On the Evolution of Rough Set Exploration System

Jan G. Bazan[1], Marcin S. Szczuka[2], Arkadiusz Wojna[2], and Marcin Wojnarski[2]

[1] Institute of Mathematics, University of Rzeszów
Rejtana 16A, 35-959 Rzeszów, Poland
bazan@univ.rzeszow.pl
[2] Faculty of Mathematics, Informatics and Mechanics, Warsaw University
Banacha 2, 02-097, Warsaw, Poland
{szczuka,wojna}@mimuw.edu.pl, mwojnars@ns.onet.pl

Abstract. We present the next version (ver. 2.1) of the Rough Set Exploration System – a software tool featuring a library of methods and a graphical user interface supporting variety of rough-set-based and related computations. Methods, features and abilities of the implemented software are discussed and illustrated with examples in data analysis and decision support.

1 Introduction

Research in decision support systems, classification algorithms, in particular those concerned with application of rough sets requires experimental verification. To be able to make thorough, multi-directional practical investigations and to focus on essential problems one needs an inventory of software tools that automate basic operations. Several such software systems have been constructed by various researchers, see e.g. [13, vol. 2]. That was also the idea behind creation of the Rough Set Exploration System (**RSES**).

It is already almost a decade since the first version of RSES appeared. After several modifications, improvements and removal of detected bugs it was used in many applications. Comparison with other classification systems (see [12, 1]) proves its value. The RSESlib, which is a computational backbone of RSES, was also used in construction of the computational kernel of ROSETTA — an advanced system for data analysis (see [19]).

The first version of Rough Set Exploration System (RSES v. 1.0) in its current incarnation and its further development (RSES v. 2.0) were introduced approximately four and two years ago, respectively (see [3, 4]). The present version (v. 2.1) introduces several changes, improvements and, most notably, several new algorithms – the result of our recent research developments in the area of data analysis and classification systems.

The RSES software and its computational kernel maintains all advantages of previous versions. The algorithms have been re-mastered to provide better flexibility and extended functionality. New algorithms added to the library follow

S. Tsumoto et al. (Eds.): RSCTC 2004, LNAI 3066, pp. 592–601, 2004.

the current state of our research. Improved construction of the system allows further extensions and supports augmentation of RSES methods into other data analysis tools.

The re-implementation of the RSES core classes in Java^{TM} 2 and removal of legacy code is further fostered in the RSES v. 2.1. The computational procedures are now written in Java using its object-oriented paradigms. The migration to Java simplifies some development operations and, ultimately, leads to improved flexibility of the product permitting migration of RSES software to operating systems other than Windows (currently e.g. Linux).

In this paper we briefly show the features of the RSES software, focusing on recently added algorithms and methods. The changes in GUI and improvements in existing components are also described. We illustrate the presentation of new methods with examples of applications in the field of classification systems.

2 Basic Notions

To give the reader a better understanding of the RSES' description, we bring here some basic notions that are further used in the presentation of particular methods.

The structure of data that is the central point of our work is represented in the form of *information system* or, more precisely, the special case of an information system called *decision table*.

Information system is a pair of the form $\mathbf{A} = (U, A)$ where U is a *universe* of *objects* and $A = \{a_1, ..., a_m\}$ is a set of *attributes* i.e. mappings of the form $a_i : U \to V_{a_i}$, where V_{a_i} is called *value set* of the attribute a_i. The decision table is also a pair of the form $\mathbf{A} = (U, A \cup \{d\})$ with a distinguished attribute d. In the case of decision table the attributes belonging to A are called *conditional attributes* or simply *conditions* and d is called *decision*. We will further assume that the set of decision values is finite. The i-th *decision class* is a set of objects $C_i = \{o \in U : d(o) = d_i\}$, where d_i is the i-th decision value taken from the decision value set $V_d = \{d_1, ..., d_{|V_d|}\}$

A reduct is one of the most essential notions in rough sets. $B \subset A$ is a *reduct* of information system if it carries the same indiscernibility information as the whole A, and no proper subset of B has this property. In case of decision tables a *decision reduct* is a set of attributes $B \subset A$ such that it cannot be further reduced and carries the same indiscernibility information as the decision.

A *decision rule* is a formula of the form $(a_{i_1} = v_1) \wedge ... \wedge (a_{i_k} = v_k) \Rightarrow d = v_d$, where $1 \leq i_1 < ... < i_k \leq m$, $v_i \in V_{a_i}$. Atomic subformulae $(a_{i_1} = v_1)$ are called *conditions*. We say that the rule r is *applicable* to an object, or alternatively, an object *matches* a rule, if its attribute values satisfy the premise of the rule. With a rule we can connect some numerical characteristics such as *matching* and *support*, that help in determining rule quality (see [1, 2]).

By *cut* for an attribute $a_i \in A$, such that V_{a_i} is an ordered set we will denote a value $c \in V_{a_i}$. With the use of a cut we may replace the original attribute a_i with a new, binary attribute which depends on whether the original attribute value for an object is greater or lower than c (more in [10]).

Template of **A** is a propositional formula $\bigwedge(a_i = v_i)$ where $a_i \in A$ and $v_i \in V_{a_i}$. A generalised template is the formula of the form $\bigwedge(a_i \in T_i)$ where $T_i \subset V_{a_i}$. An object *satisfies* (matches) a template if for every attribute a_i occurring in the template the value of this attribute on the considered object is equal to v_i (belongs to T_i in case of a generalised template). The template induces in natural way the split of original information system into two distinct subtables. One of those subtables contains objects that satisfy the template, the other the remainder. Decomposition tree is a binary tree, whose every internal node is labelled by a certain template and external node (leaf) is associated with a set of objects matching all templates in a path from the root to the leaf (see [10]).

3 Contents of RSES v. 2.1

3.1 Input/Output Formats

During operation certain functions belonging to RSES may read and write information to/from files. Most of these files are regular ASCII files.

Slight changes from the previous RSES versions were introduced in the format used to represent the basic data entity i.e. the decision table. The new file format permits attributes to be represented with use of integer, floating point number or symbolic (text) value. There is also a possibility of using "virtual" attributes, calculated during operation of the system, for example derived as a linear combinations of existing ones. The file format used to store decision tables includes a header where the user specifies size of the table, the name and type of attributes. The information from header is visible to the user in the RSES GUI e.g., attribute names are placed as column headers when the table is being displayed.

RSES user can save and retrieve data entities such as rule sets, reduct sets etc. The option of saving the whole workspace (project) in a single file is also provided. The project layout together with underlying data structures is stored using dedicated, optimised binary file format.

3.2 The Algorithms

The algorithms implemented in RSES fall into two main categories.

First category gathers the algorithms aimed at management and edition of data structures. It covers functions allowing upload and download of data as well as derived structures, procedures for splitting tables, selecting attributes etc. There are also procedures that simplify preparation of experiments, such as an automated n fold cross-validation.

The algorithms for performing rough set based and classification operations on data constitute the second essential kind of tools implemented inside RSES.

Most important of them are:
Reduction algorithms i.e. algorithms allowing calculation of the collections of reducts for a given information system (decision table). In the version 2.1 the method for calculation of dynamic reducts (as in [1]) is added.
Rule induction algorithms. Several rule calculation algorithms are present. That includes reduct-based approaches (as in [2]) as well as evolutionary and covering methods (cf. [17, 8]). Rules may be based on both classical and dynamic reducts. Calculated rules are accompanied with several coefficients that are further used while the rules are being applied to the set of objects.
Discretisation algorithms. Discretisation permits discovery of cuts for attributes. By this process the initial decision table is converted to one described with simplified, symbolic attributes; one that is less complex and contains the same information w.r.t. discernibility of objects (cf. [1, 10]).
Data completion algorithms. As many real-life experimental data contains missing data, some methods for filling gaps in data are present in RSES. For more on data completion techniques see [9].
Algorithms for generation of new attributes. New attributes can be generated as linear combinations of existing (numerical) ones. Such new attributes can carry information that is more convenient in decision making. The proper linear combinations are established with use of methods based on evolutionary computing (cf. [4, 14]).
Template generation algorithms provide means for calculation of templates and generalised templates. Placed side by side with template generation are the procedures for inducing table decomposition trees (cf. [11]).
Classification algorithms used to determine decision value for objects with use of decision rules, templates and other means (cf. [1, 2, 11]). Two major new classification methods have been added in RSES version 2.1. They belong to the fields of instance-based learning and artificial neural networks, respectively. They are described in more detail further in this paper (Sections 4.1 and 4.2). The classification methods can be used to both verifying classifiers on a test sample with given decision value and classifying new cases for which we do not know decision value.

3.3 The RSES GUI

To simplify the use of RSES algorithms and make it more intuitive the RSES graphical user interface was further extended. It is directed towards ease of use and visual representation of workflow. Version 2.0 (previous one) undergone some face lifting. There are some new gadgets and gizmos as well. Project interface window has not change much (see Fig. 1). As previously, it consists of two parts. The visible part is the project workspace with icons representing objects created during our computation. Behind the project window there is the history window, reachable via tab, and dedicated to messages, status reports, errors and warnings. While working with multiple projects, each of them occupies a separate workspace accessible via tab at the top of workplace window.

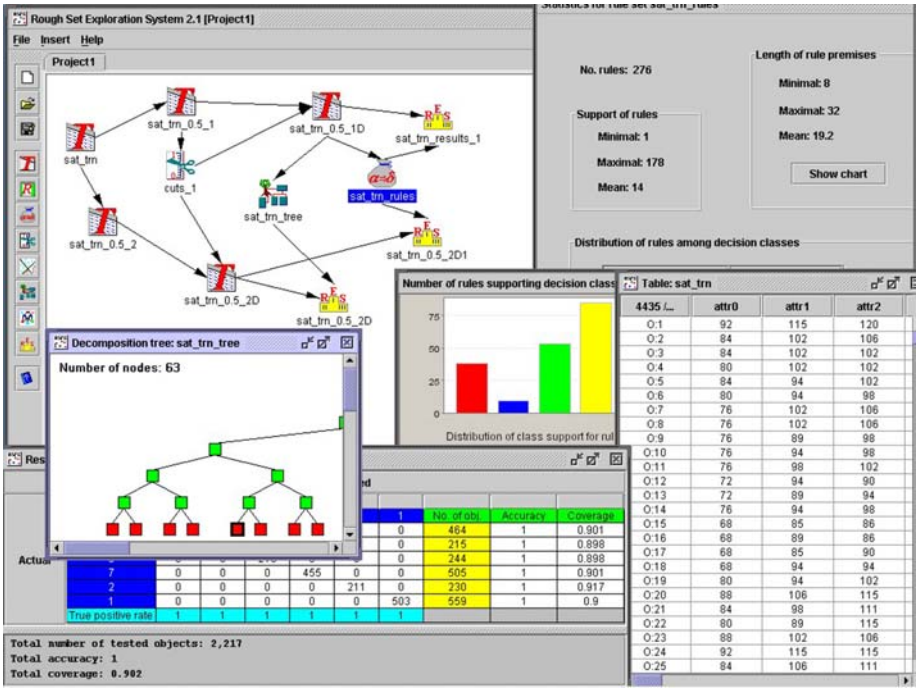

Fig. 1. The project interface window

It was designers' intention to simplify the operations on data within project. Therefore, the entities appearing in the process of computation are represented in the form of icons placed in the upper part of workplace. Such an icon is created every time the data (table, reducts, rules,...) is loaded from the file. User can also place an empty object in the workplace and further fill it with results of operation performed on other objects. Every object appearing in the project have a set of actions associated with it. By right-clicking on the object the user invokes a context menu for that object. It is also possible to invoke an action from the general pull-down program menu in the main window. Menu choices allow to view and edit objects as well as include them in new computations. In many cases a command from context menu causes a new dialog box to open. In this dialog box the user can set values of parameters used in desired calculation. If the operation performed on the object leads to creation of a new object or modification of existing one then such a new object is connected with edge originating in object(s) which contributed to its current state. Placement of arrows connecting icons in the workspace changes dynamically as new operations are being performed. In the version 2.1 the user has the ability to align objects in workspace automatically, according to his/her preferences (eg. left, horizontal, bottom).

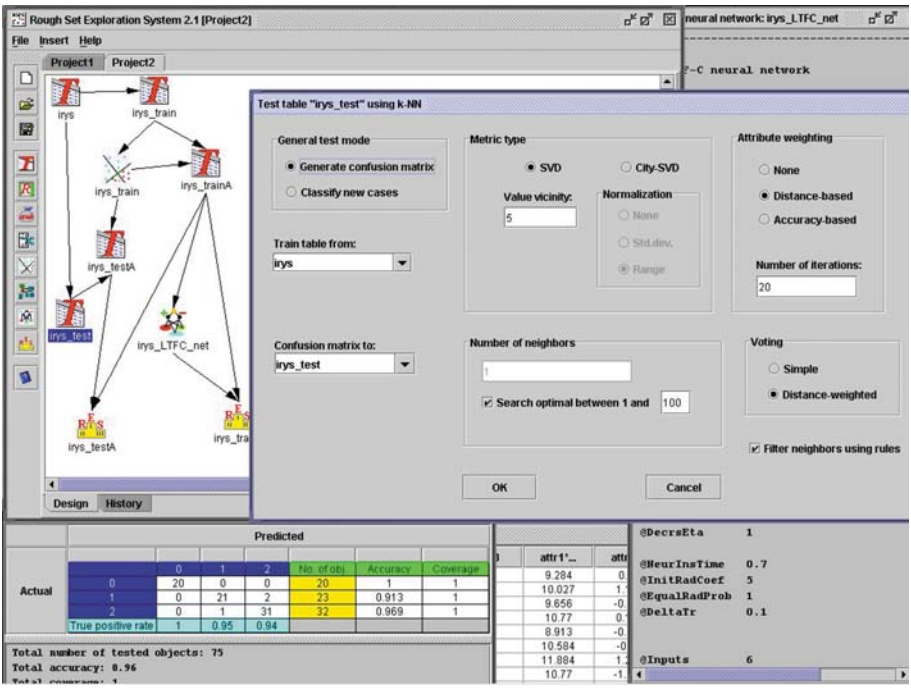

Fig. 2. Instance based classification in the RSES GUI

An important new GUI feature added in the version 2.1 is the possibility to display some statistical information about tables, rules and reducts in a graphical form (see Fig.1).

4 New Methods

In the current version two new classification methods have been added.

4.1 Instance Based Method

As an instance based method we implemented the special, extended version of the k nearest neighbours (k-nn) classifier [6]. First the algorithm induces a distance measure from a training set. Then for each test object it assigns a decision based on the k nearest neighbours of this object according to the induced distance measure.

The distance measure ρ for the k-nn classifier is defined as the weighted sum of the distance measures ρ_a for particular attributes $a \in A$:

$$\rho(x, y) = \sum_{a \in A} w_a \cdot \rho_a(a(x), a(y)).$$

Two types of a distance measure are available to the user. The City-SVD metric [5] combines the city-block Manhattan metric for numerical attributes with the Simple Value Difference (SVD) metric for symbolic attributes.

The distance between two numerical values $\rho_a(a(x), a(y))$ is the difference $|a(x) - a(y)|$ taken either as an absolute value or normalised with the range $a_{\max} - a_{\min}$ or with the doubled standard deviation of the attribute a on the training set. The SVD distance $\rho_a(a(x), a(y))$ for a symbolic attribute a is the difference between the decision distributions for the values $a(x)$ and $a(y)$ in the whole training set. Another metric type is the SVD metric. For symbolic attributes it is defined as in the City-SVD metric and for a numerical attribute a the difference between a pair of values $a(x)$ and $a(y)$ is defined as the difference between the decision distributions in the neighbourhoods of these values. The neighbourhood of a numerical value is defined as the set of objects with similar values of the corresponding attribute. The number of objects considered as the neighbourhood size is the parameter to be set by a user.

A user may optionally apply one of two attribute weighting methods to improve the properties of an induced metric. The distance-based method is an iterative procedure focused on optimising the distance between the training objects correctly classified with the nearest neighbour in a training set. The detailed description of the distance-based method is described in [15]. The accuracy-based method is also an iterative procedure. At each iteration it increases the weights of attributes with high accuracy of the 1-nn classification.

As in the typical k-nn approach a user may define the number of nearest neighbours k taken into consideration while computing a decision for a test object. However, a user may use a system procedure to estimate the optimal number of neighbours on the basis of a training set. For each value k in a given range the procedure applies the leave-one-out k-nn test and selects the value k with the optimal accuracy. The system uses an efficient leave-one-out test for many values of k as described in [7].

When the nearest neighbours of a given test object are found in a training set they vote for a decision to be assigned to the test object. Two methods of nearest neighbours voting are available. In the simple voting all k nearest neighbours are equally important and for each test object the system assigns the most frequent decision in the set of the nearest neighbours. In the distance-weighted voting each nearest neighbour vote is weighted inversely proportional to the distance between a test object and the neighbour. If the option of filtering neighbours with rules is checked by a user, the system excludes from voting all the nearest neighbours that produce a local rule inconsistent with another nearest neighbour (see [7] for details).

The k-nn classification approach is known to be computationally expensive. The crucial time-consuming task is searching for k nearest neighbours in a training set. The basic approach is to scan the whole training set for each test object. To make it more efficient an advanced indexing method is used [15]. It accelerates searching up to several thousand times and allows to test datasets of a size up to several hundred thousand of objects.

Table 1. Classification error of k-nn classifiers with the estimation of the optimal value of k from a training set

Nearest neighbors voting Filtering with rules			Simple No filtering		Dist-weighted No filtering		Simple Filtering		Dist-weighted Filtering	
Dataset	Trn set	Test set	est. k	error	est. k	error	est. k	error	est. k	error
segment	1 540	770	1	2,47%	1	2,47%	1	2,47%	4	2,73%
splice (DNA)	2 000	1 186	1	5,99%	1	5,99%	1	5,99%	1	5,99%
chess	2 131	1 065	1	2,45%	1	2,45%	1	2,45%	20	1,6%
satimage	4 435	2 000	5	9,45%	4	9,85%	5	9,45%	4	9,35%
mushroom	5 416	2 708	1	0%	1	0%	1	0%	1	0%
pendigits	7 494	3 498	1	2,84%	5	2,26%	1	2,84%	4	2,29%
nursary	8 640	4 320	13	1,95%	15	0,75%	19	0,91%	13	0,31%
letter	15 000	5 000	1	3,22%	5	2,92%	1	3,22%	8	2,82%
census94	30 162	15 060	27	15,95%	151	16,44%	76	16,27%	160	16,29%
shuttle	43 500	14 500	1	0,06%	3	0,06%	1	0,06%	2	0,05%

Table 1 presents the classification accuracy for 10 data sets from the UCI repository [21]. The data sets provided as a single file (*segment, chess, mushroom, nursery*) have been randomly split into a training and a test part with the ratio 2 to 1. The remaining data sets (*splice, satimage, pendigits, letter, census94, shuttle*) have been tested with the originally provided partition. In the experiment the City-SVD metric with the distance based attribute weighting method were used. We tested four k-nn based classifiers: all combinations of simple and distance weighted voting with and without filtering neighbours with rules. To make the results comparable all classifiers were tested with the same instance of a distance measure and the same partition for each data set. The values of k used in the experiments were selected from the range between 1 and 200 by the procedure delivered with the system.

The results from Table 1 show that the accuracy of the k-nn classifiers is comparable to other well-known classifiers like C5.0 [7]. The classification error is similar for different parameter setting but in general the k-nn with distance-weighted voting and rule-based filtering seems to have a little advantage over the k-nn classifiers with the other setting.

4.2 Local Transfer Function Classifier

Local Transfer Function Classifier (LTF-C) is a neural network solving classification problems [16]. Its architecture is very similar to this of Radial Basis Function neural network (RBF) or Support Vector Machines (SVM) – the network has a hidden layer with gaussian neurons connected to an output layer of linear units. There are some additional restrictions on values of output weights that enable to use an entirely different training algorithm and to obtain very high accuracy in real-world problems.

The training algorithm of LTF-C comprises four types of modifications of the network, performed after every presentation of a training object:

1. changing positions (means) of gaussians,
2. changing widths (deviations) of gaussians, separately for each hidden neuron and attribute,
3. insertion of new hidden neurons,
4. removal of unnecessary or harmful hidden neurons.

As one can see, the network structure is dynamical. The training process starts with an empty hidden layer, adding new hidden neurons when the accuracy is insufficient and removing the units which do not positively contribute to the calculation of correct network decisions. This feature of LTF-C enables automatic choice of the best network size, which is much easier than setting the number of hidden neurons manually. Moreover, this helps to avoid getting stuck in local minima during training, which is a serious problem in neural networks trained with gradient-descend.

LTF-C shows a very good performance in solving real-world problems. A system based on this network won the first prize in the EUNITE 2002 World Competition "Modelling the Bank's Client behaviour using Intelligent Technologies". The competition problem was to classify bank customers as either active or non-active, in order to predict if they would like to leave the bank in the nearest future. The system based on LTF-C achieved 75.5% accuracy, outperforming models based on decision trees, Support Vector Machines, standard neural networks and others (see [20]) .

LTF-C performs also very well in other tasks, such as handwritten digit recognition, breast cancer diagnosis or credit risk assessment (details in [16]).

5 Perspective

The RSES toolkit will further grow as new methods and algorithms emerge. More procedures are still coming from current state-of-the-art research. Most notably, the work on a new version of the RSESlib library of methods is well under way. Also, currently available computational methods are being integrated with DIXER - a system for distributed data processing.

The article reflects the state of software tools at the moment of writing, i.e. beginning of March 2004. For information on most recent developments visit the Web site [18].

Acknowledgement

Many persons have contributed to the development of RSES. In the first place Professor Andrzej Skowron, the supervisor of all RSES efforts from the very beginning. Development of our software was supported by grants 4T11C04024 and 3T11C00226 from Polish Ministry of Scientific Research and Information Technology.

References

1. Bazan, J.: A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Tables, In [13], vol. 1, pp. 321–365
2. Bazan, J.G., Nguyen, S.H, Nguyen, H.S., Synak, P., Wróblewski, J.: Rough Set Algorithms in Classification Problem. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds), Rough Set Methods and Applications, Physica-Verlag, Heidelberg, 2000 pp. 49–88.
3. Bazan, J., Szczuka, M.,: RSES and RSESlib - A Collection of Tools for Rough Set Computations. Proc. of RSCTC'2000, LNAI 2005, Springer-Verlag, Berlin, 2001, pp. 106–113
4. Bazan, J., Szczuka, M., Wróblewski, J.: A New Version of Rough Set Exploration System. Proc. of RSCTC'2002, LNAI 2475, Springer-Verlag, Berlin, 2002, pp. 397–404
5. Domingos, P.: Unifying Instance-Based and Rule-Based Induction. Machine Learning, Vol. 24(2), 1996, pp. 141–168.
6. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis, Wiley, New York, 1973.
7. Góra, G., Wojna, A.G.: RIONA: a New Classification System Combining Rule Induction and Instance-Based Learning. Fundamenta Informaticae, Vol. 51(4), 2002, pp. 369–390.
8. Grzymała-Busse, J.: A New Version of the Rule Induction System LERS. Fundamenta Informaticae, Vol. 31(1), 1997, pp. 27–39
9. Grzymała-Busse, J., Hu, M.: A Comparison of Several Approaches to Missing Attribute Values in Data Mining. Proc. of RSCTC'2000, LNAI 2005, Springer-Verlag, Berlin, 2001, pp. 340–347
10. Nguyen Sinh Hoa, Nguyen Hung Son: Discretization Methods in Data Mining. In [13] vol.1, pp. 451-482
11. Hoa S. Nguyen, Skowron, A., Synak, P.: Discovery of Data Patterns with Applications to Decomposition and Classfification Problems. In [13] vol.2, pp. 55-97.
12. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood, London, 1994
13. Skowron A., Polkowski L.(ed.): Rough Sets in Knowledge Discovery vol. 1 and 2. Physica-Verlag, Heidelberg, 1998
14. Ślęzak, D., Wróblewski, J.: Classification Algorithms Based on Linear Combinations of Features. Proc. of PKDD'99, LNAI 1704, Springer-Verlag, Berlin, 1999, pp. 548–553.
15. Wojna, A.G.: Center-Based Indexing in Vector and Metric Spaces. Fundamenta Informaticae, Vol. 56(3), 2003, pp. 285-310.
16. Wojnarski, M.: LTF-C: Architecture, Training Algorithm and Applications of New Neural Classifier. Fundamenta Informaticae, Vol. 54(1), 2003, pp. 89–105
17. Wróblewski, J.: Covering with Reducts - A Fast Algorithm for Rule Generation. Proceeding of RSCTC'98, LNAI 1424, Springer-Verlag, Berlin, 1998, pp. 402-407
18. Bazan, J., Szczuka, M.: The RSES Homepage, `http://logic.mimuw.edu.pl/~rses`
19. Ørn, A.: The ROSETTA Homepage, `http://www.idi.ntnu.no/~aleks/rosetta`
20. Report from EUNITE World competition in domain of Intelligent Technologies, `http://www.eunite.org/eunite/events/eunite2002/competitionreport2002.htm`
21. Blake, C.L., Merz, C.J.:UCI Repository of machine learning databases. Irvine, CA: University of California, 1998, `http://www.ics.uci.edu/~mlearn`