GADBMS - Restricted Genetic Algorithm to induce a set of rules from Relational Databases

Andréa de Fátima Cavalheiro
Federal University of Paraná, Computer Science Department,
Curitiba, Brazil, 81531-990
andrea.cavalheiro@gvt.net.br

and

Aurora Ramirez Pozo
Federal University of Paraná, Computer Science Department,
Curitiba, Brazil, 81531-990
aurora@inf.ufpr.br

Abstract

The present work introduces the GADBMS, a system that uses a special restricted genetic algorithm to induce a set of rules from relational databases. The choice of GA paradigm is partially justified by its great capacity in dealing with noise, invalid or inexact data, and its easy adaptation to different domains of data. The GA algorithm uses Tabu lists to restrict the selection process. This restriction allows the creation of a set of potential rules for the classifier tool. This tool was tested in four datasets and compared to other twenty-three rules based algorithms. After that, noise was added to the databases and a new set of experiments was performed. The results prove that the proposed algorithm is efficient and robust. And the strategy used to maintain the diversity was considered valid, since the algorithm was able to keep its accuracy in categorization even for smaller populations.

Keywords: Genetic Algorithms, Tabu Search, Classification Task and Knowledge Discovery in Databases.

1 Introduction

Data mining deals with the discovery of hidden knowledge and unexpected patterns in large databases. Data Mining can be considered to be an inter-disciplinary field involving concepts from machine learning, database technology, statistics and mathematics among others. While the main concern of database technologists are to find efficient ways of storing, retrieving and manipulating data, the main concern of the machine learning is to develop techniques for learning knowledge from data. In data mining the classification task is a very important method that use approaches based on machine learning among which decision trees, neural networks, rule induction and genetic algorithms (GA).

Genetic algorithms are suitable for complex problems, but some improvement must be done in the basic algorithm. Many approaches have been proposed. Some studies aims at improving the performance of GA searching capabilities using hybridization [1, 2, 3]. In hybridization, GA is used together with one of the following paradigms: simulated annealing, tabu search, artificial neural networks and expert systems.

This paper presents a classifier induction tool based on genetic algorithms restricted by Tabu lists, which allows the generation of a set of potential rules. Some reasons to use genetic algorithms include their scalability, their ability to handle noisy data, configuration of parameters according to the task, and the possibility of parallel processing [4].

This strategy was proposed in [4] for multimodal and multiobjective function optimization and used in [5] for data mining tasks, where it uses two lists of restrictions: long-term and short-term Tabu lists. We extended the work done by Lopes and Pozo [5] using the same strategy proposed by them, but with x long-term and short-term Tabu lists, where x corresponds to the number of classes in the datasets. The rules for the different classes are available in separated Tabu lists. In our approach, each individual in the population is treated as a SQL command making possible the evaluation of the rules directly in the DBMS.

This paper is organized as follows. Section 2 begins with a brief overview of data mining and genetic algorithms. Section 3 shows the implemented algorithm, and the results for some tests are discussed in section 4. Section 5 concludes this paper.

2 Data Mining and Genetic Algorithms

Data mining is commonly used defined as the search for useful patterns in data. It consists in the application of data analysis and discovery algorithms to produce patterns or models from the data. There are a large variety of data mining approaches [6, 7, 8, 9] with different searching methods aiming to discover different kinds of knowledge. The classification task using rule learning technique is one of these methods. Rules are commonly used in expressing knowledge and are easily understood by human. Rules are also commonly used in expert systems for decision making. Rule learning is the process of inducing rules from a set of training examples. The problem of data mining can be formulated as conducting a search for novel, useful, and interesting knowledge. The search can be accomplished by several techniques, for instance, using genetic algorithms.

Genetic algorithm is a machine learning technique that uses the metaphor of natural selection to simulate the evolution of individual structures via processes of selection and reproduction. These processes depend on the perceived performance (fitness) of the individual structures as defined by an environment. Each individual in the population is evaluated, receiving a measure of its fitness in the environment. Selection focuses attention on high-fitness individuals, exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for exploration. Although simplistic from a biologist‘s viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms. For more information about genetic algorithms and its operators, a good source is [10].

A genetic classifier consists of a population of classifying elements that compete to make predictions. Elements that do not present a good performance are discarded while the ones that make good predictions proliferate producing new offspring. During the training the force of the elements are subjected to changes to maintain the predictions correct and/or to punish mistakes. All population goes through an apprenticeship phase where the weakest elements die and new elements are created. During this process random mutation happens, characterized through changes in the elements, and crossovers in which two elements cross to form a new element.

Work on genetic classifier algorithm has traditionally been grouped into one of two general approaches. The Pittsburg approach [11] uses a traditional genetic algorithm in which each entity in the population is a set of rules representing a complete solution to the learning problem. The Michigan approach [12] has generally used a distinctly different evolutionary mechanism in which the population consists of individual rules, each of which represents a partial solution to the overall learning task. That makes it necessary to develop some strategy to extract a group of non-redundant rules from the GA population [12, 13]. Different research methods were developed to overcome this problem.

The first group of studies [14, 15, 16] focuses on the maintenance of the diversity in the population and include (1) sharing methods, that use sharing functions to avoid the convergence to similar individuals; (2) crowding methods, which constrain the replacements of new individuals; and (3) crossover restrictions. However, these methods are difficult to apply to practical problems: (1) Unsuitable sharing functions often prevent individuals from exploiting optimal regions; (2) the crowding often failed to avoid the early convergence; (3) the method seems too artificial.

The second group aims at improving the performance of GA searching capabilities using hybridization [1, 2, 3]. In this approach, GA is used aside one of the following paradigms: simulated annealing, tabu search, artificial neural networks and expert systems. However, most of the studies in the literature have focused on the global search via GA, while the local search has been made using one of the previously referred techniques.

The last group of studies focuses on either function optimization problems, or problems to find Pareto optimal solutions [17, 18, 19]. These studies include: (1) methods to divide individuals into subgroups, where each subgroup corresponds to one objective function, (2) methods to rank Pareto optimal individuals not to be covered by other individuals, (3) combination of tournament and sharing methods, and (4) methods to divide Pareto solutions to some ranges. These previous studies have the common goal of improving GA by using methods to maintain diversity in the population.

In this paper, a classifier induction tool based on genetic algorithms restricted by Tabu lists is presented. This strategy was proposed in [4] for multimodal and multiobjective problems and used in [5] for data mining tasks. The implemented algorithm use *x* long-term and *x* short-term Tabu lists, where *x* corresponds to the number of classes in the database, which differs from the work done by Lopes and Pozo [5], that use two lists of restrictions, one long-term and one short-term Tabu lists. In [5], it is necessary to execute *n* times the program to find all the possible rules (where *n* is the number of classes), in our approach, the rules are stored in *n* separated Tabu lists, making it possible to discover the rules in just one execution.

Tabu Search is an optimization method that uses a form of short-term memory to prevent the search to become trapped in local minima. A Tabu list that keeps track of recent solutions is formed. At each iteration, in the optimization process, solutions are checked against the Tabu list. A solution that is on the list will not be chosen for the next iteration (unless it overrules its Tabu condition by what is called an aspiration condition). The Tabu list forms the core of Tabu search and keeps the process from cycling in one neighborhood of the solution space.

At each iteration, a steepest-descent solution that does not violate the Tabu condition is chosen. If no non-Tabu improving solution exists, the best non-improving solution is taken. The combination of memory and gradient descent allows for diversification and intensification of the search. Local minima in the search space are avoided while good areas are well explored. In the next section, we present how these concepts are used in GADBMS.

3 Overview of the System Implemented

This work proposes the GADBMS, a classifier induction tool based on genetic algorithms restricted by Tabu lists. The tool was implemented in C++ and we use Oracle database. Furthermore, each individual in the population is treated as a SQL command; in this way it becomes possible to evaluate the rules directly in the DBMS. The basic architecture can be seen in the Fig. 1.:

To accomplish its goal, the GADBMS was implemented in three main modules: Configuration Module, GATabu and Extracted Rules. The following explains how the modules work.

3.1 Configuration Module

The main parameters that need to be provided to GADBMS are: a file that contains all the attributes; classes and valid values of the database; crossover, mutation and unrestricted attribute rates; population; long-list and short-list size; number of generations and runs; and similarity threshold that can be in phenotypic or genotypic space. After providing these parameters, the most important part of the process is the searching of rules, which will be explained in the next paragraph.

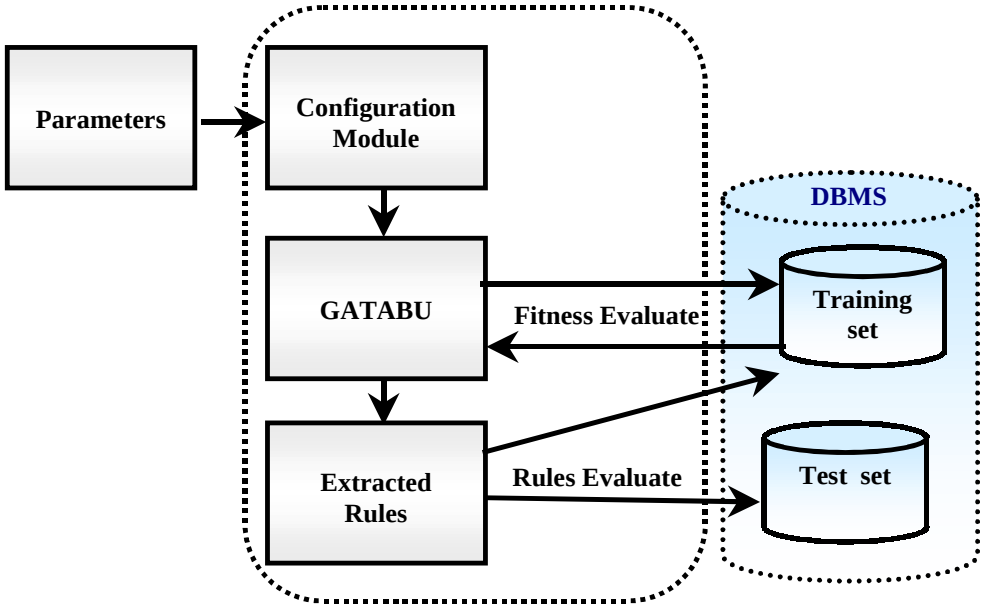

Fig. 1. The GADBMS

3.2 GATabu

The Classifier Tool uses the genetic algorithm restricted by Tabu lists to evolve a population of rules. After the execution, the rules stored in the long list will be used to create a classifier.

3.2.1 Initial Population

Based on the domains features and GA parameters, we create the initial population. When the individuals are created, an approximated number of unrestricted attributes are used. The attributes that are unrestricted in the individual are randomly chosen, which avoids the generation of invalid rules (rules that do not cover any example) in the starting population. The proportion of unrestricted attributes is determined by the unrestricted attribute rate.

3.2.2 Fitness Function

After creating the population it is necessary to calculate the fitness of each individual in the population. The fitness function for this problem must be able to qualify the rules as partial classifiers, so the accuracy of a rule is more important than its ability to cover all training instances. In our GADBMS we used de Laplace Function [20], which is described in equation 1:

$$\text{Fitness Function} = (VP + 1)/(VP+FP+K). \qquad (1)$$

Where K is the number of classes in the domain, VP are positive examples correctly classified by the rule and FP is negative examples incorrectly classified as positive. The fitness is calculated directly in the training set using the same individual as a SQL command. After this, a genetic algorithm restricted by Tabu lists is applied in the population that was created randomly.

3.2.3 Genetic Algorithm Restricted by Tabu Lists

The algorithm works with *x* long list with size *m* and *x* short list with size *n*, where *x* will be the amount of classes in the dataset, *m* and *n* will be the number of maximum individuals stored in the lists. In Fig. 2 we can see a basic idea of the Tabu GA used to classify the best rules of the population.

The long-list with size *m* will only store the individuals that cannot have identical or similar phenotype/genotype. The best individuals of all the previous generations will remain stored in this list. The short-list with size *n*, will only store these individuals of the most recent iterations. When the list is completely filled, a new individual will replace the oldest one. In this list, the individuals can have the same phenotype or genotype.

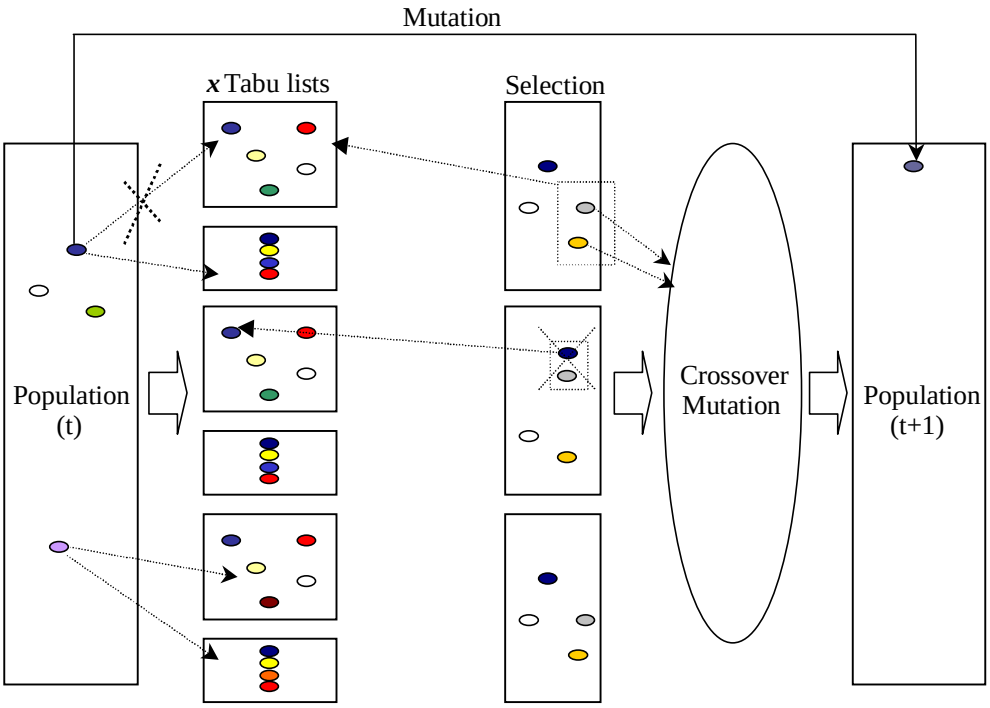

Fig. 2. The TABU – GA

In each generation we verify if the best individual of the population is stored in the correspondent Tabu lists, according to the class that individual belongs to. The individual will only be stored in the long-list if a similar individual does not exist in it. If an individual with a similar phenotype or genotype needs to be added to the long-list, it will only be added if it has a function value higher than the other individual. This individual will be removed from the list, suffer mutation and be put back in the population to participate in the next generations.

The individuals are chosen by tournament selection method and the list of restrictions can be applied to only one parent. When selecting parents candidate, we refer to the Tabu lists in order not to select individual with similar genotypes or phenotypes.

The algorithm used crossover in two-points, which generates two offsprings and mutation. When mutation is applied on a rule, it randomly chooses one of its attributes and changes it value. When applied on discrete attributes, mutation consists on choosing another value from some positive training instance. For continuous attributes, two values *Min* and *Max* are sampled from different positive examples, and a restriction in the form $Min <= A <= Max$ is created, where *A* is the attribute and $Min <= Max$.

3.24 Similarity

In our classifier the similarity metric can be based on either phenotype or genotype space. The phenotype similarity measure corresponds to the amount of similar attributes in the rule. If two attributes of different rules are identical we added 1.0 and if two attributes of different rules are similar we added 0.5. Adding all the values of each attribute allows as to compare them to the similarity threshold previously defined.

The genotype similarity is measured based on the positive training instances correctly classified by the rules. We computed the similarity on the following way: the individuals covered by the rule that is being evaluated are separated in a group, resulting in an amount x. For each rule that composes the long list it is executed on that group and results in an amount y of covered individuals. With these two values (x and y), we calculated the percentile of individuals covered by the rule that is being evaluated and we compared the result with the similarity threshold previously defined.

3.3 Extracted Rules

After having finished the searching process, in this module the rules are organized in an appropriate way, seeking to obtain a good precision for the classifier.

A set of rules formed by the individuals of the long list will be evaluated for the classifier. In this stage all the rules belonging the long list will be analyzed and only the rules that classify at least one individual will be used. Finally, the precision of the classifier will be calculated using the test sets.

In this approach we used ordered rules set, all rules are assembled in a particular order. To classify a new example, each rule is tried in order until some rule that covers the example is found. In that case, the predicted class is given by the activated rule. If none of the rules cover the instance, a default class is used.

Ordered rules set have the property of being logical, in the sense that clashes between rules cannot occur as only one rule can be used at a time. The fitness of the rules is used to define their order.

4 Experiments and Results

In order to allow comparisons, we evaluated four datasets that were used by [5], which were compared with the methodology used by [21]. In this methodology, 33 classification algorithms were compared in different datasets in terms of classification accuracy and many other characteristics. In the following we choose twenty-three of these algorithms based on trees and rules. Table 1 describes the datasets used in our experiments.

Table 1. Analyzed data, where *N* is a numerical attribute, *C* is a categorical attribute and the symbol + represents a database with noise.

Name	Size	Class	Attr (*N*)	Attr(*C*)	Description
Bld	345	2	6		Predict whether or not a male patient has a liver disorder based on blood tests and alcohol consumption.
Bld+	345	2	15		
Pid	532	2	7		Predicted female diabetes patient.
Pid+	532	2	15		
Smo	1855	3	3	5	Predicted attitude toward restrictions on smoking in the workplace.
Smo+	1855	3	10	5	
Vot	435	2		16	Classify a Congressman as Democrat or Republican.
Vot+	435	2		30	

For each database, one variation has been created by the addition of noise, that is, more attributes with random values. We used two methods to estimate the error rate:

- For large databases (more than 1000 examples) we used the test set to estimate de error rate.
- For smaller databases we used 10-fold cross-validation: The database is randomly divided in 10 disjoint subsets, each containing approximately the same proportion of records from each class. For each subset, a classifier is constructed using the records that are not in it. The classifier is then tested on the withheld subset to obtain a cross-validation estimation of its error rate. The error rate is the average of all cross-validation estimates.

For all the simulations we used 200 generations. However, in most of the executions, the generations did not pass 50. In this algorithm the number of generations is stopped automatically at 50 attempts of doing crossover, if the long list already contain a similar individual.

We used population size and long-term Tabu list with length 200 and short-term Tabu list with 20. The crossover rate, mutation rate and similarity threshold were adjusted according to the database that were evaluated. Initial tests were made to find better parameters to each database.

4.1 Population Size

One previous experiment was performed to verify the effects of the population size on the accuracy of the system. An important point that should be mentioned is that for all the evaluated sets we executed the GADBMS on a population of 200 individuals and the classifier obtained almost the same precision as when we executed on a population of 2000 individuals. In Table 2 it is possible to see the results obtained in two of these datasets.

Table 2. Effects of the population size on the accuracy of the system

Database	Smo		Pid	
Population Size	2000	200	2000	200
Rules founded	3	3	4	4
Error Rate	0,3050	0,3050	0,3076	0,2993
Time (hh:mm.ss)	00:09:17	00:01:56	00:16:25	00:02:25

4.2 Similarity Space

The second experiment was performed to verify the effects of the similarity space and is summarized in Table 3. Working with genotypic space we could see that the execution time is very fast and the amount of extracted rules is less than when we used the similarity measure in phenotypic space. However, in some of the analyzed bases, the type of similarity applied was not significant to arrive to the best precision of the classifier.

Table 3. Similarity applied on Smo dataset.

	Genotypic	Phenotypic
Rules founded	3	10
Error Rate	0,3050	0,3050
Distance	4	4
Time (hh:mm.ss)	00:01:56	00:16:58

4.3 Error Rates Comparisons

Another important measure is the standard error, given by $(p\ (1- p)/n)^{½}$, where *p* is the error estimate for the database and n is the number of training examples. The result of a classifier is considered close to the best if the difference between its result and the best for the database is not greater than the standard error [21]. GADBMS was close to the best classifier in two databases.

The GADBMS has been tested on the databases, obtained a mean error rate of 0,2553. In Table 4 we can see the errors rates averages between the implemented algorithm and the results obtained by [5, 21]. Compared to the mean rates of the 23 algorithms presented by [5, 21], this result is not significantly different (at the 10% level) from the best of the other 23 classifiers, because does not differ more than 0.058 [22].

Table 4. Comparison of the results between 23 algorithms based on trees and rules and the implemented algorithm.

	LIM; LOH; SHIH Results Report in [21]			LOPES Results Report in [5]	GADBMS
Dataset	Min.	Max.	Median	Mean	Mean
Bld	0,2790	0,4320	0,3220	0,3775	0,3862
Bld+	0,3130	0,4410	0,3490	0,3475	0,4219
Pid	0,2210	0,3100	0,2380	0,2593	0,2576
Pid+	0,2210	0,3180	0,2500	0,2778	0,2705
Smo	0,3040	0,4240	0,3050	0,3008	0,3050
Smo+	0,3050	0,4110	0,3050	0,3068	0,3050
Vot	0,0364	0,0580	0,0457	0,0582	0,0503
Vot+	0,0412	0,0662	0,0469	0,0611	0,0455
Mean	0,2151	0,3075	0,2327	0,2486	0,2553

During the experiments, the accuracy did not have significant variations between runs when the same genetic parameters were used in the same database. The addition of noise attributes does not appear to increase the error rates significantly. The usage of equipment with different hardware configurations made it impossible to have a reliable comparison between processing times in the database for the experiments. It is important to note that statistical algorithms are in general more efficient but the classification using rules presents models that are more meaningful for the final user. This was one of the reasons that encouraged our study.

5 Conclusion

In this work, we have introduced a classifier induction tool based on genetic algorithms restricted by Tabu lists. In our approach the possible rules can be found in just one run and stored in separate lists, which can be a good way to work with different classes in a single database.

The system displayed good efficiency working with smaller populations reducing the processing time and producing the same results as when using larger populations. In our experiments very few parameters combinations were tried, which means the accuracy in many databases can be significantly improved if better parameters are found.

The use of SQL queries is very interesting, mainly due to the fact of being possible to evaluate the rules directly in the database. Once the population size is quite small, the processing time did not seem to be significant. This can be a great advantage when used on large amounts of data.

It is suggested for future works, to execute the tool in larger databases, analyzing the execution time and the rules found and use a new similarity measure, which can be applied regardless the types of attributes being evaluated.

Acknowledgments

The first author is grateful for the financial support provided by GVT (Global Village Telecom) to present this work.

References

[1] Costa, D.: An Evolutionary Tabu Search Algorithm and the NHL Scheduling Problem. Information Systems and Operational Research, 33 (3): (1995) 161-178.

[2] Glover, F.: Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). Discrete Applied Mathematics, (1994), 49:231-255

[3] Glover, F., Kelly, J., Laguna, M. Genetic Algorithms and Tabu Search: Hybrid for Optimization. Computers and Operations Research, (1995), 22 (1):111-134.

[4] Kurahashi, S., Terano, T.: A Genetic Algorithm with Tabu Search for Multimodal and Multiobjective Function Optimization. Proc. of the Second International Conference on Genetic Algorithms, (2000), 291-298

[5] Lopes, F. M., Pozo, A. R.: Genetic Algorithm Restricted by Tabu lists in Data Mining. Proc of the XXI International Conf. of the Chilean Computer Science Society, (2001), 178-185

[6] Ramakrishnan, N., Grama, A.Y.: Data Mining: From Serendipity to Science. IEEE Computer, (1999), 32(4):34-37.

[7] Ganti, V., Gehrke, J., Ramakrishman, R.: Mining Very Large Databases. IEEE Computer, (1999), 32(4):38-45

[8] Han, J., Lakshmanan, V. S., Ng, T.: Constraint-Based, Multidimensional Data Mining. IEEE Computer, (1999), 32(4):46-50

[9] Hellerstein, J. M., Avnur, R., Chou, A., Hidber, C., Raman, V., Roth, T., Hass, P. J.: Interactive Data Analysis: The Control Project. IEEE Computer, (1999)., 32(4):51-59.

[10] Mitchell, M.: An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press, (1997), p.207

[11] Smith, S. F.: A Learning System Based on Genetic Adaptive Algorithms. PhD thesis, University of Pittsburg, (1980)

[12] Holland, J. H.: Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel rule-Based Systems. Machine Learning: an AI approach. Los altos: Morgan Kaufmann, (1986), 2:593-623

[13] Wilson, S.: Classifier Systems and the Animate Problem. Machine Learning 2, Kluwer Acad, (1987), 199-228

[14] Eshelman, L., Shaffer, J.: Preventing Premature Convergence in Genetic Algorithm by Preventing Incest. Proc. of the Fourth ICGA, (1991), 115-122

[15] De Jong, K., Spears, W.: Using genetic Algorithms to Solve NP-Complete Problems. Proc. 3rd ICGA, (1989)

[16] Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Alabama: Addison Wesley, (1989), 413.

[17] Cantu-Paz, E.: Topologies, Migration Rates, and Multi-Population Parallel Genetic Algorithms. Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99), (1999), 1:91-98.

[18] Coelho, A.: An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends. Evolutionary Computation, 1-11, (1999)

[19] Schaffer, D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proc. of the First International Conference on Genetic Algorithms, (1985). 93-100

[20] Niblett, T.: Constructing Decision Trees in Noisy Domains. In Proceedings European Working Session on Learning, (1987), 2:67-78

[21] Lim, T. S., Loh, W. Y., Shih, Y. S.: A comparison of prediction accuracy, complexity and training time of 33 old and new classification algorithms. Machine Learning Journal, Boston, (1999)

[22] Ruppert, G., Miller, Jr.: Simultaneous Statistical Inference. Springer-Verlag, New York, 2nd edition, 1981.