RSES and RSESlib - A Collection of Tools for Rough Set Computations

Jan G. Bazan[1] and Marcin Szczuka[2]

[1] Institute of Mathematics, Pedagogical University of Rzeszów
Rejtana 16A, 35-310 Rzeszów, Poland
e-mail: bazan@univ.rzeszow.pl
[2] Institute of Mathematics, Warsaw University
Banacha Str. 2, 02-097, Warsaw, Poland
e-mail: szczuka@mimuw.edu.pl

Abstract. Rough Set Exploration System - a set of software tools featuring a library of methods and a graphical user interface is presented. Methods, features and abilities of the implemented software are discussed and illustrated with a case study in data analysis.

1 Introduction

Research in decision support systems, classification algorithms in particular those concerned with application of rough sets requires experimental verification. At certain point it is no longer possible to perform every single experiment using software designed for a single purpose. To be able to make thorough, multi-directional practical investigations one have to posess an inventory of software tools that automatise basic operations, so it is possible to focus on the most essential matters. That was the idea behind creation of Rough Set Exploration System, further referred as RSES for short.

First version of RSES and the library RSESlib was released several years ago. After several modifications, improvements and removal of detected bugs it was used in many applications. The RSESlib - library of tools for rough set computations was successfully used for data analysis with encouraging effects. Comparison with other classification systems (see [10]) proves its value. The RSESlib is also used in construction of computational kernel of ROSETTA - an advanced system for data analysis (see [15]).

The RSES software and its computational kernel - the new RSESlib 2.0 library maintains all advantages of previous version. The algorithms from the first incarnation of library are now re-mastered to provide better flexibility, extended functionality and ability to process massive data sets. New algorithms added to the library reflect the current state of our research in classification methods originating in rough sets theory. Improved construction of library allows further extensions and supports augmentation of RSESlib into different data analysis tools.

The library of functions is not sufficient as an answer to experimenters' demand for helpful tool. Therefore the RSES user interface was constructed. This interface allows to use RSESlib interactively. User can construct experiments by easy point-and-click objects in the workplace. It is also possible to construct the RSES scripts that describe compound, sometimes exhaustive operations.

2 Basic notions

In order to provide clear description further in the paper and avoid any misunderstandings we bring here some essential definitions from Rough Set theory.

The structure of data that is central point of our work is represented in the form of *information system* [11] or, more precisely, the special case of 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 $a_i : U \to V_a$, where V_a is called *value set* of the attribute a_i. The decision table is also a pair of the form $\mathbf{A} = (U, A \cup \{d\})$ where the major feature that is different from the information system is the distinguished attribute d. In case of decision table the attributes belonging to A are called *conditional attributes* or simply *conditions* while d is called *decision* (sometimes *decision attribute*). 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 decision value set $V_d = \{d_1, ..., d_{rank(d)}\}$

For any subset of attributes $B \subset A$ *indiscernibility relation* $IND(B)$ is defined as follows:

$$xIND(B)y \Leftrightarrow \forall_{a \in B} a(x) = a(y) \tag{1}$$

where $x, y \in U$.

Having indiscernibility relation we may define the notion of reduct. $B \subset A$ is a *reduct* of information system if $IND(B) = IND(A)$ and no proper subset of B has this property. In case of decision tables the notion of *decision reduct* is handy. The decision reduct is a the set $B \subset A$ of attributes such that it cannot be further reduced and $IND(B) \subset IND(d)$.

Decision rule is a formula of the form

$$(a_{i_1} = v_1) \wedge ... \wedge (a_{i_k} = v_k) \Rightarrow d = v_d \tag{2}$$

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 rule r is *applicable* to object, or alternatively, the object *matches* rule, if its attribute values satisfy the premise of the rule. With the rule we can connect some characteristics. *Support* denoted as $Supp_{\mathbf{A}}(r)$ is equal to the number of objects from $\mathbf{A}$ for which rule r applies correctly i.e. premise of rule is satisfied and the decision given by rule is similar to the one preset in decision table. $Match_{\mathbf{A}}(r)$ is the number of objects in $\mathbf{A}$ for which rule r applies in general. Analogously the notion of matching set for a rule or collection of rules may be introduced (see [1], [3]).

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 cut we may replace original attribute a_i with new, binary attribute which tells as whether actual attribute value for an object is greater or lower than c (more in [7]).

Template of $\mathbf{A}$ 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 considered object is equal to v_i (belongs to T_i in case of 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 oder those that do not. Decomposition tree is a binary tree, whose every internal node is labeled by some template and external node (leaf) is associated with a set of objects matching all templates in a path from the root to a given leaf (see [6] for more details).

3 The RSES library v. 2.0

The RSES library (RSESlib) is constructed according to the principles of object oriented programming. The programming language used in implementation is Microsoft Visual C++ compliant with ANSI/ISO C++ (ISO/IEC 14882 standard). The library and accompanying tools are primarily targeted at the users of Intel/Windows platform, however the versatile construction and universal programming language used allow to compile and link the library with other operating systems (e.g. Linux, Solaris) with only cosmetic changes.

3.1 Input/Output formats

During operation certain functions belonging to RSES may read and write information to/from files. Most of the files that can be read or written are regular ASCII text files. They particular sub-types can be distinguished by reviewing the contents or identifying file extensions (if used).

These are:

- **Decision table**, with default file extension **.tab**, stores the contents of information system (decision table) in the form of rectangular table. Rows in the table correspond to objects, columns are the attributes. Last column is assumed to be decision attribute, but it can be changed using library function after the table is loaded. First row in the table file contains two integers being the number of rows and columns in the table respectively.
- **Cut file**, with default extension **.cut** is used to store cuts for attributes used in the process of discretisation. Every row in the file contains the number of attribute and the value of a cut for this attribute.
- **Reduct file**, with default extension **.red**, stores the reducts obtained in the process of data reduction. First row of a file contains the information on the number of reducts and the reducts are listed one by in the consequent rows of the file.

- **Rule file**, with default extension **.rul**, is used to store derived decision rules together with additional information. Every decision rule is accompanied with a set of numbers (coefficients) reflecting rules' behaviour with respect to the decision table it was induced from.
- **Results file**, with default extension **.res** is used to store results of experiments performed on consider data sets.
- Two more file types are used to store RSES GUI projects (extension **.rsp**) and RSES scripts (extension **.rss**).

3.2 The algorithms

The algorithms that have been implemented in the RSES library fall into two general categories.

First category gathers the algorithms aimed at management and edition of data structures that are present in the library. Functions allowing upload and download of data as well as derived structures, procedures for adding/removing objects, setting decision attributes, calculating statistical information about data and others are provided in order to make basic manipulation with library objects easier, more natural and better co-ordinated with other available operations.

The algorithms for performing Rough Set theory based operations on data constitute the second, most essential kind of tools implemented inside RSES library. To give the idea what apparatus is given to the user we describe shortly the most important algorithms.

Reduction algorithms i.e algorithms allowing calculation of the collections of reducts for a given information system (decision table). The exhaustive algorithm for calculation of all reducts is present, however such operation may be time-consuming due to computational complexity of such task (see [12]). Therefore approximate and heuristic solutions such as genetic or Jonhson algorithms were implemented (see [13], [5] for details). The library methods for calculation of reducts allow setting initial conditions for number of reducts to be calculated, required accuracy, coverage and so on. Basing on calculated reduct it is possible to calculate decision rules (see [3]). Procedures for rule calculation allow user to determine some crucial constrains for the set of decision rules. Rules received are accompanied with several coefficients that are further used while the rules are being applied to the set of objects (see [2], [1]). In connection with algorithms for reduct/rule calculation appear the subclass of algorithms allowing shortening of rules and reducts with respect to different requirements (see [2]).

Discretisation algorithms allow to find cuts for attributes. In this way initial decision table is converted to one described with less complex, binary attribute without lose of information about discernibility of objects. Foundations for discretisation algorithms preserving information are given in [9], [7], [1].

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. Details of that algorithms may be reviewed in [6] and [8].

Classification algorithms used for establishing decision value with use of decision rules and/or templates. Operations for voting among rules with use of different schemes fall into this category. Thorough discussion of such algorithms can be found in [2], [8], [3] and [1].

4 The RSES GUI.

To simplify the use of RSES library and make it more intuitive a graphical user interface was constructed. This interface allows interaction with library methods in two modes. First is directed towards ease of use and visual representation of workflow. The other is intended to provide tools for construction of scripts that use RSES functionality and allow to perform multiple, complicated and thorough experiments.

4.1 The project interface.

Project interface window(see Figure 1) consists of two parts. Upper part is the project workspace where icons representing objects occurring during our computation are presented. Lower part is dedicated to messages, status reports, errors and warnings produced during operations triggered from upper window. It was designers intention to simplify the operations on data within project. Therefore, the entities appearing in the process of rough set based 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. The objects that may exist in the workplace are: decision table, collection of reducts, set of rules, decomposition tree, set of cuts and collection of results. Every object (icon) appearing in the project have a set of actions connected with it. By right-clicking on the object the user invokes a context menu for that object. It is the easiest way to perform the operation on object, however it is also possible to call the necessary action form general pull-down program menu in the main window. Menu choices allow to view and edit objects as well as make them input to some computation. In many cases choice of some option from context menu will cause a new dialog box to open. In this dialog box user can set values of coefficients used in desired calculation, in particular, designate the variable which will store the results of invoked operation. If the operation performed on the object leads to creation of new object or modification of existing one then such a new object is connected with edge originating in object (or objects) which contributed to its current state. Such edges (arrows) make interpretation of information workflow easier. Setting of arrows connecting icons in the workspace changes dynamically as new operations are being performed.

The entire project can be saved to file on disk to preserve results and information about current setting of coefficients used in the calculation process. That also allows to re-create the entire work on other computer or with other data.

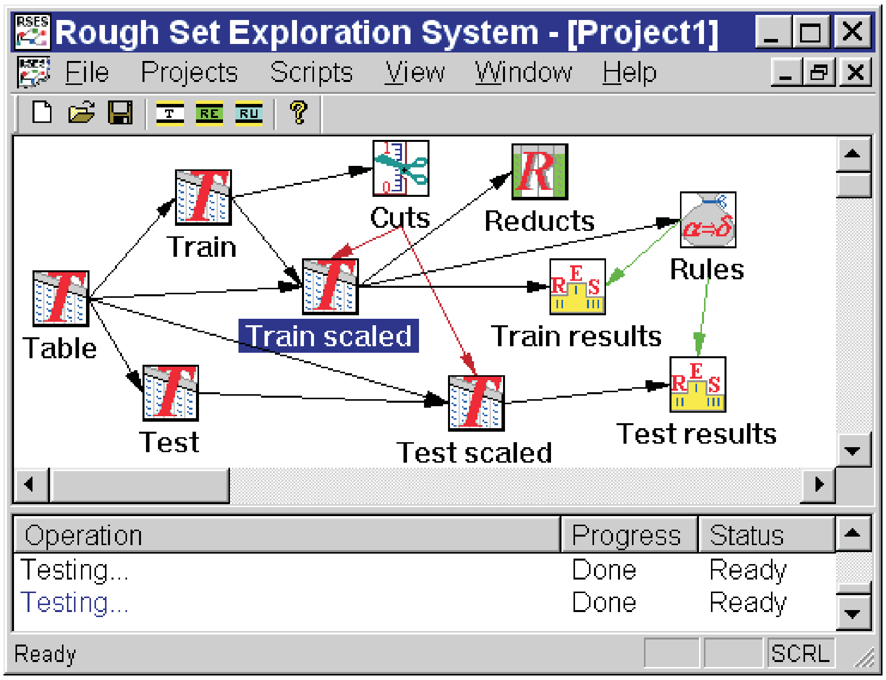

Fig. 1. The project interface window.

4.2 Scripting interface.

In case we have to perform many experiments with different parameter settings and changing data (e.g. cross-validation test) it is more convenient to plan such a set of operations in advance and then let computer calculate. The idea of simplifying the preparation and execution of compound experiments drove the creation of RSES scripting mechanism.

The mechanism for performing script-based operations with use of RSES library components consists of three major parts. The scripting interface is the part visible to user during script preparation. Other two are behind the scenes and perform simple syntax checking and script execution. We will not describe checking and executing in greater detail since such a matter my be interesting rather from software engineering than script writers' point of view.

The user interface for writing RSES based scripts is quite simple. Main window is split into two parts of which upper contains workplace where scripts are being edited and lower contains messages (errors, status, warnings, progress etc.) generated during script execution (see Figure 2).

The RSES scripting language constructs available to user are:

- Variables. Any variable is inserted to script with predefined type. The type may be either standard (e. g. integer, real) or RSES-specific (e.g. variable representing set of cuts).

– Functions. User can use all the functions from RSES armory as well as standard arithmetic operations. Function in RSES script always returns a value.
– Procedures. The procedures from RSES library, unlike functions, may not return a value. The procedures correspond to major operations such as: reduct calculation, rule shortening, loading and saving objects.
– Conditional expressions. The expressions of the form **If ... then ... else** may be used. The user defines condition with use of standard operations.
– Loops. Simple loop can be used within RSES script. The user is required to designate loop control variable and set looping parameters.

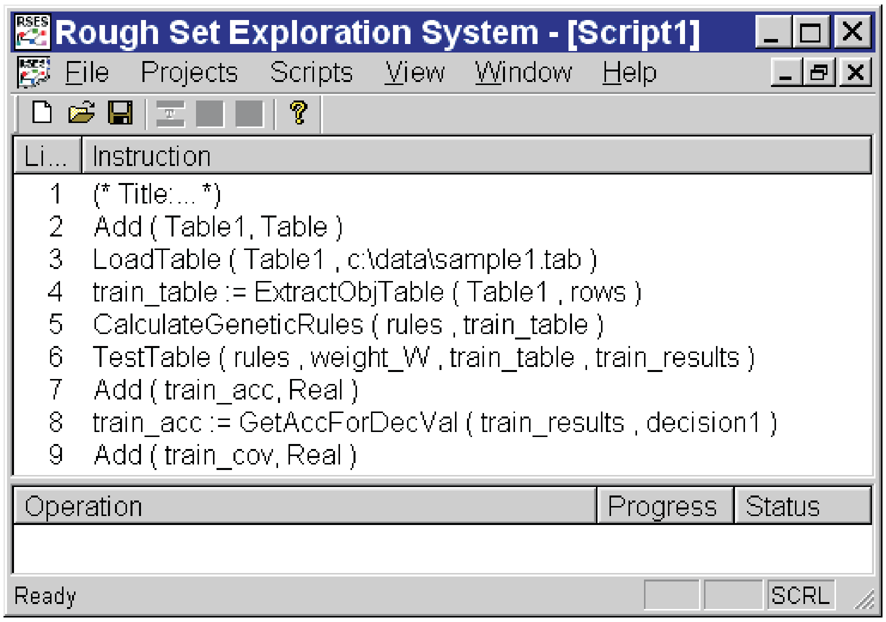

Fig. 2. The script interface window.

While preparing a script the user may not freely edit it. He/she can only insert or remove one of the constructs mentioned above using context menu which appears after right-clicking on the line in the script where insertion/removal is due to occur. The new construct may also be inserted with use of toolbar. The operation of inserting the new construct involves choosing the required operation name and setting all required values such as names of variables, conditions etc. For example, to extract subtable from a given table (RSES function **ExtractObjTable**) we have to determine name of variable storing original table, list of objects (lines) to be extracted and a name of variable in which the resulting table should be stored. All this is done by easy point-and-click operations within appropriate dialog box supported by pull-down lists of available names. All the edition operations are monitored by checker to avoid obvious errors such as type mismatch.

Once the script edition is finished it can be executed with menu command **Run**. Before execution the syntax is checked once again. The behaviour of currently running script may be seen in the lower part of interface window.

5 Case study

As already mentioned, the ability of dealing with large data sets is one of key new features of RSESlib 2.0. To deal with such a massive data we use decomposition based on template-induced trees.

5.1 Data decomposition by template tree

Decomposition is a problem of partitioning a large data table into smaller ones. One can use templates extracted from data to partition data table into blocks of objects with common features. We consider here decomposition schemes based on a template tree (see [6]). The main goal of this method is to construct a decomposition tree. Let $\mathbf{A}$ be a decision table. The algorithm for decomposition tree construction can be presented as follows:

Algorithm 1 *Decomposition by template tree* (see [6])
Step 1 Find the best template T in $\mathbf{A}$.
Step 2 Divide $\mathbf{A}$ onto two subtables: $\mathbf{A}_1$ containing all objects satisfying $\mathbf{T}$
and $\mathbf{A}_2 = \mathbf{A} - \mathbf{A}_1$.
Step 3 If obtained subtables are of acceptable size
(in the sense of rough set methods)
then stop
else repeat 1-3 for all "too large" subtables.

This algorithm produces a binary tree of subtables with corresponding sets of decision rules for subtables in the leaves of the tree.

The decision tree produced by algorithm presented below can be used to classify a new case to proper decision class. Suppose we have a binary tree created in the process of decomposition. Let u by a new object and $\mathbf{A}(T)$ be a subtable containing all objects matching template T. We classify object u starting from the root of the tree as follows:

Algorithm 2 *Classification by template tree* (see [6])
Step 1 If u matches template T found for $\mathbf{A}$
then: go to subtree related to $\mathbf{A}(T)$
else: go to subtree related to $\mathbf{A}(\neg T)$.
Step 2 If u is at the leaf of the tree then go to 3
else: repeat 1-2 substituting $\mathbf{A}(T)$ (or $\mathbf{A}(\neg T)$) for $\mathbf{A}$.
Step 3 Classify u using decision rules for subtable attached to the leaf

This algorithm uses a binary decision tree, however it should not be mistaken for C4.5, ID3 or other algorithms using decision trees. As we told before, in our experiments (see Section 5.3), a rough set methods have been used for classifying algorithm construction in leaves of the decomposition tree (see [1] and [2] for more details).

5.2 Forest CoverType data

The Forest CoverType data used in our experiments were obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data (see [4] [16] for more details).

Data is in rectangle form (information system) with 581012 rows (objects) and 56 columns (55 attributes and decision). There are 7 decision values.

The Forest CoverType data has been studied before using Neural Networks and Discriminant Analysis Methods (see [4]).

5.3 Experiments with data decomposition

The Forest CoverType data original decision table was divided into a training set (11340 objects), a validation set (3780 objects) and a testing set (565892 objects) (see [4]). In our experiments we used algorithms presented in Section 5.1 that are implemented in RSESlib. The decomposition tree for data have been created only by reference to the training set. The validation set was used for adaptation of classifying algorithms which obtained in the leafs of the decomposition tree (ses [1] and [2] for more details about adaptation of classification algorithms). The classification algorithm was applied to new cases from the test set. As a measure of classification success (or failure) we use the most common measure called *accuracy* (see e.g. [6], [4]). The accuracy we define as the ratio of the number of properly classified new cases to the total number of new cases. In table 1 we give the results of experiments in applying three different classification systems to the Forest CoverType data.

Paradigm	Accuracy
Neural Networks	0.70
Discriminant Analysis	0.58
Rough Set (RSES)	0.73

Table 1. A comparison results

6 Perspective

The RSES toolset will further grow as new methods and algorithms emerge. More procedures are still coming from current state-of-the-art research. The article reflect the state of software tools at the moment of writing, i.e. end of April. For information on most recent developments visit the Website [14].

Acknowledgement: In the first place the special tribute should be paid to Professor Andrzej Skowron who, for all these years, is the *spiritus movens* of RSES evolution. We want to stress our gratitude to colleagues who greatly contributed providing their algorithms: Adam Cykier, Nguyen Sinh Hoa, Nguyen Hung Son, Jakub Wróblewski, Piotr Synak.

Development of our software was significantly supported by European ESPRIT project 20288 - CRIT 2. M. Szczuka was also supported by the Wallenberg Foundation in frame of the WITAS project.

References

1. Bazan J., A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Tables, In: Skowron A., Polkowski L.(ed.), Rough Sets in Knowledge Discovery 1, Physica Verlag, Heidelberg, 1998, pp. 321-365
2. Bazan J., Approximate reasoning methods for synthesis of decision algorithms (in Polish), Ph. D. Thesis, Department of Math., Comp. Sci. and Mechanics, Warsaw University, Warsaw, 1998
3. Bazan J., Son H. Nguyen, Trung T. Nguyen, Skowron A. and J. Stepaniuk, Decision rules synthesis for object classification. In: E. Orłowska (ed.), Incomplete Information: Rough Set Analysis, Physica - Verlag, Heidelberg, 1998, pp. 23-57.
4. Blackard, J.,A., Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types, Ph. D. Thesis, Department of Forest Sciences, Colorado State University. Ford Collins, Colorado, 1998.
5. Garey M., Johnson D., Computers and Intarctability: A Guide to the Theory of NP-completness, W.H. Freeman&Co., San Francisco, 1998, (twentieth print)
6. Nguyen Sinh Hoa, Data regularity analysis and applications in data mining. Ph. D. Thesis, Department of Math., Comp. Sci. and Mechanics, Warsaw University, Warsaw, 1999
7. Nguyen Sinh Hoa, Nguyen Hung Son, Discretization Methods in Data Mining, In: Skowron A., Polkowski L.(ed.), Rough Sets in Knowledge Discovery 1, Physica Verlag, Heidelberg, 1998, pp. 451-482
8. Hoa S. Nguyen, A. Skowron and P. Synak, Discovery of data patterns with applications to decomposition and classfification problems. In: L. Polkowski and A. Skowron (eds.), Rough Sets in Knowledge Discovery 2, Physica-Verlag, Heidelberg, 1998, pp. 55-97.
9. Nguyen Hung Son, Discretization of real value attributes. Boolean reasoning approach. Ph. D. Thesis, Department of Math., Comp. Sci. and Mechanics, Warsaw University, Warsaw, 1997
10. Michie D., Spiegelhalter D. J., Taylor C. C., Machine Learning, Neural and Statistical Classification, Ellis Horwood, London, 1994
11. Pawlak Z., Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer, Dordrecht, 1991
12. Rauszer C., Skowron A., The Discernibility Matrices and Functions in Information Systems, In: Słowiński R. (ed.), Intelligent Decision Support, Kluwer, Dordrecht 1992.
13. Wróblewski J., Covering with Reducts - A Fast Algorithm for Rule Generation, Proceeding of RSCTC'98, LNAI 1424, Springer Verlag, Berlin, 1998, pp. 402-407
14. Bazan J., Szczuka M., The RSES Homepage, `http://alfa.mimuw.edu.pl/~rses`
15. Ørn A., The ROSETTA Homepage, `http://www.idi.ntnu.no/~aleks/rosetta`
16. Bay, S. D. , The UCI KDD Archive, `http://kdd.ics.uci.edu`