ABSTRACT
This paper describes the application of a multiobjective GRASP to rule selection, where previously generated simple rules are combined to give rule sets that minimize complexity and misclassification cost. As rule selection performance depends heavily on the diversity and quality of the previously generated rules, this paper also investigates a range of multiobjective approaches for creating this initial rule set and the effect on the quality of the resulting classifier.

Categories and Subject Descriptors
I.2.8 [Database Management]; Database Applications—Data mining; G.1.6 [Numerical Analysis]; Optimization; I.2.8 [Artificial Intelligence]; Problem Solving, Control Methods and Search—Heuristic methods

General Terms
Algorithms, Experimentation, Performance

Keywords
Multiobjective optimization, GRASP, data mining, rule induction, rule selection

1. INTRODUCTION
In previous work [13, 15], multiobjective algorithms were applied to the task of partial classification (or rule induction), generating simple rules of the form

\[
\text{if } \text{hairColour} = \text{red}, \text{exposure} \geq 1 \text{ hr and month} = \text{May then sunburn} = \text{yes}
\]

where the antecedent is a conjunction of simple attribute tests and the consequent, describing the class of interest, is the same for all rules generated. Such simple rules may have high confidence or high coverage, and may provide useful information to the client. However, the rule structure is too limited to allow effective optimization of both confidence and coverage. If we wish to produce a binary classifier that accurately determines whether a record belongs to the class of interest or not, we must consider more complex structures.

One approach is to optimize more complex structures from scratch. However, if a set of simple rules is already available, it may be easier to select a small subset of these rules to act as a classifier. Selecting from previously generated (and presumably strong) rules may confer additional advantages over generating rule sets from scratch. A set of rules that are individually strong is likely to be easier to recall than a rule set with weak components that match few records.

The use of GRASP (Greedy Randomized Adaptive Search Procedure) [3, 12] was motivated by previous work using a multiobjective GRASP (MOG) for partial classification [15]. Designed to be guided solely by the concept of Pareto dominance, MOG uses neither linear combinations of objectives nor utility functions. While this algorithm outperformed other multiobjective algorithms on a simple partial classification problem, it was perhaps less flexible, requiring a multiobjective greedy construction algorithm on which to base the GRASP. However, in this paper we show how this algorithm may be adapted for the problem of rule selection.

The quality of the resulting binary classifier depends not only on the design of the rule selection algorithm but also on the rule set from which it selects. This raises two questions.

- If an initial rule set does not already exist, how should one be generated so as to maximize the subsequent performance of the rule selection algorithm?
- If an initial rule set is already available, how likely is it that the rule selection algorithm will be able to produce an effective binary classifier?

Previous work on multiobjective rule induction [13] explored a range of methods for encouraging rule diversity, resulting in significantly different rule sets. A second contribution of this paper, therefore, is an evaluation of how the use of these and other techniques in the rule induction phase affect the performance of subsequent rule selection.

Section 2 describes the methods used to generate the initial set of simple rules from which MOG selects, including the use of novelty measures to encourage diversity in the rule set. The rule selection problem itself is outlined in section 3. Section 4 describes the multiobjective GRASP, with further implementation details in section 5. Section 6 presents results, followed in sections 7 and 8 by a discussion of the results obtained and concluding remarks.
2. MULTIOBJECTIVE RULE INDUCTION AND NOVELTY MEASURES

While the rule selection algorithm described in section 4 can be applied to any set of rules, it is assumed that the initial set is produced by an algorithm that favours high quality rules. For this paper, initial rule sets were generated by the multiobjective algorithm NSGA II, using a range of techniques described in previous work [13] and some new adaptations. This section describes the types of rule generated and the techniques used to generate and assess them.

Later (in section 6), a collection of rule sets generated using these techniques are used as input to the rule selection algorithm. This allows us to investigate how differing methods for the production of the initial rule set affect the quality of the classifier produced by subsequent rule selection.

2.1 Simple Rules

Simple rules are of the form

\[ \text{antecedent} \rightarrow \text{consequent}, \]

where both antecedent and consequent are constructed from attribute tests.

Let \( Q \) be a finite set of attributes, corresponding with the fields in the database. Each \( q \in Q \) has an associated domain, \( \text{Dom}(q) \). An attribute test (AT), \( b \), consists of an attribute, \( a(b) \in Q \), and a value set, \( \text{Val}(b) \subseteq \text{Dom}(a(b)) \), and may be written \( a(b) \in \text{Val}(b) \). A record satisfies this test if its value for attribute \( a(b) \) belongs in the set \( \text{Val}(b) \).

An algorithm may allow only certain types of value sets \( \text{Val}(b) \), corresponding to different types of AT. In this paper, ATs are limited to two types of categorical AT and one type of numeric AT, as follows:

- **Value:** \( \text{Val}(b) = \{ v(b) \} \), where \( v(b) \in \text{Dom}(a(b)) \). This may be written \( a(b) = v(b) \).

- **Inequality:** \( \text{Val}(b) = \{ x \in \text{Dom}(a(b)) : x \neq v(b) \} \), where \( v(b) \in \text{Dom}(a(b)) \). This may be written \( a(b) \neq v(b) \).

- **Binary partition:** \( \text{Val}(b) = \{ x \in \text{Dom}(a(b)) : x \leq v(b) \} \) or \( \text{Val}(b) = \{ x \in \text{Dom}(a(b)) : x \geq v(b) \} \), where \( v(b) \in \text{Dom}(a(b)) \). In this case, the AT may be written \( a(b) \leq v(b) \) or \( a(b) \geq v(b) \), respectively.

Each rule created has the same consequent, consisting of just one AT that defines the class of interest. Rule antecedents are conjunctions of ATs, none of which may share the same attribute as the consequent.

2.2 Support, Confidence and Coverage

Define the support set, \( S(M) \), of any conjunction, \( M \), of ATs to be the set of records which satisfy \( M \). Further define the support, \( \text{sup}(M) \), to be the cardinality of this set, i.e. \( \text{sup}(M) = |S(M)| \). Given a rule, \( r \), we designate the antecedent of the rule \( r^a \) and the consequent \( r^c \).

The support set of \( r \), \( S(r) \), is defined as \( S(r^a \land r^c) \). The support for \( r \), \( \text{sup}(r) \), is defined as \( |S(r)| \).

The confidence and coverage of \( r \) are defined by

\[ \text{conf}(r) = \frac{\text{sup}(r)}{\text{sup}(r^a)}, \quad \text{cov}(r) = \frac{\text{sup}(r)}{\text{sup}(r^c)}. \]

2.3 Simple Optimality Criteria and Dominance

The basic approach to comparing two rules is to compare according to confidence and coverage. Rule \( q \) is dominated by rule \( r \) according to \( cc \)-dominance (or equivalently to \( sc \)-dominance [1]) if and only if

\[ \text{cov}(q) \leq \text{cov}(r) \land \text{conf}(q) < \text{conf}(r), \quad \text{or} \]

\[ \text{cov}(q) < \text{cov}(r) \land \text{conf}(q) \leq \text{conf}(r). \]

Within the constraints imposed on the rules, it is possible to search for \( cc \)-optimal rules, i.e. those rules that are not dominated by any other rule that satisfies the constraints.

2.4 Rule Novelty and Modified Dominance

The use of \( cc \)-dominance may result in useful rules being dominated. A rule may match a set of records unmatched by other rules, yet have relatively low confidence and coverage values. In an effort to allow some such rules to survive, the dominance relation was modified to take into account the novelty of one rule with respect to another.

Of the novelty measures devised [13], those of primary interest in this paper consider the support sets of the two rules. If \( C \) is the set of rules in the class of interest, then:

\[ \text{nov}(r,q) = \frac{|S(r^a) - S(q^a)|}{|S(r)|}. \]

These are the absolute novelty and relative novelty of rule \( q \) with respect to rule \( r \).

If rule \( q \) is \( cc \)-dominated by rule \( r \), then the dominance margin is defined as

\[ \text{dm}(r,q) = \min(\text{conf}(r) - \text{conf}(q), \text{cov}(r) - \text{cov}(q)). \]

By combining the use of rule novelty and the dominance margin, a new dominance relation can be defined. Rule \( q \) is now dominated by rule \( r \) if and only if

\[ r > cc q \quad \text{and} \quad \lambda \text{nov}(q,r) \leq \text{dm}(r,q), \]

where \( \lambda > cc \) indicates \( cc \)-dominance and \( \lambda \) indicates either of the novelty measures. This allows previously dominated rules to survive if they provide sufficient novelty. The parameter \( \lambda \) allows the user to control the value of this novelty.

2.5 Epsilon Dominance

Ishibuchi et al. [6, 5] have used the alternative approach of \( \epsilon \)-dominance to increase rule set diversity. Rule \( q \) is \( \epsilon \)-dominated by rule \( r \) if and only if

\[ \text{cov}(q) + \epsilon \leq \text{cov}(r) \land \text{conf}(q) + \epsilon < \text{conf}(r), \quad \text{or} \]

\[ \text{cov}(q) + \epsilon < \text{cov}(r) \land \text{conf}(q) + \epsilon \leq \text{conf}(r). \]

While this increases the number of non-dominated rules, no effort is made to ensure that the additional rules provide useful new information.

2.6 A Third Objective

Previous work focused on the use of two objectives, confidence and coverage. However, this might lead to simple rules being under-represented, since such rules may be marginally dominated by more complex rules. To produce simple and accurate classifiers, a rule selection algorithm needs access to these simple rules. Therefore, the number of ATs was added as a third objective, to be minimized. Rule \( q \) is dominated by rule \( r \) according to \( cc \)-dominance if and only if

\[ \text{cov}(q) \leq \text{cov}(r), \text{conf}(q) \leq \text{conf}(r) \land \text{comp}(q) > \text{comp}(r), \quad \text{or} \]

\[ \text{cov}(q) \leq \text{cov}(r), \text{conf}(q) < \text{conf}(r) \land \text{comp}(q) \geq \text{comp}(r), \quad \text{or} \]

\[ \text{cov}(q) < \text{cov}(r), \text{conf}(q) \leq \text{conf}(r) \land \text{comp}(q) \geq \text{comp}(r). \]
where $\text{comp}(r)$ indicates the complexity of rule $r$, given by the count of ATs.

If the novelty measures described in section 2.4 are used then rule $q$ is dominated by rule $r$ if and only if $r >_{\text{ccc}} q$ where $>_{\text{ccc}}$ indicates ccc-dominance and $\lambda_{\text{nov}}(q, r) \leq \delta_{\text{dm}}(r, q)$, where the dominance margin is defined as before.

3. MULTIOBJECTIVE RULE SELECTION

Given a large set of simple rules, the task of rule selection is to select a subset of rules that may be used as a binary classifier, distinguishing between those records that are in the class of interest and those that are not. A record is predicted to belong to the class of interest if it matches the antecedent of any one of the rules selected. Otherwise it is predicted not to belong to the class of interest.

The algorithm described in this paper minimizes two objectives: misclassification cost and rule set complexity. In the experiments, the misclassification cost is the simple error rate, i.e. the number of records misclassified, divided by the total number of records; while rule set complexity is simply the number of ATs in the rule set. However, the algorithm can handle alternative misclassification cost functions, such as the balanced error rate, and different measures of rule set complexity, without modification.

4. MULTIOBJECTIVE GRASP

4.1 GRASP

The metaheuristic known as GRASP (Greedy Randomized Adaptive Search Procedure) [3, 12] has been applied to a wide range of optimization problems [4]. Each iteration of the algorithm consists of two phases:

1. The application of a randomized greedy algorithm, usually constructive in nature;
2. The subsequent application of local search to improve the solution constructed.

In a typical implementation, the first phase constructs a solution one element at a time. Candidate elements are assessed using a greedy evaluation function and the best elements form a restricted candidate list (RCL). The element to be added is then selected at random from the RCL.

While GRASP has been applied to a wide range of single objective problems, it is rarely used in a multiobjective setting. Furthermore, in many of these instances, GRASP is merely used as a single objective component of a hybrid system [9, 10, 11], making true multiobjective implementations of GRASP rarer still. Vianna and Arroyo [18] designed a simple GRASP for the multiobjective knapsack problem. However, the heavy use of weighted linear utility functions raised concerns about the suitability of the approach for problems with a non-convex objective space. This led to research on a multiobjective GRASP for partial classification [15] that removed the need for weighted linear utility functions, being based solely on Pareto dominance. This paper provides a second application of this algorithm.

4.2 A Multiobjective GRASP

Pseudo-code for the multiobjective GRASP is presented in figure 1. Here $G$ represents the number of generations, $N$ represents the number of times the greedy algorithm is applied in each generation while $I$ gives the number of iterations of local search per generation.

Local search is applied to fronts, rather than individual solutions. As a result, utility functions are not required to guide the search; solutions are accepted whenever they improve the quality of the front. Note that the greedy algorithm may be called multiple times before local search is applied. This prevents local search from being applied to poor quality starting solutions. (Setting $N = 1$ produces a more standard algorithm, with local search being applied to the results of each application of the greedy algorithm.)

The implementation of the function GreedyRun is omitted, as it is difficult to see how a practical, generic implementation could be written without the use of utility functions to give a direction in which to search. However, section 4.3 shows that in the specific case of rule selection, it is possible to create a multiobjective version of GreedyRun, without the need for utility functions.

4.3 Completing the Multiobjective GRASP for Rule Selection

If a rule set is constructed one rule at a time, a partially constructed rule set is also a rule set. Hence GreedyRun can
Function GreedyRun()

\[ \begin{align*}
    & \text{function } \text{GreedyRun}() \\
    & \text{ \hspace{1em} \begin{array}{l}
            \text{front} := \emptyset \\
            \text{r}_0 := \text{the empty rule set} \\
            \text{Add } \text{r}_0 \text{ to front} \\
            i := 1 \\
            \text{finished} := \text{false} \\
            \text{while (not finished)} \\
            \hspace{1em} \begin{array}{l}
                \text{Select a value for } \kappa_i, \\
                \text{ \hspace{1em} \begin{array}{l}
                    \text{R}_i := \{\text{Rule sets obtained by adding a rule to } \text{r}_{i-1}\} \\
                    \text{C}_i := \{s \in \text{R}_i : \text{cost}(s) < \text{cost}(\text{r}_{i-1}) - \kappa_i\} \\
                    \text{if } (\text{C}_i \neq \emptyset) \\
                    \hspace{1em} \begin{array}{l}
                        \text{B}_i := \{s \in \text{C}_i : \text{comp}(s) = \min_{s \in \text{C}_i} (\text{comp}(t))\} \\
                        \text{Choose } \text{r}_i \in \text{B}_i \text{ that minimizes } \text{cost}(\text{r}_i) \\
                        \text{Add } \text{r}_i \text{ to front} \\
                    \hspace{1em} \text{else} \\
                    \hspace{2em} \text{finished} := \text{true} \\
                    \end{array} \\
                    \text{else} \\
                    \hspace{1em} \begin{array}{l}
                        \text{C}_i := \{s \in \text{R}_i : \text{cost}(s) < \text{cost}(\text{r}_{i-1})\} \\
                        \text{if } (\text{C}_i \neq \emptyset) \\
                        \hspace{1em} \begin{array}{l}
                            \text{B}_i := \{s \in \text{C}_i : \text{cost}(s) = \min_{s \in \text{C}_i} (\text{cost}(t))\} \\
                            \text{Choose } \text{r}_i \in \text{B}_i \text{ that minimizes } \text{comp}(\text{r}_i) \\
                            \text{Add } \text{r}_i \text{ to front} \\
                        \hspace{1em} \text{else} \\
                        \hspace{2em} \text{finished} := \text{true} \\
                        \end{array} \\
                    \end{array} \\
                \end{array} \\
            \end{array} \\
            i := i + 1 \\
            \text{endwhile} \\
        \end{array} \]
    \]
\]

Figure 2: Basic pseudo-code for GreedyRun, minimizing rule set complexity and misclassification cost.

start with a simple, inaccurate rule set and progressively increase both rule set complexity and accuracy through the addition of rules, as shown in figure 2. The result is a multi-objective greedy construction algorithm for rule sets.

In each step of the greedy algorithm, the required reduction in misclassification cost, \( \kappa_i \), is selected at random. If a rule can be added to the rule set in such a way as to produce this reduction in cost, the rule that does so while minimizing the increase in rule set complexity is chosen. Ties are broken on misclassification cost. If the required improvement in cost is unobtainable, a rule is added so as to maximize the reduction in misclassification cost, breaking ties on rule set complexity. If no rule produces a reduction in misclassification cost, the algorithm halts.

Figure 3 illustrates how this algorithm is equivalent to selecting from a set of Pareto-optimal additions to the rule set (though the probability of rule selection will vary from rule to rule), where the additions are evaluated according to the objective values of the resulting rule set. The diamonds represent the objective values for the rule sets generated through the addition of a single rule. If the dashed line represents the cost threshold, then the rule selected appears below the line and as far to the left as possible. Here there are two such rules, so the tie is broken on cost, resulting in the selection of \( B \), rather than the dominated addition \( C \). By setting the cost threshold to different values, it is possible to select any of the Pareto-optimal additions corresponding to the black diamonds, and only such additions.

The value of \( \kappa_i \) is selected at random from an exponential distribution in each iteration of the algorithm. The mean of this distribution, \( \lambda \), is generated uniformly at random each time the greedy algorithm is run, from a range provided as a parameter of the algorithm by the user. This produces a useable algorithm, but it may be difficult for the user to determine the best value for this parameter. The potential cost improvement will be larger early in the construction process, much smaller towards the end, and is likely to be highly problem dependent. As a result, we also consider an alternative approach, as outlined in figure 4.

Here, instead of a threshold in cost improvement, a limit on the increase in rule set complexity, \( l_i \), is imposed. The rule that satisfies the complexity constraint and maximizes the improvement in misclassification cost is selected, provided there is an improvement. Ties are broken on complexity. If no such rule exists, the simplest rule that produces an improvement in cost is selected, breaking ties on complexity. As before, this can be shown to be equivalent to choosing from the set of Pareto-optimal rule additions.

A value for \( l_i \) is selected at random in each iteration of the greedy algorithm, from a truncated geometric distribution

\[ P(X=k) = (1 - \alpha)a^{k-1}/(1 - \alpha^n), \quad k = 1, \ldots, n, \]

where \( n \) is the maximum rule size. The value for \( \alpha \) is generated from a uniform distribution each time the greedy algorithm is run, with the range provided by the user. As the increase in rule set complexity is easier to predict, it is hoped that the best value for this parameter will be easier to determine and less problem dependent.

Note that both approaches can be adapted to cater for an upper limit on the permitted complexity of the rule set, by preventing overly complex rule sets from being added to \( R_t \). Since very large rule sets are unlikely to be well understood by the client, using such a complexity limit focuses the search on those smaller rule sets that might be of interest.

5. IMPLEMENTATION DETAILS

This section describes a number of different techniques that have been used to speed up the operation of the greedy phase of the algorithm. In addition, we describe the operators used by the local search.

In the following, a rule is considered to match a record (and vice versa) if the record satisfies the rule antecedent. A rule correctly matches a record if the record also satisfies the consequent. “Remaining” records are those currently unmatched by the rule set under construction.
match counts for a rule, the number of remaining records
rapid updating, a list of the newly matched records is cre-
match counts are updated lazily, i.e. only when the rule
the match table. However, whenever a rule is added to the
initial costs) of the rule set upon the addition of a rule. Initial
update of the match counts (and hence the misclassification
ing match counts for unselected rules enables the efficient
5.2 Match Count Update
Once the data and rules have been read in, a match table
the complexity of each of the rules.
\begin{itemize}
  \item which rules match which records;
  \item which records are in the class of interest;
  \item the complexity of each of the rules.
\end{itemize}
Once the data and rules have been read in, a match table
containing this information is created. The algorithm then
need only refer to the match table.

\subsection{Match Count Update}
Let the (correct) match count of a rule be the number of
remaining records that (correctly) match the rule. Maintaining
match counts for unselcted rules enables the efficient update of the match counts (and hence the misclassification costs) of the rule set upon the addition of a rule. Initial match counts for each rule are produced upon creation of the match table. However, whenever a rule is added to the rule set by the greedy algorithm, match counts for the remaining rules must be updated.

To significantly reduce the number of updates required, match counts are updated lazily, i.e. only when the rule in question is being considered for addition. To facilitate rapid updating, a list of the newly matched records is created whenever a rule is added to the rule set. To update the match counts for a rule, the number of remaining records
is compared with the number of records eliminated since
the rule’s last update. If there are fewer remaining records, the match counts are reevaluated by scanning these records. Otherwise, the eliminated records are scanned, with the counts being decremented for every matching record that has been eliminated since the last update.

\section{Rule Ordering}
Rules are grouped in buckets for each level of complexity. Within each bucket, rules are ordered according to the count of correct matches. While these match counts will be out of date, they are used to prioritize rules, reducing the number of rules considered in each iteration and preventing match count updates being performed on poor quality rules.

When using cost thresholds, the simplest rules are scanned first, with equally complex rules scanned in decreasing order of the count of correct matches. In the complexity limit approach, rules are scanned in order of decreasing match count. Rules with the same counts are scanned in order of increasing complexity. Once it becomes clear that further scanning cannot result in a better rule, scanning is halted.

Rules that result in too many false positives to improve the rule set, regardless of how many false negatives are eliminated, are removed from consideration.

\subsection{Clustering}
Multiobjective rule induction may produce thousands of rules. Many of these will be similar, especially when the data includes numeric attributes. Furthermore, rules that may appear to be different may match similar sets of records.

While this is likely to have only a limited impact on the local search phase of the GRASP, the exhaustive nature of the rule reevaluations in the greedy phase means that multiple near-copies of the same rule may lead to a loss in algorithm efficiency. It is therefore worth considering the elimination of some of these near-copies before each run of the greedy algorithm. This is achieved by considering a clustering of the rules and selecting only one rule from each cluster.

Rule clusters were generated using the hierarchical clustering algorithm AGNES (AGglomerative NESting) [8], with inter-cluster distances calculated using the weighted average linkage [17] for reasons described by Reynolds et al. [16]. Distances between rules were calculated by summing the novelty of each rule with respect to the other, i.e.
\begin{align*}
d(q,r) &= \text{nov}_a(r,q) + \text{nov}_a(r,q) \\
when using absolute novelty and \[ j(q,r) = \text{nov}_r(q,r) + \text{nov}_r(r,q), \]
when using relative novelty. \( d(q,r) \) is the simple matching coefficient [8], while \( j(q,r) \) is the Jaccard coefficient [7].

\section{Cluster Representative Selection}
The first method considered for selecting a cluster repre-
sentative was simply to select a rule at random. Selecting representatives prior to each run of the greedy algorithm en-
sures that each run will produce different rule sets, increasing
the range of rule sets produced. However, simple rules, consisting of only one or two ATs, may be overlooked by this method, due to the greater numbers of more complex rules. Therefore, two other methods were also employed. The first considers only the smallest rules in a cluster, selecting from these at random. In the second ‘balanced’ approach, the
size of the rule to be selected is first chosen uniformly at random. Then a rule of this size is selected, at random, from the cluster. This increases the bias towards small rules without eliminating the possibility of selecting larger ones.

5.6 Local Search

Moves used in local search are of three types: rule removal, rule addition and a rule swap. If the current rule set is empty, only additions are considered. Otherwise, each move type is assigned a probability of one third. After the move, if the rule set exceeds the complexity limit, rules are removed at random until the complexity constraint is met.

6. RESULTS

6.1 Parameter Training

Two phases of experiments were performed to find suitable values for the parameters, using one core of an Intel Core 2 Duo processor T5500 at 1.66GHz, with 2 Gb of available RAM. 80% of the training portion of the adult data set [2] was used, with rules generated using two objectives, absolute novelty valued at \( \lambda = 0.7 \) and a limit of 6 ATs per rule. The rule consequent was \( \text{salary} > 50k \). The complexity limit for the rule set was set to 20 ATs.

- **Phase 1**: Parameter settings for the greedy component of the algorithm were evaluated. The cost threshold and complexity limit approaches were compared, using a range of step sizes, different numbers of clusters and each cluster representative selection method. Thirty two-minute runs were performed using each combination of the following parameter values:
  - Step size parameter, \( \lambda \) (cost threshold approach): 0.00025–0.005 (T1), 0.0005–0.01 (T2), 0.001–0.02 (T3), 0.002–0.04 (T4), 0.004–0.08 (T5), 0.008–0.16 (T6);
  - Step size parameter, \( \alpha \) (complexity limit approach): 0.25–0.5 (L1), 0.5–1 (L2), 0.5–2 (L3), 0.667–1.333 (L4), 1–2 (L5), 2–4 (L6);
  - No. clusters: No clustering, 500, 200, 100;
  - Representative selection method: Random, Smallest, Balanced.

- **Phase 2**: Experiments were performed altering the ratio of local search to greedy construction and the number of generations of the GRASP. The best parameter values from phase 1 were used, with each combination of the following parameter values:
  - No. generations: 1, 2, 5, 10;
  - Time in LS: 0%, 25%, 50%, 75%, 90%, 95%.

Thirty five-minute runs were performed with each parameter combination.

Of the 120 combinations of parameters for phase 1, table 1 presents the best 10, ranked by the sum of rule costs at the twenty complexity levels, averaged over the thirty runs. Figure 5 shows the difference in performance between using no clustering and using 500 clusters with the three cluster representative selection methods. The rule selection algorithm was run thirty times, with five minutes per run, on each of the rule sets. Figure 6 shows the effect of using novelty measures on the sum of rule set costs. Using novelty-enhanced dominance relations clearly results in better performance from the subsequent application of the rule selection algorithm. It is also apparent that considering rule complexity as a third objective in the rule induction algorithm results in improved rule sets.

Table 1: Best parameter combinations for phase 1.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>limit 0.5–2</td>
<td>500</td>
<td>balanced</td>
<td>3.4711</td>
<td></td>
</tr>
<tr>
<td>threshold 0.008–0.16</td>
<td>500</td>
<td>balanced</td>
<td>3.4713</td>
<td></td>
</tr>
<tr>
<td>limit 1–2</td>
<td>500</td>
<td>balanced</td>
<td>3.4714</td>
<td></td>
</tr>
<tr>
<td>threshold 0.004–0.08</td>
<td>500</td>
<td>balanced</td>
<td>3.4717</td>
<td></td>
</tr>
<tr>
<td>limit 0.667–1.333</td>
<td>500</td>
<td>balanced</td>
<td>3.4717</td>
<td></td>
</tr>
<tr>
<td>threshold 0.002–0.04</td>
<td>500</td>
<td>balanced</td>
<td>3.4724</td>
<td></td>
</tr>
<tr>
<td>limit 0.5–1</td>
<td>500</td>
<td>balanced</td>
<td>3.4729</td>
<td></td>
</tr>
<tr>
<td>limit 2–4</td>
<td>500</td>
<td>balanced</td>
<td>3.4738</td>
<td></td>
</tr>
<tr>
<td>limit 0.667–1.333</td>
<td>None</td>
<td>-</td>
<td>3.4742</td>
<td></td>
</tr>
<tr>
<td>limit 0.5–2</td>
<td>None</td>
<td>-</td>
<td>3.4742</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Comparison between using no clustering and using 500 clusters with the three cluster representative selection methods. Error bars give 95% confidence intervals.

6.2 Alternative Initial Rule Sets

Having decided on the parameter values to be used, the algorithm was applied to 18 different rule sets generated from 80% of the adult data set, as indicated in table 2. Each rule set was produced by NSGA II, limited to 50,000 rule evaluations. Values for \( \lambda \) and \( \epsilon \) were chosen so as to produce rule sets of roughly equal sizes for the three methods, although no value for \( \epsilon \) could be found to produce rule sets of similar size to those produced with the lowest values of \( \lambda \).

Rule clustering was performed using the rule dissimilarity metric that matched the novelty measure, where appropriate. When no novelty measure or \( \epsilon \)-dominance was used, rules were clustered according to both metrics. The rule selection algorithm was run thirty times, with five minutes per run, on each of the rule sets. Figure 6 shows the effect of using novelty measures on the sum of rule set costs. Using novelty-enhanced dominance relations clearly results in better performance from the subsequent application of the rule selection algorithm. It is also apparent that considering rule complexity as a third objective in the rule induction algorithm results in improved rule sets.
is the best obtained from using the two rule dissimilarity measures when clustering. These results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.

7. DISCUSSION

When searching for rule sets that minimize both cost and complexity, insisting that component rules are selected from a predefined set might be expected to limit the ability of the algorithm to find effective solutions. Indeed, results of this work and work by Ishibuchi et al. [6, 5] confirm this, if rules are required to be non-dominated according to confidence and coverage. However, careful use of measures to encourage rule diversity when performing the original rule induction greatly reduces this problem. When evaluated on the training data, using genetic programming to produce rule sets from scratch [14] marginally outperforms the rule induction algorithm. However, alternative novelty measures [13] can be calculated more quickly — use of these measures is a matter for further research.

Ishibuchi et al. have used thresholds, on both confidence and coverage measures when clustering. The results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.

7. DISCUSSION

When searching for rule sets that minimize both cost and complexity, insisting that component rules are selected from a predefined set might be expected to limit the ability of the algorithm to find effective solutions. Indeed, results of this work and work by Ishibuchi et al. [6, 5] confirm this, if rules are required to be non-dominated according to confidence and coverage. However, careful use of measures to encourage rule diversity when performing the original rule induction greatly reduces this problem. When evaluated on the training data, using genetic programming to produce rule sets from scratch [14] marginally outperforms the rule selection approach. However, the difference in performance is small compared with the change in cost when the rule sets are evaluated on the test data (see figure 7).

Novelty based dominance relations also improved performance of the rule selection algorithm, when compared with the use of $\epsilon$-dominance. This is as expected, as $\epsilon$-dominance does nothing to discourage the production of near duplicates of rules already present. Hence, with rule sets of approximately equal size, those generated using the novelty measures are likely to contain a more diverse collection of rules. A drawback of using support set based novelty measures is the increase in time required to perform each iteration of the rule induction algorithm. However, alternative novelty measures [13] can be calculated more quickly — use of these measures is a matter for further research.

Ishibuchi et al. have used thresholds, on both confidence and coverage measures when clustering. The results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.

7. DISCUSSION

When searching for rule sets that minimize both cost and complexity, insisting that component rules are selected from a predefined set might be expected to limit the ability of the algorithm to find effective solutions. Indeed, results of this work and work by Ishibuchi et al. [6, 5] confirm this, if rules are required to be non-dominated according to confidence and coverage. However, careful use of measures to encourage rule diversity when performing the original rule induction greatly reduces this problem. When evaluated on the training data, using genetic programming to produce rule sets from scratch [14] marginally outperforms the rule selection approach. However, the difference in performance is small compared with the change in cost when the rule sets are evaluated on the test data (see figure 7).

Novelty based dominance relations also improved performance of the rule selection algorithm, when compared with the use of $\epsilon$-dominance. This is as expected, as $\epsilon$-dominance does nothing to discourage the production of near duplicates of rules already present. Hence, with rule sets of approximately equal size, those generated using the novelty measures are likely to contain a more diverse collection of rules. A drawback of using support set based novelty measures is the increase in time required to perform each iteration of the rule induction algorithm. However, alternative novelty measures [13] can be calculated more quickly — use of these measures is a matter for further research.

Ishibuchi et al. have used thresholds, on both confidence and coverage measures when clustering. The results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.

7. DISCUSSION

When searching for rule sets that minimize both cost and complexity, insisting that component rules are selected from a predefined set might be expected to limit the ability of the algorithm to find effective solutions. Indeed, results of this work and work by Ishibuchi et al. [6, 5] confirm this, if rules are required to be non-dominated according to confidence and coverage. However, careful use of measures to encourage rule diversity when performing the original rule induction greatly reduces this problem. When evaluated on the training data, using genetic programming to produce rule sets from scratch [14] marginally outperforms the rule selection approach. However, the difference in performance is small compared with the change in cost when the rule sets are evaluated on the test data (see figure 7).

Novelty based dominance relations also improved performance of the rule selection algorithm, when compared with the use of $\epsilon$-dominance. This is as expected, as $\epsilon$-dominance does nothing to discourage the production of near duplicates of rules already present. Hence, with rule sets of approximately equal size, those generated using the novelty measures are likely to contain a more diverse collection of rules. A drawback of using support set based novelty measures is the increase in time required to perform each iteration of the rule induction algorithm. However, alternative novelty measures [13] can be calculated more quickly — use of these measures is a matter for further research.

Ishibuchi et al. have used thresholds, on both confidence and coverage measures when clustering. The results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.

7. DISCUSSION

When searching for rule sets that minimize both cost and complexity, insisting that component rules are selected from a predefined set might be expected to limit the ability of the algorithm to find effective solutions. Indeed, results of this work and work by Ishibuchi et al. [6, 5] confirm this, if rules are required to be non-dominated according to confidence and coverage. However, careful use of measures to encourage rule diversity when performing the original rule induction greatly reduces this problem. When evaluated on the training data, using genetic programming to produce rule sets from scratch [14] marginally outperforms the rule selection approach. However, the difference in performance is small compared with the change in cost when the rule sets are evaluated on the test data (see figure 7).

Novelty based dominance relations also improved performance of the rule selection algorithm, when compared with the use of $\epsilon$-dominance. This is as expected, as $\epsilon$-dominance does nothing to discourage the production of near duplicates of rules already present. Hence, with rule sets of approximately equal size, those generated using the novelty measures are likely to contain a more diverse collection of rules. A drawback of using support set based novelty measures is the increase in time required to perform each iteration of the rule induction algorithm. However, alternative novelty measures [13] can be calculated more quickly — use of these measures is a matter for further research.

Ishibuchi et al. have used thresholds, on both confidence and coverage measures when clustering. The results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.

7. DISCUSSION

When searching for rule sets that minimize both cost and complexity, insisting that component rules are selected from a predefined set might be expected to limit the ability of the algorithm to find effective solutions. Indeed, results of this work and work by Ishibuchi et al. [6, 5] confirm this, if rules are required to be non-dominated according to confidence and coverage. However, careful use of measures to encourage rule diversity when performing the original rule induction greatly reduces this problem. When evaluated on the training data, using genetic programming to produce rule sets from scratch [14] marginally outperforms the rule selection approach. However, the difference in performance is small compared with the change in cost when the rule sets are evaluated on the test data (see figure 7).

Novelty based dominance relations also improved performance of the rule selection algorithm, when compared with the use of $\epsilon$-dominance. This is as expected, as $\epsilon$-dominance does nothing to discourage the production of near duplicates of rules already present. Hence, with rule sets of approximately equal size, those generated using the novelty measures are likely to contain a more diverse collection of rules. A drawback of using support set based novelty measures is the increase in time required to perform each iteration of the rule induction algorithm. However, alternative novelty measures [13] can be calculated more quickly — use of these measures is a matter for further research.

Ishibuchi et al. have used thresholds, on both confidence and coverage measures when clustering. The results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.

7. DISCUSSION

When searching for rule sets that minimize both cost and complexity, insisting that component rules are selected from a predefined set might be expected to limit the ability of the algorithm to find effective solutions. Indeed, results of this work and work by Ishibuchi et al. [6, 5] confirm this, if rules are required to be non-dominated according to confidence and coverage. However, careful use of measures to encourage rule diversity when performing the original rule induction greatly reduces this problem. When evaluated on the training data, using genetic programming to produce rule sets from scratch [14] marginally outperforms the rule selection approach. However, the difference in performance is small compared with the change in cost when the rule sets are evaluated on the test data (see figure 7).

Novelty based dominance relations also improved performance of the rule selection algorithm, when compared with the use of $\epsilon$-dominance. This is as expected, as $\epsilon$-dominance does nothing to discourage the production of near duplicates of rules already present. Hence, with rule sets of approximately equal size, those generated using the novelty measures are likely to contain a more diverse collection of rules. A drawback of using support set based novelty measures is the increase in time required to perform each iteration of the rule induction algorithm. However, alternative novelty measures [13] can be calculated more quickly — use of these measures is a matter for further research.

Ishibuchi et al. have used thresholds, on both confidence and coverage measures when clustering. The results suggest that, for rule sets of roughly equal size, relative novelty produces the most useful initial rule set, followed by absolute novelty.

Rule sets of up to 8 ATs, generated from a typical run of the rule selection algorithm applied to rules generated using three objectives and relative novelty rated at $\lambda = 0.3$, are shown in table 4.
and coverage, to help eliminate some of the problems with \( \epsilon \)-dominance [5]. However, preliminary experimentation suggests that this is not sufficient to allow \( \epsilon \)-dominance to outperform the use of novelty measures. The use of thresholds in combination with novelty-enhanced dominance relations will be examined in future work.

While using clusters improves the performance of the greedy algorithm, the time required to calculate the dissimilarity matrix is likely to deter users from creating them, unless they are already available. However, clustering may be performed for other reasons. For example, the results of hierarchical clustering are useful when exploring the rule set. Additionally, simpler methods for calculating rule dissimilarity would result in much faster clustering.

8. CONCLUSION

A multiobjective GRASP for rule selection has been developed that produces rule sets that minimize misclassification cost and rule set complexity. Provided the set from which the algorithm selects rules is sufficiently diverse, results can be obtained that are competitive with those obtained when generating rule sets from scratch. The required diversity in the initial rule set can be produced by using a multiobjective rule induction algorithm, provided the dominance relation is modified in a way that encourages such diversity, for example by utilizing novelty measures and dominance margins [13].

9. REFERENCES