URL: http://www.elsevier.nl/locate/entcs/volume82.html 12 pages

Incremental Induction of Decision Rules from Dominance-based Rough Approximations

Jerzy Błaszczyński [1] and Roman Słowiński [1]

Institute of Computing Science, Poznań University of Technology
60-965 Poznań, Poland

Abstract

An incremental algorithm generating satisfactory decision rules and a rule post-processing technique are presented. The rule induction algorithm is based on the Apriori algorithm. It is extended to handle preference-ordered domains of attributes (called criteria) within Variable Consistency Dominance-based Rough Set Approach. It deals, moreover, with the problem of missing values in the data set. The algorithm has been designed for medical applications which require: (i) a careful selection of the set of decision rules representing medical experience and (ii) an easy update of these decision rules because of data set evolving in time, and (iii) not only a high predictive capacity of the set of decision rules but also a thorough explanation of a proposed decision. To satisfy all these requirements, we propose an incremental algorithm for induction of a satisfactory set of decision rules and a post-processing technique on the generated set of rules. User's preferences with respect to attributes are also taken into account. A measure of the quality of a decision rule is proposed. It is used to select the most interesting representatives in the final set of rules.

Key words: Rough set theory, multiple-criteria decision support, knowledge discovery, decision rules, incremental learning.

1 Introduction

The aim of scientific analysis of a decision problem is to answer two basic questions. The first question is to explain decisions in terms of the circumstances in which they were made. The second is to give a recommendation how to make a good decision under specific circumstances. Knowledge discovery techniques aim to induce from data describing decision situations a set of decision rules or decision trees useful for both explanation and recommendation of past and future decisions, respectively.

[1] Email: Roman.Slowinski@cs.put.poznan.pl, Jerzy.Blaszczynski@cs.put.poznan.pl

In this paper we consider decision problems of the classification type, where a set of objects described by a set of regular attributes and criteria is to be assigned to a finite set of preference ordered pre-defined decision classes.

Rough Set Theory introduced by Pawlak [8,9] proved to be suitable mathematical tool to address the problem of inconsistencies in analyzed data sets involving only attributes with no defined preference-order in their domains. An extension of the original Rough Set Theory was made by Greco, Matarazzo and Słowiński [5,7] to deal, moreover, with inconsistencies caused by violation of the dominance principle when the analyzed data set involves criteria and preference-ordered decision classes. The proposed approach was called Dominance-based Rough Set Approach (DRSA). Further extensions were made by the same authors to adapt DRSA to analysis of data containing some missing values and to allow partially inconsistent rough approximations (Variable Consistency Rough Set Approach, VC-DRSA) [6]. The above characteristics of data sets (inconsistencies of both types and missing values) are frequent in practical applications, such as decision support in medicine. This explains our interest in the VC-DRSA methodology.

The decision rules are useful in decision support because unlike, for example, artificial neural networks, they are easily understandable for the DM. In decision support concerning medical problems this feature is particularly important. In these problems, it is essential that the recommendation is followed by a thorough explanation of suggested decisions. Moreover, the specificity of decision support in medicine requires the search of the most interesting representatives in the set of rules.

Another specific feature of medical decision problems is a data set evolving in time due to frequent updates, for example via Internet. These frequent updates require algorithms for incremental induction of decision rules. In this paper, we propose an incremental algorithm updating the rule set when new classification examples are added to the data set. The algorithm is called *DomAprioriUpp*. The rules are induced with restrictions to minimum number of supporting objects, minimum confidence and maximum size.

We also propose a post-processing technique for the generated set of rules. It takes into account user's preferences with respect to attributes and criteria and finds out the best representatives of rules covering the same subset of objects. The set of the best representatives is called reduced set of rules.

Both the reduced set of rules and the original set of rules are used to recommend the assignment of new objects to decision classes. Explanation of the reasons of recommendation is relatively straightforward as the rules are self-explanatory and their support is known (learning examples).

This paper is structured as follows. In the next section a detailed description of *DomAprioriUpp* algorithm is given. In section 3 the rule post-processing technique is presented and some measures of interestingness and usefulness of rules are proposed. Section 4 presents results of a computational experiment. Section 5 groups conclusions.

2 DomAprioriUpp algorithm

DomAprioriUpp is an algorithm inducing decision rules from rough approximations of decision classes defined within Variable Consistency Dominance-based Rough Set Approach (VC-DRSA). The algorithm was inspired by Agrawal et al. [1], Cheung et al. [4] and previous work on *DomApriori* [3]. *Apriori* proposed by Agrawal et al. is a well known procedure for mining association rules. Cheung et al. proposed a general updating technique for maintaining association rules discovered in a data set. The algorithm proposed by Cheung et al. is called *FUP*. *DomApriori* is a version of *Apriori* algorithm adapted to operate in VC-DRSA. The main improvement of *DomAprioriUpp* with respect to *DomApriori* is that it is incremental. This means that it uses information stored during its former runs to increase efficiency of updating the set of rules when new objects are added or removed from the data set.

Although *DomAprioriUpp* extends *DomApriori* by incremental rule induction, it generates the same minimal satisfactory rule set as *DomApriori*. Induction of satisfactory rule set takes into account user's requirements imposed on generated rules. In case of *DomAprioriUpp* these requirements concern: minimum support s by learning examples, minimum confidence c and maximum size max_size. A rule set is minimal if all its members are minimal. Since decision rules are implications, a decision rule is *minimal* if there is no other implication in the rule set with an antecedent of at least the same weakness (i.e. using a subset of selectors or/and weaker selectors) and consequent of at least the same strength (i.e. assigning to the same union of decision classes or to a subset of union of decision classes). Moreover, *DomAprioriUpp* extends *DomApriori* by dealing with missing values. This is achieved by generation of a set of robust rules.

DomAprioriUpp was designed as an algorithm operating on a database. It tends to minimize the number of database scans (i.e. checks of all the objects in the database to determine the support of the conditional part of a rule) and stores information about supports in the database.

In particular, for the conditional part of a rule, the prior support (i.e. support in the database before update) is known or it is certain that this support was not sufficient to make a rule. This knowledge makes it possible to reduce the number of scans of the database. For some of considered conditional parts of a rule, only recomputation of supports on the modified part of the database is needed.

2.1 DomAprioriUpp pseudo-code

Let us recall that in VC-DRSA, the rough approximations and decision rules concern, so called, upward and downward unions of preference-ordered decision classes. An upward union with respect to class Cl_t is a union of classes at least as good as Cl_t, denoted by $Cl_t^{\geq}$. A downward union with respect to class Cl_s is a union of classes at most as good as Cl_s, denoted by $Cl_s^{\leq}$. The conditional

part of a rule (i. e. complex) consists of selectors represented by elementary conditions. For regular attributes without preference order these selectors are in form $f(x,a) = v_a$, where x denotes object and v_a belongs to domain of attribute a. For criteria elementary conditions selectors are in two forms: $f(x,a) \geq v_a$ or $f(x,a) \leq v_a$. Let us also remind that the complex being candidate for a rule is called simply candidate.

We will use the following notation for description of the support of a candidate. The prefixes *pos*, *neg* indicates positive and negative support. Then the symbols r, nr in the subscript indicates *robust* and *non-robust* type of support. The symbol σ points to the support in the whole database D. The symbols following σ are $'$ and $*$ both in superscript. The first denotes support in the updated database. The latter, upper bound of support in the database (calculated for particular candidates). As to symbol δ, it denotes the support in changed part of the database. The added part of the database is denoted by Δ^+, the removed part by Δ^-. Further explanations of the notation are presented in section 2.2

(i) For each lower approximation of an upward or downward union of decision classes perform the following steps:
  (a) Create set of candidates C'_k. Only candidates from L'_{k-1} satisfying condition $\frac{pos\sigma'_{x_k}}{pos\sigma'_{x_k}+neg\sigma'_{x_k}} \neq 1$ are considered as a base for the k-size candidates. Stop if $C'_k = \emptyset$ or $k > max\_size$.
  (b) Partition set C'_k into subsets P_k, S_k, N_k and Q_k. For elements of set S_k calculate their $pos\sigma^*_{x_k}$ support and treat it as $pos\sigma_{x_k}$.
  (c) Calculate $posb^+_{x_k}$ support, for all candidates x_k from P_k, S_k and Q_k.
  (d) Remove from sets P_k, S_k candidates having $pos\sigma_{x_k} + posb^+_{x_k} < |d'| * s$.
  (e) Remove from set Q_k candidates having $posb^+_{x_k} \leq (|\Delta^+| - |\Delta^-|) * s$.
  (f) For each element of sets P_k, S_k and Q_k, calculate $pos\delta^+_{x_k}$ ($pos_r\delta^+_{x_k}$ and $pos_{nr}\delta^+_{x_k}$).
  (g) Remove from set S_k candidates x_k for which $pos_r\sigma_{x_k} < 1 \wedge pos_r\delta^+_{x_k} < 1$.
  (h) For each element of sets P_k, S_k and Q_k, calculate $pos\delta^-_{x_k}$ ($pos_r\delta^-_{x_k}$ and $pos_{nr}\delta^-_{x_k}$).
  (i) Remove from P_k, S_k sets candidates x_k for which $pos\sigma_{x_k} + pos\delta^+_{x_k} - pos\delta^-_{x_k} < |d'| * s$.
  (j) Remove from set Q_k those candidates x_k for which $pos\delta^+_{x_k} - pos\delta^-_{x_k} \leq (|\Delta^+| - |\Delta^-|) * s$.
  (k) Remove those candidates $x_k \in P_k$ for which $pos_r\sigma_{x_k} + pos_r\delta^+_{x_k} - pos_r\delta^-_{x_k} < 1$.
  (l) Scan D^* and get *robust* and *non-robust* supports of each candidate $x_k \in S_k \bigcup Q_k$. Add calculated supports to $pos\delta^+_{x_k}$, $pos_r\delta^+_{x_k}$ to get $pos\sigma'_{x_k}$ i $pos_r\sigma'_{x_k}$.
  (m) Scan D' and get *robust* supports and *non-robust* positive supports $pos_r\sigma'_{x_k}$, $pos_{nr}\sigma'_{x_k}$ of each candidate $x_k \in N_k$.
  (n) Remove candidates $x_k \in S_k \bigcup Q_k \bigcup N_k$, for which $pos_r\sigma'_{x_k} < 1$.

(o) Remove candidates $x_k \in S_k \bigcup Q_k \bigcup N_k$ satisfying condition $pos'\sigma_{x_k} < |d'| * s$.
(p) For each candidate $x_k \in P_k$ calculate $neg\delta_{x_k}$.
(q) For $l = k-1$ remove from set P_k those candidates x_l, for which $\frac{pos\sigma_{x_l}+pos\delta_{x_l}}{pos\sigma_{x_l}+pos\delta_{x_l}+neg\sigma_{x_l}+neg\delta_{x_l}} < |d'| * c$.
(r) For each candidate $x_k \in S_k \bigcup N_k \bigcup Q_k$, scan D' to get $neg\sigma'_{x_k}$ support.
(s) For $l = k-1$ remove from $S_l \bigcup Q_l \bigcup N_l$ those candidates for which $\frac{pos\sigma'_{x_l}}{pos\sigma'_{x_l}+neg\sigma'_{x_l}} < |d'| * c$.
(t) Add to set L'_k candidates x_k remaining in $P_k \bigcup S_k \bigcup N_k \bigcup Q_k$ after the pruning operations.
(u) Check minimality in L'_k.

(ii) Check minimality for all L'_k sets.

2.2 DomAprioriUpp description

DomAprioriUpp algorithm uses the breadth-first search strategy. Sets C_k of candidates x_k for conditional parts of decision rules are created iteratively. The candidate x_k is also called a complex of size k or a list of elementary conditions. Candidates are growing in subsequent iterations (as k increases). They are created separately for each lower approximation of an upward or downward union of decision classes. L'_k denotes a set of large candidates (i.e. lists of elementary conditions) generated during a current iteration. Each new candidate is composed of a candidate from L'_{k-1} and a new added elementary condition. All candidates x_{k-1} from L'_{k-1}, for which $\frac{pos\sigma'_{x_{k-1}}}{pos\sigma'_{x_{k-1}}+neg\sigma'_{x_{k-1}}} \neq 1$ (i.e. their confidence level is lower than 1), are considered as a base for new candidates. Even rules are extended to new candidates if they do not fulfill this condition. Positive support (denoted by pos) stands for a number of objects in the corresponding union of decision classes covered by the complex x_k. Negative support results from covering objects not belonging to the corresponding union of decision classes and is denoted by neg. The symbol σ' used in the formula above indicates that considered support concerns the updated database.

Partition of the candidate set C_k into subsets P_k, S_k, N_k and Q_k takes place in point (b) of the *DomAprioriUpp* algorithm. Set P_k is composed of candidates x_k which were found in the database to be large itemsets (they were members of set L_k in database before update D, thus $pos\sigma_{x_k} \geq d*s$). They also were minimal, robust and had sufficient confidence level. Components of set S_k are candidates that were large but did not meet other necessary conditions to become a rule. Those candidates are found by searching for subcomplexes in L_i, $i = 1, ..., k$, in the database. As it is done in the *DomApriori* algorithm, complexes not being minimal are not extended in future iterations. We can found them only by searching for their subsets in the database. For elements of set S_k, support $pos\sigma^*_{x_k}$ is calculated. It is taken to be the minimal

support of all maximal-length sub-complexes of candidate x_k (it is then an upper bound of candidate's x_k support in D). The candidates belonging to set N_k are build on base of examples that were present in the database before update but were not part of the lower approximation of the considered union of decision classes. The rest of candidates become elements of Q_k. Those candidates were not large in D.

The follo wing lemmas justify thesteps of the algorithm.

Lemma 2.1 *If $x_k \in L_k$ and $pos\sigma_{x_k} + pos\delta^+_{x_k} - pos\delta^-_{x_k} < |d'| * s$, then $x_k \notin L'_k$, thus x_k can b e removed.*

As to notation used in the Lemma, d' denotes the union of decision classes calculated for the database after update. L_k is the set of large candidates in the database before update. The supports $pos\sigma_{x_k}$, $pos\delta^+_{x_k}$ and $pos\delta^-_{x_k}$ denotes support calculated before update of the database and supports calculated for added and removed part of the considered union of decision classes, respectively. The upper bound of support $pos\delta^+_{x_k}$, denoted b y $posb^+_{x_k}$, is used in computations in points (d) and (e). The bound is calculated as minimal support of all sub-complexes of length $k-1$ in the added part of the considered union of decision classes Δ^+ (the support $pos\delta^+_{x_{k-1}}$ was calculated in a prior iteration of the algorithm). This is due to Lemma 2.2.

Lemma 2.2 *F or allsublists (sub-complexes) of candidates x_k and candidates y_i, $i = 1, ..., k$, such that y_i ar e sublists (sub-omplexes) of x_k, which is denoted by $y_i \subset x_k$, $pos\delta^+_{y_i} \geq pos\delta^+_{x_k}$. The same is also true for $pos\delta^-_{x_k}$.*

Lemma 2.3 *If $x_k \notin L_k$ and $pos\delta^+_{x_k} - pos\delta^-_{x_k} \leq (|\Delta^+| - |\Delta^-|) * s$, then $x_k \notin L'_k$.*

In point (e) Lemma 2.3 is used in pruning tests on set Q_k. At this stage of computations, upper bound $posb^+_{x_k}$ can no longer be used. In point (f) the exact support of candidates from all sets (P_k, S_k and Q_k) is evaluated. This support, denoted by $pos\delta^+_{x_k}$, concerns only objects added to the database (part of the database denoted Δ^+). It is divided into two numbers, the *robust* support $pos_r\delta^+_{x_k}$ and the *non-robust* support $pos_{nr}\delta^+_{x_k}$. These supports are needed to distinguish *robust* rules. A *robust* rule has the value of $pos_r\sigma'_{x_k} \geq 1$, $pos_r\sigma'_{x_k}$ indicates the n umber of objects supporting the rule and having no missing values on the attributes corresponding to the ones from its conditional part. Naturally, $pos\delta^+_{x_k} = pos_r\delta^+_{x_k} + pos_{nr}\delta^+_{x_k}$

Next, candidates from set S_k which were not *robust* in D and cannot be *robust* in D' are remov ed (the point (g)). At this point support of complexes in the removed part of database Δ^- is needed. Therefore, in point (h), support $pos\delta^-_{x_k}$ (i.e. positive support in the remov ed part of the considered union of decision classes) is calculated for all candidates remaining in sets P_k, S_k and Q_k. Similarly to point (f), both *robust* and *non-robust* supports are calculated.

In point (i) Lemma 2.1 is applied again on sets P_k and S_k; this time it takes into account newly computed supports $pos\delta^+_{x_k}$ and $pos\delta^-_{x_k}$. Similar steps are followed in point (j).

The removal of candidates from set P_k in point (k) is a result of direct application of the definition of *robust* complexes. It is important to notice that supports of candidates removed in this point are stored in the database in L'_k as they are large complexes. The value of their support $pos_r\sigma'_{x_k}$ indicates that they were removed for being *non-robust*.

The unchanged part D^* of the database is scanned to determine supports $pos\sigma'_{x_k}$ and $pos_r\sigma'_{x_k}$ of all remaining elements of the S_k and the Q_k sets. Naturally , if it occurs that support $pos\sigma^*_{x_k}$ of the candidate x_k from set S_k is known for size k (not sub-complexes of x_k), this support is equivalent to $pos\sigma_{x_k}$. In this case, for those elements of set S_k the scan of D^* is unnecessary. The supports of candidates belonging to set N_k must be evaluated during scan of all objects in the database after update D'. That is done in point (m) of the algorithm.

In point (n) of the algorithm, elements of sets S_k, N_k and Q_k, which are *non-robust*, are stored in the database as elements of L'_k.

At this phase of the decision rules generation process, sets P_k, S_k, N_k and Q_k consist of candidates of length k that were proven to be large and *robust* (i.e. $pos_{nr}\sigma'_{x_k} \geq 1$). Therefore, it remains to check their confidence level. The negative support (i.e. the number of objects supporting x_k and not belonging to the considered union of decision classes) $neg\delta_{x_k}$ of elements of set P_k is calculated. That step requires a scanning of the added part of the database (Δ^+) and the part of database which was removed (Δ^-). In point (p), support $neg\delta_{x_k}$ of the candidates from the other subsets of L_k is evaluated.

In point (q), the candidates from set P_{k-1}, having no sufficient level of confidence are removed. This operation is possible after new candidates are generated. The support of removed candidates is stored in L'_{k-1}.

Analogous steps are taken in point (r). The negative supports $neg\sigma'_{x_k}$ of elements of sets S_k, N_k and Q_k are calculated. Those supports will be necessary in point (s) to remove elements with insufficient confidence level.

In the final stage of computations, precisely in point (t), elements remaining in sets P_k, S_k, N_k and Q_k after all pruning operations are added to set L'_k. Those candidates are then checked for minimality (in points (u) and (ii)). The remaining candidates are rules discovered by the *DomAprioriUpp* algorithm. Candidates removed for being non-minimal, as well as candidates being rules, are stored in the database in $L'k$.

3 Post-processing technique

Reasons for which one rule can be considered as more interesting than others can be inferred from information gathered during classification of new test examples. In that way ,the information concerning the quality of recommendation made by the rules (i.e. quality of prediction) is used to post-process the set of rules constituting the classifier, so that it is becoming an adaptive system in a sense.

Moreover, in some cases the considered regular attributes and/or criteria have a different relevance for the DM. This relevance depends on how much the DM trusts the information supplied by the corresponding attribute (regular attribute or criterion), and on how expensive is this information in terms of money or other consequences (e.g. how much its acquisition is harmful for the patients). In medical classification problems, the objects correspond to patients and attributes usually stand for results of medical examinations. An order of relevance can be defined on the attribute set of that problem. It can represent relative costs of medical examinations. The order can be specified through weights of attributes, denoted by C_A. To determine the weights several techniques can be applied (for example, the playing cards method introduced by Simos [10] for multi-criteria decision problems).

3.1 A measure of the accuracy of prediction for set A of attributes

For a set of regular attributes and criteria A, a measure of the accuracy of prediction by a set of decision rules (classifier) involving these attributes can be calculated upon a number of classification tests. An example of such a measure with respect to a set of regular attributes is the measure proposed by Wróblewski in [11]. Our measure $Pred_A$ extends this proposal on the case of multiple-criteria classification . This measure is defined as a product of the predictive ability $Val(A)$ and cover $Cov(A)$.

$$Pred_A = Val(A) * Cov(A) \tag{1}$$

The predictive ability $Val(A)$ of set A is defined as:

$$Val(A) = \sqrt[n]{\prod_{t=1}^{n-1} \frac{Prop(A, Cl_t^{\leq})}{Prop(A, Cl_t^{\leq}) + Bad(A, Cl_t^{\leq})}} * \sqrt[n]{\prod_{s=2}^{n} \frac{Prop(A, Cl_s^{\geq})}{Prop(A, Cl_s^{\geq}) + Bad(A, Cl_s^{\geq})}} \tag{2}$$

In the above formula, $Prop(A, Cl_t^{\leq})$ denotes the number of objects correctly classified by decision rules using in their condition parts the attributes and criteria from set A, to the downward union of decision classes $Cl_t^{\leq}$. $Prop(A, Cl_s^{\geq})$ is defined analogously. Moreover, $Bad(A, Cl_t^{\leq})$ denotes the number of objects incorrectly classified by the same rules having to the downward union of decision classes $Cl_t^{\leq}$. $Bad(A, Cl_s^{\geq})$ is defined analogously.

The cover $Cov(A)$ of set A is defined as:

$$Cov(A) = \frac{\sum_{t=1}^{n-1} Prop(A, Cl_t^{\leq}) + \sum_{s=2}^{n} Prop(A, Cl_s^{\geq})}{|U|} \tag{3}$$

3.2 A measure of the quality of a decision rule

Basing on the measure of the accuracy of prediction defined (1) for set A of regular attributes and criteria, and on the weights of attributes from set A specified by the DM, the following measure of the quality of a decision rule r can be proposed.

$$W(r) = (1 - C_A) * Pred_A * Pred_r, \tag{4}$$

where $Pred_r$ is defined as follows:

$$Pred_r = \frac{Prop(r)}{Prop(r) + Bad(r)} \tag{5}$$

In the above formula, $Prop(r)$ and $Bad(r)$ stand for the numbers of correctly and incorrectly classified objects by rule r, respectively.

3.3 Post-processing of a set of rules

Application of the rule post-processing technique leads to reduction of the size of the set of rules.

Clusters consisting of rules covering approximately the same subset of objects are evaluated. "Approximately the same" means that at least some p percent of objects covered by the rules creating a cluster are the same.

For each cluster, only the best rule with respect to the quality measure $W(r)$ is taken to the final reduced set of rules.

4 Experiments

Computational experiments have been performed to verify the usefulness of the *DomAprioriUpp* algorithm and the rule post-processing technique.

The artificial data sets generated by computer program were used in these experiments to ensure that instances of the database differing in size will have the same distribution of objects over decision classes. In other words, in the artificial data set, regardless of its size, all the classes contain almost the same number of objects and these objects are well shuffled. The only changing parameter of the data set during the experiment is the size of the instance. This feature is hard to achieve in experiments involving real-life data sets.

In the first group of experiments *DomAprioriUpp* has been compared with non-incremental *DomApriori*. More precisely, the comparison concerned *DomAprioriDB* - a version of *DomApriori* reimplemented to work with a database. The two programs were tested on an artificial data set described by 6 attributes (criteria and regular attributes).

The results of experiments show the effects of adding (figure 1a) and removing (figure 1b) objects from the database. In both experiments the minimum support threshold was set to 85%. When considering addition of objects, for initial instances of the data set, *DomAprioriUpp* is slower than *DomAprioriDB* and then, with subsequent updates of the database it is outperforming

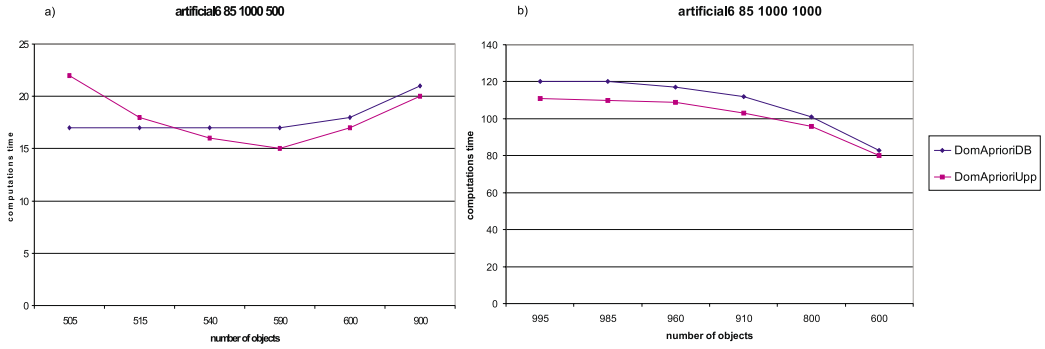

Fig. 1. Experimental comparison of *DomAprioriDB* and *DomAprioriUpp* for adding and removing the same number of objects, and for adding the objects only

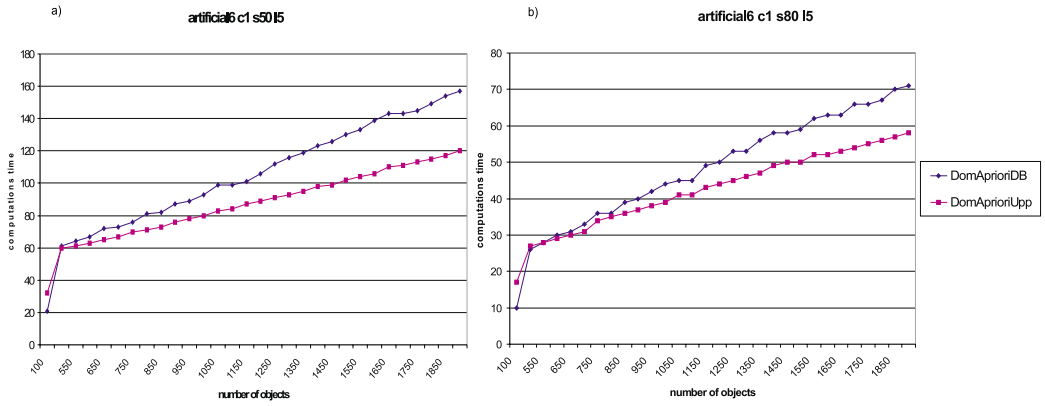

Fig. 2. Experimental comparison of *DomAprioriDB* and *DomAprioriUpp*for minimal support of 50% and 80%, and maximal rule length equal to 5

the referent algorithm. The cause of this effect is the need to perform additional scans of the database to deal with missing values (*DomAprioriDB* is not performing those scans). This effect is also visible in the results of experiments shown on figures 2a,b. The real advantage of *DomAprioriUpp* over *DomAprioriDB* (i.e. the efficiency of algorithm regardless of additional scans needed to handle missing values) is thus hard to determine. The results of experiments with removing objects from the database prove that *DomAprioriUpp* is faster. Figures 2a,b presents additional experiments carried out for larger instances of the data set in order to show the influence of change of the minimal support on the time of computations. In this experiment, the initial number of objects in the database was 100. Then, after generation of rules, 400 objects were added, and the rules was generated once more. Next, in each step, a bundle of 50 objects was added to the database, and rules were generated. The *max_size* parameter was set to 5, the minimum support threshold to 50% and then 80%, respectively (see section 2 for definitions). In both experiments *DomAprioriUpp* outperforms *DomAprioriDB*. Expectedly, the smaller the minimal support of induced rules, the greater the advantage of emphDomAprioriUpp over *DomAprioriDB*.

Data set	Common Cov.%	Avg. N. of rul.	Err. rate	No dec.
Intestine cancer	no post-proc.	132.7	0.29	0.02
	adaptive 10%	36.4	0.18	0.07
	adaptive 60%	65.2	0.27	0.02
	adaptive 90%	80.4	0.29	0.02
Buses	no post-proc.	110.57	0.01	0
	adaptive 10%	110.34	0.03	0
	adaptive 60%	110.34	0.03	0
	adaptive 90%	110.34	0.01	0
Iris plant	no post-proc.	14.6	0.04	0.28
	adaptive 10%	10.3	0.04	0.28
	adaptive 60%	14	0.04	0.28
	adaptive 90%	14.6	0.04	0.28

Table 1
Results of 10-fold cross validation tests with original and reduced sets of rules

The original set of rules and the reduced set of rules were compared in a series of 10-fold cross validation tests. These tests were conducted on modified real-life data sets: *Iris plant* (166 objects, 5 attributes and 3 preference ordered decision classes), technical diagnostics of buses (76 objects, 9 attributes and 3 preference ordered decision classes) and *Intestine cancer* (105 objects, 9 attributes and 5 preference ordered decision classes). The results of 10-fold cross validation tests are presented in table 1. The examined characteristics of the sets of rules included: average number of induced rules (Avg. N. of rul.), average error rate (Err. Rate) and average number of examples for which no decision was found (No dec.). Moreover, the percentage of common cover (Common Cov.) relates to parameter p defined in section 3.3 as the percentage of objects covered by the rules creating a cluster. The results show that application of the rule post-processing technique (with p=90%) leads to sets of rules with at least the same prediction ability as the original set of satisfactory decision rules.

5 Conclusions

In this paper, a new method of incremental rule induction, called *DomAprioriUpp*, and a technique of post-processing of decision rule sets are proposed. For the rule induction algorithm, the key idea is to reduce the number of necessary scans of the database as the process of rule generation is repeated on updated dataset. In case of the rule post-processing technique, the key points are the definitions of measures of aquracy of prediction of attribute set and rule quality. The measure of the accuracy of prediction of attribute set is used to define the measure of the quality of a rule. The measure of quality of a rule is essential in post-processing of rules technique. This technique finds the best representative rule for each cluster of rules covering a similar subset of

objects.

Experiments show that the *DomAprioriUpp* algorithm and the post-processing technique give, in general, better results than their known competitors. *DomAprioriUpp* generates rules faster than *DomApriori* in case of the database evolving in time. The new algorithm deals, moreover, with missing data. The rule post-processing technique discovers the most interesting rules in a larger set of satisfactory rules. The reduced set of rules gives classification results not worse than the original set. Finally, the application of *DomAprioriUpp* and the rule post-processing technique is possible via the Internet in the *MedAssist* system [3].

References

[1] R. Agrawal, H. Manilla, R. Srikant, H. Toivonen, A.I. Verkamo *Fast Discovery of Association Rules* In: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Ulthurusamy *Advanes in Knowledge Discovery and Data Mining*, AAAI Press, 1996

[2] R. Agrawal, T. Imielinski, A. Swami *Mining Association Rules between Sets of Items in Large Databases.* In *Proceedings, ACM SIGMOD Conference on Management of Data*, 207-216. Washington, D.C. 1993

[3] J. Błaszczyński *Internet-based rough set analysis of medical decision problems* Master's thesis, Poznan University of Technology, 2002

[4] D. W. Cheung, S. D. Lee, B. Kao *A General Incremental Technique for Maintaining Discovered Association Rules.* In Proceedings: Database Systems for Advanced Applications, pp. 185-194, 1997

[5] S. Greco, B. Matarazzo, R. Słowiński *Rough sets theory for multicriteria decision analysis.* European Journal of Operational Research 129 (2001) 1-47

[6] S. Greco, B. Matarazzo, R. Słowiński: *Handling missing values in rough set analysis of multi-attribute and multi-criteria decision problems* In: N. Zhong, A. Skowron and S. Ohsuga (eds): *New Directions in Rough Sets, Data Mining and Granular-Soft Computing.* Lecture Notes in Artificial Intelligence vol. 1711, Springer-Verlag, Berlin, 1999, pp. 146-157.

[7] S. Greco, B. Matarazzo, R. Słowiński *A new rough set approach to multicriteria and multiattribute classification.* In: Polkowski L., Skowron A. (eds.) *Rough sets and Current Trends in Computing.* Springer, pp. 60-67, 1998

[8] Z. Pawlak *Rough sets. Theoretical Aspects of Reasoning about Data.* Kluwer, Dordrecht, 1991

[9] Z. Pawlak *Rough sets* International Journal of Information & Computer Science 11 (1982) 341-356

[10] J. Simos *Evaluer L'impact sur l'environement. Une approche originale par l'analyse multicritère et la négotiation* Presses Polytechniques et Universitaires Romandes, Lausanne, 1991

[11] J. Wróblewski *Adaptacyjne metody klasyfikacji obiektów* Ph.D. thesis, Warsaw University, 2002