G3PARM: A Grammar Guided Genetic Programming Algorithm for Mining Association Rules

José María Luna, José Raúl Romero and Sebastián Ventura

***Abstract*— This paper presents the G3PARM algorithm for mining representative association rules. G3PARM is an evolutionary algorithm that uses G3P (*Grammar Guided Genetic Programming*) and an auxiliary population made up of its best individuals who will then act as parents for the next generation. Due to the nature of G3P, the G3PARM algorithm allows us to obtain valid individuals by defining them through a context-free grammar and, furthermore, this algorithm is generic with respect to data type. We compare our algorithm to two multiobjective algorithms frequently used in literature and known as NSGA2 (*Non dominated Sort Genetic Algorithm*) and SPEA2 (*Strength Pareto Evolutionary Algorithm*) and demonstrate the efficiency of our algorithm in terms of running-time, coverage and average support, providing the user with high representative rules.**

I. Introduction

With the rapid growth in the size and number of available databases, mining for knowledge, regularities or high level information from data became essential to support decision making and predict future behavior. Association rule mining is an important task involving data mining and knowledge discovery in databases. Basically it is the process of finding some relationships among the attributes and attribute values of a large database. Within the huge collection of data stored in a database, there could be a lot of relationships between the many attributes. Discovery of such relationships within a vast amount of data could greatly help in decision-taking. These relationships can be represented as a relation $A \rightarrow C$ where A and C symbols refer to the antecedent and consequent, respectively.

Existing algorithms for mining association rules [1], [2] are mainly based on the approach suggested by *Agrawal et al.* [3], [4] called the *Apriori* algorithm. A limitation of this algorithm is that it works in two phases and also its computational cost is very high. The first phase is for frequent itemset generation, and the second generates the rules from frequent sets. To reduce its computational cost, this algorithm establishes that if an itemset is frequent, all its subsets must also be frequent. However, supersets of an infrequent itemset are not frequent, so the computational cost is reduced by removing such results. Another limitation is the encoding scheme, where separate symbols are used for each possible value of an attribute. This encoding scheme may be suitable for encoding the categorically valued attributes, but not for encoding the numerically valued attributes as they may have different values in each record. To avoid this situation, some ranges of values may have to be defined by using discretization techniques.

Many studies [5], [6] have already proposed Evolutionary Algorithms (EAs) [7] for rules/knowledge extraction from databases, as they consider this kind of algorithm, and especially Genetic Algorithms (GAs) [8], as one of the most successful search techniques applied in complex problems, and they have proved to be an important technique for learning and mining knowledge. GAs are robust and flexible search methods. In fact, the same GA can be executed using different representations. In addition, GAs also allow feasible solutions to be obtained within specified time limits. This is why data mining experts have shown an increasing interest in both EAs and GAs. Genetic programming (GP) is a methodology based on EA to find computer programs by employing operations inspired by biological evolution. The main disadvantage of the process of generating individuals by GP is the generation of invalid individuals. An extension of GP is G3P (*Grammar Guided Genetic Programming*) [9], [10], [11] which allows valid individuals to be obtained by defining them as context-free grammars so that they formally describe the syntactic constraints of the problem to be solved.

Many real-world problems involve simultaneous optimization of often competing objectives. Often, there is no single optimal solution, but rather a set of alternative solutions. These solutions are optimal in the wider sense that no other solutions in the search space are better than they are at taking all objectives into consideration. They are known as Pareto-optimal solutions. Some researchers suggest that multiobjective search and optimization might be a problem area where EAs do better than other blind search strategies. Existing studies [12], [13] treat association rule mining as a multiobjective problem rather than as a single objective one. The objective functions like confidence, comprehensibility, interestingness, etc., can be thought of as different criterion of an association rule mining problem.

This paper presents an evolutionary algorithm using G3P for mining association rules. This algorithm, called G3PARM, is able to generate valid individuals, not larger than a predefined size, and to obtain representative rules from datasets in a short time. Furthermore, G3PARM can be used with numerical attributes by simply changing the grammar from which individuals are obtained, so the algorithm presented can be used for any type of dataset. In this study, we make a comparison between G3PARM and two multiobjective algorithms

J.M. Luna (*Student Member, IEEE*), J.R. Romero (*Member, IEEE*) and S. Ventura (*Senior Member, IEEE*) are with the Dept. of Computer Science and Numerical Analysis, University of Córdoba, Rabanales Campus, Albert Einstein building, 14071 Córdoba, Spain. Email: {i32luarj, jrromero, sventura}@uco.es.

$G = (\Sigma_N, \Sigma_T, Rule, P)$ with:
Σ_N = {Rule, Antecedent, Consequent, Comparison, Categorical Comparator,
Categorical Attribute Comparison }
Σ_T = {AND, "! =", "=", "name", "value"}
P = {Rule = Antecedent, Consequent ;
Antecedent = Comparison | AND, Comparison, Antecedent ;
Consequent = Comparison ;
Comparison = Categorical Comparator, Categorical Attribute Comparison ;
Categorical Comparator = "! =" | "=" ;
Categorical Attribute Comparison = "name", "value" ;}

Fig. 1. Context-free grammar expressed in Extended BNF.

frequently used in literature: NSGA2 (*Non dominated Sort Genetic Algorithm*) [14], [15] and SPEA2 (*Strength Pareto Evolutionary Algorithm*) [16], [17] to demostrate the efficiency of our proposal in terms of running-time, coverage and average support. For multiobjective algorithms, support and confidence measures will be used as different objectives of the rule mining problem.

This paper is structured as follows: Section II describes the algorithm conceived and its main characteristics; Section III describes the datasets used in the experiments and the algorithms used for comparing; Section IV describes both the execution and results; finally, some concluding remarks are underscored and future research lines that we expect to tackle are discussed.

II. G3PARM ALGORITHM

This section presents our model along with its major characteristics: how individuals are represented, its genetic operators, the evaluation process and the algorithm used.

A. Individual representation

A context-free grammar G is defined as a 4-tuple $G = (\Sigma_N, \Sigma_T, Rule, P)$ where $\Sigma_N \cap \Sigma_T = \emptyset$, Σ_N is the alphabet of nonterminal symbols, Σ_T is the alphabet of terminal symbols, $Rule$ represents the start symbol, and P is the set of production rules, written in Extended BNF. Individuals are defined as the derivation syntax-tree where the root is the start symbol $Rule$, the internal nodes contain only nonterminal symbols and the leaves nodes contain only terminal symbols. The series of derivation steps that generate a sentence that is a possible solution to the problem is represented by the derivation syntax-tree. Therefore, an individual codifies a sentence of the language generated by the grammar as a derivation syntax-tree. Figure 1 shows the context-free grammar that represents the rules codified by population individuals. The chromosome encodes its expression using a preorder route. It should be noted that the terminal grammar symbol "name" is determined by the dataset attributes used each time. Moreover, for each grammar attribute, the value that is assigned is determined by the range of that attribute's values in the dataset.

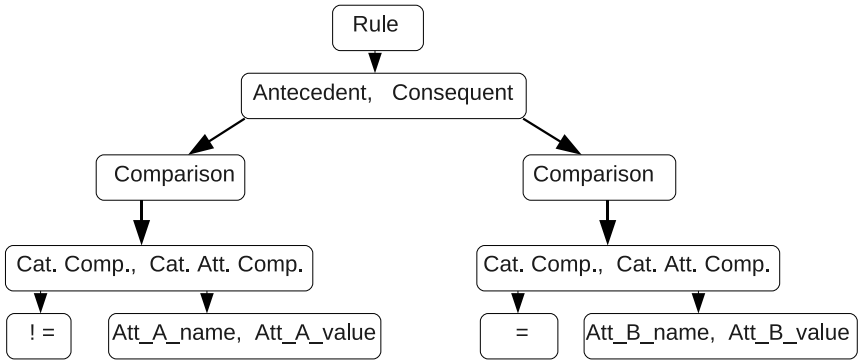
Each individual is represented by a syntax-tree structure (Figure 2(a)) according to the defined grammar. Individuals are composed of two distinct components: a genotype, encoded with a syntax-tree structure using G3P with limited depth to avoid infinite derivations, and a phenotype (Figure 2(b)), that represents the complete rule consisting of an antecedent and a consequent. The phenotype is obtained by eliminating genotype nonterminal symbols. The antecedent of each rule is formed by a series of conditions that contain the values of certain attributes that must all be satisfied. By contrast, the consequent is composed of a single condition. This is because G3PARM looks for rules with high support and confidence and, the lower the number of conditions of the consequent, the higher the confidence of the rule.

To carry out the derivation of the symbols that appear in the grammar, we use the cardinality concept. The number of derivation chains that can be generated from a context-free grammar is infinite. However, we can group them into sets generated by *d* derivations. The cardinality concept is defined as the number of elements generated in a set. The cardinality of each nonterminal symbol will be based on the set generated in *d* derivations. If a nonterminal symbol can be derived in several ways, the cardinality of the nonterminal symbols will be determined by the sum of the cardinalities of each of the possible derivations of that symbol. If a derivation has more than one nonterminal symbol, the cardinality of the set formed by the symbols will be determined by the product of the cardinalities of each nonterminal symbol present in the derivation.

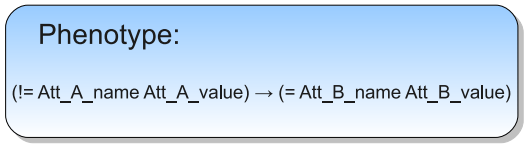
Each individual is generated from the initial grammar symbol $Rule$ through the random application of production rules P until a valid derivation chain is reached. The number of derivations is determined by the maximum derivation size provided in algorithm configuration parameters. To generate an individual, the algorithm starts with the initial grammar symbol and maximum derivation size. From this symbol, the productions sought are obtained based on derivation size and one is chosen at random taking into account that, the larger the cardinality of a symbol, the greater the probability this symbol has of deriving. The algorithm continues recursively for each nonterminal symbol, reducing the maximum derivation size at each iteration.

B. Genetic operators

We use two genetic operators to generate new individuals in a given generation of the algorithm: crossover and mutation.



(a) Example of genotype of an individual represented in a syntax-tree structure.



(b) Example of phenotype of an individual.

Fig. 2. How to representate the individuals.

Both operators work according to defined grammar.

Crossover: this operator creates new individuals by exchanging two parent derivation subtrees from two randomly selected compatible nodes in each of them. Two nodes are compatible if they belong to the same nonterminal symbol under the condition that this node not be the initial grammar symbol.

If two compatible nodes are found, the crossover is completed by swapping the two subtrees underneath these two nodes. In addition, an overhead checking is carried out to ensure that the individuals produced by crossover satisfy size constraints, since the size of the individuals produced must remain within a predefined range between the minimum and maximum size.

If one parent, or both, exceed the maximum size, this parent is kept intact and, therefore, the parent and the offspring are equal.

Mutation: this operator randomly selects a syntax-tree node and will act based on the symbol type. If the selected node is a nonterminal symbol, a new production of the grammar is used to derive a new subtree. We must take into account that, if the selected item is a nonterminal symbol, and because it is possible to make a derivation different from that undertaken in the first instance, the number of derivations needed to reach a terminal symbol may vary in order to prevent exceeding the maximum derivation size imposed by the algorithm. If, however, the selected node is a terminal symbol, it changes the value of the terminal symbol at random.

C. Evaluation

Before evaluating an individual, we must carry out its decoding, namely, by finding the association rule that corresponds to the genotype syntax-tree. The decoding process is to build an in-depth expression of the syntax-tree and remove nonterminal symbols that appear in the genotype. It is also necessary to verify that individuals do not have the same attribute in rule antecedent and consequent. The evaluation process of individuals is performed by obtaining the fitness function value. It will be the support, which is defined as the ratio (in percentage) of records that contains A and C to the total number of records in the database. Here, A is called antecedent, and C consequent. The support of a rule R is defined as:

$$sup(R) = \frac{sup(A \bigcup C)}{|D|}$$

where $sup(A \bigcup C)$ is the support of the itemsets A and C, and the D symbol refers to the dataset. The support of a rule ($sup(R)$) always takes continuous values between 0 and 1, while the support of any item will be any positive integer.

Another heuristic that we will use is the confidence rule generated by the decoding process. This is defined as the ratio (in percentage) of the number of records that contain A and C to records that contain A. The confidence of a rule R is defined as:

$$conf(R) = \frac{sup(A \bigcup C)}{sup(A)}$$

and, as with the support, confidence always takes continuous values between 0 and 1.

D. Algorithm

The G3PARM algorithm uses an auxiliary population with the best individuals, who will act as parents for the next generation. Thus the best individuals obtained are not lost with the passing of generations. Furthermore, because the individuals in the auxiliary population may act as parents in subsequent generations, the algorithm tends to converge to the best individuals. In the initial generation, the auxiliary population will be empty.

The algorithm starts producing the population by randomly generating individuals from the context-free grammar defined in Figure 1 and by fulfilling the maximum number of derivations. Based on the population, individuals are selected via a binary tournament. This selector works by selecting two individuals randomly from the current population and after comparing them, it keeps the best of them. Individuals are selected to act as parents for the crossover based on their probability, being more probable the higher the probability. The next step is to perform the mutation of the individuals selected. Like crossover, mutation will depend on its probability. Once we have the new population by crossover and mutation, we update the auxiliary population. To update the auxiliary population, the previous auxiliary population and the current population are combined. Then, the individuals are ranked according to the support and individuals with the same genotype are eliminated. The G3PARM algorithm takes two individuals as equals if, having different genotypes, they contain the same attributes. For example, the algorithm considers that the case of the rules A AND $B \rightarrow C$ and B AND $A \rightarrow C$ are equal. From the resulting set, we select the individuals that exceed a certain threshold of support and confidence.

The G3PARM algorithm is represented by the pseudocode in Algorithm 1. Algorithm 2 shows how the auxiliary population is updated.

Algorithm 1 G3PARM algorithm

Require: $max\_generations, N$
Ensure: A
  $P_0 \leftarrow random(N)$
  $A_0 \leftarrow \emptyset$
  while $num\_generations < max\_generations$ **do**
    Select parents ($P_t \cup A_t$)
    Crossover (P')
    Mutation (P')
    $P \leftarrow P'$
    Update auxiliary population (A_t)
    $num\_generations++$
  end while
  return A

As indicated in Section II-C, the support is the fitness, so individuals attempt to maximize support over generations, while the confidence measure is maximized in the auxiliary population. Thus, the algorithm tries to maximize both measures over generations.

Algorithm 2 Update auxiliary population

Require: A
Ensure: A
  $A' \leftarrow P' + A_t$
  Order (A')
  Eliminate duplicate (A')
  $A_t \leftarrow$ Threshold(A')
  return A

The algorithm terminates when it covers all the instances from the dataset or when it reaches a certain number of generations. When this occurs, the auxiliary population individuals are returned.

III. Experimentation

To evaluate the performance of our proposal for mining association rules, several experiments have been carried out on different datasets using different algorithms. First, the two algorithms that are compared are explained in detail and subsequently we show the different datasets used.

A. Multiobjective algorithms

This section presents two multiobjective algorithms frequently used in the literature, SPEA2 and NSGA2, which use the support and confidence measures of rules as objectives to optimize. When the algorithms finish, the rules located in the Pareto front are returned.

1) SPEA2: This algorithm allows us to find or approximate the Pareto-optimal set for multiobjective optimization problems. The fitness of each individual takes into account the number of individuals which it dominates and also the number of individuals which are dominated by it. The first step is to calculate the number of individuals it dominates. This is the strength value of each individual. Next, we calculate the raw fitness which is the sum of the strength values of the individuals that dominate it.

Another fitness component that we need is the density of information. This is a nearest neighbor density estimation. If individuals in the population establish a few dominance relationships among themselves (for example, all lie on the Pareto front), large groups of individuals with the same fitness will be formed, so the algorithm chooses individuals with a high degree of randomness. Because of this, a density estimator called the *K-th* nearest neighbor is added to the raw fitness.

First, for each individual i we must compute the Cartesian distance between it and every other individual. This gives us a vector of distances which must be ranked from smallest to largest. Then we take the *K-th* element of the vector and use the equation:

$$D(i) = \frac{1}{distance(k) + 2}$$

This value is adding to the raw fitness to obtain the final fitness of this individual.

2) NSGA2: This algorithm organizes the population in fronts of nondominated individuals, assigning to each individual the value of the front to which it belongs. The next step is to get an estimate of the solution density that surrounds a particular solution in the population (we calculate the average distance of points on both sides of this point along each of the objectives).

Next, we update the new population, assigning individuals from the first front towards the last front, until the population reaches its complete size. Assume that every individual in the population has two attributes: nondomination front and crowding distance. Hence, between two solutions with differing nondomination fronts, we prefer the solution with the better front. Otherwise, if both solutions belong to the same front, then we prefer the solution that is located in a less crowded region.

Finally, we assign the new population of size N to the next generation.

B. Datasets used for the experimentation

Table I shows the different datasets used indicating name, number of instances and number of attributes. Numerical data has been preprocessed using the *equal-frequency binning*[1] [18] discretization technique in five and ten intervals.

WDBC, *WPBC* and *WDatabaseBC* datasets correspond respectively to the sets of data: *Wisconsin Diagnostic Breast Cancer*, *Wisconsin Prognostic Breast Cancer* and *Breast Cancer Database*. *HH* concerns a study to predict the median price of houses in a region by considering both demographic composition and the state of the housing market. This data was collected as part of the 1990 US census. Finally, *Soybean* and *Mushroom* datasets were obtained from the UCI[2] repository.

TABLE I
DATASETS PROPERTIES.

Name	Records	Attributes
HH	22784	17
$Mushroom$	8124	22
$Soybean$	683	36
$WDatabaseBC$	683	11
$WDBC$	569	31
$WPBC$	194	34

The G3PARM algorithm and the two multiobjective algorithms presented have been developed using JCLEC software (*Java Class Library for Evolutionary Computation*) [19], that serves as a framework for the development of evolutionary computation applications. To obtain the configuration parameters, a series of tests has been carried out to check the behavior of the algorithms. The configuration parameters obtained from these tests are shown in Table II.

The algorithms wind up their execution when the set of rules obtained (from the auxiliary population or from the Pareto front, as appropriate) covers 100% of the dataset instances. Because some performances fail to finish within a reasonable time, if the execution reaches 1000 generations without being able to cover 100% of the dataset instances, the algorithm execution will end in that generation.

All the experiments were performed using an Intel Core i7 with 12GB of memory and running CentOS 5.4.

[1]This method involves dividing the values range in constant frequency intervals

[2]*The UCI Machine Learning repository can be reached at:* `http://archive.ics.uci.edu./ml/datasets.html`

TABLE II
CONFIGURATION PARAMETERS FOR THE ALGORITHMS.

G3PARM	
Individuals	75
Crossover Probability	70%
Mutation Probability	10%
Maximum Derivation Size	24
External Population Size	20
External Confidence Threshold	90%
External Support Threshold	70%

SPEA2	
Individuals	150
Crossover Probability	85%
Mutation Probability	20%
Maximum Derivation Size	24
Neighbours Proximity	2

NSGA2	
Individuals	50
Crossover Probability	60%
Mutation Probability	15%
Maximum Derivation Size	24

IV. RESULTS

This section compares the results obtained with G3PARM, SPEA2 and NSGA2 algorithms for each dataset. The results are shown in Table III, where *average_sup* is the average support of the rule set; *average_conf* is the average confidence of the rule set; *global_sup* represents the support of the rule set obtained (the percentage of records covered by these rules over the total records in the dataset).

It should be noticed that the results obtained by the proposed algorithms are the average results obtained running our algorithms with ten different seeds. These seeds are used for the generation of random individuals by creating different individuals based on the seed used, and therefore it is necessary to use several seeds and not rely on only one.

Analyzing the results presented in Table III, we can note that the G3PARM algorithm obtains rules that cover 100% of dataset instances. Only in the WDBC5 dataset does the algorithm not cover all instances, although it comes close (99.19%), but it does manage to obtain better coverage than the other algorithms. Furthermore, G3PARM optimizes the support better than multiobjective algorithms, achieving a much greater average support for different datasets. Because maximizing the support involves a maximization of confidence, G3PARM manages to optimize average confidence by maximizing the support (fitness function) and is helped by auxiliary population thresholds. G3PARM reaches an average confidence of over 92% in most cases. By contrast, multiobjec-

TABLE III

RESULTS OBTAINED BY THE ALGORITHMS.

G3PARM

Name	average_sup	average_conf	global_sup
$HH5$	0.7481	0.9120	100.00%
$HH10$	0.8010	0.9141	100.00%
$Mushroom$	0.7998	0.9283	100.00%
$Soybean$	0.8110	0.9476	100.00%
$WDatabaseBC5$	0.8025	0.9212	100.00%
$WDatabaseBC10$	0.8604	0.9516	100.00%
$WDBC5$	0.7550	0.9448	99.19%
$WDBC10$	0.7956	0.9085	100.00%
$WPBC5$	0.7687	0.9614	100.00%
$WPBC10$	0.8046	0.9144	100.00%

SPEA2

Name	average_sup	average_conf	global_sup
$HH5$	0.5563	0.9723	94.92%
$HH10$	0.6543	0.9868	98.81%
$Mushroom$	0.9371	0.9945	100.00%
$Soybean$	0.8622	0.9859	99.59%
$WDatabaseBC5$	0.6430	0.9825	99.21%
$WDatabaseBC10$	0.6738	0.8835	89.25%
$WDBC5$	0.6772	0.9942	81.86%
$WDBC10$	0.6158	0.9793	97.54%
$WPBC5$	0.7379	0.9755	98.04%
$WPBC10$	0.5778	0.9854	99.54%

NSGA2

Name	average_sup	average_conf	global_sup
$HH5$	0.4662	0.9545	96.33%
$HH10$	0.5783	0.9704	99.75%
$Mushroom$	0.8521	0.9891	100.00%
$Soybean$	0.6723	0.9763	100.00%
$WDatabaseBC5$	0.6685	0.9637	100.00%
$WDatabaseBC10$	0.8010	0.9904	100.00%
$WDBC5$	0.5854	0.9567	96.17%
$WDBC10$	0.5013	0.9521	100.00%
$WPBC5$	0.6582	0.9510	98.61%
$WPBC10$	0.6865	0.9648	99.69%

TABLE IV

RUNTIME AND NUMBER OF RULES OBTAINED BY THE ALGORITHMS.

G3PARM

Name	n_generations	n_rules	runtime
$HH5$	169.5	10.9	171079.7
$HH10$	50.7	9.3	54413.4
$Mushroom$	129.3	8.6	133273.5
$Soybean$	34.3	7.5	1976.1
$WDatabaseBC5$	28.8	9.4	5726.7
$WDatabaseBC10$	7.7	7.0	922.4
$WDBC5$	528.0	13.6	19655.6
$WDBC10$	7.9	6.4	693.1
$WPBC5$	434.6	11.9	8409.3
$WPBC10$	3.8	5.5	344.2

SPEA2

Name	n_generations	n_rules	runtime
$HH5$	908.7	39.4	5894740.9
$HH10$	762.4	21.7	4263430.2
$Mushroom$	3.6	2.2	6166.6
$Soybean$	220.8	3.8	25892.6
$WDatabaseBC5$	563.5	15.1	2219500.7
$WDatabaseBC10$	405.3	5.9	38863.9
$WDBC5$	913.3	12.7	133064.9
$WDBC10$	335.1	9.4	46556.5
$WPBC5$	800.0	5.5	31489.6
$WPBC10$	408.0	10.0	15581.5

NSGA2

Name	n_generations	n_rules	runtime
$HH5$	1000	9.9	2376091.9
$HH10$	482.7	12.7	1068132
$Mushroom$	27.4	2.9	14512.9
$Soybean$	94.1	4.5	4434.3
$WDatabaseBC5$	169.2	8.0	8766.1
$WDatabaseBC10$	84.3	5.2	4226.0
$WDBC5$	741.2	5.5	39235.4
$WDBC10$	114.1	7.7	5506.1
$WPBC5$	408.9	5.2	6235.5
$WPBC10$	331.6	5.8	5340.7

tive algorithms have both confidence and support as objectives to maximize. Multiobjective algorithms move by fronts and attempt to maximize the two objectives. Bearing in mind the concept of dominance in the multiobjective algorithms, and because maximizing confidence is easier than maximizing support, we deduce that multiobjective algorithms offer an average confidence that is quite high compared to average obtained support.

If we focus on Table IV and the number of rules (*n_rules*) obtained, we can see that the G3PARM algorithm manages to cover all instances of the datasets with between 5 and 14 rules. However, multiobjective algorithms require a range of rules of $[2, 40]$ for SPEA2 and $[2, 13]$ for NSGA2 and fail to cover all instances in the datasets.

If we look at the NSGA2 algorithm, we can observe how the number of rules obtained is similar to the G3PARM algorithm and, moreover, manages to cover almost 100% of the instances. If we focus only on global support and the number of rules, we could say that their behavior is similar to that of the G3PARM algorithm. However, this statement is not correct because the average support obtained with NSGA2 is much lower than that obtained with the generational algorithm.

This is where we question how, with approximately the same number of rules, NSGA2 covers almost 100% of the instances although it has lower average support. The answer is that: NSGA2 gets very good rules but also very bad rules, so the average falls and yet, at the same time these bad rules help to cover the dataset instances.

Finally, the running-time is much less in the G3PARM algorithm than in multiobjective algorithms. 8 out of 10 experiments have better times for the G3PARM algorithm than for multiobjective algorithms. This is because the G3PARM algorithm covers all the instances without going to the maximum number of iterations. Figure 3 shows the running-times (in milliseconds) for each dataset on a logarithmic scale.

To compare the results obtained and to analyze if there are any significant differences between the three algorithms, we use the *Friedman* test. This test first ranks the jth of k algorithms on the ith of N datasets, and then calculates the average rank according to the *F-distribution* (F_F) throughout all the datasets, and calculates *Friedman* statistics. If the *Friedman* test rejects the null-hypotehsis, we go on to carry out a *Bonferroni-Dunn* test to reveal the differences between algorithms. Using the *Friedman* test, we evaluate the perfor-

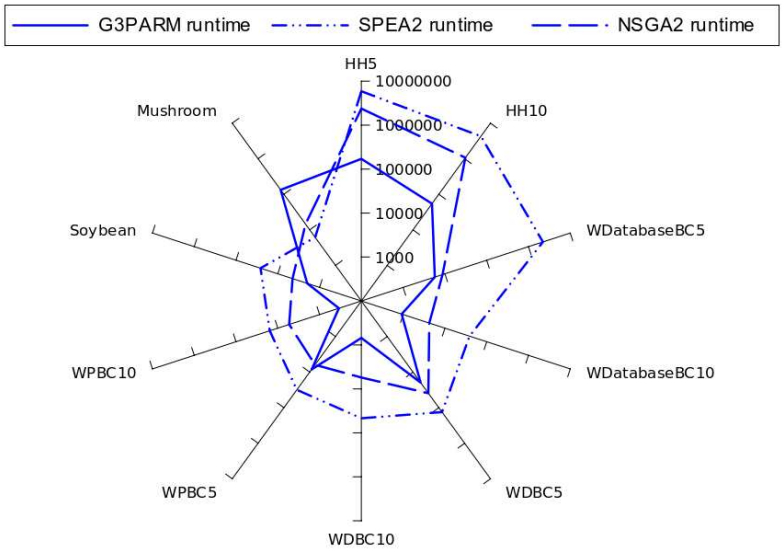


Fig. 3. Runtime for each dataset.

mance of G3PARM by comparing it to the other algorithms using average support, global support and the running-time of each algorithm in all the datasets. Average rankings of all the algorithms considered are summarized in Table V, where we can see that the computed control algorithm (the algorithm with the lowest ranking) is our proposal.

The *Friedman* average ranking statistic for support measure distributed according to F_F with $k-1$ and $(k-1)(N-1)$ degrees of freedom is 6.7894, which does not belong to the critical interval $[0, (F_F)_{0.05,2,18} = 3.5545]$. On the other hand, if we focus on measuring global support, the *Friedman* average rankings statistic is 18.2727, which does not belong to the critical interval $[0, (F_F)_{0.05,2,18} = 3.5545]$. Thus, we reject the null-hynothesis that all algorithms perform equally well for support and global support measures.

TABLE V
AVERAGE RANKING OF THE ALGORITHMS.

Support

Algorithm	Ranking
$G3PARM$	1.3
$SPEA2$	2.1
$NSGA2$	2.6

Global support

Algorithm	Ranking
$G3PARM$	1.3
$SPEA2$	2.9
$NSGA2$	1.8

Due to the significant differences between the three algorithms, we use the *Bonferroni-Dunn* test to reveal the difference in performance and the Critical Difference (CD) value is 1.2553 considering $p = 0.01$.

The results obtained indicate that, for the support measure at a significance level of $p = 0.01$ (i.e., with a probability of 99%), there are significant differences between the G3PARM and NSGA2 algorithms, the performance of G3PARM being statistically better than that of NSGA2. G3PARM is also competitive with SPEA2 in terms of support measure. If we focus on global support measure, the results indicate that, at a significance level of $p = 0.01$ (i.e., with a probability of 99%) there are significant differences between G3PARM and SPEA2, the performance of G3PARM being statistically better than that of SPEA2. G3PARM is also competitive with NSGA2 in terms of global support measure.

V. CONCLUSIONS AND FUTURE WORK

This paper has presented a comparison between the G3PARM algorithm to discover association rules based on an auxiliary population, and two multiobjective algorithms frequently used in literature and known as NSGA2 and SPEA2. By evaluating the results obtained in Section IV, the following conclusions can be drawn with respect to the effectiveness of our proposal:

- The association rules obtained by our proposal maintain a high support and a high confidence level, providing the user with high representative rules.
- Our proposal lets us obtain a reduced set of association rules, since the number of rules is restricted by the size of the auxiliary population. Also, with this small association rule set, we managed to cover all the instances in the dataset.
- The running-time of our proposal is much shorter than that needed by multiobjective algorithms.
- The algorithm proposed can be used with both numerical and categorical attributes by simply changing the grammar that obtains the individuals, so the algorithm that we present can be used for any type of dataset.

Future work includes new approaches that can be used in association with multiobjectives; we have not taken into account in this study as can be interestingness measure. In the future, we will explore the use of rare itemsets [20], [21] by modifying our algorithm to work with this type of patterns. In this field, we will verify the performance of multiobjective algorithms as we have done in the present study.

ACKNOWLEDGMENT

This research is subsidized by the Regional Andalusian Government and Ministry of Science and Technology projects P08-TIC-3720, TIN2008-06681-C06-03, and FEDER founds.

REFERENCES

[1] F. Bodon, "A tire-based apriori implementation for mining frequent item sequences," in *1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementation*, 2005, pp. 56–65.

[2] C. Borgelt, "Efficient implementations of Apriori and Eclat," in *FIMI'03, 1st Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, USA*, December 2003.

[3] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile*, September 1994, pp. 487–499.

[4] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *SIGMOD Conference, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C.*, May 1993, pp. 207–216.

[5] J. Mata, J. L. Alvarez, and J. C. Riquelme, *Discovering Numeric Association Rules via Evolutionary Algorithm*, ser. Lecture Notes in Computer Science, 2002, vol. 2336/2002, pp. 40–51.

[6] X. Yan, D. Zhang, and S. Zhang, "Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support," *Expert Systems with Applications*.

[7] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer-Verlag New York, LLC, 2003.

[8] X. Zhu, Y. Yu, and X. Guo, "Genetic algorithm based on evolution strategy and the application in data mining," in *ETCS'09, International Workshop on Education Technology and Computer Science, Wuhan Shi, China*, March 2009, pp. 848–852.

[9] P. A. Whigham, "Grammatically-based genetic programming," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, California, USA*, J. P. Rosca, Ed., July 1995, pp. 33–41.

[10] M. L. Wong and K. S. Leung, *Data Mining Using Grammar-Based Genetic Programming and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[11] J. Couchet, D. Manrique, J. Ríos, and A. Rodríguez-Patón, "Crossover and mutation operators for grammar-guided genetic programming," *Soft Comput.*, vol. 11, no. 10, pp. 943–955, 2007.

[12] A. Ghosh and B. Nath, "Multi-objective rule mining using genetic algorithms," *Inf. Sci.*, vol. 163, no. 1-3, pp. 123–133, 2004.

[13] S. Dehuri, A. Jagadev, A. Ghosh, and R. Mall, "Multi-objective genetic algorithm for association rule mining using a homogeneous dedicated cluster of workstations," *American Journal of Applied Sciences*, vol. 11, no. 3, pp. 2086–2095, 2006.

[14] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.

[15] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, Paris, France*. Springer-Verlag, September 2000, pp. 849–858.

[16] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[17] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, K. Giannakoglou *et al.*, Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[18] J. Han and M. Kamber, *Data Mining. Concepts and Techniques*, 2nd ed., ser. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2006.

[19] S. Ventura, C. Romero, A. Zafra, J. Delgado, and C. Hervás, *JCLEC: a framework for evolutionary computation*, ser. Soft Computing. Springer Berlin / Heidelberg, 2007, vol. 12, pp. 381–392.

[20] Y. Koh and N. Rountree, *Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection*. Information Science Reference, Hershey, New York, 2010.

[21] M. Adda, L. Wu, and Y. Feng, "Rare itemset mining," in *ICMLA'07, Sixth International Conference on Machine Learning and Applications, Cincinnati, Ohio*, December 2007, pp. 73–80.