A parallel Genetic Programming algorithm for classification

Alberto Cano, Amelia Zafra, and Sebastián Ventura

Department of Computing and Numerical Analysis, University of Córdoba
14071 Córdoba, Spain
{i52caroa,azafra,sventura}@uco.es

Abstract. In this paper a Grammar Guided Genetic Programming-based method for the learning of rule-based classification systems is proposed. The method learns disjunctive normal form rules generated by means of a context-free grammar. The individual constitutes a rule based decision list that represents the full classifier. To overcome the problem of computational time of this system, it parallelizes the evaluation phase reducing significantly the computation time. Moreover, different operator genetics are designed to maintain the diversity of the population and get a compact set of rules. The results obtained have been validated by the use of non-parametric statistical tests, showing a good performance in terms of accuracy and interpretability.

Keywords: Genetic Programming, Classification

1 Introduction

The general idea of discovering knowledge in large amounts of data is both appealing and intuitive, but technically it is significantly challenging and difficult, especially in the fields where really huge amounts of relational data have been collected over last decades. In this paper we focus on classification which is a well-known task in data mining.

The classification task has been overcomed with numerous computer techniques (rule learning, instance based learning, neural networks, support vector machines, statistical classifiers and so on). These include crisp rule learning [10, 6], decision trees [15], evolutionary algorithms [3, 4, 13, 20] and specifically Genetic Programming (GP) algorithms [7, 11, 18, 19]. One of the best advantage of rule-based system is that they provide interpretable solutions to the user. When considering a rule-based learning system, the different genetic learning methods follow two approaches in order to encode rules within a population of individuals. The first one represents an individual as a rule set, this proposal is known as Pittsburgh approach [17]. The second one represents an individual as a single rule, and the whole rule set is provided by combining several individuals in a population (rule cooperation) or via different evolutionary runs (rule competition). In turn, within the individual as a single rule approach, there are three

generic proposals: Michigan, Iterative Rule Learning and Genetic Cooperative-Competitive Learning).

The main advantage of the Pittsburgh model compared to other approaches is that it allows to address the cooperation-competition problem, dealing the interaction among rules in the evolutionary process. However, its main problem is controlling the number of rules of the individuals, as the total number of rules in the population can grow quite, increasing the computational cost and becoming unmanageable problems. On the other hand, the other approches provide good results but are inefficient methods and have addressed the cooperation-competition problem by not dealing the interaction between rules in the evolutionary process.

In this paper we propose a Grammar Guided GP (G3P) based algorithm that learns disjunctive normal form (DNF) rules generated by means of a context-free grammar, coded as one rule base decision list [16] per individual (Pittsburgh approach). This provides easily interpretable and understandable rules, and it also considers the interaction among the rules in the evolutionary process. The genetic operators are designed to work on two levels. On the one hand, it allows the optimization of particular rules by combining the rules to obtain the bests disjunction and conjunction of attribute-value comparisons. On the other hand, it address the problem of cooperation-competition, considering the interaction that occurs among the rules when these are combined to form the final classifiers.

Evolving full classifiers allows us to evaluate the relationship among rules but introduces greater complexity and computation time. To solve this problem and reduce the computation time, the evaluation phase for fitness computation is parallelized using the GPU or multiple CPU threads. Furthermore, the problem of controlling the number of rules is solved by setting a parameter with the maximum number of rules per class. This parameter allows the user to decide to obtain simpler or more complex classificators by limiting the number of rules.

An experimental study involving 18 datasets and 11 well-known classification algorithms shows that the algorithm obtains accurate and comprehensible rules. Non-parametric statistical methods have been used to compare and analyze the accuracy of the experimental results. They show the good performance in terms of accuracy of our approach compared to other traditional methods analyzed. Moreover, the suitability of some components such as the use of specific genetic operators and the use of full classifier take advantages with respect to the rest of GP and G3P-based methods considered in the study.

The remainder of this paper is organized as follows. Section 2 presents the proposed GP classification algorithm and discusses the genetic operators, the fitness function and the generational model. Section 3 describes the experimental study. In Section 4, the results will be announced and finally the last section presents the final remarks of our investigation and outlines future research work.

2 Algorithm

This section details the algorithm proposed. It describes the encoding of the individual's genotype, the genetic operators, the fitness function, the initialization criterion and the generational model.

2.1 Individual representation

One of the main advantages of GP is the flexibility to represent the solutions. Thus, GP can be employed to construct classifiers using different kinds of representations, e.g. decision trees, classification rules, discriminant functions, and many more [9]. In our case, the individuals in the Pittsburgh GP algorithm are classifiers where each individual is a set of classification rules generated by means of a context-free grammar and whose expression tree is composed by terminal and non-terminal nodes. A classifier can be expressed as a set of IF-antecedent-THEN-consequent rules in a decision rules list. The antecedent of the rule represents a disjunction and conjunction of attribute-value comparisons and the rule consequent specifies the class to be predicted for an instance that satisfies all the conditions of the rule antecedent. The terminals set consists of the attribute names and attribute values of the dataset being mined, logical operators (AND, OR, NOT), relational operators (<, =, <>, >) and the interval range operator (IN). We must ensure that the classifier contains at least one rule for each class.

2.2 Genetic operators

This subsection describes the genetic crossover and mutation operators which modify the genotype of the individuals throughout the evolutionary process.

Crossover operator

As mentioned, an individual represents a classifier as a set of rules. Taking advantage of this representation, the crossover operator is designed to optimize both the rules and the interaction among them in the classifier. So on the one hand, the crossover applied on specific rules operates on two rules from the individual and produces two new rules. Two random compatible nodes are selected from within each rule and then the resultant sub-trees are swapped, generating two child rules. These crossed rules become the decision list of a new individual. On the other hand, the crossover applied on classifers acts over the individuals swapping their rules. Given two parent individuals, two crossing points are chosen (one by parent) so the rules are swapped from those points, building two new individuals different from their parents. Therefore, selected crossing points will ensure that at least one rule of a classifier will cross with the classification rules from the other, i.e., it is not allowed to swap all the rules of a classifier.

Mutation operator

The mutation operator can also be applied to the rules and the individuals. The rules mutation operates either on a function node or a terminal node. It randomly selects a node in a sub-tree and replaces it with a new randomly created sub-tree. The mutation of an individual is determined by the random elimination of a rule of the rule set with a probability degree.

2.3 Fitness function

The algorithm has two fitness functions. The first one evaluates the rules of each individual independently. The second one evaluates the individuals checking the success rates of the classifiers over the training set. It is necessary to evaluate the rules of each individual first and then evaluate the classifiers success rates.

Fitness function applied on particular rules

The fitness function we use on particular rules is the proposed by Bojarczuk et al. [11]. Specifically, each rule is evaluated over each instance and each class. This obtains the results of the quality of the predictions of the rules for each class. Thus, the consequent of a rule is reassigned to the class that has produced better results. The rules fitness function combines two indicators that are commonplace in the domain, namely the sensitivity (Se) and the specifity (Sp), which employ the true positive (t_p), false positive (f_p), true negative (t_n) and false negative (f_n) values from the match of the class of the instance and the predicted class.

$$Se = \frac{t_p}{t_p + f_n} \qquad Sp = \frac{t_n}{t_n + f_p} \qquad ruleFitness = Se * Sp \tag{1}$$

Fitness function applied on rule set (the classifier)

The classifiers fitness function is performed once the best consequent for each rule is calculated. The fitness of a classifier is simply the success rate of the classifier over the training set. Each instance is submitted to the decision list to find the first rule that covers the instance. If the consequent of the rule matches the class of the instance it is a hit, otherwise it is a fail.

The activation process of the rules defines in which order the rules are evaluated to determine which rule classifies each instance. The activation method employed is a decision list. Thus, an instance is classified by the first rule of the classifier that covers the instance and whose order is defined in the individual's genotype. If no rule covers the instance, it is classified using the default class (most frequent class or defined by the user).

Parallel evaluation phase

Many studies have proved the evaluation phase is by far the most expensive [5] since it requires evaluating each rule over each instance. As mentioned, a disadvantage of Pittsburgh approaches is their high computational complexity. In fact, if you analyze our proposal, the total number of rules is the addition of the number of rules for each individual. This number can be large and at least the total number of rules is the product of the number of classes and the population size. An evolutionary system that preforms crossover, mutation, evaluation and selection on so many rules is really expensive.

To solve this problem, numerous studies have parallelized the evaluation phase of these algorithms. Among the references there are two main ways to perform the parallelization. One is the use of different threads and the second one in more recent works is the use of GPUs [14]. The most efficient approaches conclude they can speed up the evaluation phase more than 800 times [5]. Therefore, we will take advantage of its parallelization capability to solve the high computational cost problem of the evaluation phase, specifically in Pittsburgh approaches, by using the GPU.

Our system divides the evaluation phase into two functions: rules fitness function and individuals fitness function. The rules evaluation can be performed completely parallel using GPUs, but also the evaluation of individuals can be parallelized. Each individual can be tested as a rule over the training set independently. Therefore, our model will take advantage of both parallel fitness functions to reduce the computational cost. The full description of the parallel model would exceed the scope and the number of pages of this paper but it is detailed in [5].

2.4 Initialization

The initialization of the algorithm with a genetically diverse population is crucial for a successful outcome. The problem of the fitness function we use to evaluate the rules is that a minority class can ignored. Therefore, we must ensure that individuals contains at least one rule for each class. One way to do this, once the individuals are created, is to complete with new rules of the classes that have not yet been covered by at least one rule. As it might be very expensive to get at least one rule for each and every one of the classes, the process of completing the individual is performed up to a number of times depending on the number of classes. This way, we try most of the classes to be represented in the classifier. Finally, among all the rules of all individuals we decide to keep the best rule for each class that will help us later to fill the classifiers with the rules of the classes that could be missing.

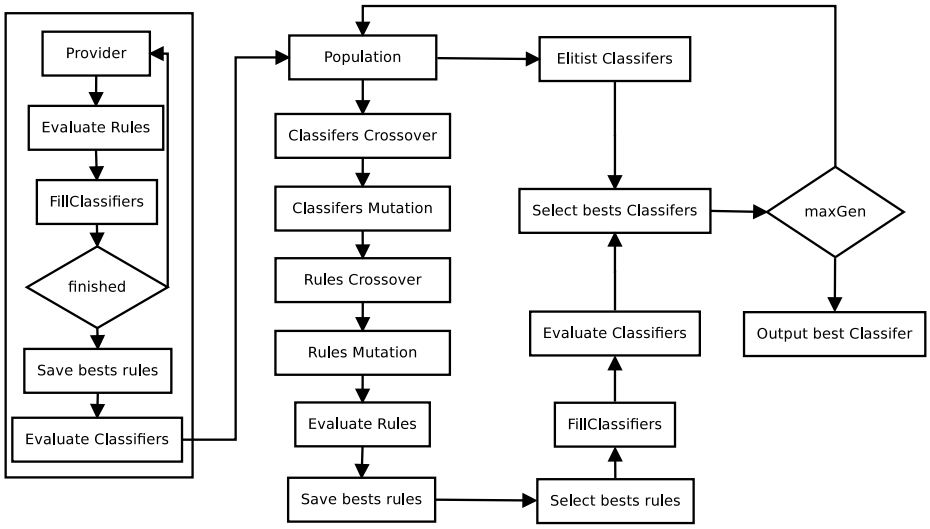

Fig. 1. Algorithm flowchart.

2.5 Generational model

In this section we describe the generational model represented in Fig. 1. The left box represents the initialization of individuals described before. Once the population is initialized, the algorithm iteratively proceeds to apply genetic operators crossover and mutation described in the section 2.2. Specifically, the algorithm performs the individuals crossover swapping subsets of rules. The individuals can mutate eliminating some of its rules. The rules within each individual are crossed together and mutated, obtaining new classification rules. These new rules must be evaluated to get their consequents and fitness values.

To ensure the survival of the best rules, the algorithm checks in each generation and for each class if any new rule is better than the one stored for that class, if so the rule is replaced by the new one. As the crossover operator may have created individuals that exceed the maximum number of rules allowed, we can simplify by selecting the best rules subset.

The individual must be completed by rules from uncovered classes, taking the best rules from the pool to cover them. Once completed, each individual must be evaluated to get its fitness using the training data. We employ elitism to keep a subset of the best individuals in each generation to ensure the survival of the fittest individuals. The algorithm finishes when it has found an individual that correctly classifies all the training instances or when the algorithm has iterated a certain number of generations.

3 Experimental study

This section describes the details of the experiments, discuss the application domains, the algorithms used and the settings of the tests.

The experiments performed compare the results of 11 different classification algorithms using 18 datasets. These algorithms are available on the JCLEC [21] website and KEEL [2] website. The datasets employed have been selected from the KEEL repository website[1]. These datasets are very varied considering different degrees of complexity, number of classes, number of features and number of instances. Thus, the number of classes ranges up to 10, the number of features ranges from 4 to 60 and the number of instances ranges from 101 to 58000.

To properly evaluate the performance of the algorithm proposed it is considered to make a comparative study with some evolutionary crisp rule learning algorithms for classification (De Falco et al. [7], Bojarczuk et al. [11], Tan et al. [18, 19], MPLCS [3], ILGA [13], CORE [20] and UCS [4]), two rule learning algorithms widely extended (PART [10] y RIPPER [6]) and a classic decision tree algorithm (C4.5 [15]). For each algorithm, the values of the parameters to be configured by the user were set to the default values provided by the authors. The population size and the number of generations for our proposal are both set to 200, i.e. the algorithm deals with 200 candidate full classifiers, and the maximum number of rules per class is set to 3. The results are validated using 10-fold cross validation and 10 runs with different seeds for stochastic methods. The results provided are the average of the 100 executions. A CPU time comparison analysis would exceed the number of pages of this paper and the execution time of the proposal is definitely higher than the other algorithms.

The experiments were executed in an Intel i7 quadcore machine with 12 GB DDR3-1600 and two NVIDIA GTX 480 GPUs, which are 3 billion transistors GPUs with 480 cores and 1.5 GB GDDR5. The operating system was Ubuntu Linux 10.10 64 bits, the GPU computing software was NVIDIA CUDA 3.1 and the compiler GCC 4.4.4.

4 Results

In this section we provide the results of the experiments and discuss the performance of the algorithms over the datasets. Table 1 shows the average results of the predictive accuracy obtained running each algorithm ten times in each of the datasets tests folds. The last but one row shows the average accuracy over all the datasets.

Analyzing the results of the table notice that the algorithm proposed obtains better average accuracy results and ranking. Its accuracy is higher than most algorithms and is very close and slightly higher than the algorithms MPLCS, UCS and C4.5. In those datasets in which our algorithm does not achieve the best results, its accuracy is usually quite competitive. Moreover, the algorithm does not stand out negatively in any dataset.

In order to analyze the results and discover the existence of significant differences between the algorithms, a series of statistical tests [8, 12] were carried

Table 1. Experiments results: predictive accuracy (%)

Dataset	Proposal	Falco	Bojarczuk	Tan [18]	Tan [19]	MPLCS	ILGA	CORE	UCS	C4.5	PART	RIPPER
Zoo	95.90%	61.73%	86.31%	93.43%	92.75%	95.50%	84.67%	94.58%	**96.50%**	92.80%	86.13%	93.41%
Iris	95.82%	76.20%	84.40%	89.67%	75.34%	96.00%	93.33%	94.67%	92.00%	**96.67%**	33.33%	94.00%
Hepatitis	**88.51%**	68.85%	74.44%	73.50%	83.43%	85.15%	77.92%	83.43%	80.74%	85.66%	84.17%	79.09%
Wine	94.58%	63.72%	79.33%	75.81%	66.24%	91.54%	89.28%	**95.49%**	91.57%	94.90%	63.98%	93.79%
Sonar	75.09%	61.90%	66.31%	60.57%	59.05%	76.81%	71.57%	53.38%	**77.83%**	70.54%	60.53%	72.45%
Glass	**69.39%**	31.79%	39.98%	36.88%	48.85%	65.88%	52.40%	50.97%	66.07%	68.86%	46.50%	63.47%
New-thyroid	**94.40%**	69.18%	75.54%	73.65%	78.66%	91.65%	90.71%	91.21%	93.55%	93.52%	77.22%	93.05%
Heart	81.85%	66.78%	74.37%	72.55%	76.30%	**83.70%**	64.44%	71.11%	80.37%	78.14%	57.77%	76.29%
Dermatology	94.96%	45.61%	73.59%	77.29%	76.56%	95.53%	60.81%	43.86%	**96.36%**	94.35%	73.12%	94.42%
Haberman	**73.82%**	65.92%	68.43%	52.70%	70.18%	72.17%	71.89%	70.88%	72.18%	71.19%	73.53%	46.72%
Ecoli	78.92%	51.72%	57.63%	50.48%	65.19%	**80.67%**	63.10%	65.78%	79.49%	77.37%	44.67%	72.63%
Australian	85.41%	72.42%	84.29%	75.77%	78.41%	**86.96%**	85.07%	83.33%	86.09%	85.21%	60.72%	81.44%
Pima	75.01%	69.82%	67.59%	62.98%	66.68%	74.60%	73.19%	72.28%	**76.04%**	72.53%	65.11%	69.53%
Vehicle	70.59%	31.11%	41.67%	36.04%	41.52%	71.15%	57.24%	40.05%	**72.45%**	66.66%	37.85%	70.44%
Contraceptive	**55.61%**	40.75%	42.68%	40.37%	44.00%	55.47%	43.59%	45.01%	49.49%	51.19%	42.90%	50.78%
Thyroid	99.22%	68.05%	51.39%	52.92%	92.43%	94.72%	94.10%	68.00%	96.99%	**99.56%**	92.58%	99.37%
Penbased	83.32%	25.54%	40.29%	35.76%	44.90%	91.80%	53.25%	15.69%	14.28%	94.89%	15.86%	**96.15%**
Shuttle	**99.97%**	61.16%	75.51%	63.55%	89.51%	99.60%	93.67%	91.60%	99.62%	99.96%	99.60%	99.96%
Average (%)	**84.02%**	57.34%	65.76%	62.44%	69.44%	83.82%	73.34%	68.40%	78.98%	83.00%	61.97%	80.38%
Ranking	**2.22**	10.56	8.72	9.72	8.08	3.03	7.00	7.25	3.56	3.58	9.19	5.14

out. We use the Iman and Davenport test to rank the k algorithms over the N datasets. The average rank according to the F-distribution throughout all the datasets is shown in the last row of the table. Notice that the best global position, the lowest ranking value, corresponds to the obtained by our proposal.

The Iman and Davenport statistic for average accuracy distributed according to F-distribution with $k-1=11$ and $(k-1)(N-1)=187$ degrees of fredom is 30.1460. This value does not belong to the critical interval $[0, F_{0.01,11,187} = 2.3439]$ for $p=0.01$. Thus, we reject the null-hypothesis that all the algorithms perform equally well. In order to analyse if there are significant differences among the algorithms, we use the Bonferroni-Dunn test to reveal the difference in performance using a critical difference (CD) value for $p=0.01$ equal to 3.9865.

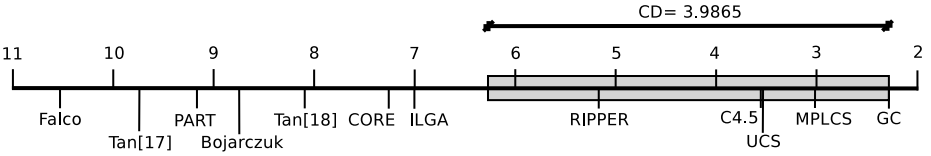

Fig. 2. Bonferroni-Dunn test. The noncritical range is shown shaded.

The results indicate that a significance level of $p=0.01$ (i.e., with a probability of 99%), there are significant differences between our algorithm and De Falco et al., Bojarczuk et al., Tan et al. [18, 19], PART, IGLA and CORE algorithms being our algorithm statistically better. These differences are shown in Fig. 2. The figure represents the algorithm's ranking and the critical distance interval. Regarding to the other algorithms, the test does not indicate significant differences. However, our proposal obtains the lowest ranking value, indicating that considering all datasets, it obtains better results in a greater number of them than other proposals.

5 Conclusions

In this paper we have proposed a G3P algorithm that optimizes the class prediction accuracy of a decision list as a set of interpretable DNF rules. The encoding of the individuals have allowed us to design specific crossover and mutation operators that consider the interaction among the rules in the evolutionary process. The algorithm overcomes the high computational complexity of Pittsburgh approaches by the parallelization of the evaluation phase. Nevertheless, in spite of this improvement, the complexity of the algorithm is still high. The experiments carried out compare the results of our proposal with other classic algorithms for solving classification problems considering 11 algorithms and 18 datasets. A statistical study on results validates the model showing that it provides the most accurate classifiers since obtaining the lowest ranking in the general comparative.

The algorithm design is complex and the execution time is high even using GPUs for the evaluation of the individuals. The individual representation is as a set of classification rules and it is necessary to evaluate both rules and classifiers every generation. Therefore, the algorithm performs slow regarding to the other algorithms from the experimental. Nevertheless, the idea of this work was to experiment a highly complex G3P Pittsburgh algorithm for classification accelerated using GPUs, since GPUs have been successfully used in Michigan approaches to speed up the evaluation phase [5]. Therefore, there is no a sequential implementation of the algorithm to compare with, since it would require excessive CPU time.

Currently, databases are moving toward sparse data, unbalanced, multiple labels, instances. It would be interesting to check the performance of the algorithm in these contexts and to improve the efficiency of the algorithm to perform faster.

Additional information of the paper such as the algorithm's code, the experimental setup and the datasets are published and available on the website: `http://www.uco.es/grupos/kdis/kdiswiki/HAIS11`

Acknowledgments This work has been financed in part by the TIN2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (CICYT), the P08-TIC-3720 project of the Andalusian Science and Technology Department and FEDER funds.

References

1. Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17, 255–287 (2011)
2. Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M., Ventura, S., Garrell, J., Otero, J., Romero, C., Bacardit, J., Rivas, V., Fernández, J., Herrera, F.: KEEL:

A Software Tool to Assess Evolutionary Algorithms for Data Mining Problems. Soft Computing - A Fusion of Foundations, Methodologies and Applications 13, 307–318 (2009)
3. Bacardit, J., Krasnogor, N.: Performance and efficiency of memetic pittsburgh learning classifier systems. Evolutionary Computation 17(3), 307–342 (2009)
4. Bernadó-Mansilla, E., Garrell, J.: Accuracy-based learning classifier systems: Models and analysis and applications to classification tasks. Evolutionary Computation 11(3), 209–238 (2003)
5. Cano, A., Zafra, A., Ventura, S.: Solving classification problems using genetic programming algorithms on GPUs. In: Hybrid Artificial Intelligence Systems. Lecture Notes in Computer Science, vol. 6077, pp. 17–26. Springer (2010)
6. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning. pp. 115–123. Morgan Kaufmann (1995)
7. De Falco, I., Della Cioppa, A., Tarantino, E.: Discovering interesting classification rules with genetic programming. Applied Soft Comput. 1(4), 257–269 (2001)
8. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
9. Espejo, P.G., Ventura, S., Herrera, F.: A Survey on the Application of Genetic Programming to Classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C 40(2), 121–144 (March 2010)
10. Frank, E., Witten, I.: Generating accurate rule sets without global optimization. In: Proceedings of the 15th International Conference on Machine Learning. pp. 144–151 (1998)
11. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2002)
12. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. Journal of Heuristics 15, 617–644 (December 2009)
13. Guan, S., Zhu, F.: An incremental approach to genetic-algorithms-based classification. IEEE Transactions on Systems and Man and and Cybernetics and Part B 35(2), 227–239 (2005)
14. Harding, S.: Genetic programming on graphics processing units bibliography, `http://www.gpgpgpu.com/`
15. Quinlan, J.: C4.5: Programs for Machine Learning (1993)
16. Rivest, R.L.: Learning decision lists. Mach. Learn. 2, 229–246 (1987)
17. Smith, S.F.: A Learning System Based on Genetic Adaptive Algorithms. Phd thesis, University of Pittsburgh (1980)
18. Tan, K.C., Tay, A., Lee, T.H., Heng, C.M.: Mining multiple comprehensible classification rules using genetic programming. In: Proceedings of the Evolutionary Computation CEC '02. pp. 1302–1307. IEEE Computer Society, Washington, DC, USA (2002)
19. Tan, K.C., Yu, Q., Heng, C.M., Lee, T.H.: Evolutionary computing for knowledge discovery in medical diagnosis. Artificial Intelligence in Medicine 27(2), 129–154 (2003)
20. Tan, K., Yu, Q., Ang, J.: A coevolutionary algorithm for rules discovery in data mining. International Journal of Systems Science 37(12), 835–864 (2006)
21. Ventura, S., Romero, C., Zafra, A., Delgado, J.A., Hervás, C.: JCLEC: a Java framework for evolutionary computation. Soft Comput. 12, 381–392 (October 2007)