A NICHED GENETIC ALGORITHM FOR CLASSIFICATION RULES DISCOVERY IN REAL DATABASES

MARCONI A. PEREIRA1,2, JOÃO A. DE VASCONCELOS2

1Departamento de Computação, CEFET/MG
Av. Amazonas, 7675, 30510-000, Belo Horizonte, Brazil
E-mail: marconi@decom.cefetmg.br

2Laboratório de Computação Evolucionária, Departamento de Engenharia Elétrica, UFMG
Av. Antonio Carlos, 6627, 31270-010, Belo Horizonte, MG, Brasil
E-mail: jvasconcelos@ufmg.br

Abstract— This paper presents a Niched Genetic Algorithm (NGAE) that uses Elitism and other techniques that makes it efficient for classification rule mining problems using real databases. This implementation was compared to other classical tools of classification. The results obtained with NGAЕ were similar to those obtained with the classical tools. A great advantage of the proposed Niched Genetic Algorithm is that it can work efficiently with real databases that have a large volume of data. This is because the proposed implementation doesn’t need to load all data at once. The classical implementations of classification tools usually work with a small volume of data. This paper brings too, as a new contribution a SQL modeling to implement and evaluate the individuals.

Keywords— Association Rules, Data Mining, Dissolved Gas Analysis (DGA)

Resumo— Este trabalho apresenta um Algoritmo Genético (NGAE) baseado em técnicas de Nicho e que usa Elitismo, além de outras técnicas que faz com que seja eficiente para problemas de mineração de regras de classificação, utilizando bases de dados reais. Esta implementação foi comparada a outras ferramentas clássicas de classificação e o resultado apresentado foi semelhante aos instrumentos clássicos. Uma grande vantagem da proposta desse novo algoritmo é que ele pode trabalhar eficientemente com bases de dados reais que possuem um grande volume de dados. Isso ocorre porque a execução proposta não precisa carregar para a memória principal todos os dados de uma única vez. As implementações clássicas de ferramentas de classificação geralmente trabalham com um pequeno volume de dados. Este artigo traz também, como uma nova contribuição, uma modelagem utilizando SQL, a fim de implementar e avaliar os indivíduos.

Palavras-chave— Regras de Associação, Mineração de Dados, Análise de Gases Dissolvidos (AGD)

1 Introduction

Mining Rules is one task of the process of Knowledge Discovery Databases (KDD) (Fayyad et al, 1996) and have been studied because of the large amount of practical applications, which could use the generated information. For example:

- **Commercial applications**: to know more about their costumers;
- **Industrial applications**: to prevent faults in its equipments;
- **Educational applications**: to know in which kind of situation students fail.

The task of Mining Rules consists of discovery rules to classify or to associate data. Evolutionary tools are applied with success in this kind of problem (Goldberg, 1989). In this paper, a Niched Genetic Algorithm (NGAE) with Elitism especially designed for real databases are shown. In particular, this algorithm is applied in a Dissolved Gas Analysis (DGA) problem to diagnose incipient faults in power transformers (Morais; Rolim, 2006).

The DGA problem was modeled as an optimization problem and the implemented tool was compared with three other tools: Decision Tree, Support Vector Machine (SVM) and Radial Basis Function (RBF). The results showed that the NGAЕ obtained a satisfactory performance. Also, this paper shows, as a new contribution, a SQL modeling to implement and evaluate each instance (individual).

2 Related Works

In (Dehuri et al, 2008) is shown a Multi-objective Genetic Algorithm with Elitism for the generation and classification (evaluation) of rules (individuals) from a database. In this work, the calculation of the objective functions takes into account the accuracy, specificity and sensitivity. These values are used to find the rules that maximize the number of true positives and minimize the number of false positives extracted from databases. The results showed that the rules have a high accuracy for the databases used in the tests.

In (Srinivasa et al, 2007) is described a model of SAMGA (Self-Adaptative Migration Model Genetic Algorithm), where the chances of mutation, crossover and selection of each individual are dynamically adjusted to the size of the population. This algorithm is used to extract rules for classification of patterns. According to Srinivasa et al, Genetic Algorithms used in the classification can be divided into two classes, depending on how the rules will be codified in the population of individuals:

- **Michigan Approach**: Each individual in the population is a simple rule;
- **Pittsburg Approach**: Each individual encodes a set of prediction rules;
Each rule can be represented by a string of binary characters.

The Michigan approach is used in SAMGA. It was compared with a simple Genetic Algorithm and the results showed the superiority of SAMGA.

In (Sikora; Piramuthu, 2007) a framework for the selection of rules in a genetic algorithm is applied to data mining based on selecting best individuals for a given domain. The algorithm considers each individual as a rule. The fitness function is calculated by counting the number of patterns that match with each rule. This counting considers both positive and negative responses. Each individual has, in addition to the attributes, a sequence of bits to say whether the attribute is considered or not by the rule. The Genetic Algorithm receives the training data to generate the rules for classification. To evaluate the quality of the rules, they were applied to the test data. The proposed framework was compared with a simple Genetic Algorithm. The results showed that the framework performance was better, giving greater accuracy and reducing the execution time.

In (Tan et al, 2005) is shown a distributed co-evolutionary algorithm for extracting data classification rules. The algorithm was validated using the databases available in the UCI Machine Learning. The results showed high accuracy and a decrease of the processing time as the number of processing cores increases.

3 The Problem Modeling

The chosen problem in this work is to find the characteristics of power transformers when they could present a failure. One of the main techniques to find pending failures in this equipment is to analyze the gases dissolved in the mineral oil used to insulate the transformer (Morais; Rolim, 2006). This oil is exposed to heat so that gases are produced. The analysis of the concentration of these gases identifies the conditions in which failures can happen.

Thus, the DGA problem consists of finding rules that identify power transformers that are normal and others that may present an impending failure.

A classical way to measure the effectiveness of a classifier is to produce a confusion matrix of the results (Table 1).

<table>
<thead>
<tr>
<th>Table 1 - Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
</tr>
<tr>
<td>Positive</td>
</tr>
<tr>
<td>Negative</td>
</tr>
</tbody>
</table>

Table 2 - Example of Confusion Matrix

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Normal</th>
<th>Electrical Faults</th>
<th>Thermal Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>40</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Electrical Fault</td>
<td>6</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>Thermal Fault</td>
<td>4</td>
<td>2</td>
<td>23</td>
</tr>
</tbody>
</table>

In this example the number of correct classifications is in the principal diagonal: 40 normal, 16 electrical faults and 23 thermal faults. The other values are wrong classifications. For example, 6 transformers have been classified as normal but they had electrical fault.

The quality of the classifier of each class can be measured using some metrics: Accuracy, Sensitivity and Specificity, where

\[
\begin{align*}
\text{Accuracy} (A) & = \frac{TP + TN}{TP + FN + FP + TN}, \\
\text{Sensitivity} (A) & = \frac{TP}{TP + FN}, \\
\text{Specificity} (A) & = \frac{TN}{TN + FP}.
\end{align*}
\]

In the example given in Table 2, we have for class Normal:

\[
\begin{align*}
TP & = 40, \\
FN & = 2 + 5 = 7, \\
FP & = 6 + 4 = 10, \\
TN & = 16 + 2 + 2 + 23 = 43. \\
\text{Accuracy} (\text{Normal}) & = \frac{40 + 43}{40 + 7 + 10 + 43} = 0,83, \\
\text{Sensitivity} (\text{Normal}) & = \frac{40}{40 + 7} = 0,85, \\
\text{Specificity} (\text{Normal}) & = \frac{43}{43 + 10} = 0,81.
\end{align*}
\]

Similarly the accuracy, sensitivity and specificity can be calculated for the other classes: Electrical Fault and Thermal Fault.

To find the best set of classification rules the problem was modeled in 2 steps: find the set of rules that identify items of all classes and maximize the following function for each class:

\[
f(X) = \text{Accuracy}(X) \times \text{Sensitivity}(X) \times \text{Specificity}(X)
\]  

4 The Niched Genetic Algorithm

To maximize the function (1) and therefore finding the best set of rules that identifies power transformers of all classes was used an algorithm based on the Niched Genetic Algorithm. This algorithm works with individuals that are selection predicate (SQL WHERE-clauses). Thus, it can discover a set of rules that classify the transformers of the database. In Figure 1 is shown the flowchart describing how to classify power transformers. A training set (70% selected randomly of the total data) is used in order to find the best rules to classify the power trans-
formers. The best rules founded are validated using the test set, composed of all the others 30% of data.

Figure 1 - Flowchart of the algorithm proposed.

The pseudocode of the tool is shown in Algorithm 1. This algorithm receives the training set and returns the best rules of each class. The validation using the test set is made using the rules (SQL WHERE-clauses) that will be shown below.

Algorithm 1. Pseudocode of Niched Genetic Algorithm
1. Generate initial population;
2. while number of generations < max_gen do
3. for each individual of population do
4. Look for the individual’s fitness in the cache memory;
5. if (memory cache don’t have this fitness )
6. Access the database and calculate the fitness;
7. Store the fitness in the cache;
8. end if;
9. end for;
10. if (number of generations > 1)
11. Apply elitism;
12. end if;
13. for each niche do
14. Select the best individuals of this niche;
15. end for;
16. Apply the following genetic operations: selection, crossover and mutation;
17. Increment de generation count;
18. end while;
19. return the best individuals.

4.1 The Individual Definition
The individual was modeled as a SQL WHERE-clause. All the tables of data are structured as

\[ \text{chromatography}(F1, F2, F3, F4, F5, \text{Class}) ; \]

where F1, F2, F3, F4 and F5 are respectively the gas concentrations of H2, CH4, C2H2, C2H4 and C2H6, and Class indentifies the classification of the electrical transformer status: A = Normal, B = Electrical Fault and C = Thermal Fault.

In this classification problem, the rules are like the following:

\[ \text{if } F1 > x_1 \land F2 < x_2 \land F3 > x_3 \land F4 > x_4 \land F5 < x_5 \text{ then Class=A} \]

where x1, x2, x3, x4 and x5 are real values generated randomly in the interval \([x_{\text{min}}, x_{\text{max}}]\). Also, x_{\text{min}} and x_{\text{max}} are respectively the minimum and maximum values in the database for the attribute Fi, i \in \{1, 2, 3, 4, 5\}. The symbols greater than ‘>’ and less than ‘<’ are generated randomly as follows:

\[ r = \text{floor}(2\times\text{rand}()); \text{ If } r = 0 \text{ use ‘>’; else use ‘<’.} \]

The sixth individual’s attribute concerns about the class of the individual and is generated based on the existing classes of the database. In the studied DGA problem there are three possible values:

- Class = ‘A’
- Class = ‘B’
- Class = ‘C’

To generate one of these values, the following idea is used:

\[ s = \text{floor}(3\times\text{rand}()); \text{ if } s = 0 \text{ use Class = ‘A’; if } s = 1 \text{ use Class = ‘B’; otherwise, use Class = ‘C’}. \]

As stated previously, the individual was modeled as a SQL WHERE-clause. For the given example above, the individual is:

\[ F1 > x_1 \text{ AND } F2 < x_2 \text{ AND } F3 > x_3 \text{ AND } F4 > x_4 \text{ AND F5 < x_5 AND Class = ‘A’}. \]

This individual is coded in this work as shown in Figure 2.

\[ \begin{array}{cccccccc}
F1 \text{ > x1} & F2 \text{ < x2} & F3 \text{ > x3} & F4 \text{ > x4} & F5 \text{ < x5} & \text{Class = ‘A’} \\
1 & 1 & 1 & 1 & 1 & \\
\end{array} \]

Figure 2 - Database individual

The individuals are composed of six chromosomes where the first five, can be disabled during the execution of the algorithm. The last one, regarding to the class, is never disabled. Thus, this individual can represent, after crossovers and/or mutations, a rule with a reduced number of conditions. An example of an individual with fewer conditions is given in Figure 3.

\[ \begin{array}{cccccccc}
F1 \text{ > x1} & F2 \text{ < x2} & F3 \text{ > x3} & F4 \text{ > x4} & F5 \text{ < x5} & \text{Class = ‘A’} \\
1 & 0 & 1 & 0 & 1 & \\
\end{array} \]

Figure 3 - Individual with conditions Disabled

The correspondent rule of Figure 3 is:

\[ \text{if } F1 > x_1 \land F3 > x_3 \land F5 < x_5 \text{ then Class = A} \]
4.2 The Fitness Evaluation

The calculation of the fitness function is totally based on SQL queries to be held in the database.

In this paper, the queries are shown using relational algebra and SQL-99. The relational algebra elements used are shown in Table 3.

The calculation of fitness function refers to only one individual and can be divided into:

- Calculation of the confusion matrix coefficients;
- Fitness function evaluation, using equation (1).

### Table 3 - Relational Algebra operators

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{\texttt{\textbf{F}}} )</td>
<td>Function aggregate. This operator applies a function (count, max, min, etc.) in the list of attributes. This list can refer to all attributes, using the word all.</td>
</tr>
<tr>
<td>( \text{\texttt{\textbf{\sigma}}} )</td>
<td>Selection. This operator applies the logic predicate of selection in the relation R.</td>
</tr>
<tr>
<td>(-)</td>
<td>Set Difference (represented by minus ‘-’ signal) is used in the Relations. For example, (A - B) results in a set that have the elements of A that are not in B.</td>
</tr>
</tbody>
</table>

Considering that the relation R (F1, F2, F3, F4, F5, Class) contains the gas concentration and the class of each electrical transformer and I is the set of conditions given by the individual of the Genetic Algorithm, the calculation of the confusion matrix values and the calculation of the performance are shown below.

#### A) Calculation of the Confusion Matrix Values

The value of **True Positives** is the number of tuples covered by the individual and belonging to its class. This number of tuples is obtained by the following query using relational algebra:

\[
\text{\texttt{F}} \text{countall} (\sigma(R)),
\]

where I is the set of conditions of an individual of the Niched Genetic Algorithm and R is the relation (table) that contains the information about the electrical transformers.

Using SQL-99, this query can be expressed as:

```sql
SELECT COUNT (*) FROM R WHERE I
```

The **True Negative** value is calculated by counting the actual number of items that are not covered by the individual and don’t belonging to its class. The query that identifies this quantity is:

\[
\text{\texttt{F}} \text{countall} (\sigma_{\text{InotClass}}(R - \sigma_{\text{IwithoutClass}}(R))),
\]

where:

- **InotClass** is the negative of the class coded in individual I. Considering the example in Figure 3, **InotClass** is “Class <> ‘A’”.
- **IwithoutClass** is the set of conditions coded in individual I, without the class. Considering the example in Figure 3, **IwithoutClass** is “F1 > x1 AND F3 > x3 AND F5 < x5”.

Using SQL-99, this query can be expressed as:

```sql
SELECT COUNT (*) FROM (SELECT * FROM R) AS TrueNegative
WHERE InotClass
```

The **False Positive** value is calculated by counting the number of items classified by the individual that don’t belong to its class. Considering the example in Figure 3, the calculation of the False Positive consists in counting the number of items that satisfies the conditions: “F1 > x1 AND F3 > x3 AND F5 < x5” and don’t belong to the class A, that is “Class <> ‘A’”. The query that identifies this quantity is:

\[
\text{\texttt{F}} \text{countall} (\sigma_{\text{IwithoutClass} \text{AND InotClass}}(R))
\]

Using SQL-99, this query can be expressed as:

```sql
SELECT COUNT (*) FROM (SELECT * FROM R) AS Unin
WHERE IwithoutClass
```

The **False Negative** value is the number of items not obtained by the individual and that belongs to its class. The query that identify this quantity is:

\[
\text{\texttt{F}} \text{countall} (\sigma_{\text{IClass} \text{AND IwithoutClass}}(R)),
\]

where **IClass** is the class of the individual. In the example given in Figure 3, this value is “Class = ‘A’”.

Using SQL-99, this query can be expressed as:

```sql
SELECT COUNT (*) FROM (SELECT * FROM R WHERE IClass) AS Uni
WHERE IwithoutClass
```

#### B) Fitness Evaluation

After the calculation of the four values that constitute the confusion matrix, the fitness function is calculated using the formula (1):

\[
F(I, X) = \text{Accuracy}(I, X) \times \text{Sensitivity}(I, X) \times \text{Specificity}(I, X),
\]

where F(I, X) is the fitness function of the individual I that identify items of the class X.
4.3 Mutation

The mutation operator implemented in this work is Bit by Bit (Vasconcelos et al, 2001) where each gene has the same probability to be changed. The mutation can change the relational operator, the numerical value of the gas concentration, the class or simply change the activity of the individual. Considering the example of Figure 3, a possible result of the mutation operation in this individual is (Figure 4):

```
F1 < X1  F2 < X2  F3 > X3  F4 > X4  F5 < X5  Class = 'C'
```

Figure 4 - Individual after mutation operation

In the mutation process the individual has changed one gene, where the condition “F1 > X1” transformed to “F1 < X1” and the class was changed from “Class = ‘A’” to “Class = ‘C’”.

The probabilities of change in the individual selected for mutation were chosen as follows:
- For each chromosome:
  - There are 50% chance to occur a change
    - If the mutation is to be realized, each relational operator, the numeric value Xi, and the activity index has 33% of chance of changing
  - Moreover, the individual can have its class changed with probability of 25%

4.4 Crossover

The crossover operation used in this work is the Uniform Crossover (Vasconcelos et al, 2001), where all genes have the same probability to be changed. A couple of individuals always generate two others. The algorithm is implemented as follows:
- Select two (“father” and “mother”) individuals of the same niche (class), using roulette wheel;
- Generate two other individuals where the first has the same symbols and condition’s activity of the “father” and the second has the same symbols and condition’s activity of the “mother”;
- The new number in each condition is generated using crossover of real number (Goldberg, 1989);
- For each chromosome, there are 25% of chance of changing its activity.

4.5 Niches and Elitism

To obtain a set of rules, where each class of the problem has a correspondent rule of classification, the niche technique was used. In this DGA problem, where there are 3 classes of power transformers, each class was used to identify one niche. More specifically, each individual is shared in niches using its class (the last item of the individual). Thus, the crossover operations are applied in individuals of the same niche, despite the fact that the individuals generated could belong to other niche (class) (see previous section for more details).

To guarantee that the best individuals of each niche (class) will be selected to the next generation, the Elitism Technique was applied. In this work the 10% best individuals are selected, where approximately 3% come from each class. Formally the calculation of the number of individuals of each class is:

\[
\text{ceil} \left( \frac{\text{size of population} \times 0.10}{\text{number of classes}} \right)
\]

4.6 Improvements for Individuals Evaluation

As seen previously, the calculation of the fitness function for each individual is made based on SQL queries executed in databases. As these queries demand disk access, which is time consuming, a simple cache memory to store the fitness of some individuals was implemented. This cache memory consists in a Hash table where the key is the individual and the value stored is the value of the fitness function. The use of this cache reduced the execution time of the experiments of this paper by approximately 80%. This is easy to understand considering that genetic algorithm uses many times the fitness of the individuals, for example, in its operations of selection, in the roulette wheel and elitism technique.

Another technique to improve the quality and reduce the time to discover good rules is to consider the proportion of each gas and not its real value.

Each concentration is calculated as:

\[
\text{new concentration}_i = \frac{\text{actual concentration}_i}{\sum_j \text{concentration}_j}
\]

where \(i\) is the index of the gas concentration, \(n = 5\) and \(i \in \{1, 2, 3, 4, 5\}\).

For example, considering a power transformer with the following concentration of gases:

\[\text{H}_2 = 10, \text{CH}_4 = 9, \text{C}_2\text{H}_2 = 14, \text{C}_2\text{H}_4 = 10, \text{and C}_2\text{H}_6 = 1.\]

The relative proportions of the gases are:

\[\text{H}_2 = 0.227, \text{CH}_4 = 0.205, \text{C}_2\text{H}_2 = 0.318, \text{C}_2\text{H}_4 = 0.227, \text{and C}_2\text{H}_6 = 0.023\]

In this example, 3 decimal digits were used. It’s important to emphasize that using a low precision can compromise the performance of the algorithm. In this paper, the database type ‘double precision’ was used. This type has 15 decimal digits allowing higher quality of results.

Using proportional gas concentration values, instead of using the absolute ones, the result’s quality was improved around 15% in our tests.
5 Experiment and Results

The implemented algorithm was tested using three different databases, labeled as BASE 1, BASE 2 and BASE 3 respectively with 224, 51 and 149 samples. The performance was compared with the results obtained by the Support Vector Machine (SVM), Radial Basis Function Neural Network (RBF) and Decision Tree (J48). All these three tools are from Weka API. The experiment was repeated 3 times for each technique using each base. The presented results in this paper are the average of these 3 results. All the obtained results are shown below:

Table 5 and Table 7 shows the results obtained using balanced databases. The standard deviation (σ) was calculated to help the results comparison, since they are very close. The proposed NGAE presented in this analysis the smallest standard deviation for balanced databases. This means that the proposed tool has performed well to all classes of the databases showing its robustness.

6 Conclusion

This paper proposed a new evolutionary algorithm to be applied in classifications problems. This algorithm uses niche technique, elitism, cache memory, and model the individual as SQL WHERE-clause. In order to evaluate the performance of the design algorithm, a classification problem (DGA to diagnose incipient faults in power transformers) was used. Classical classifications algorithms, like Neural Networks, decision tree and Support Vector Machine, were used in order to compare the obtained results. The results show that the proposed algorithm is competitive and robust, since it presented the best results in many cases.

References