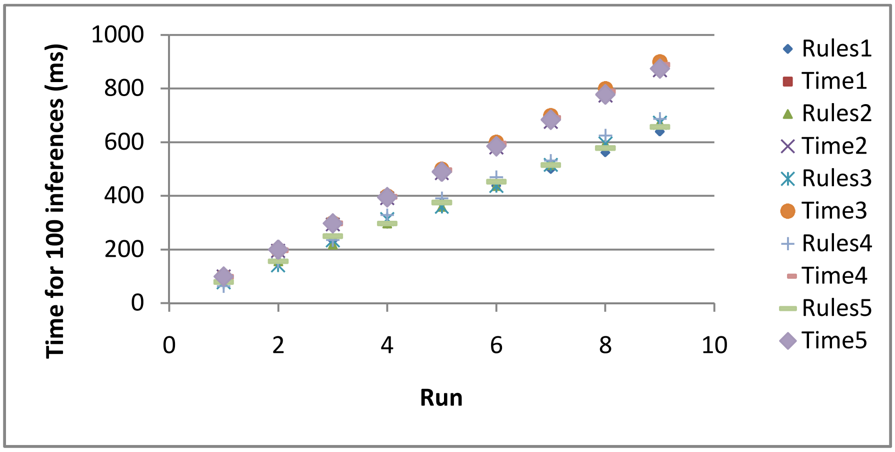

Fig. 6. The runtime of 100 inferences as the knowledge base is populated with gradually more rules

In order to demonstrate that the inference algorithm did perform linearly, proportional to the number of rules in the system, the performance of the inference algorithm alone was also measured. This was done by measuring the time it took to perform 100 consecutive inferences on a knowledge base at 9 points, after 100 rules were added, 200 rules were added etc. The result is shown in Figure 6, and indicates that the runtime of the inferences is directly and linearly proportional to the number of rules in the system, with the lower cluster of markers in each instance being the number of rules in the system, and the higher cluster being the time taken to inference that knowledge base 100 times.

The unfortunate result of the performance measured here is that MCRRR in its current state should be considered unsuitable for domains where there is a very large number of cases or rules. On current hardware, the time taken to add a rule would become unacceptable after large numbers (>1500) of cases had been seen, and rules added.

5 Conclusions and Further Work

It has been determined that the MCRRR method is currently of limited value in many traditional domains, at least in the sense that the expert will find only limited use of grouping rules. However, it is still felt that the method may be of substantial value in other domains, such as configuration or planning tasks. However, proof of this remains to be published.

Unfortunately, it appears that the method as it currently stands is unsuitable in very large domains where many rules may be required, or many cases may be examined. However, it is thought that the computational performance of adding rules can yet be substantially improved. Currently, every cornerstone case must be inferenced for every rule added. However, through the appropriate application of indexing and searching strategies it should be possible to eliminate many of the past cases from the potential cornerstone case bank for any given rule, by simply examining the conditions of the rule. Of particular interest is the possibility of indexing cornerstone cases based on the values of their attributes, as well as the classifications they have. Having produced such an index it should be possible to substantially reduce the number of cornerstone cases which must be inferred upon, since many may be eliminated immediately due to not matching the search criteria of the new rule. This could potentially enable an order of magnitude performance increase, thus making the method applicable to even very large problem domains. It is felt that this is the important next step for MCRRR, and that with this problem resolved there will be no reason to justify using MCRDR rather than MCRRR, apart from the additional complexity of implementation.

References

1. Kang, B., Compton, P., Preston, P.: Multiple Classification Ripple Down Rules: Evaluation and Possibilities. In: AIII-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems, Banff (1995)
2. Compton, P., Preston, P., Kang, B.H.: The use of simulated experts in evaluating knowledge acquisition (1995)
3. Dazeley, R., Kang, B.H.: Rated MCRDR: Finding non-Linear Relationships Between Classifications in MCRDR (2003)
4. Kang, B.H.: Validating knowledge acquisition: multiple classification ripple-down rules. In: Computer Science and Engineering. University of New South Wales (1995)
5. Compton, P., Jansen, R.: A philosophical basis for knowledge acquisition. In: European Knowledge Acquisition for Knowledge-Based Systems, Paris (1989)

6. Compton, P., Kang, B.H., Preston, P., Mulholland, M.: Knowledge Acquisition without Analysis. In: Knowledge Acquisition for Knowledge-Based Systems. Springer, Heidelberg (1993)
7. Kang, B., Compton, P.: A Maintenance Approach to Case Based Reasoning. Advances in Case-Based Reasoning (1994)
8. Preston, P., Edwards, G., Compton, P.: A 2000 Rule Expert System Without a Knowledge Engineer. In: AIII-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems, Banff (1994)
9. Compton, P., Peters, L., Edwards, G., Lavers, T.G.: Experience with ripple-down rules. Knowledge Based Systems 19(5), 356–362 (2006)
10. Sarraf, Q., Ellis, G.: Business Rules in Retail: The Tesco.com Story, cited (2007)
11. Vazey, M., Richards, D.: Troubleshooting at the Call Centre: A Knowledge-based Approach (2005)
12. Bindoff, I., Tenni, P., Peterson, G., Kang, B.H., Jackson, S.: Development of an intelligent decision support system for medication review. J. Clin. Pharm. Ther. 32(1), 81–88 (2007)
13. Mulholland, M.: The Evaluation of the Applicability of Artificial Intelligence Software to Solving Problems in Ion Chromatography, in Chemistry. University of New South Wales (1995)
14. Beydoun, G., Hoffmann, A.: NRDR for the Acquisition of Search Knowledge. LNCS, pp. 177–186. Springer, Heidelberg (1997)
15. Compton, P., Richards, D.: Extending ripple down rules (1999)
16. Compton, P., Richards, D.: Generalising ripple-down rules. In: Knowledge Engineer-ing and Knowledge Management: Methods, Models, Tools, pp. 2–6 (2000)
17. Gaines, B.R.: Exception dags as knowledge structures
18. Kahn, A.B.: Topological sorting of large networks. Communications of the ACM 5(11), 558–562 (1962)
19. Gaines, B.R., Compton, P.J.: Induction of ripple down rules (1992)
20. Tsoumakas, G., Katakis, I.: Multi-Label Classification: An Overview. International Journal of Data Warehousing and Mining 3(3), 1–13 (2007)
21. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco (2005)