Pairwise Cores in Information Systems

Jakub Wróblewski

Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland
jakubw@pjwstk.edu.pl

Abstract. A core in information system is a set of attributes globally necessary to distinct objects from different decision classes (i.e., the intersection of all reducts of the information system). A notion of a pairwise core (2-core), which naturally extends the definition of a core into the case of pairs of attributes is presented. Some useful features concerned with the graph representation of pairwise cores are discussed.
The paper presents also practical application of the notion of 2-core. It is known that a core (if exists) may be used to improve the reduct finding methods, since there exist polynomial algorithms for core construction. The same may be proven for a 2-core, which may be also used for estimation of minimal reduct size.

Keywords: reducts, core, pairwise core, reduct finding algorithms.

1 Introduction

Rough set theory [7] provides the tools for extracting knowledge from incomplete data based information. The rough set approximations enable us to describe the decision classes, regarded as the sets of objects satisfying some predefined conditions, by means of indiscernibility relations grouping into classes the objects with the same values of the considered attributes. Rough set expert systems are based on the notion of *reduct* [7] [9], a minimal subset of attributes which is sufficient to discern between objects with different decision values. A set of reducts can be used to generate decision rules, thus the reduct finding algorithms are investigated intensively (cf. [9] [1] [4] [3]). The problem of optimal reducts generation (criteria include reduct length, the number of generated rules etc. [12]) is NP-hard, however, approximate algorithms (like the genetic one [10] [8]) can be used to obtain reducts in reasonable time.

The notion of a *core* of the information system, i.e., the intersection of all reducts can be used in searching for reducts [9] [6] [3]. Our work extends this notion to the *pairwise core* (*2-core*), i.e., the set of pairs of attributes such that at least one of the attribute in a pair is necessary to discern between decision classes. The next sections are devoted to the study of connections between 2-cores and some graph-theoretic features and algorithms, as well as reduct finding algorithms employing the new notions and theorems. Section 5 describes results of experiments on benchmark data sets.

2 Cores and reducts

In rough set theory [7] [9] a sample of data takes the form of an *information system* $\mathbb{A} = (U, A)$, where each attribute $a \in A$ is a function $a : U \to V_a$ into the set V_a of all possible values on a. Reasoning about data can be stated as, e.g., a classification problem, where the values of a distinguished decision attribute are to be predicted under information over conditional attributes. In this case, we consider a triple $\mathbb{A} = (U, A, d)$, called a *decision table*, where, for the decision attribute $d \notin A$, values $v_d \in V_d$ correspond to mutually disjoint decision classes of objects.

Definition 1. *Let $\mathbb{A} = (U, A, d)$, where $A = \{a_1, \ldots, a_n\}$, be given. For any $B \subseteq A$, the* B**-indiscernibility relation** *is the equivalence relation defined by*

$$IND_{\mathbb{A}}(B) = \{(u_1, u_2) \in U \times U : \forall_{a \in B}\ a(u_1) = a(u_2)\} \tag{1}$$

Each $u \in U$ induces a B**-indiscernibility class** *of the form*

$$[u]_B = \{u' \in U : (u, u') \in IND_{\mathbb{A}}(B)\} \tag{2}$$

Definition 2. *The decision table $\mathbb{A} = (U, A, d)$ is* ***consistent*** *iff $IND_{\mathbb{A}}(A) \subseteq IND_{\mathbb{A}}(\{d\})$, i.e., if there are no identical (wrt. A) objects with different decisions.*

We will assume further that our decision tables are consistent. Most of the features and notions described in this paper hold also for the case of non-decision problems (i.e., for information systems and general, not decision reducts) as well as for inconsistent tables (using a generalized decisions [9]); however, for the sake of simplicity of notation and proofs we will restrict ourselves to the less general case.

Indiscernibility relation enables us to express global dependencies as follows:

Definition 3. *Given $\mathbb{A} = (U, A, d)$, we say that $B \subseteq A$* **defines** d **in** $\mathbb{A}$ *iff*

$$IND_{\mathbb{A}}(B) \subseteq IND_{\mathbb{A}}(\{d\}) \tag{3}$$

or, equivalently:

$$\forall_{u_1, u_2 \in U}\ d(u_1) \neq d(u_2) \Longrightarrow \exists_{a \in B} a(u_1) \neq a(u_2) \tag{4}$$

Assume that $\mathbb{A}$ is consistent. We say that $B \subseteq A$ is a **decision reduct** *iff it defines d and none of its proper subsets does it. By $\mathcal{R}_{\mathbb{A}}$ we will denote the set of all decision reducts of $\mathbb{A}$.*

The following property is widely utilized by many reduct finding algorithms [1]:

Lemma 1. *Suppose that $B \subseteq A$ defines d (note that B does not need to be a reduct, because a reduct has to be locally minimal). There exists $B' \subseteq B$ such that B' is a reduct.*

Proof. If B is a reduct itself, then $B' = B$. If $B = \emptyset$ then B is a reduct (because it is locally minimal).

Suppose that $B \neq \emptyset$ and B is not a reduct. In this case there exists $B'' \subset B$ such that B'' defines d. In this case there are two possibilities:

- *B'' is a reduct (thus $B' = B''$) and the procedure stops,*
- *B'' is not a reduct (it is not minimal), thus the same procedure of reduction may be applied to B''.*

Since B is finite and each step of the above procedure will decrease the cardinality of involved subsets, the procedure of reduction must stop.

□

A reduct denotes a minimal set of attributes which are sufficient to define (e.g. using decision rules) the value of decisions for all objects. On the other hand, a *core* is used to denote attributes which are necessary for that.

Definition 4. *Let $\mathbb{A}$ and the family of its decision reducts $\mathcal{R}_{\mathbb{A}}$ be given. By a* ***core*** *$C \subseteq A$ of the decision table $\mathbb{A}$ we will denote:*

$$C = \bigcap_{R \in \mathcal{R}_{\mathbb{A}}} R \tag{5}$$

i.e., the intersection of all reducts. (To avoid confusion with the further notation, the core will be sometimes denoted by ***1-core****).*

The core may be empty. In fact, for many of real-world data it is empty (see Table 2). It is interesting to see that, although the problem of minimal reduct finding is NP-hard [9] and $|\mathcal{R}_{\mathbb{A}}|$ may be exponential, the core C is easy to be found.

Lemma 2. *For a consistent decision table $\mathbb{A}$ and its core C the following condition holds:*

$$a \in C \iff IND_{\mathbb{A}}(A \setminus \{a\}) \not\subseteq IND_{\mathbb{A}}(\{d\}) \tag{6}$$

i.e., C is a set of all attributes which are indispensable for discernibility of decision classes in $\mathbb{A}$.

Proof. Let $a \in C = \bigcap_{R \in \mathcal{R}_{\mathbb{A}}} R$. Suppose that $IND_{\mathbb{A}}(A \setminus \{a\}) \subseteq IND_{\mathbb{A}}(\{d\})$, i.e., $A \setminus \{a\}$ defines d. Thus, according to Lemma 1, there exists a reduct $R \subseteq A \setminus \{a\}$. This contradicts the assumption that a belongs to any reduct of $\mathbb{A}$.

What is left is to show that if $A \setminus \{a\}$ does not define d, then $a \in C$. It is evident, as (by definition) in this case neither $A \setminus \{a\}$ nor any of its subset may be a reduct, thus any $R \in \mathcal{R}_{\mathbb{A}}$ must contain a.

□

Let us extend the notion of a 1-core as follows:

Definition 5. *By the* ***pairwise core*** *or* ***2-core*** *of a decision table $\mathbb{A}$, denoted by C_p, we mean a set of (unordered) pairs of attributes $C_p \subseteq A \times A$ such that:*

1. $\{a_1, a_2\} \in C_p \Longrightarrow a_1 \notin C \wedge a_2 \notin C$, *where C is a core of $\mathbb{A}$,*
2. $\{a_1, a_2\} \in C_p \Longrightarrow \forall_{R \in \mathcal{R}_\mathbb{A}}\, a_1 \in R \vee a_2 \in R$,
3. *C_p is the maximal set satisfying the above conditions.*

The pairwise core is a set of pairs $\{a_1, a_2\}$ of attributes such that neither a_1 nor a_2 belongs to a core, but at least one of them is required for discernibility between decision classes (i.e., any reduct of $\mathbb{A}$ must contain at least one attribute from every pair).

Such notions as reducts or cores are often defined using *discernibility matrix* of decision table [9], i.e., the matrix of size $|U| \times |U|$ which for any pair of objects u_1, u_2 (from different decision classes) contains a set of attributes discerning u_1 and u_2. In this formulation a 1-core may be found by collecting all attributes present in the discernibility matrix as singletons. On the other hand, a 2-core is a collection of all two-attribute elements of the matrix (not containing core attributes).

As a natural extension of 1-core (i.e., a classical core) and 2-core one may define a notion of k-core, i.e., the set of k-element subsets of attributes, which do not contain elements of any lower l-core ($l < k$) and every reduct must contain at least one attribute from every element of the k-core. Alternatively, the k-core contain all k-element cells of indiscernibility table (not covered by any l-core, $l < k$). We will not consider k-cores for $k > 2$ because of two reasons: first, the most of the results presented in the next sections are not easily extendable to the k-cores, and second, the time of computation of k-core (see below) rises significantly.

As stated above, the problem of reduct finding is generally hard. In contrast, both core and 2-core can be calculated quickly, i.e., in polynomial time wrt. both the number of attributes and objects:

Fact 1 *(algorithm for finding cores). The following procedure may be applied to find both a core C and a pairwise core C_p of a consistent decision table $\mathbb{A} = (U, A, d)$:*

```
1. $C := \emptyset$, $C_p := \emptyset$
2. For all $a \in A$ do
3.         $R := A \setminus \{a\}$
4.         If $R$ does not define $d$, then $C := C \cup \{a\}$
5. End For
6. For all $a_i, a_j \in A \setminus C$, $i < j$, do
7.         $R := A \setminus \{a_i, a_j\}$
8.         If $R$ does not define $d$, then $C_p := C_p \cup \{\{a_i, a_j\}\}$
9. End For.
```

To determine whether R defines d or not, one may use the following procedure, based on an algorithm for reduct finding [10] [1]:

1. *Sort the set of objects U according to values of attributes from R.*
2. *Determine the indiscernibility classes $[u]_R$, $u \in U$ (linear scan, as U is sorted).*
3. *If all $[u]_R$ have uniform values of decision d, then R defines d.*

Alternatively, the discernibility matrix can be used to obtain both 1-core and 2-core (refer to the discussion above). But the time and space complexity of indiscernibility matrix calculation is $O(m \times n^2)$ where $m = |A|$, $n = |U|$. For a large data set the factor of n^2 is too high. On the other hand, the algorithm described above is $O(m^2 \times n \log n)$ in a case of efficient implementation [1], which is much more acceptable. Note that the procedure may be easily extended to k-cores, but it will have the complexity of $O(m^k \times n \log n)$.

3 Pairwise core graph

The following notion allows us to use wide range of graph-theoretic tools and heuristics for efficient reduct finding:

Definition 6. *The **pairwise core graph** for a consistent decision table $\mathbb{A} = (U, A, d)$ is an undirected graph $G_{\mathbb{A}} = (V, E)$ such that:*

$$V = A, \qquad E = C_p$$

where C_p is a 2-core for $\mathbb{A}$.

The notion of pairwise core graph allows us to express connections between reducts and 2-cores as some graph-theoretic features of $G_{\mathbb{A}}$. One of the most interesting results is concerned with the *vertex covering* of graph $G_{\mathbb{A}}$ (i.e., finding a subset of vertices of $G_{\mathbb{A}}$ such that every edge connects at least one vertex).

Theorem 1. *Suppose that $G_{\mathbb{A}}$ is a pairwise core graph for a consistent decision table $\mathbb{A}$. Let $R \in \mathcal{R}_{\mathbb{A}}$ be a reduct. Then R is a vertex cover of $G_{\mathbb{A}}$.*

Proof. The proof is straightforward from Definitions 5 and 6, since for every edge (a_i, a_j) at least one of the attributes a_i, a_j must be contained by R.

$\square$

Unfortunately, the opposite property does not hold, i.e., not all vertex covers (even minimal ones) are reducts. The simplest example is a decision table for which $C_p = \emptyset$ thus the minimal cover is empty, whereas the table may still have nonempty reducts. Moreover, if $B \subseteq A$ is a vertex cover of $G_{\mathbb{A}}$, then B is not necessarily a subset of any reduct. For the decision table $\mathbb{A}$ presented below, we have $C_p = \{\{a_1, a_2\}, \{a_3, a_4\}, \{a_5, a_6\}\}$ and $B = \{a_1, a_3, a_5\}$; as we can see, B is a minimal vertex cover and is not a reduct (nor a subset of any of the 7 reducts of $\mathbb{A}$):

a_1	a_2	a_3	a_4	a_5	a_6	d
0	0	0	0	0	0	0
1	1	0	0	0	0	1
0	0	1	1	0	0	1
0	0	0	0	1	1	1
0	2	0	2	0	2	2

The following properties will be useful for reduct finding task:

Corollary 1. *Suppose that $G_{\mathbb{A}}$ contains a clique $K = \{a_{k_1}, ..., a_{k_i}\}$. Then any reduct $R \in \mathcal{R}_{\mathbb{A}}$ must contain at least $i - 1$ attributes from K.*

Corollary 2. *Suppose that $G_{\mathbb{A}}$ is a full graph: $G_{\mathbb{A}} = K_n$ where $n = |A|$. Then:*

$$\mathcal{R}_{\mathbb{A}} = \{A \setminus \{a_i\}, \quad i = 1, ..., n\}$$

Proof. It is immediate from Corollary 1. Note that the set A cannot be a reduct itself (because in this case $C = A$, thus $C_p = \emptyset$) and the only possible sets containing at least $n - 1$ attributes are defined as above.

□

Note that Theorem 1 provides a lower bound on the reduct size: every reduct must be at least as large as the minimal vertex covering of $G_{\mathbb{A}}$. Fortunately, we do not need to find the minimal covering (which is NP-hard [2]) to obtain a weaker lower bound:

Corollary 3. *Suppose that $G_{\mathbb{A}}$ may be divided into a set of disjoint cliques K_1, ... K_i. Let R be an arbitrary reduct of $\mathbb{A}$. Then:*

$$|R| \geq (|K_1| - 1) + \cdots + (|K_i| - 1) \tag{7}$$

It is interesting to see that a structure of 2-core may be very rich:

Theorem 2. *Let G be an arbitrary graph, $|E| \geq 1$. Then, there exists $\mathbb{A}$ such that $G_{\mathbb{A}} = G$.*

Proof. Consider $G = (V, E)$. Let $\mathbb{A} = (U, A, d)$ be defined such that $A = V$, $a : U \to \{0, 1\}$ for each $a \in A$, and $U = \{u_0, u_1, ..., u_k\}$ for $k = |E|$, and:

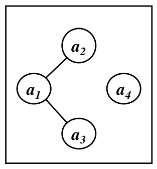
$$\begin{array}{ll} \forall_i & d(u_0) = 0 \wedge a_i(u_0) = 0 \\ \forall_{e_j \in E} & d(u_j) = 1 \wedge a_i(u_j) = \begin{cases} 1 & \text{where } e_j = (a_{k_1}, a_{k_2}),\ i = k_1 \vee i = k_2 \\ 0 & \text{otherwise} \end{cases} \end{array}$$

In this decision table the 1-core is empty, because the decision table have exactly two "1"-s in each row u_j, $j > 0$ and thus reduction of any single attribute is not enough to make u_0 and u_j indiscernible.

On the other hand, any reduct R of $\mathbb{A}$ must contain at least one attribute for all edges $e_j = (a_{k_1}, a_{k_2})$. If not, a pair $\{u_0, u_j\}$ (having opposite decision values) will be indiscernible by R.

□

An example presented in Table 1 illustrates the proof of Theorem 2.



$\longrightarrow$

	a_1	a_2	a_3	a_4	d
u_0	0	0	0	0	0
u_1	1	1	0	0	1
u_2	1	0	1	0	1

Table 1. Example of a graph G and a decision table $\mathbb{A}$ such that $G_{\mathbb{A}} = G$.

4 Application of the pairwise cores for reducts finding

Finding a 1-core or a 2-core in an information system may lead to two kinds of profits. Firstly, an information about importance of attributes and their influence into decision value is stated, which may be interesting in descriptive data analysis. Secondly, due to the properties presented in Section 3, the 2-core graph may be very helpful in reduct finding task.

Let $\mathbb{A}$ be given, let C and C_p will be a 1-core and a 2-core of $\mathbb{A}$, respectively. Suppose we have a method for finding all reducts being supersets of a set $B \subseteq A$. One may use e.g. techniques adopted from [1] (i.e., reducing a subset of attributes until a reduct is found, omitting attributes from B), or Apriori-like algorithm [4] for all reducts finding. The following hints may be used for finding all reducts of $\mathbb{A}$:

1. Let $B = C$, since all reducts must contain the core.
2. Divide $G_{\mathbb{A}}$ into a family of disjoint cliques K_1, ... K_m.
3. For each K_i omit exactly one attribute and add the rest of them into B. Then find all reducts being supersets of B.
4. Cycle through the above steps checking all possible combinations of omitted attributes.

Any reduct of $\mathbb{A}$ must contain at least $|K_i| - 1$ attributes from every clique K_i (Corollary 1). Thus, all reducts may be found by checking all supersets of sets generated by the above procedure.

The procedure in the worst case is exponential (as the number of reducts may be exponential, and the problem of finding maximal clique is NP-hard, and the number of all possible combinations checked in step 4 may be exponential). Nevertheless, the procedure may be treated as a heuristics which may highly restrict the search space. The main advantage of the pairwise core graphs is that they are in most cases considerably small. Even for large data sets (see Section 5) these graphs can be analyzed exhaustively. For larger graphs one may use

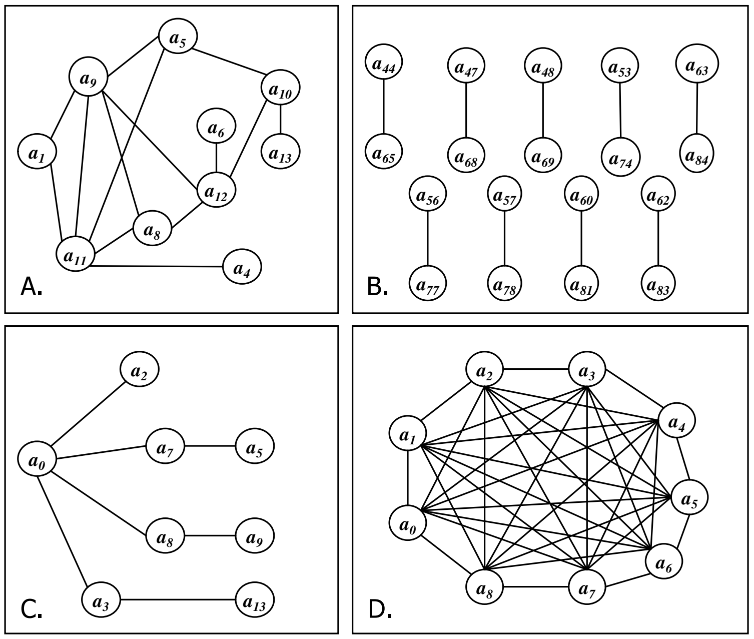


Fig. 1. The pairwise core graphs for benchmark data sets (numbering of attributes starts with 0). A. `Letter`, B. `CoIL'2000`, C. `Soybean`, D. `Tic-Tac-Toe`.

e.g. greedy heuristics for decomposition into cliques: find the largest clique (by selecing a vertex with maximal rank and finding a clique around it), then remove it from the graph and iterate these steps until there is no clique (even a single edge) left. This heuristics may be extended in many ways, e.g. by randomization.

Another direction of searching for all reducts is to generate all locally minimal vertex covers of $G_{\mathbb{A}}$ and to analyze theirs supersets. This algorithm may be harder to use as it is much easier to find one decomposition into cliques than all minimal vertex covers.

5 Experimental results

Several data sets were tested to find a 2-core and $G_{\mathbb{A}}$ (see Table 2). We have selected these decision tables because they are widely used as machine learning benchmarks [5] and in most cases they contain mainly nominal (not numerical) attributes which is preferable for reduct-based analysis of data. One of these tables may be regarded as a relatively large one (`Covtype`). Results presented in Table 2 show, that more than a half of these tables have nonempty 2-cores, whereas in only 5 out of 14 cases a 1-core was nonempty. It means that the notion of 2-core may be virtually more helpful for reduct finding tasks than the 1-core. For the largest decision table `Covtype` we found its 2-core in about 6 minutes (Celeron 1 GHz machine) which is still acceptable.

The variety of structures of pairwise core graphs of these decision tables is surprisingly rich, what can be seen in Figure 1. Note that these vertices, which are not present in any 2-core pair, were omitted in Figure 1.

Finally, the possibility of estimating the size of minimal reduct in the tables was analyzed. Note that even for very large data sets their pairwise core graphs are often relatively simple. For all analyzed data sets we may quickly perform decomposition into disjoint cliques (even manually) and obtain a lower bound of minimal reduct size due to Corollary 2. The results of these estimations are presented in Table 2. For `Tic-Tac-Toe` data set we have obtained a full graph, so we apply Corollary 2 directly and obtain the set of all reducts immediately. For the rest of data sets, we obtained a lower bound which in some cases (`CoIL 2000`, `Letter`, `Soybean`) are not far from actual minimal reduct size.

The usefulness of the pairwise core graphs may be justified by the following experiment. Suppose we have to check whether known 4-attribute reduct of `Covtype` decision table is the minimal one. The straightforward method is to check all 3-attribute subsets, i.e., to perform 24804 sortings of more than half million objects of the table [1]. On the other hand, the pairwise core for this data set is $C_p = \{\{a_0, a_5\}, \{a_0, a_9\}, \{a_4, a_9\}, \{a_5, a_9\}\}$. It means that it is enough to check only 3-attribute supersets of all possible vertex covers of pairwise core graph, i.e., 52 supersets of $\{a_0, a_9\}$, 52 supersets of $\{a_5, a_9\}$ and the set $\{a_0, a_4, a_5\}$. We perform 105 sortings, which is 236 times faster than without use of the pairwise core graph.

6 Conclusions

The notion of pairwse core (2-core) was introduced and discussed. This notion may be interesting and helpful for reduct finding. One of the most interesting results of this paper is Theorem 1 and its corollaries concerning connections between reducts of decision tables and vertex covers of particular graph. The size of minimal reduct may also be estimated due to Corollary 3.

The efficient algorithms for generating all reducts needs further research. The algorithms may use a wide set of heuristics designed for clique finding and graph covering, adopted to the special case of reduct finding. Results of experiments on a set of benchmark tables (presented in Section 5) are very promising.

Acknowledgements. Supported by the Research Center of the Polish-Japanese Institute of Information Technology.

Data set	Size (obj.×attr.)	Size of C	Size of C_p	Min. reduct estim./known
Australian credit	690×14	1	0	1/3
CoIL 2000	5822×85	9	9	18/22
Optical digits	3823×64	0	0	
Pen-based digits	7494×17	0	0	
DNA splices	2000×180	0	0	
German credit	1000×24	0	1	1/5
Letter	15000×16	3	14	7/10
Pima	768×8	0	0	
Shuttle	43500×9	1	0	1/4
Covtype	581012×54	0	4	2/4
Soybean	307×35	2	7	6/9
Tic-tac-toe	958×9	0	36	**8/8**
Mushroom	8124×22	0	1	1/4
Lymnography	148×18	0	1	1/6

Table 2. Experimental results: benchmark tables, the size of the core and the pairwise core, the estimated (basing on a core and 2-core graph) minimal size of reducts and actual minimal (or minimal known) size of reducts.

References

1. Bazan J., Nguyen H.S., Nguyen S.H., Synak P., Wróblewski J.: Rough Set Algorithms in Classification Problem. In: L. Polkowski, S. Tsumoto, T.Y. Lin (ed.): Rough Set Methods and Applications. Physica-Verlag, Heidelberg, New York (2000) 49–88.
2. Garey M.R., Johnson D.S.: Computers and Intractability, a Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco (1979).
3. Hu X., Lin T. Y., Han J.: A New Rough Sets Model Based on Database Systems. Fundamenta Informaticae **59** (2,3), IOS Press (2004) 135–152.
4. Kryszkiewicz M., Cichon K.: Towards Scalable Algorithms for Discovering Rough Set Reducts. In: J.F. Peters et al. (eds.), Transaction on Rough Sets vol. I. Springer (LNCS 3100), Berlin, Heidelberg (2004) 120–143.
5. UCI Machine Learning Rep.: http://www.ics.uci.edu/∼mlearn/MLRepository.html.
6. Miao D., Hou L.: An Application of Rough Sets to Monk's Problems Solving. Proc. of the RSFDGrC2003, Springer (LNAI 2639), Berlin, Heidelberg (2003) 138– 145.
7. Pawlak, Z.: Rough sets – Theoretical aspects of reasoning about data. Kluwer Academic Publishers, Dordrecht (1991).
8. RSES – Rough Set Exploration System: http://logic.mimuw.edu.pl/∼rses/
9. Skowron A., Rauszer C.: The discernibility matrices and functions in information systems. W: R. Slowinski (ed.). Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory. Kluwer, Dordrecht (1992) 311–362.
10. Wróblewski J.: Finding minimal reducts using genetic algorithms. Proc. of the Second Annual Join Conference on Information Sciences, Wrightsville Beach, NC (1995) 186–189, http://www.mimuw.edu.pl/∼jakubw/bib/

11. Wróblewski J.: Covering with reducts – a fast algorithm for rule generation. Proc. of RSCTC'98, Warsaw, Poland. Springer-Verlag (LNAI 1424), Berlin Heidelberg (1998) 402–407, http://www.mimuw.edu.pl/∼jakubw/bib/
12. Wróblewski J.: Ensembles of classifiers based on approximate reducts. Fundamenta Informaticae **47** (3,4), IOS Press (2001) 351–360.