Missing Template Decomposition Method and Its Implementation in Rough Set Exploration System

Jan G. Bazan[1], Rafał Latkowski[2] and Marcin S. Szczuka[3]

[1] Institute of Mathematics, University of Rzeszów
Rejtana 16A, 35-959 Rzeszów, Poland
bazan@univ.rzeszow.pl

[2] Institute of Informatics, Warsaw University
Banacha 2, 02-097, Warsaw, Poland
rlatkows@mimuw.edu.pl

[3] Institute of Mathematics, Warsaw University
Banacha 2, 02-097, Warsaw, Poland
szczuka@mimuw.edu.pl

Abstract. We describe a method of dealing with sets that contain missing information in case of classification task. The described method uses multi-stage scheme that induces and combines classifiers for complete parts of the original data. The principles of the proposed Missing Template Decomposition Method are presented together with general explanation of the implementation within the RSES framework. The introduced ideas are illustrated with an example of classification experiment on a real data set.

1 Introduction

The hard task of dealing with data imperfection in inductive learning methods was addressed in the area of data impreciseness by Pawlak in early 80's [9]. He proposed a *Rough Set* approach that made possible to precisely express facts about imprecise data in a formal way. The main concept of Rough Sets, the *indiscernibility relation*, proved to be very useful for analysis of decision problems concerning objects described in a data table by a set of conditional attributes and a decision attribute [10, 11]. However, original definition of the rough set theory does not capture the situation where some of the attribute values are missing. In last twenty years a great research effort has been made in the area of data incompleteness to develop methods inducing classifiers for data with missing attribute values. Some approaches making possible to handle missing attribute values have been developed within the rough sets framework, see [5, 6, 14].

One can identify three major approaches to the issue of handling missing data in classification tasks. These are:

– Modification of indiscernibility relation by adopting it to handle missing attribute values (see [6, 14]).

- Modification of classifier induction algorithms like, e.g., in case of *LEM1* and *LEM2* (see [5]).
- Imputation — replacement of missing values with regular ones (see [3, 4]).

The Missing Template Decomposition method (MTD) represents an approach that cannot be strictly classified to any of the three streams of research listed above. It is devised to make it possible to reason on the basis of data with missing attribute values without modification of the inductive learning algorithm itself. The empirical evaluation of core MTD method has been already presented in, e.g, [7] showing that MTD can improve not only the reasoning quality, but also it can reduce complexity of classifier.

In this paper we present a brief description of the principles of Missing Template Decomposition classifier that has been implemented with use of previous experiences (see [7]) and added to the collection of methods that are available within the framework of Rough Set Exploration System (RSES).

The Rough Set Exploration System (RSES) is a free software tool for analysis and exploration of data with use of methods originating in the Rough Set theory. It is being developed for several years and provides a stable platform for experiments with data (see [2]). It can be downloaded from [16].

This paper first presents the general concepts about data, missing values, and templates. Then we introduce the principles of MTD method and the classification method based upon it. The method and its implementation in RSES is illustrated with an example of experiment on *head injury* (`hin`) data.

2 Basic notions

As usual in Rough Set approach, we start with data set represented in the form of *information system* or, more precisely, the special case of information system called *decision table*.

Information system is a pair of the form $\mathbb{A} = (U, A)$ where U is a *universe* of *objects* and $A = \{a_1, ..., a_m\}$ is a set of *attributes* i.e. mappings of the form $a_i : U \to V_a \cup \{?\}$, where V_a is called *value set* of the attribute a_i and ? denotes missing value. The decision table is also a pair of the form $\mathbb{A} = (U, A \cup \{d\})$ with 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*. 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 $V_d = \{d_1, ..., d_{|V_d|}\}$.

For any subset of attributes $B \subset A$ *indiscernibility relation* $IND(B)$ for $x, y \in U$ is defined as follows:

$$x\, IND(B)\, y \iff \forall_{a \in B}\, a(x) = a(y). \qquad (1)$$

The indiscernibility relation, as an equivalence relation, induces decomposition of objects into *indiscernibility classes* in which all objects are identically described on attributes from subset B. The above, classic, definition of indiscernibility relation is capable to handle missing attribute values only in exactly the same way as regular values. We will use K to denote a number of all indiscernibility classes $[x^1]_{IND(B)}, \ldots, [x^K]_{IND(B)}$, and $M \leq K$ to denote a number of inconsistent indiscernibility classes $[x^{j_1}]_{IND(B)}, \ldots, [x^{j_M}]_{IND(B)}$, where an inconsistent indiscernibility class $[x^{j_m}]_{IND(B)}$ contains objects from more than one decision class (i.e., $card(\{d(x) : x \in [x^{j_m}]_{IND(B)}\}) > 1$).

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 rule r is *applicable* to an object, or alternatively, the object *matches* rule, if its attribute values satisfy the premise of the rule. With the rule we can connect some numerical characteristics such as *matching* and *support* (see [1]).

Missing template t (also called *total template*) of $\mathbb{A}$ is a propositional formula $\bigwedge(a_i \neq ?)$ where $a_i \in A$. An object *satisfies* (matches) a template if for every attribute a_i occurring in the missing template the value of this attribute on considered object is defined (i.e., different from ?). A *width* of template t denoted as $w(t)$ is a number of attributes occurring in the template. A *height* of template t denoted as $h(t)$ is the number of objects satisfying the template. The missing template t induces in natural way a subtable $\mathbb{S}_t = (U_t, A_t \cup \{d\})$ of original information system $\mathbb{A} = (U, A \cup \{d\})$ consisting of set objects U_t that satisfy the missing template t and set of attributes A_t occurring in the template (c.f. [7]). Obviously, $h(t) = card(U_t)$, $w(t) = card(A_t)$ and the subtable $\mathbb{S}_t$ is complete, i.e. totally described, while all objects satisfying a template are described on attributes occurring in the template.

We will also use a normalization factor $\rho = \frac{card(U)}{card(U_t)} = \frac{card(U)}{h(t)}$ to normalize heuristic measures of different missing templates, D to denote a number of decision classes occurring in subtable $\mathbb{S}_t$ and D_i to denote a number of decision classes occurring in i-th indiscernibility class $[x^i]_{IND(A_t)}$ of $\mathbb{S}_t$.

3 Missing Template Decomposition

The Missing Template Decomposition method (MTD), as it was indicated in introduction, differs from main streams of research on reasoning with incomplete object description. It is meant to meet two requirements. The first one is to adapt many well-known classifier induction methods, that are initially not capable of handling missing attribute values, to the case of incomplete data. In other words, MTD makes it possible to analyze incomplete information systems by previously known and implemented classification methods without the need for their modification. The second requirement is that MTD shall be able to cope with the problem of incomplete data without making an additional assumption of independent random distribution of missing values and without using data imputation methods [3, 4]. The second requirement comes from the fact that many real world applications have shown that appearance of missing values may be governed by very complicated dependencies. Missing attribute values are frequently not uniformly distributed but, their distribution is determined by the hidden nature of investigated phenomenon, just like in the case of regular values. Hence, the application of an arbitrary method for data imputation can reduce accuracy of a classifier.

3.1 Classifier Induction

The MTD tries to avoid the necessity of reasoning on data with missing attribute values. The original incomplete data is decomposed into data subsets which contain no missing values. Next, methods for classifier induction are applied to these complete subsets. Finally, a conflict resolving method is used to obtain final solution from partial classifiers constructed on subtables.

In the data decomposition phase the original decision table with missing attribute values is partitioned into a number of decision subtables which contain no missing values. This data decomposition should reveal patterns in distribution of missing attribute values. Ideally, the complete subtables that are result of the decomposition should correspond to natural subproblems of the whole problem domain. With the help of the concept of total template introduced earlier, we can define data decomposition phase as generation of set of total templates t_1, $\ldots$, t_T and extraction of subtables $\mathbb{S}_{t_1}$, $\ldots$, $\mathbb{S}_{t_T}$ that satisfy these templates (see Fig. 1).

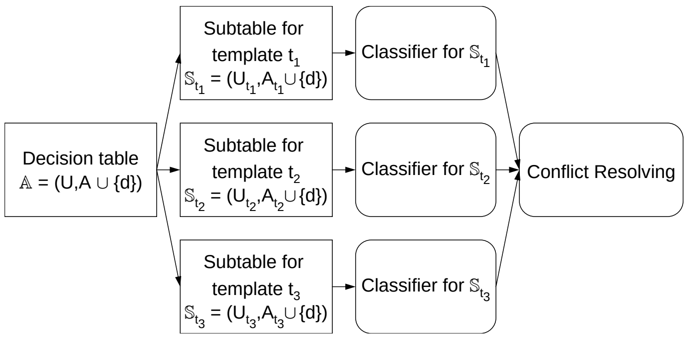

Decision table
$\mathbb{A} = (U, A \cup \{d\})$
Subtable for template t_1
$\mathbb{S}_{t_1} = (U_{t_1}, A_{t_1} \cup \{d\})$
Subtable for template t_2
$\mathbb{S}_{t_2} = (U_{t_2}, A_{t_2} \cup \{d\})$
Subtable for template t_3
$\mathbb{S}_{t_3} = (U_{t_3}, A_{t_3} \cup \{d\})$
Classifier for $\mathbb{S}_{t_1}$
Classifier for $\mathbb{S}_{t_2}$
Classifier for $\mathbb{S}_{t_3}$
Conflict Resolving

Fig. 1. Missing Template Decomposition Method.

Once we have data decomposed into complete decision subtables, we perform classifier induction from these subtables and classifier fusion (refer to figure Fig. 1). For classifier induction one can apply arbitrary method of inductive learning. In our current implementation it is possible to utilize decision tree or decision rules' classifier. For decision tree classifier we can play with two parameters: minimal required confidence (purity) of each leaf and minimal nonterminal node size. For decision rule classifier we can select between optimal exhaustive decision rules (all rules, see [2, 12]) and LEM2 rules (see [13]). For classifier fusion we could apply any method for conflict resolution. Current implementation provides standard voting method and decision tree method for this purpose.

The decision subtables $\{\mathbb{S}_t = (U_t, A_t \cup \{d\})\}$ contain exactly the same decision attribute as the original decision table $\mathbb{A} = (U, A \cup \{d\})$. This fact determines the course of classifier induction. All classifiers induced for decision subtables are classifying objects into the same decision classes, but for a given object some of these classifiers may be not applicable. Note, that during classification process the (subtable-induced) classifier is applicable to an object only if this object satisfies total template related to the considered classifier. On the other hand, there may be objects for which more than one classifier is applicable. That is why after classifier induction we need a mechanism for conflict resolution.

Conflict resolving shall result in creation of the final answer. In case of standard voting this answer, depending of requirements, may be obtained in one of two ways. First approach to conflict resolving takes into account only one final decision value (definite decision class assignment) for each partial classifier on the examined object. The object is assigned a decision value that has been selected by majority of partial classifiers. In the second approach conflict resolving is done on vector of decision class assignment probabilities. The final result (class assignment) is reached by taking the decision with highest cumulative probability.

In the case of conflict resolving with use of decision tree, the tree is induced from a virtual decision table that consist of all classifier answers for objects from training table. Once decision tree is constructed, it is utilized to merge answers from all partial classifiers.

We can briefly summarize the missing template decomposition method as follows:

- Create set of templates $t_1, \ldots, t_T$ for missing values in the following way:
  - Create a temporary set of objects $U' := U$ from the original decision table and repeat two following steps until the temporary set U' becomes empty:
  - Generate the best missing template t_i for objects U' according to a chosen criterion;
  - Remove from the temporary set U' objects that are covered by missing template t_i;
- Create complete decision subtables $\mathbb{S}_{t_1}, \ldots, \mathbb{S}_{t_T}$ that correspond to previously generated set of templates;
- Induce classifiers over complete decision subtables;
- Select a conflict resolving method (or learn a conflict resolving strategy) to get the final answer.

3.2 Data Decomposition Criteria

Subsets $\mathbb{S}_t$ of original decision table $\mathbb{A}$ must satisfy some requirements in order to achieve good quality of inductive reasoning as well as applicability in case of methods that cannot deal with missing attribute values. We expect the decision subtables to exhaustively cover the input table (at least in the terms of objects, i.e., $\bigcup_{i=1}^{T} U_{t_i} = U$). They should contain no missing values. It is also obvious that the quality of inductive reasoning depends on a particular partition and some partitions are better than others.

In current implementation of MTD the search for promising set of total templates $t_1, \ldots, t_T$ is done with help of heuristic functions and genetic algorithm with variable population size. The library utilized by RSES software provides several heuristic functions for total template evaluation. These heuristic functions join properties of standard template evaluation measures with feature selection measures, especially measures based on rough sets. The implemented heuristic functions are of the form $q(t) = w(t)^\alpha \cdot h(t) \cdot f(t)^\beta$, where $q(t)$ is considered heuristic function, called also quality function of template, $f(t)$ is an additional template evaluation measure, and α, β are exponents for controlling the impact of different components of quality function. Currently there are 8 template evaluation measures implemented in RSES and ready to be used for this purpose. These are:

– S — size measure only, $f(t) = 1$, the template quality function has form $q(t) = w(t)^\alpha \cdot h(t)$,
– C — conflict measure that counts conflicts in inconsistent indiscernibility classes, $f(t) = \frac{maxc(t)-c(t)}{maxc(t)}$, where $c(t)$ is a function similar to conflict $c(t) = \rho \cdot \sum_{i=1}^{M} \prod_{d=1}^{D_i} card(\{x \in [x^{j_i}]_{IND(A_t)} : d(x) = d\}))$ and $maxc(t) = \rho \cdot \prod_{d=1}^{D} card(\{x \in U_t : d(x) = d\})$ is a function that estimates maximal possible $c(t)$ value from the top,
– I — inconsistency measure, $f(t) = \frac{h(t)-i(t)}{h(t)}$, where $h(t)$ estimates $i(t)$ value from the top and $i(t) = \rho \cdot \sum_{i=1}^{M} \sum_{d=1}^{D_i} card(\{x \in [x^{j_i}]_{IND(A_t)} : d(x) = d\})$,
– D — average ratio of maximal purity within indiscernibility classes, $f(t) = \frac{1}{K} \sum_{i=1}^{K} \frac{max_{d \in V_d} card(\{x \in [x^i]_{IND(A_t)} : d(x)=d\})}{card(\{x \in [x^i]_{IND(A_t)}\})}$,
– E — proportion of maximal purity within indiscernibility classes to template size, $f(t) = \sum_{i=1}^{K} \frac{max_{d \in V_d} card(\{x \in [x^i]_{IND} : d(x)=d\})}{h(t)}$,
– F — $f(t) = \frac{1}{max(1, c(t))}$, where $c(t)$ is defined above,
– G — $f(t) = \sum_{i=1}^{K} \frac{max_{d \in V_d} card(\{x \in [x^i]_{IND(A_t)} : d(x)=d\})}{card([x^i]_{IND(A_t)})}$ (i.e., $G = K \cdot D$),
– H — $f(t) = \frac{1}{K} \sum_{i=1}^{K} \frac{max_{d \in V_d} card(\{x \in [x^i]_{IND} : d(x)=d\})}{h(t)}$ (i.e., $E = K \cdot H$),
– P — predictive measure, $f(t)$ is an accuracy of decision tree classifier trained and tested on table $\mathbb{S}_t$.

4 Example of Experiment with MTD

To bring the functionality of MTD closer to reader's intuition we present an experiment performed with the RSES' implementation of MTD. Our experiment is carried out with use of `hin` data. It is a set of data describing head injuries data with three possible decision values (moderate disability or good recovery, severe disability, dead or vegetative), 6 conditional attributes and 1000 observations. This dataset was selected because of the quantity of missing information. In the `hin` data table 40.5% of all objects are incomplete (contain at least one ?) and 9.8% of all values are missing. This data was originally split into ten separate train&test pairs for ten fold cross-validation (c.f. [8]), but for simplicity we use only the first pair of train&test datasets.

To begin with, we have to load the training and test data tables into the RSES system (see Fig. 2). Once we have data loaded we can start experiments. In particular we can induce *Missing Template Decomposition Classifier* (*MTD-C*). This is easily done by selecting appropriate option from context menu for an icon representing training table.

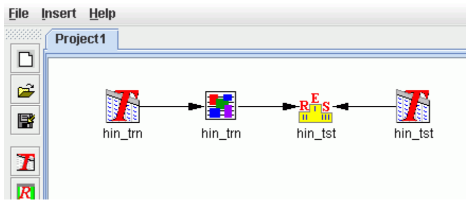
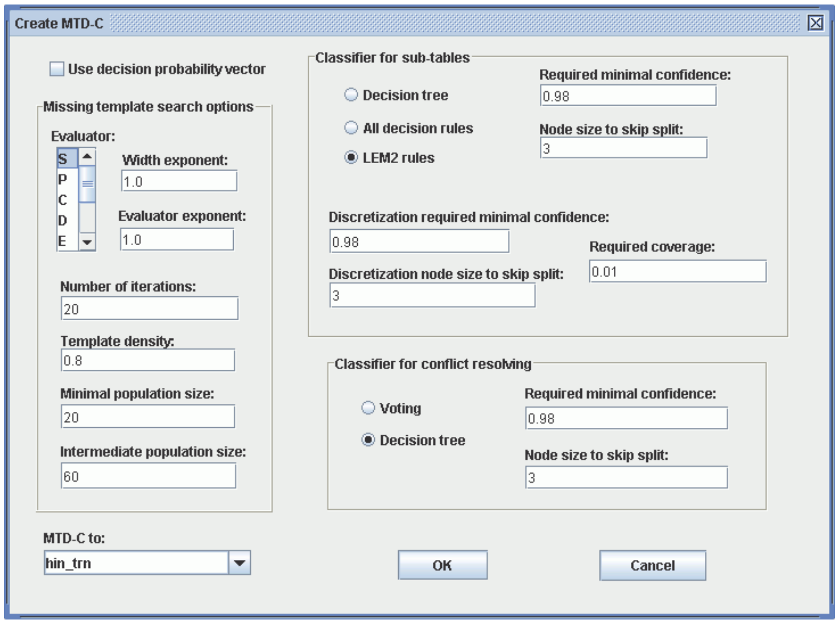


Fig. 2. Simple project diagram of RSES system with decision tables for training and testing, MTD induced from train table and classification results of MTD over test table.

Selecting this option causes the RSES system to show the dialog box with settings for MTD, as presented in Fig. 3. Most of these settings were described in the previous section, so here we just briefly point at their location. In the top left part a checkbox for selecting the type of answers gathered from subclassifiers is located. If it is unchecked, then each classifier outputs only one decision value for an object. Checking it causes RSES to use decision probability vectors. Most of the left part of the dialog is related to parameters for missing template generation method. The user can select an additional evaluation function as described previously as well as exponent factors for selecting importance of different components in the the template quality function. There are also settings for genetic algorithm with variable population size that may be used to search for templates. The user can adjust the number of genetic algorithm iterations, probability of including each attribute in initial randomized population, and minimal and maximal population sizes. In the field placed at the bottom left corner the user can select a name for our MTD, which will be used to identify it in RSES project diagram.

The right side of the dialog is devoted to settings for classifiers and conflict resolving method. In the upper part the user can select the algorithm for induction of classifiers for subtables (decision tree, all rules or LEM2 rules) and their settings. Below classifier selection menu there is a section that controls some basic settings for internal discretization algorithms. The internal discretization is required if some attributes in data are numeric and is done automatically for each subtable before rule induction. In our example the head injury data contains no numeric attributes, so discretization is not required. In RSES we have also option of discretizing the entire data table with appropriate algorithm before invoking the MTD module. Finally, the bottom-right part of the dialog is related to selection of conflict resolving method. The user has choice between voting and decision tree learning methods. For decision tree conflict resolution method the user can also specify parameters such as confidence level and minimal size of a leaf.



Create MTD-C

☐ Use decision probability vector

Missing template search options

Evaluator: [✓ S] P C D E

Width exponent: 1.0

Evaluator exponent: 1.0

Number of iterations: 20

Template density: 0.8

Minimal population size: 20

Intermediate population size: 60

Classifier for sub-tables

Decision tree All decision rules [✓ LEM2 rules]

Required minimal confidence: 0.98

Node size to skip split: 3

Discretization required minimal confidence: 0.98

Required coverage: 0.01

Discretization node size to skip split: 3

Classifier for conflict resolving

Voting [✓ Decision tree]

Required minimal confidence: 0.98

Node size to skip split: 3

MTD-C to: hin_trn

Fig. 3. Configuration dialog of RSES system for Missing Template Decomposition Method.

Let us assume that we would like to construct a MTD using the parameter choice presented in Fig. 3. These settings assume no additional template evaluation function ($S = w(h) \cdot h(t)$), LEM2 [1] rule induction algorithm with required coverage factor of 0.01 (1%) for classifier induction and decision tree used for conflict resolving. After clicking OK the RSES system will induce MTD which which will be accessible via a newly created icon within RSES project interface. By double clicking on the MTD icon the user may open the results window and see the "machinery" for constructed MTD classifier. It is possible to display the set of missing templates as well as induced rule sets and decision tree used for conflict resolving. We can test the accuracy of our MTD model on a test table. The easiest way to do that is by choosing the "Classify/Test table using

[1] Please note, that RSES implementation of LEM2 algorithm does not reflect its current status and is based on description from [13].

MTD-C" option from the context menu associated with the previously loaded test table and selecting the name of MTD to be used. The classification results are then stored in the standard RSES object that makes it possible to examine and analyze classification quality.

In the table below we present results of several experiments carried out on `hin` data with use of various classification algorithms available in the RSES system. These results exemplify usefulness of the MTD, at least for this data set. More results of MTD application, as well as more thorough explanation of underlying algorithms can be found in [7].

Description	Accuracy	Precision	Coverage	Classifier Complexity
All rules on original table	0.535	0.581	0.921	734 rules
LEM2 rules on original table	0.426	0.614	0.693	392 rules
All rules on table imputed with most common value	0.475	0.822	0.578	824 rules
LEM2 rules on table imputed with most common value	0.257	0.650	0.396	323 rules
All rules on table imputed with most common value w.r.t. decision class (*)	0.554	0.622	0.891	898 rules
LEM2 rules on table imputed with most common value w.r.t. decision class	0.268	0.628	0.426	289 rules
All Rules (*) shortened with factor 0.7	0.673	0.708	0.950	259 rules
MTD $q = w \cdot h$, All rules, voting	0.554	0.554	1.000	352 rules/4 classifiers
MTD $q = w \cdot h$, All rules, decision trees	0.663	0.663	1.000	352 rules/4 classifiers + 155 tree nodes
MTD $q = w \cdot h$, LEM2 rules (1%), decision tree	0.782	0.782	1.000	8 rules/4 classifiers + 61 nodes

Acknowledgement

This work is partly supported by grant 3T11C00226 from the Polish Ministry of Education and Science.

References

1. Bazan, J.G., Nguyen, H.S., Nguyen, S.H., 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 (2000) 49–88
2. Bazan, J.G., Szczuka, M.S.: The rough set exploration system. In Peters, J.F., Skowron, A., eds.: Transactions on Rough Sets III. Volume 3400 of Lecture Notes in Computer Science., Springer (2005) 37–56

3. Gediga, G., Düntsch, I.: Maximum consistency of incomplete data via non-invasive imputation. Artificial Intelligence Review **19**(1) (2003) 93–107
4. Grzymała-Busse, J.W., Hu, M.: A comparison of several approaches to missing attribute values in data mining. [15] 378–385
5. Grzymała-Busse, J.W., Wang, A.Y.: Modified algorithms LEM1 and LEM2 for rule induction from data with missing attribute values. In: Proceedings of 5th Workshop on Rough Sets and Soft Computing (RSSC'97) at the 3rd Joint Conference on Information Sciences, Research Triangle Park (NC, USA) (1997) 69–72
6. Kryszkiewicz, M.: Properties of incomplete information systems in the framework of rough sets. [10] 422–450
7. Latkowski, R.: On decomposition for incomplete data. Fundamenta Informaticae **54**(1) (2003) 1–16
8. Lim, T.: Missing covariate values and classification trees. http://www.recursive-partitioning.com/mv.shtml, Recursive-Partitioning.com (2000)
9. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences **11** (1982) 341–356
10. Polkowski, L., Skowron, A., eds.: Rough Sets in Knowledge Discovery 1: Methodology and Applications. Physica-Verlag (1998)
11. Polkowski, L., Skowron, A., eds.: Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems. Physica-Verlag (1998)
12. Skowron, A.: Boolean reasoning for decision rules generation. In Komorowski, H.J., Raś, Z.W., eds.: Methodologies for Intelligent Systems — ISMIS 1993. LNCS 689, Springer (1993) 295–305
13. Stefanowski, J.: On rough set based approaches to induction of decision rules. [10] 500–529
14. Stefanowski, J., Tsoukiàs, A.: Incomplete information tables and rough classification. International Journal of Computational Intelligence **17**(3) (2001) 545–566
15. Ziarko, W., Yao, Y.Y., eds.: Rough Sets and Current Trends in Computing, Second International Conference, RSCTC 2000 Banff, Canada, October 16-19, 2000, Revised Papers. In Ziarko, W., Yao, Y.Y., eds.: RSCTC 2000. LNCS 2005, Springer (2001)
16. The Rough Set Exploration System Homepage
http://logic.mimuw.edu.pl/ rses