Data Mining Rules Using Multi-Objective Evolutionary Algorithms

Beatriz de la Iglesia, Mark S. Philpott,Anthony J. Bagnall and Vic J. Rayward-Smith
University of East Anglia,
Norwich NR4 7TJ,
England
(bli,m.s.philpott,ajb,vjrs)@sys.uea.ac.uk

Abstract- In data mining, nugget discovery is the discovery of interesting classification rules that apply to a target class. In previous research, heuristic methods (Genetic algorithms, Simulated Annealing and Tabu Search) have been used to optimise a single measure of interest. This paper proposes the use of multi-objective optimisation evolutionary algorithms to allow the user to interactively select a number of interest measures and deliver the best nuggets (an approximation to the Pareto-optimal set) according to those measures. Initial experiments are conducted on a number of databases, using an implementation of the Fast Elitist Non-Dominated Sorting Genetic Algorithm (NSGA), and two well known measures of interest. Comparisons with the results obtained using modern heuristic methods are presented. Results indicate the potential of multi-objective evolutionary algorithms for the task of nugget discovery.

1 Introduction

1.1 Defining a nugget

Classification is a common task for data mining. In classification, a model is sought which can assign a class to each instance in the dataset. Classification algorithms often use overall classification accuracy as the guiding criteria to construct a model. Partial classification [2], also termed nugget discovery, seeks to find patterns that represent a description of a particular class. This data mining task is particularly relevant when some of the classes in a database are minority classes, i.e. those with few representative instances in the database, as in those cases high overall classification accuracy can be achieved even when minority classes are misclassified by the model induced.

A nugget, or partial description, is often presented as a simple conjunctive rule, $\alpha \Rightarrow \beta$, where the precondition or antecedent of the rule, α, represents a conjunction of tests on the attributes or fields, $AT_1, \ldots, AT_n$, of the database, D, and the postcondition or consequent of the rule, β, represents the class assignment. Association rules are of a similar format, but the consequent is not fixed to be a particular attribute-value pair. Also, in association rule induction [1] the target of the discovery is to generate *all* rules that meet certain constraints.

In the case of conjunctive rules, the antecedent is of the following form:

$$\alpha = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_m.$$

For a categorical attribute a conjunct, α_l, is a test that can take the following forms:

Simple value: $AT_j = v$, where v is a value from the domain of AT_j, Dom_j, for some $1 \leq j \leq n$. A record x satisfies this test if $x[AT_j] = v$.

Subset of values: $AT_j \in \{v_1, \ldots, v_k\}$,$1 \leq j \leq n$, where $\{v_1, \ldots, v_k\}$ is a subset of values in the domain of AT_j, for some $1 \leq j \leq n$. A record x satisfies this test if $x[AT_j] \in \{v_1, \ldots, v_k\}$.

Inequality test: $AT_j \neq v$, for some $1 \leq j \leq n$. A record x satisfies this test if $x[AT_j] \neq v$.

For a numeric attribute, a conjunct, α_p, is a test that can take the following form:

Simple value: $AT_j = v$, $1 \leq j \leq n$, as for categorical attributes.

Binary partition: $AT_j \leq v$ or $AT_j \geq v$, for some $1 \leq j \leq n$, $v \in Dom_j$. A record x satisfies these tests if $x[AT_j] \leq v$ or $x[AT_j] \geq v$ respectively.

Range of values: $v_1 \leq AT_j \leq v_2$ or $AT_j \in [v_1, v_2]$, for some $1 \leq j \leq n$ and $v_1, v_2 \in Dom_j$. A record x satisfies this test if $v_1 \leq x[AT_j] \leq v_2$.

A record x satisfies a conjunction of tests, $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_m$, if x satisfies all the tests, $\alpha_1, \alpha_2, \ldots, \alpha_m$.

The consequent of the rule is just the specification of the class that the rule is describing, chosen from a set of predefined classes.

Note that other rule formats can be defined, however this would represent an increase in the size of the search space and also the rules found may be more complex. It is assumed in this research that conjunctive rules are sufficient for a class description.

Note also that although the rules described use a relatively simple syntax, the introduction of numeric attributes without pre-discretisation and the use of disjunction of nominal attribute values increases the complexity of the problem dramatically. Many of the commonly proposed algorithms for rule induction (e.g.[1, 3, 4]) can only discover rules using simple value tests.

For such a simple rule, we can define some measures based on the cardinalities of the different sets defined by the rule. Each conjunct defines a set of data points (or records) for which the test specified by the conjunct is true, and the intersection of all those sets, or the set of points for which

all the conjuncts are true defines the applicability of the rule in the database. We will refer to this set as A, and $|A| = a$. The set of data points for which the consequent of the rule is true, or, in other words, the set of data points that belong to the class specified by the rule will be referred to as B, and $|B| = b$. Finally, the set of points for which both the antecedent and consequent of the rule are true will be called C, and $|C| = c$. In summary,

$$A = \{x \in D | \alpha(x)\},$$

$$B = \{x \in D | \beta(x)\} \text{ and}$$

$$C = \{x \in D | \alpha(x) \wedge \beta(x)\}.$$

Note that $c \leq a$ and $c \leq b$, as $C \subseteq A$ and $C \subseteq B$. Also $a \leq d$ and $b \leq d$, since $A, B \subseteq D$. In nugget discovery, both b and d are fixed.

1.2 Measuring Interest

In order to assess the quality of a nugget the following properties are often examined.

Accuracy (Confidence): $\text{Acc}(r) = \frac{c}{a}$
This measure represents the proportion of records for which the prediction of the rule (or model in the case of a complete classification) is correct, and it is one of the most widely quoted measures of quality, specially in the context of complete classification.

Coverage: $\text{Cov}(r) = \frac{c}{b}$
This measure is defined here as the proportion of the target class covered by the rule. Since b is fixed in nugget discovery, coverage is proportional to c.

There are other measures proposed and used by data mining algorithms, and they can often be presented in terms of $a, b, c,$ and d.

It is intuitive that we wish to find nuggets that maximise both accuracy and coverage. We can define a partial ordering with respect to accuracy and coverage,$\leq_{ca}$, such that if two rules had the same accuracy, the more covering rule would be preferred. Equally, if two rules had the same coverage, the more accurate rule would be preferred. Such a partial ordering is discussed in [7]. A similar ordering was also proposed in [3] in the context of association rules. The partial ordering would define an upper accuracy/coverage border which contains potentially interesting rules. Nugget discovery algorithms should be able to sample this front effectively looking for accurate rules or for general rules.

We propose to use multi-objective metaheuristic methods for the problem of nugget discovery from classification databases with nominal and numerical attributes. For the bi-objective problem of discovering nuggets of high accuracy/coverage, the multi-objective treatment should allow the algorithm to return an approximation to the upper accuracy/coverage border, containing solutions that are spread across the border. Other objectives, such as simplicity, can be incorporated in the search if the approach proves effective.

2 Modern Heuristics for Nugget Discovery

In previous research [6, 14] we presented an interest measure, the *fitness measure*

$$f(r) = \lambda c - a, \text{ where } \lambda \in \Re$$

which was capable of partially ordering rules according to accuracy and coverage under certain constraints. The λ parameter establishes an *accuracy threshold*, defined by $\frac{1}{\lambda}$, above which the fitness measure orders rules correctly with respect to the $\leq_{ca}$ partial ordering. Importantly, when two rules cannot be compared under the partial ordering, the λ parameter can be used to establish a preference for more accurate rules or more widely covering rules. At low values of the λ parameter, accurate rules will be fitter than widely covering rules, whereas at high values of the λ parameter, widely covering rules will be fitter than highly accurate rules. Generally, the λ range we experiment with are between $(1, d/b]$. Hence, by running modern heuristics techniques such as Simulated Annealing (SA), Tabu Search (TS) and Genetic Algorithms (GAs) to search for nuggets that maximise the fitness measure with different levels of λ we are able to approximate the upper accuracy/coverage border for a particular nugget discovery problem.

Results of applying the modern heuristic algorithms across a range of datasets from the UCI repository showed that interesting nuggets could be obtained at different levels of accuracy/coverage.

We now propose to extend this research by using Multi-objective Evolutionary Algorithms (MOEA)[5]. This should allow us to deliver a set of rules that lie in the Pareto optimal front, or in the upper accuracy/confidence border if we choose those as the measures to be optimised, in one single run of the algorithm. Additionally, if MOEA proves to be more successful than the standard GA/SA/TS approach, then other measures of interest could be introduced in the search as additional objectives.

The disadvantages to the modern heuristics approach that we may wish to overcome by using other type of MOEAs are the following. First, we need to run the algorithm several times with varying levels of λ as it is not known in advance what kind of nuggets may be obtainable by a particular λ value, and it is generally not known in advance what are the preferences of the decision maker. Second, only accuracy and coverage can be taken into consideration with the present approach. To optimise nuggets in terms of simplicity, for example, we run a post-processing algorithm which removes conjuncts in a greedy manner choosing the one that produces no deterioration in accuracy, one at a time. Hence, there is no simple way to introduce other interest criteria, other than by replacing the fitness measure by a different measure which encapsulates the alternative criteria. Also, there is no real encouragement for the algorithm to produce different rules along the upper coverage/accuracy border, hence many copies of the same rule may be produced even with variations in the λ parameter.

3 MOEAs for Nugget Discovery

In order to create a nugget discovery system that is more flexible, we propose to use Pareto-based MOEA to deliver nuggets that are in the Pareto optimal set according to some measures of interest which can be chosen by the user.

The proposed system will present to the user a number of measure of interest from which some measures can be selected. As well as those presented in section 1.2, measures of the simplicity of a nugget, based for example on the Minimum Description Length Principle [13], or simply on the number of conjuncts of the nugget may also be used. The system will represent nuggets using the same approach as in [6, 7]. The process of nugget evaluation may also be similarly conducted. The main difference is that the overall objective of the MOEA will be to find a set of diverse non-dominated solutions, which should constitute a good approximation to the Pareto optimal front. Note that in most cases we do not know the exact composition of the Pareto optimal front.

We expect that there may be some merit in evaluating the performance of various Pareto-based MOEA approaches for this problem, for example: the Niched Pareto Genetic Algorithm [9]; the Pareto Archive Evolution Strategy (PAES) algorithm proposed by Knowles and Corne [11] and the Fast Elitist Non-Dominated Sorting Genetic Algorithm (NSGA) proposed by Deb et al. [8]. However, in this paper we limit ourselves to the evaluation of one of the MOEA, the NSGA (version II) algorithm [8], against the conventional GA/SA/TA algorithms. Hence we are trying to establish the benefit of using a Pareto-based approach, against the aggregating function approach.

In this paper we also limit ourselves to the use of accuracy and coverage as our measures of interest. Since results are encouraging we plan to extend the research by using other measures.

4 Implementation of the NSGA II for Nugget Discovery

The NSGA II algorithm [8] is a variation of the original NSGA algorithm proposed in [15]. The new algorithm addresses some of the criticisms of the previous version including: the lack of elitism; the need for specifying a sharing parameter to ensure diversity in the population; and, the high computational complexity of non-dominated sort.

The NSGA II algorithm is fully explained in [8]. The initial population is created by the initialisation procedure described in a later section. The first iteration is slightly different to the rest. In the main function, a child population is created at each stage using binary tournament selection, single-point cross-over and mutation. Both populations are combined and the resulting population of size $2N$ (where N is the size of the initial population) is then sorted according to non-domination into different fronts. Hence solutions that belong to the first front are non-dominated solutions. Those are then discounted to find the second front, and so on. Within each front, solutions are sorted according to crowding distance. The crowding distance specifies the size of the largest cuboid enclosing a point in the population i, but not including any other points in the population. A new population is then created by taking solutions from the combined population in terms of rank first, and crowding distance as a secondary sorting criterion, until the new population is of size N. The process goes back then to the creation of a child population and continues through an specified number of generations.

In the following sections we discuss some of the details of the implementation that are necessary to adapt the NSGA II algorithm to solve the nugget discovery problem as presented before. Most of these implementation details are shared with the modern heuristic algorithms.

4.1 Representing a Solution

The solution to be represented is a conjunctive rule or nugget following the syntax described previously. A binary string is used for this as follows.

The first part of the string is used to represent the numeric fields or attributes. Each numeric attribute is represented by a set of Gray-coded lower and upper limits, where each limit is allocated a user-defined number of bits, p ($p = 10$ is the default). There is a scaling procedure that transforms any number in the range of possible values using p bits $[0, 2^p - 1]$ to a number in the range of values that the attribute can take. The procedure works as follows. When the data is loaded the maximum value, max_i, and minimum value, min_i, for each attribute i are stored. A weight for each attribute is then calculated as

$$w_i = \frac{max_i - min_i}{2^p - 1}.$$

When the string representing a nugget is decoded, the upper and lower limit values for each attribute are calculated by

$$limit_i = (ss * w_i) + min_i,$$

where ss represents the decimal value of an p bit Gray-coded substring extracted from the binary string, which corresponds to one of the limits.

The second part of the string represents categorical attributes, with each attribute having v number of bits, where v is the number of distinct values (or the number of labels) that the categorical attribute can take. If a bit assigned to a categorical attribute is set to 0 then the corresponding label is included as an inequality in one of the conjuncts.

4.2 Initialising the Population

The initial approach to this problem was to initialise each solution randomly, with the help of a random number generator. This however proved to be ineffective, as many of the solutions obtained were of very poor quality, and NSGA II was unable to produce good results. A similar problem was encountered in the implementation of the modern heuristics.

A more effective initialisation procedure was to use mutated forms of the default rule as initial solutions. The default rule is the rule in which all limits are maximally spaced and all labels are included. In other words, it predicts the class without any pre-conditions. For the initialisation, all solutions in the population pool were initialised to be copies of the default rule and then some of the bits were mutated according to a parameter representing the probability of mutation. Experimentation was carried out to establish a good setting for this parameter.

4.3 Evaluating a Solution

To evaluate a solution, the bit string is first decoded, and the data is scanned through.

When a bit string is decoded as a nugget it will acquire the following format:
IF
$l_1 \leq AT_1 \leq u_1$ AND
$l_2 \leq AT_2 \leq u_2$ AND
...
$l_i \leq AT_i \leq u_i$ AND
$AT_p \neq labelX$ AND
$AT_r \neq labelY$
THEN $Class_j$

where l_1 is given by the first p bits of the binary string, u_1 is given by the following p bits, etc. If a lower limit for any attribute i is set to its lowest possible value for the attribute, min_i, or the upper limit is set to its highest possible value max_i, then there is no need to include that limit in the decoded nugget. If both limits are excluded in that way, then the attribute is obviously also excluded. Equally, if a categorical attribute has a value of 1 for all the bits allocated to its labels, then there is no need to include the attribute. In this way the algorithm can perform its own feature selection by ignoring some attributes.

For each record the values of the fields are compared against the nuggets, and the class is also compared. The counts of c and a are updated accordingly. The counts of b and d are known from the data loading stage.

Once all the data has been examined the measures of interest, in this case the coverage and accuracy, are calculated for each nugget. In the modern heuristic approach, at this stage the fitness of the rule would be calculated.

5 Parameter Experimentation

In this section we introduce the parameters that were experimented with in order to improve the performance of the NSGA II algorithm, and their effect on the quality of solutions. We also discuss the parameters that were used to run the modern heuristic algorithms.

5.1 Parameters for the Modern Heuristics

In our previous work [7], three different algorithms, a Genetic Algorithm, Simulated Annealing and Tabu Search (TS), were used to solve the nugget discovery problem. It was found that the three algorithms performed similarly, with Tabu Search slightly outperforming the others in some occasions, even though only a simple implementation was produced. In the experiments that follow, therefore, we will focus on the Tabu Search algorithm to produce the set of results to be compared to those of NSGA II.

The TS used included very simple recency and frequency memory structures and aspiration criteria. No intensification or diversification techniques, or any other TS enhancements were implemented for this algorithm.

At each step, all neighbours of a solution can be generated, or alternatively, a subset of x neighbours can be generated, where x is set by the user as a parameter.

Various neighbourhood operators were tested (including flip-one-bit, flip-two-bits, swap, reverse and move operators) and the algorithm performed well with all those in terms of the quality of solutions produced. If efficiency is considered (measured by the number of evaluations required to reach the best solution) along with quality of solutions, the flip-one-bit or move operators seem to give best results.

The neighbourhood operators were tested using recency memory with a tabu tenure of 10 iterations, and a subset of 20 neighbours generated. The recency memory was implemented by recording previously visited solutions (i.e. the whole solution) in a list of size n, where n is the tabu tenure, and disallowing a visit to any solution in the tabu list.

After this, frequency memory was tested on the best three operators: flip-one-bit, flip-two-bits and move. Frequency memory used a threshold of 20, that is a single element (or, in a binary representation, a bit) can only be changed 20 times before further changes on that element are disallowed. For all three neighbourhood operators there was no marked improvement in the quality of solutions, but there was an improvement in the average number of evaluations to reach a good solution when frequency memory was used.

Next, different tabu tenures were tried in the context of recency memory. Tenures of 15 and 20 were tried in combination with the three neighbourhood operators previously chosen. The solution quality or efficiency of the search did not improve markedly with greater tabu tenure.

An extra set of experiments was carried out to observe the effect of increasing the size of the subset of neighbours produced, and it was found that this did not produce any clear gains. Increasing the threshold that controls the frequency memory produced no clear gains either.

The final parameters chosen and used in our experiments are:

- Neighbourhood operator: flip-one-bit
- Recency memory with a tabu tenure of 10
- Frequency memory with a threshold of 20
- A subset of 20 neighbours generated

- Stopping after 250 iterations without change in the best solution value.

5.2 Parameters for the MOEA algorithms

The parameters that were experimented with in this case were:

- Cross-over rate: This was varied from 60% to 90% in steps of 10%.
- Stopping condition: in this algorithm the stopping condition was implemented as a prefixed number of generations set by the user. This was varied from 50 to 100 in steps of 10.
- Population size: This was varied from 100 to 160 in steps of 10.
- Initial mutation rate: This affects the mutation of individual bits when solutions are created initially. Mutation also occurs when solutions are reproduced to form new solutions. It was varied between 0% and 4% in steps of 1%.

In terms of parameter experimentation, it was found for all the databases that the mutation rate had the most profound effect on solution quality. A rate of 2% produced the best results in each case consistently. Other values had a marked effect on the quality of the fronts produced. For example, with mutation rate of 4% the quality of solutions was distinctly inferior in each case. This may be due to the use of mutation in the initialisation procedure, in which various copies of the default rule are altered according to it. It may point to the importance of a strong initial population, hence the performance of the algorithm may be improved in the future by introducing some known "good" rules in the initialisation process.

The other parameters did not seem to affect the quality of solutions as dramatically, hence the algorithm was found to be robust to changes on those parameters. Best results were obtained with cross-over rates that varied between 60% and 80%; stopping criteria between 80 and 100 generations, and a population of about 120 solutions.

6 Results

The databases used for this experiments were extracted from the UCI repository. Table 1 represents their main characteristics, including number of classes, and number of numerical and categorical attributes (for more details see [12]). Both the TS and the NSGA II algorithms were used to extract knowledge from the databases of Table 1.

Table 1: Description of standard databases

Name	Records	Class	Num	Cat
Adult	45,222	2	6	8
Mushroom	8,124	2	0	21
Contraception	1,473	3	2	7

The TS algorithm was run using the parameters described in the previous section with a range of λ values for each dataset. This was done in order to compare the spread and quality of solutions obtainable by varying the λ value with those obtainable using NSGA II. For each dataset, λ was set at the value of 1.01 initially (equivalent to an accuracy threshold of 99%) and incremented in small steps (of 0.05 in most cases) to the value of $\frac{d}{b}$, and in some occasions to higher values in order to obtain widely covering solutions of higher accuracy than the default rule. For each λ value, 5 runs of the algorithm were performed and the accuracy and coverage of the best rule obtained in each run was recorded. The modern heuristics can, at times, get stuck in inferior solutions, so in past experiments the best of 5 runs was chosen as the best nugget obtainable with a particular λ value. For this exercise, however, we include all the solutions in order to observe the overall quality and spread of the solutions produced. No attempt has been made to prune any of the inferior or suboptimal solutions obtained. It is worth noting that, as quite often the algorithm returns copies of the same solution, even at different values of λ, the number of distinct solutions may be low, and, in the visualisation of results, overlapping solutions are not noticeable.

For the NSGA II algorithm the solutions plotted are those that form the first non-dominated front at the end of the execution of the algorithm, that is, all those solutions that are non-dominated in the final population.

We choose to evaluate the performance of our algorithms by using visualisation. This seems quite intuitive when there is just two dimensions to visualise. As the true Pareto front is not known, it is not possible to compare to it in order to evaluate performance. Throughout this section, we present the results of both algorithms in 2-dimensional graphs, plotting accuracy of the rules found in the y-axis and coverage in the x-axis. Each graph contains a series labeled MOMH (Multi-objective Meta Heuristic) which represents the results for the NSGA II algorithm, and a series labeled TS which represents the Tabu Search Results. Note that the scales of the y-axis varies in some graphs where the solutions found lie in a reduced accuracy range. This is to allow for improved visualisation of results. However, all solutions are always presented in the graphs.

The results presented in the graphs are those of the train data for both algorithms. Checks on the test data have shown no marked signs of overfitting for either algorithm, except in the case of some of the very accurate and specific patterns.

The Adult database contains census information from adults in the USA. Any records with missing values were removed from this database prior to the algorithm application. The database has two classes: one of them corresponds to "Income $> 50K$" with 24% default accuracy on the training data; the other corresponds to "Income$\leq 50K$" with 76% default accuracy on the training data. The results for the class "Income $> 50K$" are shown on Figure 1. The fronts generated by both algorithms are very similar in terms of

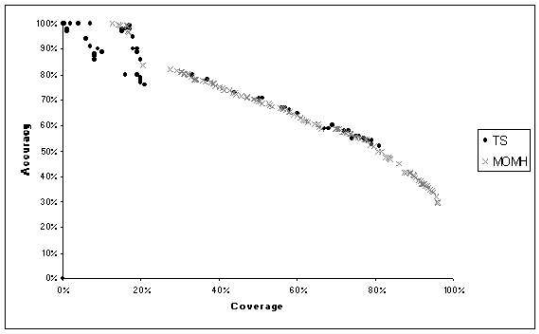

Figure 1: Adult database, class "> 50 K"

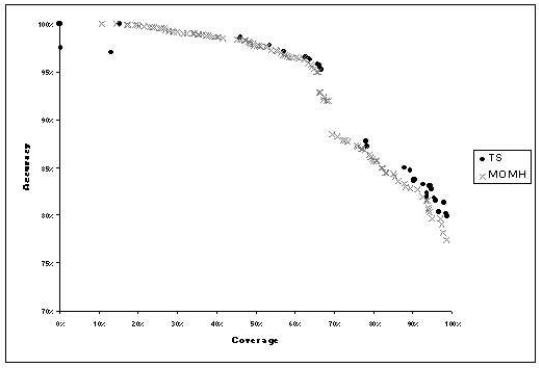

Figure 2: Adult database, class "≤ 50 K"

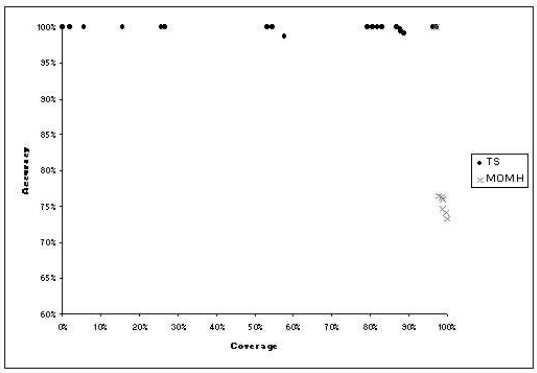

Figure 3: Mushroom database, class "Poisonous"

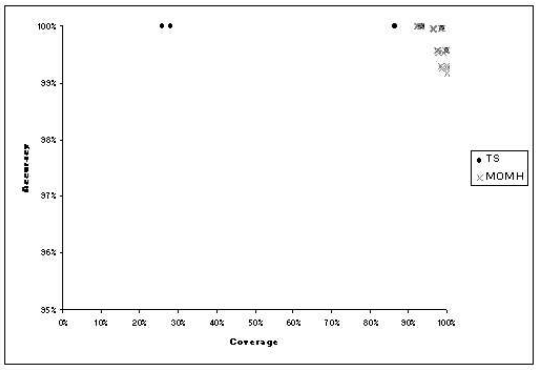

Figure 4: Mushroom database, class "Edible"

the quality of solutions obtained. However TS generates a higher proportion of solutions which are dominated by both, other TS solutions as well as NSGA II solutions. Also the spread of solutions in the front is better with the NSGA II algorithm, although this may be expected as it is a feature of the search in this particular algorithm.

The results for the class "Income $\leq$ 50 K" are shown on Figure 2. Again the results of the two algorithms are of similar quality, but in this case TS marginally outperforms NSGA II in general. Only a very small proportion of the TS solutions are dominated. The TS solutions are clustered in small areas of the front, with some regions being un-represented, hence the NSGA II algorithm has a better spread of solutions over the front.

The mushroom dataset was obtained by mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms [10]. This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. The Mushroom database has two classes: the "Poisonous" class with a default accuracy of 48% and the "Edible" class with a default accuracy of 52%. For this database, rules can be found that are nearly 100% accurate with nearly 100% coverage, hence we have some information about the Pareto front: it is expected to be small and concentrated on the right and upper border of the graph. Results for the "Poisonous" class are given in Fig. 3. The TS algorithm finds a number of dominated solutions. The best accurate rule is found by both algorithms. The NSGA II algorithm finds a cluster of rules of high coverage, which are not found by the TS. Hence, despite the Pareto front for this problem being confined to a small region, the NSGA II is successful in finding distinct solutions in the front.

The results for the "Edible" class are presented in Fig. 4. The accuracy of solutions in the non-dominated front is within a very small range (less than 1%). In this class, both algorithms produce similar quality solutions. The NSGA II algorithm produces more varied solutions in the front.

The contraceptive database is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice of a woman (no use, long-term methods, or short-term methods with default accuracies of 43%, 22% and 35% respectively) based on her demographic and socio-economic characteristics.

Results for the class "no use" are given in Fig. 5. Both algorithms produce non-dominated solutions, in different areas of the Pareto front, although the spread of non-dominated solutions across the front is better for the NSGA algorithm.

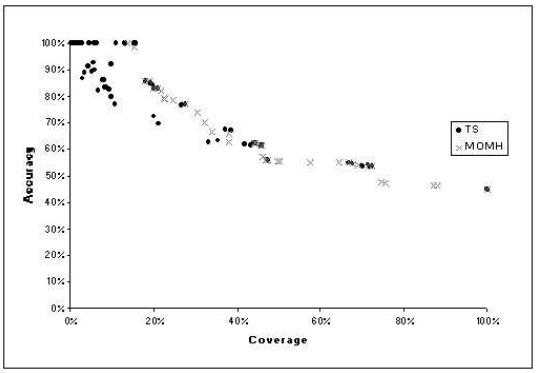

Figure 5: Contraception database, class "no use"

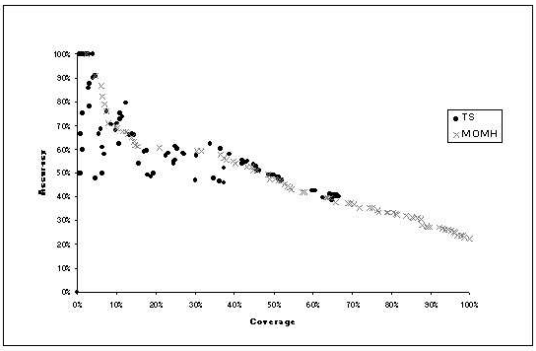

Figure 6: Contraceptive database, class "Long-term"

Results for the class "long-term use" are presented in Fig. 6. In these results TS produces more non-dominated solutions, and the spread of solutions in the front is reasonable, except for those of high coverage (greater than 65%) where NSGA II finds a large number of solutions.

Results for the class "short-term use" are presented in Fig. 7. In the Pareto front area of high accuracy (above 65%) and low coverage (below 25%) the NSGA II algorithm finds all but one of the non-dominated solutions. However, in the other area the algorithms perform very similarly both in terms of solution spread and quality.

6.1 Efficiency Comparisons

To compare the execution time of both algorithms is not straightforward in terms of computing time, since they were run on different machines. However, one method of comparison is to establish the number of evaluations that each algorithm performed to arrive to a particular set of solutions. This is possible because both algorithms use the same underlying evaluation function for a solution. This can only give a rough approximation as both algorithms perform other operations. However, evaluation is the dominating operation in both algorithms, so it can be an initial point of comparison.

The comparison was established using the Adult

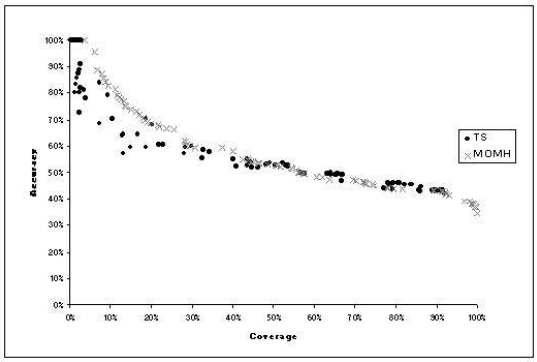

Figure 7: Contraceptive database, class "Short-term"

database. For this database, the TS required approximately 5000 evaluations on average to find one of the solutions, that is, to find a single rule which is represented as one point in the graphs above. The algorithm was run 5 times per λ value, with a range of λ values previously described. This means that 138 experiments were run to get the set of solutions shown in Figure 2. Hence the total number of evaluations performed is nearly 700,000. Even if only one run per λ value was performed, which may deteriorate the quality of the front obtained, approximately 140,000 evaluations would have to be performed. On the other hand, the NSGA II algorithm evaluates a population of $2n$ solutions, in this case $n = 120$, at each generation, and the experiments shown on graph 2 were run for 100 generations. Hence the total number of evaluations performed by this algorithm is 24,000. Therefore, we can conclude that the TS algorithm is a much less efficient way to achieve a set of results of similar quality, in terms of the number of evaluations performed.

7 Conclusions and further work

In this paper, we propose the use of Pareto-based MOEA in the extraction of rules from databases. This is novel and complements previous research in nugget discovery using heuristic techniques. The ability to present the user with a number of interest measures which may be selected, and then to search for a set of solutions which represent an approximation to the Pareto optimal front for those measures is desirable for a partial classification algorithm.

We have implemented the NSGA II Pareto-based MOEA and compared it to the TS algorithm. We have performed parameter experimentation for the former, and having found a range of suitable parameters we have applied the algorithm to a range of well known classification databases. The results have shown that, in terms of quality of individual solutions, both algorithms are comparable. However, the spread of solutions across the approximated Pareto-front found by the NSGA II algorithm was always better. Also, in terms of efficiency, the NSGA II algorithm significantly outperforms the TS approach. Therefore, we can conclude

that in order to find an approximation to the Pareto front, it is best to use the NSGA II algorithm.

Given our results, the potential of NSGA II for nugget discovery is clear. The flexibility of this algorithm will make it advantageous for solving this type of data mining problem.

As the subject of further research, the first step is to compare the performance of the NSGA II algorithm with the true Pareto optimal front. We are already working in this area and results will be reported shortly. It may be interesting to introduce more objectives into the search, such as the simplicity of a rule. It may also be worth while to compare the performance of other Pareto-based algorithms against NSGA II for this particular problem.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. In Nick Cercone and Mas Tsuchiya, editors, *Special Issue on Learning and Discovery in Knowledge-Based Databases*, number 6 in 5, pages 914–925. Institute of Electrical and Electronics Engineers, Washington, U.S.A., 1993.

[2] S. Ali, K. Manganaris and R. Srikant. Partial classification using association rules. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third Int. Conf. on Knowledge Discovery and Data Mining*, pages 115–118. AAAI Press, 1997.

[3] R. J. Bayardo and R. Agrawal. Mining the most interesting rules. In S. Chaudhuri and D. Madigan, editors. *Proceedings of the Fifth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 145–155. New York, USA, 1999. ACM.

[4] R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense datasets. In *Proc. of the 15th Int. Conf. on Data Engineering*, pages 188–197, 1999.

[5] C.A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.

[6] B. de la Iglesia, J. C. W. Debuse, and V. J. Rayward-Smith. Discovering knowledge in commercial databases using modern heuristic techniques. In E. Simoudis, J. W. Han, and U. M. Fayyad, editors, *Proceedings of the Second Int. Conf. on Knowledge Discovery and Data Mining*, pages 44–49. AAAI Press, 1996.

[7] B. de la Iglesia and V. J. Rayward-Smith. The discovery of interesting nuggets using heuristic techniques. In H. A. Abbass, R. A. Sarker, and C. S. Newton, editors, *Data Mining: a Heuristic Approach*, pages 72–96. Idea group Publishing, USA, 2002.

[8] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, Lecture Notes in Computer Science, No. 1917, pp 849–958,2000. Springer.

[9] J. Horn, N. Nafpliotis, and D. E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[10] A. A. Knopf. *The Audubon Society Field Guide to North Americal Mushrooms*. G. H. Lincoff, New York, 1981.

[11] J. Knowles and D. Corne. The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 98–105, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[12] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Coumputer Sciences, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[13] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[14] V. J. Rayward-Smith, J. C. W. Debuse, and B. de la Iglesia. Using a genetic algorithm to data mine in the financial services sector. In A. Macintosh and C. Cooper, editors, *Applications and Innovations in Expert Systems III*, pages 237–252. SGES Publications, 1995.

[15] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[16] X. Yang and M. Gen. Evolution program for bicriteria transportation problem. In M. Gen and T. Kobayashi, editors, *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, pages 451–454, Ashikaga, Japan, 1994.