RULE EXTRACTION BASED ON ROUGH SET THEORY COMBINED WITH GENETIC PROGRAMMING AND ITS APPLICATION TO MEDICAL DATA ANALYSIS

Yasser Hassan, Eiichiro Tazaki
Department of Control and Systems Engineering,
Toin University of Yokohama,
Japan.

Abstract

A methodology for using Rough Set for preference modeling in decision problem is presented in this paper; where we will introduce a new approach for deriving knowledge rules from medical database based on Rough Set combined with Genetic Programming. Genetic Programming belongs to the most newly techniques in applications of Artificial Intelligence. Rough Set Theory, which emerged about twenty years ago, is nowadays rapidly developing branch of Artificial Intelligence and soft computing. At the first glance the two methodologies we talk about have not in common. Rough Set constructs representation of knowledge in terms of attributes, semantic decision rules, etc. On the contradictory, Genetic Programming attempts to automatically create computer programs from a high-level statement of the problem requirements. But, in spite of these differences, it is interesting to try to incorporate both approaches into combined system. The challenge is to get as much as possible from this association.

1. Introduction

Knowledge discovery in database (KDD), often called data mining, aims at the discovery of useful information from large collection of data [1,2]. Rough set methodology for knowledge discovery provides a powerful tool for knowledge discovery from large and incomplete data. A number of algorithms and systems have been developed based on the rough set theory (for example see [3,4,5,6,7,8,9,10,11,12,13,2]), which may induce a set of decision rules from a given data, and may use induced rules to classify future examples. Most of them are attempting to find and select the best minimal set of rules, the use only minimal subset of attributes from the given data.

The advantage of employing various sources of knowledge and various structures of knowledge in data mining and knowledge discovery implies that new algorithmic method are desirable for hybrid systems in which rough set will be applied along with methods based on the following: Fuzzy Set; Neural Nets; Genetic Algorithms; etc. [8,2]. The main reason for combining different techniques in the hybrid systems is that a single technique is often not appropriate for every domain or dataset. Another reason is that such a hybrid system has an advantage over a single method approach because the technologies complement each other's shortcomings. Here we will present a new approach to task of incorporating rough set and genetic programming methods into one system for decision or classification support. We want not only to be determining the outcomes of new knowledge based on data, but also we are interested in analyzing the structure of the model to gain new insight into the problem at hand. By model structure we mean the type of the model's different components, how they related to each other, and how they can be interpreted. Finally we will describe application of our approach for mining decision rules on medical data.

2. Genetic Rough Induction (GRI)

2.1. Rough Set Theory

Rough set concept, which introduced by Pawlak [14], and one of its essential merits is its direct relation to classification problems [15], is founded on the assumption that each object is associated with some information (data, knowledge). Objects that characterized by the same information are said to be indiscernible in the view of available data. This induces the indiscernibly relation which is the mathematical base of rough set theory.

Information system IS=(U,C) is used for representing knowledge, where U is non-empty finite set of objects called universe, and C is non-empty finite set of attributes. Decision table A=(U, $C \cup \{d\}$) is a special case of information system introduced in rough set theory as tool to present data, where C is called condition attributes, and $d \notin C$ is called decision attribute. Let v_c be the value set for attribute c, then the attribute c can be considered as a map $c: U \to v_c$, i.e. $c(x)=v_c, \forall c \in C$.

Let $X \subseteq U$ be a set of objects and $B \subseteq C$ be a set of attributes, the indiscernibly relation is defined as:

$$I(B) = \{(x, y) \in U \times U: c(x) = c(y), \forall c \in B\}.$$

Objects x, y satisfying the relation I(B) are indiscernible by attributes from B. An order pair AS=(U,I(B)) is called an approximation space.

According to I(B), we can define two crisp sets $\underline{B}$ X and $\overline{B}$ X called lower and upper approximation of the set of objects X in the approximation space AS as:

$$\underline{B}X = \{x \in U: I_B(x) \subseteq X\}, \text{ and}$$

$$\overline{B}X = \{x \in U: I_B(x) \cap X \neq \phi\}.$$

$\underline{B}$ X consists of all objects that can be with certainty classified as elements of set X, and $\overline{B}$ X consists of all objects that can be possibly classified as elements of set X. The difference $BN_B(X)=(\overline{B}X - \underline{B}X)$ is called boundary of X, which contains all objects that cannot be classified either to X or complement of X.
The indiscernibly relation I(B) partitions the set U into number of disjoint equivalent classes denoted by $U/I(B)=\{X_1, X_2,\dots, X_n\}$, where $X_i \neq X_j$ for every $i \neq j$ and $\bigcup_{i=1}^{n} X_i = U$.
Decision rules can be perceived as data patterns, which represent relationship between attributes values of a classification system. If $A=(U,C\cup\{d\})$ is a decision table and $V=\cup\{v_c: c\in C\}\cup v_d$, is a set of values for attributes, then the decision rule is a logical form:

IF $(c_1= v_1)$ &...& $(c_n= v_n)$ THEN $(d = v_d)$.

There exist several measurements in order to evaluate the decision rule [7,8,2,12]. The classification accuracy and coverage of rule r are defined as follows:

$$Acc(r) = \frac{|\sup(r) \cap D|}{|\sup(r)|} \text{ ---------------(1)}$$

$$Cov(r) = \frac{|\sup(r) \cap D|}{|D|} \text{ ---------------(2)}$$

where |A| is the cardinality of a set A, Acc(r) is the classification accuracy of the rule r, Cov(r) is the coverage of the rule r, sup(r) is the number of cases that match the condition part of rule r, and |D| is the number of cases that match the decision part of rule r. It is clear that Acc(r) and Cov(r) belong to the interval [0,1]. Also we will interest in the measurements of set of rules beside of one rule [8]. The simple strength of set of rules is defined as:

$$sterngth(X_j, u_t) = \frac{|MRul(X_j, u_t)|}{|Rul(X_j)|}, \text{ -----(3)}$$

for $u_t \notin U$ is a tested object, Rul (X_j) is the set of all rules for decision class X_j, and MRul $(X_j,u_t) \subseteq$ Rul (X_j) is a set of rules that matching tested object u_t for decision class X_j.

2.2. Genetic Rough Induction (GRI)

Genetic programming [16,17,18,19] is one of several problem solving methods based on analogy computation to natural evolution under title evolutionary computations, developed by John Koza, which automatically creates a computer program from a high-level statement of problem's requirements. Since Genetic programming depends on tree-structure representation, we use C4.5 algorithm [20] to convert the data from decision table into tree-structure.
For our further considerations let us assume that we are given a data in the form of decision table $A=(U, C\cup\{d\})$, by running C4.5 to construct number of trees we convert the data into tree representation. Each tree T over the decision table A consists of terminal nodes (classes of decision attribute), non-terminal nodes (the set of attributes C), and edges (the attribute values V).
The complete path [2] in the tree T is a sequence s $=v_0, d_0, \dots, v_m, d_m, v_{m+1}$, where v_0 is the root of tree, v_{m+1} is the terminal one, $d_0 \dots d_m$ are the edges. If v_i is the initial, then v_{i+1} is the terminal of edge d_i, $i=0,\dots, m$. Let h(s) = m be defined as the length of path s. if PATH(T) is the set of all complete paths in the tree T, then $h(T)=\max\{h(s): s\in PATH(T)\}$ is called the depth of tree T.
A decision rule r is associated with a complete path s over the tree T which is denoted by r = rule(s). The set of paths PATH(T) gives us a set of rules S, where each rule $r\in S$ is associated with each path s $\in$PATH(T). Sometimes the rules derived from some paths may have a high error rate, or may duplicate rules derived from other paths, so the algorithm usually yields fewer rules than the number of paths in the tree.
Let ST(A) be the set of all trees over a decision table A that are constructed by run C4.5 number of times. The set of trees ST(A) represents a population in genetic programming concept, and each tree is an individual from this population. The number of trees in ST(A) is called the population size M. $ST_0(A)$ is the set of trees over A in generation 0 or initial population, and so $ST_i(A)$ is the population at generation i. The terminal nodes in each tree from ST(A) are assigned values from v_d, so we can say that the classes of decision attribute give here a set called terminal set in genetic programming. In the same manner the function set is the set of attributes C where each attribute is a function c(x). By applying genetic operators, we build a new population from old ones. We will define three types of genetic operators: crossover operator, mutation operator, and reproduction operator.
Crossover operator [18] is a mapping $C: ST(A)^2 \rightarrow ST(A)^2$, i.e. $C(T_1,T_2)=(T_1',T_2')$, where $T_1,T_2 \in ST_i(A)$ are called parents and $T_1',T_2' \in ST_{i+1}(A)$ are called offspring. It operates on two parental trees and creates two new two-offspring consisting of parts of each parent. The offspring are inserted into the new population at the next generation $ST_{i+1}(A)$. These offspring trees are typically of different sizes and shapes than their parents.

Mutation operator is a mapping M:ST(A) $\rightarrow$ ST(A). It generates a unique offspring tree T' from exist tree T, where M(T)= T' for T$\in$ST$_i$(A) and $T' \in$ ST$_{i+1}$(A). It operates on one parental tree and creates one new offspring to be inserted into the new population at the next generation.

Reproduction operator is a mapping R:ST(A)$\rightarrow$ ST(A), where it selects one individual T and makes a copy of the tree for inclusion in the next generation of the population. I.e. R(T)=T, for T $\in$ST(A).

To evaluate each individual in the population, we covert the tree into set of rules and evaluate it. Let define the significant of the set of rules μ as:

$$\mu = \frac{\sum_{t=1}^{n} strngth(X_j, u_t)}{n}, \text{ ---------(4)}$$

where strength(X_j,u_t) is defined in formula (3), but here since we only use training cases to generate rules and testing cases to measure the efficient of the method (see the experiments section), so we select $u_t \in$ U and make summation over t to all cases in the training examples where n is the number of cases. Since we divide by number of cases n, so $0 \le \mu \le 1$. The value μ can be called the fitness value of this set of rules that derived from tree T, so each individual has a value μ called the fitness value of this individual (i.e. μ_i is the fitness value of individual i). Depend on the fitness value the genetic operators select the individuals to be processed (the better the fitness, the more likely the individual is to be selected).

The task is to find all rules such that the significant of set of rules is at least a threshold. This is the maximum significant that is found in all previous generations, and there is a chance that offspring are fitter than parents. Where there is no predefined limit for combinations of attributes in the left-hand side of rule, and the right-hand side is not fixed, either; this is important so that unexpected rules are not ruled out before the processing start. Sometimes the combination of two unimportant attributes may result in a very good model, or may two attributes that are found to be important may depend on each other and combining them would not add anything. And here the search space of the rules has exponential size in the number of attributes.

We mainly achieve two points beside the strength of the set of rules in the fitness measure: The number of rules, and average rule length. This depends on the data we use, so we will define the fitness function f_i as:

$f_i = \alpha_1$(strength of rules μ) + α_2(no. of rules) + α_3 (1/average rule length) +α_4(1/no. of rules), where α_1, α_2, α_3, and α_4 are parameters and by controlling these parameters we can control which rules we will get (e.g. α_2 and α_4 control what we need: large or few number of rules as a result).

2.3. The algorithm

In this subsection we present a classification algorithm based on the techniques described in the previous subsection.

Input: Decision Table.

Output: Set of rules.

Process:

Step1:

Read the decision table; determine the upper and lower approximation for each class in the decision table, determine the thresholds λ for classification accuracy and θ for the coverage.

Step2:

Generate number of trees = M (the population size) by using C4.5 and running it M times. In each run of C4.5 method we change the probability of pruning for tree to get differ one. In the end of run C4.5 method, we store all trees as initial population.

Step3:

Iteratively perform the following sub-steps until reach the maximum of generation:

a. Evaluate each individual in the population by the following:
   - i. Convert the tree into set of rules (each rule is associated with a complete path).
   - ii. Remove the duplicated rules.
   - iii. Compute the classification accuracy and rule coverage for each rule, and remove the rule if its accuracy or coverage less than the thresholds λ and θ.
   - iv. Use the formula (4) as fitness measure to assign value for each set of rules.

b. Create a new population by applying the following operations:
   - i. Reproduce an existing individual (selected based on its fitness) and copy it into the new population.
   - ii. Create two new individuals from two existing individuals by genetically recombining randomly chosen parts of two existing trees using crossover operation.
   - iii. Create new individual from existing one by randomly mutating a randomly chosen part of selected tree using mutation operation.

Step 4:

The best-so-far individual is designated as the result of run (i.e. the set of rules).

3. Experiments

To verify the usefulness of the presented methodology, a computational experiment has been performed. We report results of experiments on the Medical data (Meningitis dataset). This data was colleted at the Medical Research Institute, Tokyo Medical and Dental University. It has 140 cases; each of which is described by 38 attributes: 19 numerical and 19 categorical. The attribute descriptions exist in Table 1. Some instances of the data are with missing values. We divided this data into 121 training cases and 19 testing cases. The problem is to find factors important for three decisions: Diagnosis (DIAG), Detection of bacteria or virus (CULT_FIND), and Prediction prognosis (COURSE).

The dataset was divided into a training set and testing set, and the computations of rules have been done only to training data. The results of computations of rules were applied to the classification the objects from the testing dataset. To evaluate the classification process, we use a measure called classifier's error rate [8], which is defined by the ratio of the number of error to the number of all cases. We will compare the results of our method with that are obtained from C4.5 and standard rough set methods. For the standard rough set, we use ROSETTA software to induce the rules from the dataset. In the classification process we use the same technique as in C4.5 that the rules are ordered and the first rule that covers a case is taken as the operative one. The default rule, rule without conditions that is put in the end of the list of rules, comes into play when no other rule covers a case. The algorithm requires a set of parameters that have to be manually specified and may have considerable impact on the performance of the algorithm. Furthermore, it is desirable to repeat the same process with different sets of parameters. We give in details the best set of parameters that we found through the run of our method.

Table 1: Attribute descriptions for medical data

	Attribute	Values of attribute
	Personal Information	
1	AGE	Numerical.
2	SEX	F, M.
	Diagnosis	
3	DIAG	ABSCESS, BACTERIA, BACTE (E), TB (E), VIRUS (E), VIRUS.
4	DIAG2	BACTERIA, VIRUS.
	Present History	
5	COLD	Numerical.
6	HEADACHE	Numerical.
7	FEVER	Numerical.
8	NAUSEA	Numerical.
9	LOC	Numerical.
10	SEIZURE	Numerical.
11	ONEST	RECURR, SUBACUTE, CHRONIC, ACUTE.
	Physical Examination at Admission	
12	BT	Numerical.
13	STIFF	0, 1, 2, 3, 4, 5.
14	KERNIG	0, 1.
15	LASEGUE	0, 1.
16	GCS	Numerical.
17	LOC_DAT	+, -.
18	FOCAL	+, -.
	Laboratory Examination at Admission	
19	WBC	Numerical.
20	CRP	Numerical.
21	ESR	Numerical.
22	CT_FIND	Normal, Abnormal.
23	EEG_WAVE	Normal, Abnormal.
24	EEG_FOCUS	+, -.
25	CSF_CELL	Numerical.
26	CELL_POLY	Numerical.
27	CELL_MONO	Numerical.
28	CSF_PRO	Numerical.
29	CSF_GLU	Numerical.
30	CULT_FIND	F, T.
31	CULTURE	-, Neisseria, Strepto, Staphylo, Tb, Influenza, Measles, Pi (B), Varicella, Rubella, Adeno, Herpes.
	Therapy and Course	
32	THERAPY2	MULTIPLE, ABPC+CZX, FMOX+AMK, ABPC, OPE, DARA_P, ABPC+FMOX, LMOX, PCG, ABPC+LMOX, PIPC+CTX, NO_THERAPY, ABPC+CTX, INH+RFP, ABPC+CEX, ZOBIRAX, ARA_A, INH, GLOBULIN.
33	CSF_CELL3	Numerical.
34	CSF_CELL7	Numerical.
35	C_COURCE	Negative, Dead, Frontal_sign, EEG_abnormal, CT_abnormal, Paralysis, Amncsia, Headche, Ataxia, Aphasia, Epilepsy, Memory_loss.
36	COURSE	N, P.
37	RISK	N, LC, Bechet, Sinusitis, Broncho, Myeloma, LC_DM, DM, Hepatits, TB.
38	RISK (Grouped)	N, P.

Decision attribute Diagnosis (DIAG={Bacteria, Virus})

We use 33 attributes (Personal Information, Present History, Physical Examination, Laboratory Examination, and Therapy and Course), and take only group attributes (e.g. we take attribute COURSE and delete attribute C_COURSE see Table 1). The best set of rules is obtained in generation 2 (Table 3) with number of rules fewer and shorter average rule length than which is obtained from C4.5 or standard rough set method. The error rate of the set of rules that is obtained from our method is the same as that is obtained from C4.5 method but the rules are obtained from standard rough set method have very high error rate with this data. The fitness function here depends on parameters α_1=0.9, α_2=0.1, α_3 =0, and α_4=0. In another run for our method we get from generation 58 the best set of rules, where it has also the same error rate = 0.00, but number of rules is 4.00 and average rule length is 1.00 (see third row of Table 3). Table 2 shows the run parameters that are used in our experiments where we mainly depend on crossover operator (probability rate is 0.8). Two sets of rules that are obtained from our method and C4.5 method are showed in Table 4.

Table 2: Run parameters for medical data (Decision Diagnosis and Prediction)

Max generation	32
Population size	600
Max depth for trees	17
Crossover rate	0.8

Mutation rate	0.1
Reproduction rate	0.1

Table 3: Results from medical data (Diagnosis decision).

	#rules	Average rule size	Error rate
C4.5	4	1.75	0.00 %
GRI	3	1.33	0.00 %
	4	1.00	0.00 %
Rough Set	16	2.00	78.9 %

Table 4: Sets of rules are resulted from experiments with medical data (decision Diagnosis).

GRI	C4.5 method
CELL_POLY >220 -→ Class BACTERIA CT_FIND = Abnormal CELL_MONO <=12 -→ Class BACTERIA CELL_MONO >12 -→ Class VIRUS	CELL_POLY >220 -→ Class BACTERIA CT_FIND =Abnormal CELL_MONO <=12 -→ Class BACTERIA CELL_POLY<=220 CELL_MONO >12 -→ Class VIRUS CT_FIND = Normal CELL_POLY <=220 -→ Class VIRUS
Default Class: VIRUS	Default Class: VIRUS

Decision attribute Detection (CULT_FIND={T, F})

We use 16 attributes only (Personal Information, Present History, Physical Examination) and ignore laboratory examination (to get new knowledge and the laboratory examination is very expansive also). The values for α-parameters of fitness function are the same as decision DIAGNOSIS. The best set of rules is obtained in generation 117 (Table 5) with number of rules better than C4.5 and shortest average rule length. Our method is the best one in error rate compared with C4.5 method and standard rough set method. The parameters for run here are shown in Table 7 where we increase mutation operator rate to 0.5. The list of rules that are obtained from our method and C4.5 method is showed in Table 6.

Table 5: Result of run in medical data (decision attribute Detection).

	# rules	Average rule size	Error rate
C4.5	1	6	26.3%
GRI	5	2.40	21.1 %
Rough Set	92	11.00	63.2 %

Table 6: Rules resulted from experiments with medical data (decision attribute Detection).

GRI	C4.5
AGE<=46 COLD <=7 NAUSEA <=9 LOC <=0 ONEST = ACUTE BT >36.1 -→ Class F GCS <=10 -→ Class F COLD >7 COLD <=9 -→ Class T ONEST=CHRONIC -→ Class T AGE >37 COLD >7 -→ Class T	AGE<=46 COLD <=7 NAUSEA <=9 LOC <=0 ONEST = ACUTE BT >36.1 -→ Class F
Default Class: F	Default Class: F

Decision attribute Prediction (COURSE ={n, p})

We use with this data 16 attributes (Personal Information, Present History, Physical Examination) and we ignore Laboratory Examination. The rules resulted from our run with new method and C4.5 method are showed in Table 9. The best set of rules is obtained in generation 4 (Table 8) with number of rules, average rule length, and error rate is same as C4.5 method, but better than standard rough set method, where from Table 9 we observe that the rules obtained from our method differ than that are obtained from C4.5 method except only one rule is the same. The parameters for run are showed in Table 2 as in decision attribute DIAGNOSIS.

Table 7: Parameters for run in medical data (Decision attribute Detection).

Max generation	200
Population size	800
Max depth for trees	17
Crossover rate	0.4
Mutation rate	0.5
Reproduction rate	0.1

Table 8: Results from medical data (Decision attribute Prediction).

	# rules	Average rule size	Error rate
C4.5	4	2.50	5.26%
GRI	4	2.50	5.26 %
Rough Set	95	14.00	73.7 %

Table 9: Rules resulted from experiments with medical data (decision attribute Prediction).

GRI	C4.5
COLD>9 FOCAL =+ -→ Class P SEX =M AGE >62 HEADACHE >2.6e-09 -→ Class P	AGE>21 FEVER <=8 AGE <=62 FOCAL =- -→ Class N COLD>9 FOCAL =+

LOC >2 STIFF = 2 -→ Class P AGE <=62 FEVER <=21 FOCAL = - -→ Class N Default Class: N	-→ Class P SEX =M AGE >62 -→ Class P LOC >2 BT <= 38.6 -→ Class P Default Class: N

Regarding the application we introduced, our approach achieves a more accuracy than heuristics of C4.5 and standard rough set theory. But C4.5 is faster, however the execution time was not a major tasks involved in its utilization. These results are suggesting that our method can be treated as a promising tool for extracting laws from experimental datasets and its performance is fully comparable with the performance of other classification systems.

4. Conclusions

On the basis of a modified rough set theory, we have presented a new algorithm, which provides an efficient and effective mechanism for knowledge discovery in database system. In the hybrid framework, rough set theory and genetic programming are integrated into hybrid system and used cooperatively to generate a set of rules from database. We believe that, successful research requires good co-operation between theoreticians and practitioners, so we presented in this paper the analysis of structure of the model for our new method and compared standard method for extracting laws from decision table based on rough set theory and that based on C4.5 algorithm with our new method on medical dataset. We observe that the rules extracted by our new method are relatively better predisposed in classification than both C4.5 and standard rough set approaches.

Reference

[1] Mannila, H., Inductive Database and Condensed Representations for Data Mining, processing of International Logic Programming Symposium 1997, pp 21-30.

[2] Ziarako, W., Ed. Rough Sets, Fuzzy Sets, and Knowledge Discovery, Springer Verlag, Berlin, 1994.

[3] Hassan, Y., and Tazaki, E., Knowledge Discovery Using Rough Set Combined with Genetic Programming, Accepted from JSAI International Workshop on Rough Set Theory and Granular Computing, 2001.

[4] Dong, J.Z., Zhong, N., and Ohsuga, S., Probabilistic Rough Induction, Yamakawa, T., and Matsumoto, G. (edt.) Methodologies for the Conception, Design and application of Soft Computing and Information/Intelligent Systems (IISUKA'98), World Scientific, 1998,pp 943-946.

[5] Dong, J.Z., Zhong, N., and Ohsuga, S., Rule Discovery by Probabilistic Rough Induction, Japanese Soc. Artificial Intelligence, 2000, pp 274-286.

[6] Garcia, A., and Shasha, D., Using Rough Sets to Order Questions Leading to Database Queries, In Nagib C. Callaos (ed.), Proceedings of the International Conference on Information Systems Analysis and Synthesis (ISAS'96), July 22-26, Orlando, USA, 1996, pp 555-560.

[7] Nakayama, H., Hattori, Y., and Ishii, R., Rule Extraction based on Rough Set Theory and its Application to Medical data Analysis, IEEE SMC'99 Conference Proceedings, 1999.

[8] Polkowski, L., and Showron, A., Rough sets in Knowledge Discovery, Physica Nerlag, 1998.

[9] Siromoney, A., and Inoue, K., Elementary Sets and Declarative Biases in a Restricted GRS-ILP model, Slovene Soc. Informatika24, 2000, pp 125-135.

[10] Stepaniuk, J., and Maj, M., Data Transformation and Rough Sets, Principles of Data Mining and Knowledge Discovery. Second European Symposium, PKDD'98, pp 441-9,1998

[11] Tsumoto, S., and Tanaka, H., Incremental learning of probabilistic rules from clinical databases. In Proceedings of the Sixth International Conference, Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96), Granada, Spain, 1996, pp 1457-1462.

[12] Tsumoto, S., and Tanaka, H., Discovery of Approximate Medical Knowledge based on Rough Set Model, Principles of Data Mining and Knowledge Discovery. Second European Symposium, PKDD'98, pp 468-76,1998

[13] Ziarako, W., Discovery Through Rough Set Theory, Communications of the ACM: Vol. 42. No. 11, 1999, pp 54-57.

[14] Pawlak, Z., Rough Sets, International Journal of Computer and Information Science, Vol. 11, No. 5, 1982, pp 341-356.

[15] Pawlak, Z., Rough Classification, International Journal of Man-Machine studies 20, 1984, pp 469-483.

[16] Kinnear, Jr., K.E., Advances in Genetic Programming, Mit Press, 1994.

[17] Koza, J.R., Genetic Programming III Darwinian Invention and Problem Solving, San Francisco, CA, 1999.

[18] Dumitrescu, D., Lazzerini, B., Jain, L.G., and Dumitrescu, A., Evolutionary Computation, CRC Press, 2000.

[19] Longdon, William B., Genetic Programming and Data Structure: Genetic Programming + Data Structure = Automatic Programming!, Amsterdam: Kluwer, 1998.

[20] Quinlan, J.R., C4.5: Programs for Machine Learning, Morgan Kaufinann, san Marteo, CA, 1993.