# Biogeography-Based Rule Mining for Classification

**Effat Farhana**
Department of Computer Science
North Carolina State University
Raleigh, United States
efarhan@ncsu.edu

**Steffen Heber**
Department of Computer Science
North Carolina State University
Raleigh, United States
sheber@ncsu.edu

## ABSTRACT

Rule-based classification is a popular approach for solving real world classification problems. Once suitable rules have been obtained, rule-based classifiers are easy to deploy and explain. In this paper, we describe an approach that uses biogeography-based optimization (BBO) to compute rule sets that maximize predictive accuracy. BBO is an evolutionary algorithm inspired by the migration patterns of species between the islands of an archipelago. In our implementation, each species corresponds to a classification rule, each island is occupied by multiple species and corresponds to a classifier, and the fitness of an island is computed as the predictive classification accuracy of the corresponding classifier. The archipelago evolves via mutation, selection, and migration of species between islands. Successful islands have a decreased immigration rate and an increased emigration rate. In general, such islands tend to resist invasion and to colonize less successful islands. This results in an evolving set of habitats that corresponds to a population of classifiers. We demonstrate the effectiveness of our approach by comparing it to several traditional and evolutionary based state-of- the-art classifiers.

## CCS CONCEPTS

•**Computing Methodologies** →Machine Learning →Supervised Learning → Supervised Learning by Classification

## KEYWORDS

Classification, Supervised Learning, Evolutionary Algorithm

## 1 INTRODUCTION

The goal of rule-based classification is to learn a set of classification rules that assigns new data points a class label. Classification rules have the form IF **P** THEN **C**, where **P** is a conjunction of data attribute tests and C is the class label. Several approaches have been developed that genetic algorithms, machine learning, fuzzy logic, or neural networks to extract classification rules for rule-based classification, see [33] for a detailed description. Once suitable rules have been extracted, rule-based classifiers are easy to understand, deploy, and explain. Due to its versatility, rule-based classification is a popular approach for solving real world classification problems, examples include analyzing stock exchange data to decide whether to buy or sell a stock [28], prostate tissue classification for recognizing cancer [9], and credit risk evaluation [39].

Evolutionary algorithms (EAs) are population-based metaheuristic optimization algorithms inspired by the principles of biological evolution. EAs have been widely used to learn classification rules, either through supervised learning or unsupervised learning [5]. In EAs, an initial population of candidate solutions is generated. Subsequently, the initial population evolves through multiple generations until a stopping criterion is met. During each iteration, a new population evolves from the previous population. The quality of each solution is evaluated, well performing solution are used to breed new candidate solutions, and poorly performing solutions are replaced. The success of the algorithm depends on the way that individuals (that is, candidate solutions) share information and collaborate with each other to move through the search space toward the optimum solution. Evolutionary rule-based systems [4] are a type of genetics based machine learning (GBML) that use sets of rules for knowledge representation [25]. In standard GBML, parents are selected randomly and a new population originates via crossover. There are two major rule-based GBML approaches: the Michigan approach and the Pittsburgh approach [7]. In the Michigan approach, each rule is encoded by an entire chromosome. On the other hand, in the Pittsburgh approach a single chromosome encodes an entire classifier that might consist of multiple classification rules.

In this paper, we describe an approach that uses biogeography-based optimization (BBO) to compute rule sets that maximize predictive accuracy. BBO is an evolutionary algorithm inspired by changes in the distribution of biological species over time and space [6]. We propose BBO-based rule miner (BBO-RM) to extract classification rules for a GBML system with Pittsburgh encoding. In BBO-RM, each island hosts multiple species


*GECCO '17*, July 15-19, 2017, Berlin, Germany

ACM ISBN 978-1-4503-4920-8/17/07..$15.00
http://dx.doi.org/10.1145/3071178.3071221

(classification rules) and encodes a classifier. Classifiers with high predictive accuracies correspond to islands with high fitness. Islands with lower fitness accept species from better performing islands, but they have a lower probability of sharing their species with others. In contrast, well performing islands tend to resist immigration and share their species generously. Over time, the average fitness of the population increases via mutation, selection and migration. We demonstrate the effectiveness of our approach by comparing it to several traditional and evolutionary based state-of the-art classifiers using fourteen selected datasets from UCI Machine Learning Repository [22].

The rest of the paper is organized as follows. First, Section 2 summarizes related work. Next, Section 3 describes the BBO framework. Section 4 contains a detailed description of BBO-RM algorithm. Section 5 presents data sets, evaluation metrics and computational experiments. Conclusions and future work are described in Section 6.

## 2 RELATED WORKS

Due to their flexibility, EAs have been used to solve problems in various different application areas, ranging from art and biology to operations research and robotics [52]. Here, we will focus mainly on data mining and classification tasks. Several EAs use the Michigan encoding, for example XCS [13], and XCS with real values as input (XCSR, [5]). These systems act as reinforcement learning agents [2]. Despite the success of these Michigan-based system, Pittsburgh encoding seems to be particularly well-suited for classification and data mining applications [41]. Systems that use Pittsburgh encoding include the Genetic Algorithm Based Concept Learner (GABIL, [11]), Genetic-Based Inductive Learning (GIL, [10]), Genetic clASSIfier sySTem (GAssist, [20]), Ordered incremental genetic algorithm (OIGA, [46]), and Incremental learning with genetic algorithms (ILGA, [47]). Hybrid GBML systems combine both encodings [50-51]. Often EA's use iterative rule learning to construct classification rules. During iterative rule learning the algorithm learns rule by rule sequentially and removes all the examples covered (matched) by a new rule from the training set. Systems that uses iterative rule learning include BioHEL [8], NAX [9], SLAVE [42], and HIDER [43]. In addition, several alternative methods for constructing classification rules have been described, examples include evolutionary programming, genetic programming [44-46]. Beyond EAs that evolve only one population, more complex co-evolutionary classification approaches that evolve multiple competing or cooperating populations in parallel have been explored [12]. A comprehensive review of EA based classifiers can be found in [14, 41].

To the best of our knowledge, this paper is the first application of the evolutionary BBO algorithm to rule-based classification. Bhugra and colleagues describe an application of BBO in combination with association rule mining [26]. In their work, the Apriori algorithm is used to generate frequent itemsets together with the corresponding association rules. Subsequently, BBO is used to enhance the quality of the derived rule set. In contrast, our method uses the (EA) BBO algorithm for classification rule extraction.

## 3 BBO FRAMEWORK

Biogeography based optimization is an example of how a natural process can be modeled to solve optimization problems [6]. The approach has been applied to approximate several benchmark functions [15], and to solve various real world problems including economic load dispatch [16], wireless network power allocation [17], flexible job shop scheduling [18] and many others. The BBO algorithm uses the migration patterns of species between the islands of an archipelago to evolve successful solutions. Each island represents a candidate solution, and contains a number of decision variables denoted as suitability index variables (SIVs). The quality of a solution is measured by the habitat suitability index (HSI) of the island, analogously to the fitness function in EAs. Like other EAs, each candidate solution in BBO probabilistically shares decision variables with other candidate solutions to improve the fitness of candidate solutions. Each island immigrates (receives) decision variables from other islands based on its immigration rate, $\lambda$, and emigrates (donates/shares) decision variables to other islands based on its emigration rate, $\mu$. A successful island has a high emigration rate and a low immigration rate. Conversely, a poor island has a low emigration rate and a high immigration rate. Following [49], the main steps of the BBO algorithm are described below:

1. Initialize population (n islands).
2. Compute island modification probabilities ($\lambda$ and $\mu$).
3a. For each island use the corresponding immigration rate $\lambda$ to probabilistically decide which islands will receive SIVs.
3b. Use emigration rates $\mu$ together with roulette wheel selection to choose SIV donor islands for the islands selected in 3a.
3c. For each selected pair of emigrating/immigrating island, migrate a decision variable (SIV) from the emigrating to the immigrating island.
4. Mutate islands.
5. Update HIS of each island.
6. Go to step 2 until termination criterion is met.

BBO differs from other EAs in important aspects. The notion of 'reproduction' is absent in BBO. Unlike most other EAs where offspring is the compilation of two parental individuals from the previous generation, in BBO a new generation is produced by modifying the current generation via species migration. As a consequence, in contrast to most EAs, the initial population survives forever, it does not 'die' at the end of each generation.

## 4 BBO RULE MINING ALGORITHM

In the following Algorithm 1 we give an overview of our BBO-RM approach, followed by a detailed description of the used encoding mechanism (Section 4.1), our initialization strategy (Section 4.2), migration and mutation operators (Sections 4.3 and 4.4), our fitness function (Section 4.5), and the stopping criteria (Section 4.6). We will refer to chromosomes as islands in the algorithm.

**Algorithm 1 Overview of BBO-RM**

1: Initialize the population.
2: Compute the fitness of each island.
3: **repeat**
4: Save the best (elite) $k$ islands.
5: Calculate immigration rate λ and emigration rate μ based on the fitness of corresponding island.
6: Apply BBO migration operator
7: Apply mutation operator.
8: Compute the fitness of islands.
9: Replace the worst $k$ islands with the elite islands of the previous generation.
10: **until** stopping criteria is fulfilled

## 4.1 Encoding Mechanism

An individual is represented by a set of *ruleN* different rules. The minimum and maximum number of rules a chromosome can have is determined by the values *rMin*, and *rMax*, respectively. The value *ruleN* is sampled from the range [*rMin; rMax*]. A rule is encoded as IF < conditions > THEN < classLabel >, where conditions are conjunctions of a set of conditions. Each condition is encoded as a triple:

$< AttrIndex > < = > < Val_j >$ [for nominal attribute]
$< AttrIndex > < \ Lower\ Bound > < \ Upper\ Bound >$ [for numerical attribute]

AttrIndex is the positional index of the attribute in the attribute-list. Fig. 1 illustrates a rule encoding. Assume that $i$th attribute has been selected. If the attribute is numerical, then $L_i$ and $U_i$ are lower and upper bounds of the associated intervals of the attribute. A data object will satisfy the condition if $< L_i \leq a_i \leq U_i >$, where $a_i$ is the value of attribute $i$ for the data object. In case of nominal attribute, $Val_{ij}$ denotes the $j$th value of the domain of $i$th attribute and the condition checks for equality of $a_i$ and $Val_{ij}$. Rules have variable lengths. The number of conditions (attributes) expressed in a rule is denoted by *attN*. This value is sampled within the range of [*attMin; attMax*]. Here *attMin* and *attMax* denote the minimum and maximum number of attributes that a rule can contain. Next, we choose relevant attributes of a rule by uniformly sampling *attN* values from the attribute lists. The maximum number of rules in a ruleset is bounded by *rMax* value. It should be noted that the parameter *rMax* prevents our system from bloating, a common problem in variable length representation of GA [20].

## 4.2 Initialization Strategy

We followed a mixed initialization approach, as suggested in GIL [10]. For each chromosome, 20% of all rules are instantiated randomly. The remaining rules are instantiated using training examples as seeds. At the beginning of the initialization stage of a rule, we determine the number of expressed attributes in it, *attN*. During random initialization, the upper and lower bounds of each real valued expressed attribute are assigned randomly within the domain size or a random categorical value is chosen.

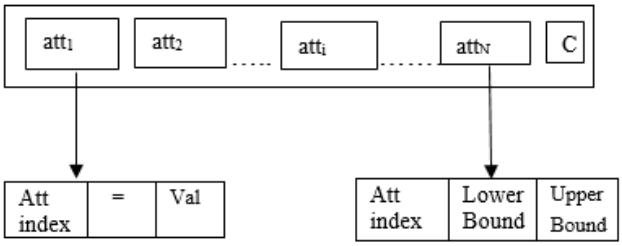


**Figure 1: Structure of a rule, att = attribute. $att_1$ is nominal and $att_N$ is numerical attribute. C is the class label. Rule is interpreted as $att_1 \wedge att_2 \wedge ... \wedge att_N \rightarrow C$**

To determine the class label for such rules, we follow the approach suggested by Freitas [4]. For each class, we calculate how many training instances a newly generated rule covers. The class label that maximizes the fitness of the rule is chosen as the rule consequent. For rules instantiated by training examples, we follow a similar approach as described in [8]. The training examples are chosen randomly to act as 'seed'. The expressed continuous attributes are encoded as centering the selected attribute value [seed-*intervalLength*/2, seed+*intervalLength*/2], where *intervalLength* is randomly initialized with uniform distribution between 25% and 75% of the domain size. Nominal attribute values are exact copy of 'seed'. Class wise sampling without replacement is used to pick up training examples as described in [19]. After initialization, the rules in each chromosome are sorted based on their confidence values.

## 4.3 Immigration and Emigration

The emigration and immigration rates of each solution (island)are used to exchange classification rules between habitats. In our method, each SIV is a rule and a population member (island) is a ruleset. For a given island $y_i$, we apply its immigration rate $\lambda_i$ to each SIV to decide whether it should be modified. If a SIV is selected to be modified, then the donor island $y_j$, is chosen randomly using the emigration rate:

$$\frac{\mu_j}{\sum_k^N \mu_k} \qquad (1)$$

where N indicates the population size (number of islands) [1]. We have used a roulette wheel mechanism to select the donor island. The donor island modifies the selected SIV by replacing it with its own SIV. Our migration procedure (Algorithm 2) is similar to the procedure described in [1]. Note that, chromosomes have variable lengths, i.e. the number of SIVs are different in each island. Therefore, we have adopted an BBO migration operator where the $i$th SIV of the receiving island is replaced by the $i$th SIV of the donor island [1]. In our algorithm, the $i$th rule of the immigrating island is replaced by the $i$th rule of the donor island (similar to [1]). In case the donor island has less than $i$ rules, a

randomly selected rule is chosen. The corresponding $\lambda$ and $\mu$ values depend on the fitness of receiver and donor islands; they are updated in each generation.

**Algorithm 2 Migration in BBO-RM**

```
1: for each candidate solution y_k do
2 : for each SIV i of y_k do
3:     Use λ_k whether to immigrate to y_k
4:     if immigrating then
5:       Use {μ} to probabilistically select
           emigrating island, y_j (equation 1)
6:      if |SIV (y_j)| ≤ i then
7:        y_k(i) ← y_j(i)
8:      else
9:        Randomly select a SIV x from y_j
           y_k(i) ← x
10:     end if
11:   end if
12: end for
13: end for
```

We have used a linear migration model in our implementation. [15, 27]. The $\lambda$ and $\mu$ values are calculated as follows:

$$\mu_i = \frac{N - r_i}{N} \quad (2)$$

$$\lambda_i = 1 - \mu_i \quad (3)$$

Here, $r_i$ is the rank of $i$th individual based on fitness, and $N$ the population size. The fittest individual has rank $r_i = 1$ and the worst individual has $r_i = N$. We have also incorporated elitism in order to retain the best solutions in the population. Immigration and emigration are used to probabilistically modify each non-elite island in the population.

Fig. 2 illustrates the situation before and after migration among three islands. Island 1 is the fittest and island 3 is the least fit island. For simplicity reasons, the rules are indexed as *r<island index><_><rule index>*. Thus *r*1_1 is the 1st rule of island 1, *r*2_1 is the 1st rule of island 2, and so on. Consider the migration of island 3: the island has 6 rules, $|SIV| = 6$. For each rule, $\lambda_3$ is used to decide whether it should be modified or not. If it is to be modified, then the donor island is chosen via roulette wheel selection (Lines 5-6 of Algorithm 2). The donor island replaces the rule as per Lines 7-11 of Algorithm 2. In island 3, the second rule is replaced by the second rule of island 1 (Lines 7-8 of Algorithm 2). The fourth rule is replaced by a randomly selected rule of island 2, in this case *r*2_3. In this example, island 1 has not allowed any invasion, as it is the fittest island. The mutated rules are colored in purple and the migrated rules have the colors corresponding to its original island.

## 4.4 Mutation

The mutation operator is applied to each chromosome. The operator selects one rule and one expressed attribute randomly with uniform probability. If the selected attribute is continuous, the operator selects one interval bound and adds a randomly generated offset to the bound, of size (picked with uniform distribution) between -30% and 30% of the attribute domain. In case this procedure generates an inconsistent value pair, we simply swap the bounds [8]. If the mutation affects categorical

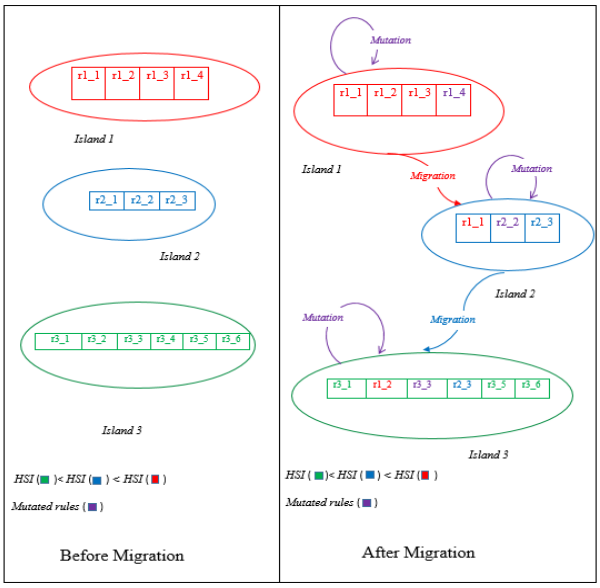


**Figure 2: Three islands before migration. Island 1(red) hosts 4 rules, island 2 (blue) hosts 3, and island 3 (green) hosts 6 rules. The habitat suitability index (HSI) measure the fitness of the individual islands.**

attribute of the rule, a new value is assigned to the attribute, picked at random from the corresponding domain. After mutation, the rules in a chromosome are sorted again according to their confidence value.

## 4.5 Fitness Function and Match Process

To evaluate the performance of a ruleset, each training instance is compared with its rules. If any rule matches the instance, the rules class label indicates the predicted class label of the instance. Then this class label is compared to the actual known class of the data point in order to identify correctly classified instances. The first matching rule determines the class of an instance. Our matching strategy is similar to the Pittsburgh based approach used in GAssist [20]. We have also included a default class which is the majority class of the dataset. If no rule matches an instance, then the default class is assigned to the data point. Note that, the default class is not encoded in the ruleset. The purpose of introducing a default class is to ensure that every instance is assigned to a class label. The fitness of a chromosome is simply the ratio of correctly classified instances to the total number of training examples.

# 5 EXPERIMENTAL RESULTS

In the following, we describe the datasets, algorithms, and the experimental setup that was used for our evaluation.

## 5.1 Dataset Properties

We tested BBO-RM and other classifiers on 14 datasets from the UCI Machine Learning Repository [22]. In particular, we selected these datasets for their structural variety; each represents a different challenge for classifiers. Our test datasets have the following characteristics: all continuous attributes (wine, sonar), all categorical attributes (vote, breast-cancer), mixed categorical and continuous attributes (adult, australian), small instances (zoo), large instances (german, adult), small number of attributes (iris), large number of attributes (sonar), small number of classes (heart disease-statlog, breast cancer-Wisconsin, pima), large number of classes (zoo). The properties of dataset are listed in Table 1.

## 5.2 Classification Algorithms

To assess the performance of our algorithm we compare BBO-RM with eight other algorithms. As BBO-RM is a rule based classifier, we focus on comparisons with rule based classifiers. Among eight algorithms two are evolutionary based rule mining algorithms (GAssist, BioHEL), non-evolutionary rule miners (PART, RIPPER, Decision Table), and three well performing state-of-the art approaches (C4.5, Support Vector Machine, Random Forest). A brief description of the algorithms is given below:

- *C4.5 (J48):* Developed by Ross Quinlan, the C4.5 algorithm uses a decision tree for classification [32].
- *PART:* the algorithm for repeatedly generates partial decision trees and infers classification rules from these trees [21].
- *RIPPER:* the algorithm is an improvement of the incremental reduced error pruning algorithm (IREP) [38].
- *GAssist:* the algorithm evolves a population of individuals via a standard genetic algorithm. Each individual represents a complete and variable-length ruleset [20].
- *BioHEL:* the algorithm is designed to handle large-scale bioinformatic datasets. Its main structure is inherited from GAssist [8].
- *Decision Table (DTable):* the algorithm computes numeric predictions from decision trees and generates an ordered set of If-Then rules [35].
- *Support Vector Machine (SVM):* the algorithm employs linear hyperplanes to infer decision boundaries among classes [30].
- *Random Forest (RF):* the algorithm uses an ensemble of randomly constructed independent decision trees [34].

## 5.3 Parameters of BBO-RM

Parameters of BBO-RM were tuned for optimum performance. We have used a population size = 50, elitism = 10, maximum number of generations = 100, and stagnation limit = 20. The other parameters are: mutation rate = 0.6, minimum and maximum number of attributes in a rule, *attMin* = 1 and *attMax* = 7, minimum and maximum number of rules in a ruleset *rMin* = 10, and *rMax* = 20. Datasets containing attributes less than seven uses attMax = all attributes (iris), *attMax* = 4, and *rMin* = 5 (thyroid gland) dataset.

**Table 1: Characteristics of Datasets**

| id | Dataset | #Ins. | MV (%) | #R | #N | #C |
|---|---|---|---|---|---|---|
| adl | adult | 48842 | 7.4 | 8 | 6 | 2 |
| aus | australian | 690 | 0.6 | 6 | 9 | 2 |
| bre | breast cancer | 286 | 0.3 | 0 | 10 | 2 |
| wis | breast-wisconsin | 699 | 0.3 | 9 | 0 | 2 |
| ger | german | 1000 | 0.0 | 7 | 13 | 2 |
| h-s | heart statlog | 270 | 0.0 | 13 | 0 | 2 |
| hep | hepatitis | 155 | 5.6 | 6 | 13 | 2 |
| irs | iris | 150 | 0.0 | 4 | 0 | 3 |
| pim | pima-indians | 768 | 0.0 | 8 | 0 | 2 |
| son | sonar | 208 | 0.0 | 60 | 0 | 2 |
| thy | thyroid-gland | 215 | 0.0 | 5 | 0 | 3 |
| vot | vote | 435 | 5.6 | 0 | 6 | 2 |
| win | wine | 178 | 0.0 | 13 | 0 | 3 |
| zoo | zoo | 101 | 0.0 | 1 | 15 | 7 |

# Ins: Total number of instances; MV: Missing values; #R = Number of numerical attributes, #N = Number of Nominal attributes, #C = Number of classes.

## 5.4 Experimental Set Up and Implementations

In the case of data with missing values, we have removed the instances before partitioning to the approach described in [12]. We also used Weka's feature selection using information gain (IG) [36, 37]. To improve accuracy, we have removed attributes of zero IG values. Subsequently, stratified tenfold cross validation was used to partition our datasets into training and test datasets. The process was repeated twice; all the accuracies reported later in the paper are the average of the accuracies obtained in the 2×10 test sets. The results of GAssist and BioHEL have been obtained by using the implementations provided by the authors [23, 24]. The default parameters have been used in GAssist, except for the number of iteration is set to 1000 (instead of 500) as suggested by the author. BioHEL configuration of the code is set according to the best setting as reported by the author in [8]. The other algorithms are obtained from Weka version 3.8.1. The default parameter values were used for all algorithm in Weka with 10-fold cross validation with two repetitions. BBO-RM is programmed in R script. To speed up the performance, we have used the RCpp package [40], which enables C++ code embedded in R. The experiments were performed on a Linux machine with Intel(R) Xeon(R) CPU processor running at 2.00 GHz with RedHat Linux (RHEL) 7 installed.

**Table 2: Accuracy results of BBO-RM with other algorithms.**

| Data | Algorithms | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **BBO-RM** | **PART** | **GAssist** | **BioHEL** | **RIPPER** | **DTable** | **J48** | **RF** | **SVM** |
| adl | 83.29 | 84.24 | 84.8 | 82.45 | 83.88 | 85.09 | **85.25** | 83.19 | 85.06 |
| aus | 86.08 | 83.70 | 86.96 | 81.00 | 85.22 | 84.71 | 85.65 | **87.39** | 85.51 |
| bre | **76.88** | 69.78 | 71.45 | 67.28 | 72.04 | 74.17 | 75.38 | 69.65 | 70.12 |
| wis | 95.82 | 93.77 | 94.13 | 94.91 | 93.49 | 92.35 | 94.28 | 95.07 | **95.85** |
| ger | 72.80 | 71.35 | 70.8 | 70.5 | 72.35 | 73.00 | 71.85 | 75.40 | **76.05** |
| h-s | 83.35 | 76.48 | 76.30 | 78.33 | 79.07 | 83.15 | 80.00 | 82.78 | **85.00** |
| hep | 91.25 | 85.00 | 85.63 | 83.13 | 83.75 | 84.38 | 85.00 | **92.50** | 84.38 |
| irs | **96.00** | 94.64 | 95.31 | 93.98 | 95.64 | 92.64 | 94.67 | 94.64 | 95.64 |
| pim | 75.40 | 74.78 | 73.66 | 72.73 | 74.78 | 74.33 | 74.71 | 76.41 | **77.25** |
| son | 76.69 | 75.25 | 74.76 | 70.97 | 75.27 | 70.71 | 79.85 | **80.13** | 77.71 |
| thy | 95.09 | 94.64 | 91.10 | 92.47 | 93.47 | 92.07 | 92.75 | **95.12** | 89.56 |
| vot | **97.62** | 96.32 | 96.30 | 94.20 | 96.33 | 95.69 | 96.56 | 96.34 | 96.77 |
| win | 94.70 | 92.06 | 91.76 | 89.03 | 92.40 | 88.17 | 93.76 | **97.43** | 98.30 |
| zoo | 94.55 | 93.14 | 93.09 | 90.59 | 88.73 | 87.73 | 92.64 | 93.09 | **96.05** |
| Avg. accuracy | **87.11** | 84.65 | 84.72 | 83.04 | 84.74 | 84.16 | 85.88 | 87.08 | 86.66 |
| Avg. rank | **2.64** | 5.82 | 5.89 | 7.93 | 5.50 | 6.53 | 4.17 | 3.42 | 3.07 |

*Boldface entries indicate the best value of the corresponding row.*

## 5.5 Results and Analysis

Table 2 compares the results of BBO-RM with the above algorithms. In addition, we have analyzed the results of BBORM, and other algorithms using Friedman test [29] for multiple comparison using BBO-RM as control algorithm. The average ranks and average accuracy values of each algorithm are shown. SVM and random forest each performs best in five datasets. BBO-RM outperforms other algorithms three datasets, J48 performs the best in one dataset. BBO-RM has the lowest average rank and also the highest average accuracy for these 14 datasets. Random forest has lower average rank than SVM but its average accuracy is higher. To test if the observed differences in average ranks correspond to a significant difference in classifier performance, we computed the Friedman statistic 38.8797 and compared it with the associated critical values for Chi square distribution for significance levels $\alpha$ =0.05 (15.507) and $\alpha$ = 0.10 (13.362). Since the Friedman statistic is greater than the associated critical values, there are significant differences among the analyzed classifiers, and additional post-hoc analysis is needed. We have used Bonferroni-Dunn's test [31] for the control algorithm BBO-RM. According to [29], the performance of classifiers is significantly different if the corresponding average ranks differ by at least the critical difference, CD (defined in section 3.2.2 of [29]). For our comparison values CD = 2.82 and CD = 2.585 for $\alpha$ = 0.05 and $\alpha$ =0.10 respectively in the two measures considered. Fig. 3 shows a graphical representation of the Bonferroni-Dunn's test with BBO-RM as control. Each algorithm is represented by a bar whose height is proportional to its average ranking. If we choose BBO-RM as control algorithm, and sum its height with the critical difference obtained by Bonferroni-Dunn (CD value), the result gives us a threshold to decide which algorithms perform significantly worse than our control. The thresholds for $\alpha$ = 0.05 and $\alpha$ = 0.1 are shown as horizontal lines in Fig. 3. The test does not reveal a significant difference between

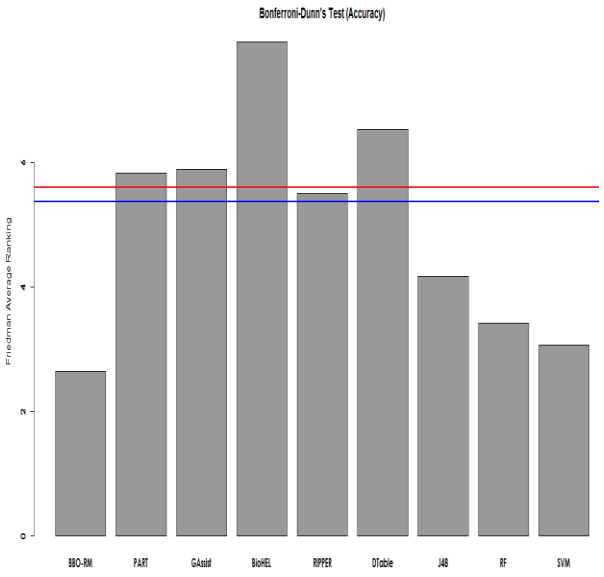


**Figure 3: Bonferroni-Dunn graphic for classification accuracy.**

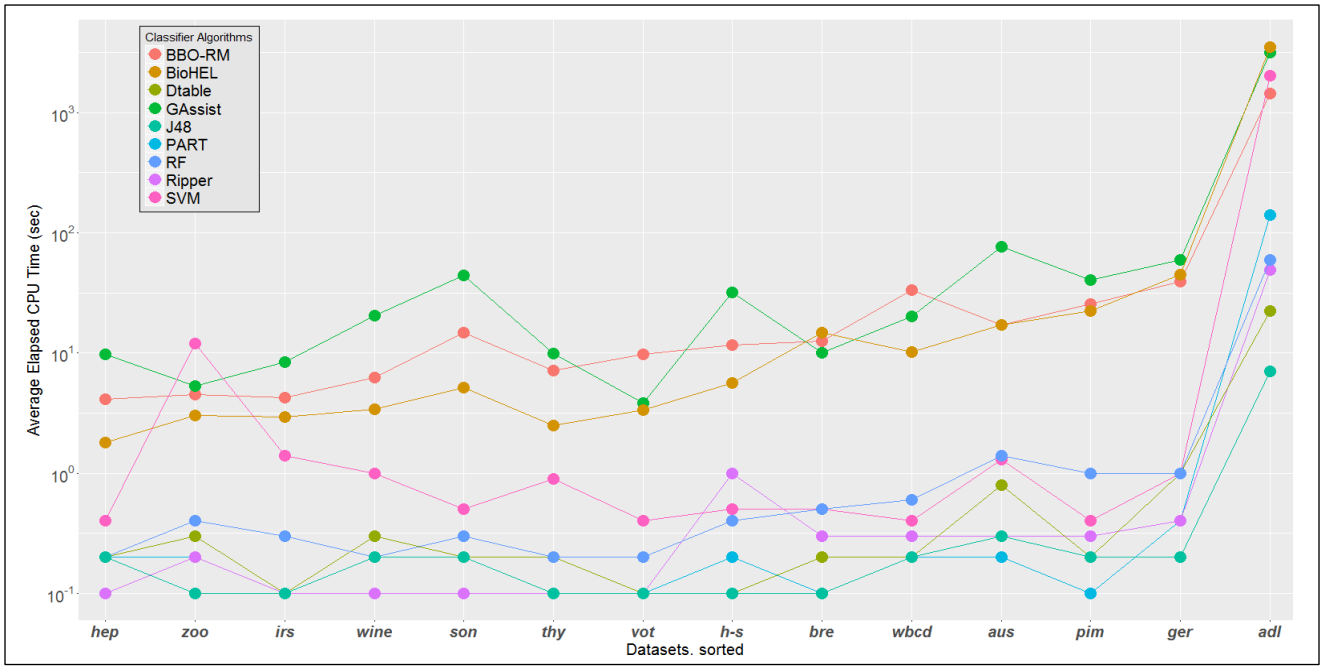


**Figure 4: CPU time for 14 datasets**

BBO-RM and J48, SVM and RF. However, remarkably, BBO-RM is significantly better than all rule based algorithms used in this experiment. BBO-RM beats PART, GAssist, BioHEL and Decision Table at both significance levels ($\alpha$= 0.05 and $\alpha$ = 0.10) and RIPPER at the significance level $\alpha$ = 0.10.

## 5.6 Running Time Analysis

To illustrate the runtime behavior of our approach, Fig. 4 displays the CPU execution time versus dataset. The datasets are sorted by their size, and time is plotted on a log scale (base 10). The time on Y axis is the average execution time of training and testing one cross validation fold. The size of the dataset reported is the size after removing instances of missing values. It can be observed that the elapsed time is not directly proportional to the size of the dataset, the number of attributes also influences the execution time. Evolutionary algorithms are naturally slower than non-evolutionary counterparts.

# 6 CONCLUSIONS AND FUTURE WORK

In this paper, we describe BBO-RM, an approach that uses an evolutionary BBO algorithm to generate rule sets for rule-based classification. The performance of BBO-RM on benchmark datasets demonstrates its suitability for solving practical classification problems. In our evaluation, BBO-RM has outperformed, or matched within 2%, eight competing approaches in eleven of fourteen test datasets. BBO-RM has the lowest average rank as well as the highest average accuracy. Using Bonferroni-Dunn test, we were able to show that these results reflect a statistical significant difference in classifier performance between BBO-RM and all tested rule-based approaches. This makes our algorithm a promising new classification approach, in particular in situations where rule-based classification is required. We hypothesize that the good performance of our algorithm results from BBO-RM's unique way of optimizing rules. In our current implementation, BBO-RM uses Pittsburgh-style learning classification system (LCS) encoding, a representation that appears to be well-suited for classification and data mining applications [41]. Pitt-style returns the *best set of rules*, and not the *set of best rules* [4]. In the future, we plan to investigate the performance of the alternative Michigan chromosome encoding in combination with iterative rule learning (IL) [8]. IL creates one rule at a time. After a new rule is obtained, the training examples that are covered by this rule are removed from the training set thus the EA is forced to explore other areas of the search space to learn additional rules. In each iteration, the best rule is inserted into the ruleset. Another possible drawback of EAs, that we have encountered in few occasions, is the convergence of the entire population towards similar individuals [4]. To promote diversity in the solution, we plan to use classification rules of the $k$ best rulesets to assemble a more diverse classifier. Another possible research direction is to explore the effect of different migration models, see [15] for a list of possible alternatives.

# REFERENCES

**[1]** H. Ma, D. Simon, M. Fei, X. Shu, and Z. Chen, 2014. Hybrid biogeography-based evolutionary algorithms, *Engineering Applications of Artificial Intelligence*, 30 (2014), 213-224.

[2] R. S. Sutton and A. G. Barto, 1998. *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA.
[3] G. H. John, and P. Langley, 1995. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 338-345.
[4] A. A. Freitas, 2002. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.
[5] S. W. Wilson, 2000. Get Real! XCS with Continuous-Valued Inputs. In *Learning Classifier Systems, From Foundations to Applications*, London, UK, 2000, 209-222.
[6] D. Simon, 2008. Biogeography-based optimization, *IEEE Transactions on Evolutionary Computation*, 12, 6, 702-713.
[7] Z. Michalewicz, 1996. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*, Springer-Verlag London, UK.
[8] J. Bacardit, E. K. Burke, and N. Krasnogor, 2009. Improving the scalability of rule-based evolutionary learning, *Memetic Computing*, 1,1, (2009), 55-67.
[9] X. Llorà, A. Priya, and R. Bhargava, 2009. Observerinvariant histopathology using genetics-based machine learning, *Natural Computing*, 8,1, (2009), 101-120.
[10] C. Janikow, 1991. *Inductive learning of decision rules in attribute-based examples: a knowledge-intensive genetic algorithm approach.* PhD dissertation. University of North Carolina at Chapel Hill.
[11] K. A. D. Jong, and W. M. Spears, 1991. Learning Concept Classification Rules Using Genetic Algorithms. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, 651-656.
[12] K. C. Tan, Q. Yu, and J. H. Ang, 2006. A coevolutionary algorithm for rules discovery in data mining, *Int'l Journal of Systems Science*, 37,1, (2006), 835-886.
[13] S. W. Wilson, 1995. Classifier Fitness Based on Accuracy, *Evol. Comput.*, 3,2, (1995), 149-175.
[14] A. Fernandez, S. Garcia, J. Luengo, E. Bernado Mansilla, and F. Herrera, 2010. Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy, and Comparative Study, *IEEE Trans. Evolut. Comput.*, 14, (2010), 913-941.
[15] H. Ma, 2014. An analysis of the equilibrium of migration models for biogeography based optimization, *Inf. Sci*, 180,18, (2010), 3444-3464.
[16] A. Bhattacharya, and P. Chattopadhyay, 2010. Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch, *IEEE Trans. Power Syst.*, 25, 4, (2010), 1955-1964.
[17] I. Boussad, A. Chatterjee, P. Siarry, and M. A. Nacer, 2011. Two-stage update biogeography based optimization using differential evolution algorithm (DBBO), *Comput. Oper. Res.*, 38, 8, (2011), 1188-1198.
[18] S. Rahmati, and M. Zandieh, 2012. A new biogeography based optimization (BBO) algorithm for the flexible job shop scheduling problem., *Int. J. Adv. Manuf. Technol.*, 58, 9, (2012), 1115-1129.
[19] J. Bacardit, 2005. Analysis of the initialization stage of a Pittsburgh approach learning classifier system. In *GECCO 2005: Proceedings of the Genetic and Evolutionary Computation Conference*, 2005, 1843-1850.
[20] J. Bacardit, 2004. *Pittsburgh genetics-based machine learning in the data mining era: Representations, generalization, and run-time.* PhD dissertation, Ramon Lull University, Barcelona, Spain.
[21] E. Frank, and I. H. Witten, 1998. Generating Accurate Rule Sets Without Global Optimization. In Proceedings of the Fifteenth International Conference on Machine Learning, 144-151.
[22] C. L. Blake, E. Keogh and C. J. Merz, 1998. UCI repository of machine learning databases., www.ics.uci.edu/mlearn/MLRepository.html.
[23] Bioinformatics-oriented Hierarchical Evolutionary Learning, 2009. http://ico2s.org/software/biohel.html.
[24] GAssist Genetic Classifier System 2004. http://ico2s.org/software/gassist.html.
[25] J. J. Grefenstette, 1993. Genetic Algorithms and Machine Learning. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, 3-4.
[26] D. Bhugra, S. Goel, and V. Singhania, 2013. Association rule analysis using biogeography based optimization. In *Computer Communication and Informatics (ICCCI)*, 1-5.
[27] G. Khademi, H. Mohammadi, and D. Simon, 2016. *Hybrid Invasive Weed / BiogeographyBased Optimization*, Submitted for publication, http://embeddedlab.csuohio.edu/BBO/IWO.html.
[28] Y. W. C. Chien, and Y. L. Chen, 2010. Mining associative classification rules with stock trading data A GA-based method, *Knowledge-Based Systems*, 23,6, (2010), 605-614.
[29] J. Dem˙sar, 2006. Statistical Comparisons of Classifiers over Multiple Data Sets, J. Mach. Learn. Res., 7 ,2006, 1-30.
[30] C. N. Shawe- Taylor J, 2000. *Support Vector Machines and Other Kernel-based Methods.* Cambridge, UK: Cambridge University Press.
[31] O. J. Dunn, 1961. Multiple comparisons among means, *Journal of the American Statistical Association*, 56 (1961), 52-64.
[32] J. R. Quinlan, 1993. *C4.5: Programs for Machine Learning*, Morgan Kaufman Publishers Inc., San Fransisco, CA, USA.
[33] W. Duch, N. Jankowski, K. Grabczewski, and R. Adamczak, 2000. Optimization and Interpretation of Rule based Classifiers. In *Proceedings of the IIS'2000 Symposium on Intelligent Information Systems*, 2000, 1-13.
[34] L. Breiman,2001. Random forests, *Machine Learning*, vol. 45, 5–32.
[35] D.L. Fisher, 1966. Data, Documentation and Decision Tables. *Comm ACM Vol.* 9 No. 1 (Jan. 1966), 26–31.
[36] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*
*Information Gain*, ser. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann.
[37] M. Hall, E. Frank, G. Holmes, B. P. Fahringer, P. Reutemann, and I. H. Witten, 2009. The weka data mining software: An update, *SIGKDD Explorations*, 11, 130–133.
[38] W. W. Cohen, 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, 115-123.Morgan Kaufmann.
[39] D. Martens, B. Baesens, T.V. Gestel and J. Vanthienen, 2007. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183 (2007), 1466–1476.
[40] Rcpp: Seamless R and C++ Integration: Version 0.12.9, 2017, https://cran.r-project.org/web/packages/Rcpp/index.html.
[41] R. J. Urbanowicz and J. H.Moore, 2009, Learning Classifier Systems: A Complete Introduction, Review, and Roadmap, *Journal of Artificial Evolution and Applications*, (Jan 2009), 1-25.
[42] A. González, R. Perez, 2001. Selection of relevant features in a fuzzy genetic learning algorithm, *IEEE Transactions on Systems and Man and Cybernetics and Part B: Cybernetics* 31 (2001) 417–425.
[43] J.S. Aguilar-Ruiz, R. Giraldez, and J.C. Riquelme, 2007. Natural encoding for evolutionary supervised learning, *IEEE Transactions on Evolutionary Computation*, 11 (2007) 466–479.
[44] W.J. Choi, and T.S. Choi, 2012. Genetic programming-based feature transform and classification for the automatic detection of pulmonary nodules on computed tomography images, *Information Sciences* 212 (2012) 57–78.
[45] P. Espejo, S. Ventura, and F. Herrera, 2010. A survey on the application of genetic programming to classification, *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 40 (2010) 121–144.
[46] A. Zafra, and S. Ventura, 2010. G3P-MI: a genetic programming algorithm for multiple instance learning, *Information Sciences* 180 (2010) 4496–4513.
[47] F. Zhu and S. U. Guan, 2004, Ordered incremental training with genetic algorithms, *Int. J. Intell. Syst.*, 19, 12, (2004) 1239–1256.
[48] S. U. Guan and F. Zhu, 2005, An incremental approach to genetic algorithms-based classification, *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 35, 2, (Apr. 2005) 227–239.
[49] D. Du, D. Simon and M. Ergezer, 2009, Biogeography-based optimization combined with evolutionary strategy and immigration refusal, *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, 2009, pp. 997-1002.
[50] S. B. Mehta, S. Chaudhury, A. Bhattacharyya, nad A. Jena, 2011, Tissue Classification in Magnetic Resonance Images Through the Hybrid Approach of Michigan and Pittsburg Genetic Algorithm, *Appl. Soft Comput.*, 11, 4, (June, 2011), 3476-3484.
[51] H. Ishibuchi, T. Nakashima, and T. Murata, 2001, Three-objective genetics-based machine learning for linguistic rule extraction, *Information Sciences*, 136, 1–4, (August 2001), 109-133.
[52] C. C. A. Coello, and G. B. Lamont, 2004, *Applications of multi-objective evolutionary algorithms.* Vol. 1. World Scientific.