DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch017

Rapid Rule Compaction Strategies for Global Knowledge Discovery in a Supervised Learning Classifier System

Jie Tan, Jason H. Moore, Ryan J. Urbanowicz

Geisel School of Medicine, Dartmouth College, 1 Medical Center Dr., Lebanon, NH 03756
Ryan.J.Urbanowicz@dartmouth.edu

Abstract

Michigan-style learning classifier systems have availed themselves as a promising modeling and data mining strategy for bioinformaticists seeking to connect predictive variables with disease phenotypes. The resulting 'model' learned by these algorithms is comprised of an entire population of rules, some of which will inevitably be redundant or poor predictors. Rule compaction is a post-processing strategy for consolidating this rule population with the goal of improving interpretation and knowledge discovery. However, existing rule compaction strategies tend to reduce overall rule population performance along with population size, especially in the context of noisy problem domains such as bioinformatics. In the present study we introduce and evaluate two new rule compaction strategies (QRC, PDRC) and a simple rule filtering method (QRF), and compare them to three existing methodologies. These new strategies are tuned to fit with a global approach to knowledge discovery in which less emphasis is placed on minimizing rule population size (to facilitate manual rule inspection) and more is placed on preserving performance. This work identified the strengths and weaknesses of each approach, suggesting PDRC to be the most balanced approach trading a minimal loss in testing accuracy for significant gains or consistency in all other performance statistics.

Introduction

Learning classifier systems (LCSs) are an adaptive rule-based class of algorithms which combine evolutionary computing with machine learning and other heuristics (Holland, 1986; Wilson, 1995). More recently, Michigan-style LCSs (M-LCSs) have been shown to be an effective approach for the detection and characterization of complex patterns of association in epidemiological data mining. This work applied M-LCSs to identify patterns of multi-locus interaction (i.e. epistasis) as well as heterogeneity when seeking to connect predictive genetic and environmental variables with human disease phenotypes (Urbanowicz and Moore, 2010; Urbanowicz et al., 2012b, 2013). LCSs yield a resulting model/solution, comprised of an entire population of rules, affording them the ability to learn iteratively and distribute learned patterns across this population. These characteristics make the application of LCSs to the problem of heterogeneous patterns particularly appealing.

A notable effect of this iterative, "one instance at a time", LCS learning is the transitional nature of the rule population, i.e. the rule population is constantly changing with offspring rules continually being added and rules of lesser fitness being eliminated. At any given learning iteration, an unknown number of rules are bound to exist in the rule population that make little or no contribution to the overall performance of the system. These include (1) rules that overlap in describing the problem space, (2) poor, recently generated rules, that the algorithm has yet to identify as poor (i.e. low accuracy), and (3) conflicting rules that harm overall performance. Additionally, a solution comprised of a population of many rules can make interpretation and knowledge discovery a considerable challenge. Interpretation of an LCS rule population had been traditionally approached with manual rule inspection, i.e. an expert would examine the best rules of a population in an attempt to extract knowledge. With this task in mind, a *rule compaction* strategy that could consolidate the rule population to a minimum set of critical, human readable rules was considered to be useful.

Wilson implemented the first LCS rule compaction strategy applied to his XCSI algorithm. This Compact Ruleset Algorithm (CRA) achieved a much smaller ruleset, yet maintained high training and testing performance when applied to Wisconsin Breast Cancer dataset (Wilson, 2002). Like most strategies that would follow, rule compaction was run following completion of the LCS algorithm as a form of post processing. Wilson's approach was designed for classifiers that had been highly trained such that the rules were maximally general and always correct in their classification of the test data. CRA was implemented purely to facilitate manual rule inspection by dramatically reducing the rule population size. Later, Fu and Davis revised CRA in order to handle less well trained, noisy classifier systems (Fu and Davis, 2002). The consideration of noisy problems wherein training and testing accuracies might never approach 100% accuracy is critical in problem domains such as bioinformatics and epidemiology. Without doing so, rule compaction is likely to sacrifice performance of the rule population in exchange for minimal size. However, the shortcomings of

CRA and Fu's approaches lie in heavy computation and time complexity, because overall rule population performance needs to be calculated each time a classifier is considered for addition or removal. In an effort to speed up rule compaction, Dixon *et. al.* developed CRA2 which focused both on speed and minimal rule population size (Dixon et al., 2003). CRA2 yielded similar performance to CRA but ran much faster when tested on the same dataset. Other similarly themed LCS rule compaction strategies include an approach for continuous-valued problem spaces (Wyatt et al., 2004), an approach for online rule compaction (Gao et al., 2006), an approach for clustering in XCS (Tamee et al., 2007), an approach which adds entropy calculation (Kharbat et al., 2008), and an approach designed for fuzzy rule representations (Shoeleh et al., 2011).

Recent efforts to move away from manual rule inspection and instead adopt a global pattern approach to knowledge discovery in LCSs have also shifted the priorities of rule compaction. In the context of complex noisy problem domains, it becomes impractical to expect rules to achieve a balance of maximum accuracy and generality. Therefore, placing too much emphasis on reducing the size of the rule population can lead to a loss of rule diversity and result in a reduction in overall performance. In (Urbanowicz et al., 2012a) a global approach to knowledge discovery is considered which introduced global evaluation statistics, and visualization strategies to achieve knowledge discovery in complex problem domains without manual rule inspection. This work focused on modeling noisy, complex patterns in supervised learning problems. In the present study we explore rule compaction somewhat differently than in previous efforts. First, we focus on LCS algorithms designed to address supervised learning domains. Specifically, while previous compaction strategies were designed to function within the context of XCS (Wilson, 1995), a reinforcement learning based LCS, we examine how rule compaction functions in supervised learning based LCSs such as UCS (Bernadó-Mansilla and Garrell-Guiu, 2003). Second, we approach rule compaction assuming that knowledge discovery will be achieved using a global approach as opposed to manual rule inspection. Lastly, we expand our evaluation of LCS performance beyond run time, training, and testing accuracy to consider the impact of rule compaction on an LCS's power to correctly prioritize predictive attributes and discover complex patterns of association as described in (Urbanowicz et al., 2012a). This expansion evaluates the compacted population's ability to yield successful knowledge discovery, as opposed to just successful classification.

The three rule compaction strategies introduced in this work focus on preserving or increasing the quality of the the final rule population rather than emphasizing a human readable population size. The first strategy, Quick Rule Compaction (QRC), is inspired by the match-covering mechanism in the third stage of CRA (Wilson, 2002). The second strategy, Parameter Driven Rule Compaction (PDRC) is largely based on Dixon's approach. Specifically, for each training instance, PCRC finds the classifier in the correct set with the largest product of accuracy, numerosity, and generality and preserves that rule in the final population. The third strategy, Quick Rule Filter (QRF) is more of a rule filter than a compaction algorithm. It simply removes any rule in the population that does not have an accuracy above 0.5. The accuracy of a rule is the frequency of correct prediction for the subset of data instances the rule matches. We compare the resulting rule population performance following that application of our new approaches to three existing approaches (Fu's two approaches and CRA2). Changes in performance statistics are evaluated relative to rule populations without any rule compaction. We consider the advantages and disadvantages of each.

Methods

In this section we describe (1) the LCS algorithm and run parameters used in this investigation, (2) the six rule compaction strategies considered in this study and (3) the experimental evaluation including data simulation, statistical analysis, and visualization.

LCS Algorithm

We begin with a brief review of LCS algorithm concepts critical to understanding rule compaction. LCSs make class predictions based on "votes" made by rules which are relevant to a given instance from the dataset. Each rule possesses a condition and a classification. For example consider a hypothetical rule (0#1## - 1). The '#' serves as a wild card. This rule would match an instance from the data that looks like (02100 - 0), but not one that looks like (12100 - 0). Notice that this first instance example matches the rule, but the rule has incorrectly predicted the class to be '1', when in fact the class of this instance is '0'. These matching rules form what is known as a match set (i.e. the subset of rules in the population which match the attribute states of the dataset instance.) All rules that both match the instance as well as make the correct prediction form a correct set. During supervised LCS learning, when a rule is included in both a match and correct set, it's accuracy and fitness will increase, while if is only involved in a match set (i.e. it matches but makes an incorrect classification) it's accuracy and fitness will decrease. For an in-depth review of LCS algorithms and how they function we refer readers to (Urbanowicz and Moore, 2009).

In order to evaluate each rule compaction strategy within a complex, noisy bioinformatics problem domain, we used an expanded Python encoding of AF-UCS (Urbanowicz et al., 2012b). AF-UCS (attribute feedback UCS), is an expanded and modified implementation of UCS (Bernadó-Mansilla and Garrell-Guiu, 2003) which incorporates a form of memory which feeds back into the genetic algorithm during learning. UCS, or the sUpervised Classifier System, is an M-LCS based largely on the popular XCS algorithm (Wilson, 1995) but replaced reinforcement learning with supervised learning. UCS was designed specifically to address single-step problems such as classification and data mining, where delayed reward is irrelevant, and showed particular promise when being applied to epistasis and heterogeneity in (Urbanowicz and Moore, 2010). This expanded version of AF-UCS incorporates expert knowledge covering as described in (Urbanowicz et al., 2012c) to speed up learning.

The selection of run parameters for this evaluation was arbitrary. We adopted mostly default M-LCS run parameters. Parameters unique to this study include: 200,000 learning iterations, a rule population size of 2000, a rule generality of 0.75 in covering, tournament selection, uniform crossover, and subsumption on. The implementation described above is available on request ($ryanurbanowicz@gmail.com$) and will be posted on the LCS and GBML Central webpage.

Rule Compaction Algorithms

The first LCS rule compaction strategy was CRA, a 3-stage algorithm aimed at dramatically reducing the number of rules and producing a human-readable ruleset (Wilson, 2002). Separate work modified the strategy for rule sorting in the first two stages of CRA such that less well trained classifiers seeking to model noisy problems could be taken into account (Fu and Davis, 2002). Two related strategies from this work are referred to throughout this paper as Fu1 and Fu2. Due to the inherent noise and complexity of our target problem domain, we have chosen to evaluate the two most successful approaches proposed by Fu and Davis as well as Dixon's CRA2, which yields similar performance but runs much faster (Dixon et al., 2003). These implementations are used as a baseline of comparison for our own proposed rule compaction strategies. In the following subsections we describe each of the six rule compaction algorithms considered, three existing strategies (Fu1, Fu2, and CRA2) as well as our three proposed strategies (QRC, PDRC, and QRF). Each have been implemented within the LCS algorithm described in the previous section, coded in Python.

Strategy 1: Fu1 Fu's first approach is summarized as:

> **Stage1:** Sort the rules by ascending numerosity (i.e. the number of copies of an identical rule). Begin with the first rule in the list, eliminate it and test the performance of the rest rules. If the performance becomes better or unchanged, continue to remove the next rule. If the performance is worse, reinsert that classifier to the ruleset and proceed to stage 2.
> **Stage2:** Continue deleting each rule orderly from the ruleset, evaluating the performance of the remaining rules. If the performance is reduced after deletion, move that rule to the new ruleset. The deleted rule is not considered in the next round of evaluation. After all rules being tested in such way, pass the new ruleset with all the rules causing performance reduction to the next stage.
> **Stage3:** Calculate the number of instances in the training set a rule matches, move the rule matching most instances to the final ruleset, and delete instances matched to that rule from training set. Repeat the above three steps until the training dataset is empty or no rules match the remaining instances.

Strategy 2: Fu2 Fu's second approach preserves the first two stages, while modifying stage 3 to take performance into consideration. The third stage can be described as follows:

> **Stage3:** First sort the list of rules obtained from stage 2 by numerosity in increasing order, remove the last rule and evaluate the performance of the remaining classifiers. If the performance drops, reinsert that rule to the top of the list, thus it is involved in the following evaluation. Repeat such test on all classifiers and the final ruleset is composed of classifiers left in the list.

Strategy 3: CRA2 Dixon's CRA2 approach avoids the need for step-wise performance evaluations. The basic idea of CRA2 is to identify the most useful rule for each instance in the training data. CRA2 examines each instance in the training data and builds a correct set from the rule population. For each instance, the most 'useful' classifier is marked for preservation in the final rule population. Dixon's approach determines the most useful rule to be the one in the correct set with the highest mathematical product of accuracy and numerosity. The original CRA2 was implemented in XCS, a reinforcement learning LCS, in which the correct set is called an action set. In the context of our implementation and evaluations, this is only a semantic difference.

Strategy 4: Quick Rule Compaction (QRC) Preliminary observations indicated that the third stage of Fu's first approach was the main cause of performance drop. QRC modifies this stage by using fitness instead of the number of instances a rule matches to retain useful rules. Additionally, QRC completely removes the first two stages utilized in both Fu1 and Fu2, eliminating the need for incremental performance evaluations of the whole rule population. Note that QRC ranks rules by fitness only once at the beginning of rule compaction. This ranking is not updated following the subsequent removal of instances. This differs from the original match-covering mechanism with the intention focusing on globally high fitness and the reduction of run time. Also, it is worth pointing out that in the LCS algorithm used, rule fitness is equal to rule accuracy. Pseudo-code for QRC is given in Algorithm 1.

```
Sort the rules decreasingly by fitness (or accuracy);
while Training dataset is not empty do
    MatchCount = 0;
    for Each instance in the training dataset do
        Determine whether it matches the first rule;
        if It matches then
            MatchCount ++;
        else
            Move the instance to new training set;
        end
    end
    if $MatchCount > 0$ then
        Copy the first rule to the final set;
    end
    Update training set to new training set;
    Delete the first rule from the sorted ruleset;
end
```

Algorithm 1: Quick Rule Compaction

Strategy 5: Parameter Driven Rule Compaction (PDRC) Our PDRC approach is quite similar to Dixon's speedy CRA2 implementation. In preliminary work we explored three different rule parameters (accuracy, numerosity, and generality) in an attempt to find the best numerical way to capture the utility of a rule. We considered them separately, the product of pairs, and the product of all three together. Keeping in mind that CRA2 utilized the product of accuracy and numerosity, we found that the product of accuracy, numerosity and generality performed best. Pseudo-code for PDRC is given in Algorithm 2.

```
for Each instance in the training dataset do
    Create MatchSet;
    Create CorrectSet;
    Find the classifier with highest product of accuracy,
    numerosity and generality in the CorrectSet;
    if The classifier is in final set then
        Pass;
    else
        Add the classifier to the final set;
    end
end
```

Algorithm 2: Parameter Driven Rule Compaction

Strategy 6: Quick Rule Filter (QRF) Our last proposed approach (QRF) is simply a filter which scans the rule population and deletes any rule with an accuracy $<= 0.5$. This is intended to remove rules that predict class no better than by random chance at the time when learning was halted. Additionally, a rule is also deleted if it covers (i.e. matches) less than two instances in the dataset. This removes rules that are likely to be blatantly overfitting training instances. However, such a deletion is prevented if the rule in question specifies only a single attribute. This accounts for the possibility of a rare variant, (i.e. a rare attribute state), which may be useful to preserve when seeking to interpret the rule population.

Experimental Evaluation

Data Simulation Consistent with the nature of our target noisy bioinformatics problem of interest, we have applied our LCS algorithm in conjunction with all rule compaction strategies to a large set of simulated datasets which concurrently model heterogeneity and epistasis as they might appear in a SNP gene association study of common complex disease Urbanowicz and Moore (2010); Urbanowicz et al. (2012b,c). All data sets were generated using a pair of distinct, two-locus epistatic interaction models, both utilized to generate instances (i.e. case and control individuals) within a respective subset of each final data set. Each two-locus epistatic model was simulated without Mendelian/-main effects, as a penetrance table as in Urbanowicz and Moore (2010). Due to the computational demands of LCSs, this study limited its evaluation to 3 heterogeneity/epistasis model combinations. For simplicity the minor allele frequency of each predictive attribute was set to 0.2, a reasonable assumption for a common complex disease SNP. The three model combinations included a pair of models with a heritability of either (0.1, 0.2, or 0.4). We considered model architectural "difficulties" of both "easy" and "hard" Urbanowicz et al. (2012d). Balanced datasets simulated from these models were generated as having four different sample sizes (200, 400, 800, or 1600) and a heterogeneous mix ratio of either (50:50 or 75:25) (e.g. 75% of instances were generated from one epistatic model, and 25% were generated from a different one). Twenty replicates of each dataset were analyzed and 10-fold cross validation (CV) was employed to measure average testing accuracy and account for over-fitting. Together, a total of 48 data set configurations (*3 Model Combos* x *4 Sample Sizes* x *2 Ratios* x *2 Difficulties*), and a total of 960 data sets (20 random seeds each) were simulated. With 10-fold CV, 9600 runs the AF-UCS-based algorithm were completed followed by the same number of runs for each of the six compaction strategies.

Statistical Analysis For each run we track training accuracy, test accuracy, rule generality, macro population size, micro population size, and the run time required for rule compaction. Unlike previous investigations of rule compaction, we also consider three power estimates: (1) the power to find both heterogeneous underlying models, (2) the power to find at least one underlying model, (3) and the power to correctly rank attribute co-occurrence (Urbanowicz et al., 2012a). Power indicates the user's ability to reliably mine knowledge from the evolved rule population. Co-occurrence power is a reflection of our ability to distinguish heterogeneous models from epistatic interactions. Results over 10-fold CV are averaged. Statistical comparisons were made using the Wilcoxon signed-rank tests due to a lack of normality in the value distributions. All statistical evaluations were completed using R. Comparisons were considered to be significant at p-value ≤ 0.05.

In order to further characterize differences between rule compaction approaches, we examine which specific rules remain following compaction. This similarity score is calculated as the ratio of rules preserved in two compacted rule sets to the size of the smaller rule set. Similarity scores were averaged over 10 CV runs.

Visualization Part of the global knowledge discovery process includes the application of intuitive visualizations to identify patterns of association within the rule population. In the present study we generate heat-maps to visualize compacted rule populations as described in (Urbanowicz et al., 2012a). Rules are encoded such that any specified attribute is coded as a 1 and any generalized attribute (#) is coded as a 0. Rule populations are visualized as a micro population, which means there are N copies of each rule reflecting respective numerosity (N). The last processing step before visualization involves the application of a clustering algorithm to the encoded and numerosity expanded population of rules. In this study we employed agglomerative hierarchical clustering based on pearson correlation. For rules or attributes having undefined pearson correlation due to uniform values, 0 is assigned for the purpose of visualization. Both clustering and 2D heat-map visualization are implemented in R using *hclut* and *gplots* packages respectively. In this paper, we generate example visualizations using arbitrarily chosen datasets with a sample size of 1600, minor allele frequencies 0.2, model heritabilities of 0.2, a “Hard” model difficulty, and a heterogeneity ratio of 50:50.

Results and Discussions

Table 1 gives a summary of the metrics evaluating rule populations following compaction. These are compared to the original rule populations prior to any form of compaction (NONE). Three existing approaches were considered (i.e. Fu1, Fu2, and CRA2) as previously described, as well as three approaches that have been proposed here (i.e. QRC, PDRC, and QRF). The color coding within table 1 makes it simple to quickly identify strategies which suffered significant losses, earned significant gains, or maintained performance relative to the original rule population. In reviewing the results, keep in mind that our proposed QRC strategy was most closely related to Fu’s approaches (Fu and Davis, 2002), and our proposed PDRC strategy was most closely related to CRA2. In this work we are most concerned with preserving or, if possible, improving the performance of a rule population while simultaneously seeking to reduce the overall rule population size.

We begin by discussing some of the more obvious trends. All approaches were successful at significantly reducing both the macro population size (i.e. the number of unique rules in the rule population) and the less interesting micro population size (i.e. the number of rules in the population, taking rule numerosity into account). The two Fu approaches yielded the most dramatic decrease in macro population size, followed by our own PDRC and CRA2. As might be expected, our QRF approach resulted in the deletion of only a handful of rules. Next, with the exception of QRC, all strategies significantly increased average rule population generality (i.e. portion of attributes that were generalized using a wild card within a given rule). This metric alone does not tell us much about performance, but all other metrics being equal, it is desirable to have a rule population that is maximally generalized. Given that within our simulated datasets, 20% of the attributes were predictive, wherein only 10% of attributes are predictive for a given heterogeneous dataset instance, we expect that the ideal rule generality would fall between 0.8 and 0.9. Turning our attention to rule compaction time, both Fu strategies took by far the most time to complete, owing to repeated accuracy evaluations of the rule population as a whole. Both CRA2 and our own PDRC strategy yielded similar, dramatically shorter run times. Our QRC approach further reduced run time by an additional order of magnitude, and lastly our QRF approach reduced run time even further by two additional orders of magnitude. Further statistical comparisons indicated that all differences in population size, generality and run time between all strategy pairs were also significant (p-value $<$ 0.001).

Next we examine the impact of each strategy on rule population accuracy and power. Keep in mind that power is a reflection of the algorithm’s success at prioritizing predictive attributes and patterns for successful knowledge discovery. Fu’s first strategy (Fu1) yielded the most significant dramatic loss in both accuracy and power (an apparent trade off with also generating the smallest rule population by far). Differently, Fu’s second strategy (Fu2) instead yielded significant increases in accuracy, but still resulted in modest, significant performance loss within all three power metrics. Comparing this to our proposed QRC strategy (most closely related to Fu1 and Fu2), we similarly observe a significant increase in accuracy (with a particularly notable increase in testing accuracy). Additionally, QRC results in a less significant and less dramatic loss in “Both-Power” and “Co-occurence Power” when compared to the Fu’s approaches, as well as a small non-significant increase in “Single Power”. Considering our multiple evaluation objectives, our QRC approach appears to outperform both of Fu’s approaches. Interestingly when examining rule population similarity, only 45.51% of Fu1 and 56.55% of Fu2 unique rules were also found in the QRC rule population. Originally, we expected that the few rules left after running

		Existing Strategies			**Proposed Strategies**		
Statistics (Averaged)	NONE	Fu1	Fu2	CRA2	QRC	PDRC	QRF
Train Accuracy	0.7865	0.7085**	**0.8056****	0.7938**	0.8027**	0.7966**	0.7873**
Test Accuracy	0.6189	0.5901**	0.6193**	0.6141**	**0.6293****	0.6160**	0.6199**
Both Power	0.2729	0.0854**	0.2167**	**0.3052****	0.2604*	0.2771**	0.2729
Single Power	0.7896	0.6969**	0.7552**	0.7781*	**0.7927**	0.7802	0.7906
Co-Occurrence Power	0.2802	0.0875**	0.2208**	**0.2896**	0.2667*	0.2854	0.2813
Generality	0.7831	**0.8392****	0.7935**	0.7977**	0.7714**	0.7928**	0.7836**
Macro Population	276.46	**13.17****	27.22**	78.54**	103.44**	60.30**	252.10**
Micro Population	2000	**208.52****	414.91**	820.38**	1010.85**	972.33**	1971.53**
Rule Compaction Time (min)	0	4.9625	5.2235	0.0243	0.0064	0.0239	$\mathbf{1.60 \times 10^{-5}}$

Table 1: A comparison of population characteristics following rule compaction. Each statistic is averaged over all 960 simulated datasets and compared with populations prior to rule compaction (i.e. NONE) to determine statistical significance using the Wilcoxon signed rank sum test.(* p-value $<$ 0.05, ** p-value $<$ 0.001). Statistics in yellow have been significantly improved from NONE, while those in red have been significantly impaired. Time values in yellow are relatively fast, while those in red are comparatively slow. For each statistic, the the top finding across all strategies is given in bold.

Fu's strategies would mostly be included in the larger QRC rule population. It turned out that half of the rules in Fu's compacted rulesets did not possess a particularly high fitness relative to rules in the original rule population.

Examining CRA2, we observe a very small but significant reduction in both testing accuracy, and "Single Power". However, CRA2 notably yielded a relatively large and significant improvement in "BothPower", while maintaining "Co-occurence Power". Overall, CRA2 performance was certainly the best of the three existing strategies examined. Comparing CRA2 to our proposed PDRC strategy (again most closely related), we observe that PDRC also yields a small significant loss in testing accuracy. However, average testing accuracy for PDRC is significantly higher than for CRA2 (p-value $<$ 0.05). PDRC also yields a significant increase in "BothPower", however it is not as large an improvement as with CRA2. PDRC does not significantly impact either "Single Power" or "Co-occurence Power". Overall, PDRC better preserves accuracy while also yielding a smaller average macro rule population than CRA2. Note that which micro population is smaller for PDRC than in CRA2, the opposite is true for micro population size. This indicates that PDRC is preferentially selecting rules with a higher average numerosity than CRA2. When examining rule population similarity, 74.89% of the unique rules commonly exist in both CRA2 and PDRC rule populations. This demonstrates CRA2 and PDRC tend to more frequently preserve the same rules within the rule population. It is different from what we observed comparing Fu's approaches to QRC.

Lastly we examine the performance of QRF. QRF is the fastest strategy, and the only one which both significantly improves accuracy (although not as much as QRC and PDRC) as well as maintaining "BothPower" and slightly improving "Single Power" and "Co-occurence Power" (however this improvement is not significant). The most obvious drawback to QRF is that it has very little effect on population size. However in the context of global knowledge discovery strategies, this may be of little importance.

Clearly there are strengths and weaknesses to each of the proposed approaches. For those interested in manual rule inspection with emphasis on obtaining the smallest rule set possible, it would appear that Fu's approaches are best. If speed and the predictive ability of the rule population are priorities, the results suggest that QRC achieves the largest improvement in testing accuracy and runs approximately 1000 times faster than Fu's strategies or approximately 10 times faster than CRA2 or PDRC. If the reader is interested in the most well rounded approach, PDRC yields the smallest rule population next to the Fu's approaches and preserves or even improves all the metrics except a minor reduction in testing accuracy. Finally, if the reader is interested in the fastest approach that has the added benefit of completely preserving accuracy and power, but also does the least to reduce the size of the rule population, we suggest QRF.

Visualization Results

Figures 1-3 present heat-maps visualizing rule population prior to rule compaction, following Fu2 compaction, and following PDRC compaction, respectively. Each row represents a rule while each column represents an attribute (X0-X19). Yellow blocks indicate attributes which have been specified within a rule, while blue indicates generalization (i.e. don't care). Within each illustration, only four attributes were modeled as predictive. One epistatic model includes attributes (X0 and X1) and the second independent model includes attributes (X2 and X3). An accurate representation of this underlying model would yield rules that concurrently specify either attribute pair. Notice in Figure 1 that while there is clearly some noise, two distinct yellow bands, correctly corresponding to (X0,X1) and (X2,X3), are apparent. Additionally hierarchical clustering most strongly links these individual attribute pairs.

In contrast to this successful interpretation of the underlying simulated patterns, Figure 2 shows the visualization of the same rule population following Fu2 compaction. Note that there are far fewer rules in this population, thus the respective row heights are larger than in Figure 1 since fewer rules are forced into the same figure height. Also, notice that while attributes (X2,X3) still form an obvious yellow band, (X0,X1) are no longer as clustered together as strongly, and the overall correct pattern is lost. The correct interpretation of this visual pattern should correspond with power estimates given in Table 1. An example of an effective compaction attempt which maintains, or perhaps, improves the interpretability of the heatmap is given by Figure 3. This figure shows the visualization of the rule population following application of PDRC to the original rule population. Notice that strong clusters are maintained between (X0,X1) and (X2,X3) attribute pairs, and some of the noise is eliminated when compared to Figure 1. Additionally, keep in mind that there are significantly fewer rules in the population visualized in Figure 3 than in Figure 1. It is worth noting that while not shown here, a similar visualization of the rule population following CRA2, yields a similarly clear illustration of patterns modeled in the dataset as seen in Figure 3.

Conclusions and Future Work

In this study, we evaluate rule compaction algorithms for learning classifier systems (LCSs) with the goal of preserving or improving performance while reducing the size of rule population and facilitating knowledge discovery. Specifically we are most interested in applying rule compaction to problems domains with complex and noisy patterns. In addition, based on new global strategies for pursuing knowledge discovery within an LCS rule population (Urbanowicz et al., 2012a), we evaluate rule compaction performance by prioritizing global interpretation rather than traditional manual rule inspection. We also seek to avoid the most obvious shortcoming of most compaction strategies, i.e. computational complexity. We have introduced two new rule compaction strategies (QRC and PDRC) along with a rule filtering strategy (QRF), and compared them to three existing rule compaction methodologies (Fu1, Fu2, and CRA2) using a broader set of performance criteria than previously considered in similar studies.

The results highlighted the strengths and weaknesses of each rule compaction strategy in the context of our complex, noisy problem domain. Fu1 and Fu2 took the longest to run, yielding the smallest rule populations at the expense of accuracy and power. These strategies may be best suited to easier problems, or in problems where manual rule inspection is preferred. Our QRC strategy ran about 1000 times faster, achieving the highest testing accuracy, but re-

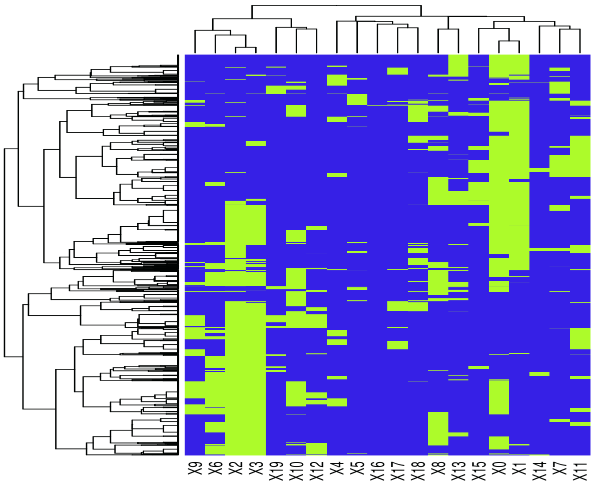

Figure 1: Rule population before compaction.

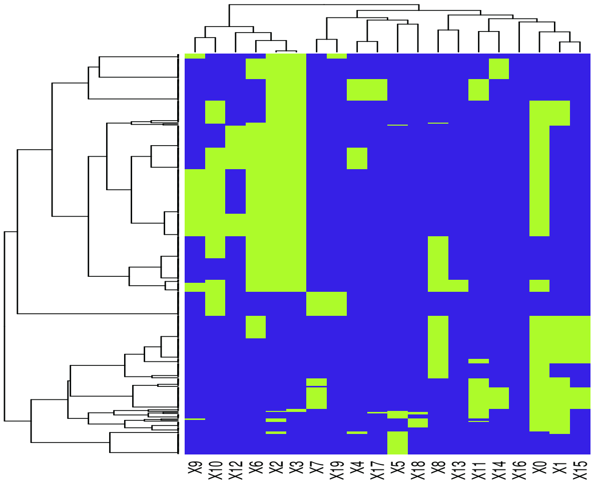

Figure 2: Rule population after Fu2 compaction.

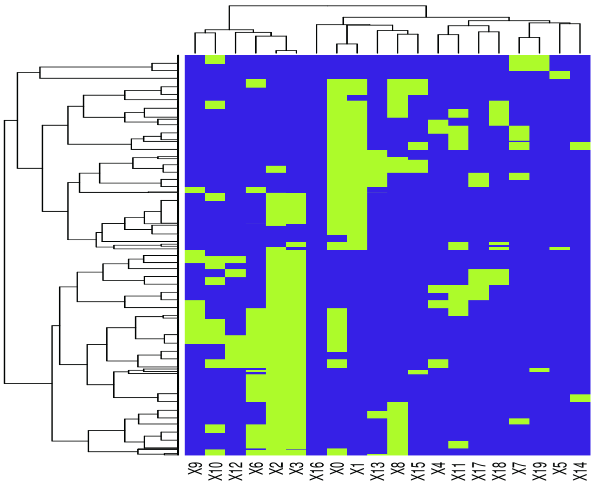

Figure 3: Rule population after PDRC compaction.

sulted in some minor losses in power. This strategy may be ideal when a very large dataset is being investigated or when the goal is classification rather than data mining. Overall our PDRC strategy yielded the most well rounded performance, with CRA2 close behind. Therefore, out of the six strategies examined, PDRC would be the best choice if a researcher wishes to effectively reduce the rule population size while preserving the overall performance of the rule population. Lastly QRF was the only approach that completely preserved or improved accuracy and power. This approach was also approximately 100000 times faster than Fu's approaches. However, this strategy did little to reduce the overall population size. This strategy would be best for situations in which rule population size is not a concern, but preservation of population performance is critical.

Future work will focus on exploring the constitution of rule sets after applying different rule compaction algorithms. Since there are distinct similarities between the compaction strategies, it would be interesting to further investigate the overlapping rules between populations formed by different strategies and better characterize the makings of an essential classifier. Also, while we have utilized a single specific supervised LCS, we would expect that other LCS implementations would similarly benefit from these proposed compaction strategies, as well as global strategies for knowledge discovery. We expect to utilize these compaction strategies in various real-world analyses and LCSs in future work.

Acknowledgements

This work was supported by the William H. Neukom 1964 Institute for Computational Science and NIH grants LM011360, LM009012 and LM010098. RJU was supported by NIH grant R25 CA134286.

References

Bernadó-Mansilla, E. and Garrell-Guiu, J. (2003). Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238.

Dixon, P., Corne, D., and Oates, M. (2003). A ruleset reduction algorithm for the xcs learning classifier system. *Learning Classifier Systems*, pages 20–29.

Fu, C. and Davis, L. (2002). A modified classifier system compaction algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 920–925. Morgan Kaufmann Publishers Inc.

Gao, Y., Wu, L., and Huang, J. (2006). Ensemble learning classifier system and compact ruleset. *Simulated Evolution and Learning*, pages 42–49.

Holland, J. H. (1986). A mathematical framework for studying learning in classifier systems. *Physica D*, 2(1-3):307–317.

Kharbat, F., Odeh, M., and Bull, L. (2008). A new hybrid architecture for the discovery and compaction of knowledge from breast cancer datasets.

Shoeleh, F., Hamzeh, A., and Hashemi, S. (2011). Towards final rule set reduction in xcs: a fuzzy representation approach. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1211–1218. ACM.

Tamee, K., Bull, L., and Pinngern, O. (2007). Towards clustering with xcs. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1854–1860. ACM.

Urbanowicz, R., Granizo-Mackenzie, A., and Moore, J. (2012a). An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems. *Computational Intelligence Magazine, IEEE*, 7(4):35–45.

Urbanowicz, R., Granizo-Mackenzie, A., and Moore, J. (2012b). Instance-linked attribute tracking and feedback for michigan-style supervised learning classifier systems. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 927–934. ACM.

Urbanowicz, R., Granizo-Mackenzie, D., and Moore, J. (2012c). Using expert knowledge to guide covering and mutation in a michigan style learning classifier system to detect epistasis and heterogeneity. *Parallel Problem Solving from Nature-PPSN XII*, pages 266–275.

Urbanowicz, R. and Moore, J. (2009). Learning classifier systems: a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009:1.

Urbanowicz, R. and Moore, J. (2010). The application of michigan-style learning classifiersystems to address genetic heterogeneity and epistasisin association studies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 195–202. ACM.

Urbanowicz, R. J., Andrew, A. S., Karagas, M. R., and Moore, J. H. (2013). Role of genetic heterogeneity and epistasis in bladder cancer susceptibility and outcome: a learning classifier system approach. *Journal of the American Medical Informatics Association*.

Urbanowicz, R. J., Kiralis, J., Fisher, J. M., and Moore, J. H. (2012d). Predicting the difficulty of pure, strict, epistatic models: metrics for simulated model selection. *BioData mining*, 5(1):1–13.

Wilson, S. (1995). Classifier fitness based on accuracy. *Evolutionary computation*, 3(2):149–175.

Wilson, S. (2002). Compact rulesets from xcsi. *Advances in Learning Classifier Systems*, pages 65–92.

Wyatt, D., Bull, L., and Parmee, I. (2004). Building compact rulesets for describing continuous-valued problem spaces using a learning classifier system. *Adaptive Computing in Design and Manufacture VI. Springer*, pages 235–248.