Chapter 2

Rough Set Algorithms in Classification Problem

Jan G. Bazan[1], *Hung Son Nguyen*[2,3], *Sinh Hoa Nguyen*[2,3]
Piotr Synak[3], *Jakub Wróblewski*[3,2]

[1] Institute of Mathematics, Pedagogical University of Rzeszów
Rejtana 16A, 35-310 Rzeszów, Poland
[2] Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
[3] Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland

Abstract: In this Chapter we present some algorithms, based on rough set theory, that can be used for the problem of new cases classification. Most of the algorithms were implemented and included in Rosetta system [43]. We present several methods for computation of decision rules based on reducts. We discuss the problem of real value attribute discretization for increasing the performance of algorithms and quality of decision rules. Finally we deal with a problem of resolving conflicts between decision rules classifying a new case to different categories (classes).

Keywords: knowledge discovery, rough sets, classification algorithms, reducts, decision rules, real value attribute discretization

1 Introduction

The term *"classification"* concerns any context in which some decision is taken or a forecast is made on the basis of currently available knowledge or information. A *classification algorithm* is an algorithm which permits us to repeatedly make a forecast on the basis of accumulated knowledge in new situations. We consider here a classification related to construction of a classification algorithm on the basis of current knowledge. Such algorithm is applied then to classify objects previously unseen. Each new object is assigned to a class belonging to a predefined set of classes on the basis of observed values of suitably chosen attributes (features).

Many approaches have been proposed for constructing classification algorithms, among them we would like to point out classical and modern statistical techniques (see e.g. [33], [20]), neural networks (see e.g. [33], [19]), decision trees (see e.g. [13], [48], [58], [60], [33], [49]), decision rules (see e.g. [14], [31], [59], [10], [32], [61], [22], [55], [34], [43]), inductive logic programming (see e.g. [17]).

In this Chapter we present some methods for extracting laws from data, based on rough set approach (see [44]) and Boolean reasoning (see [11]). Most of them were implemented and included in Rosetta system [43]. Results of performed

experiments on different kinds of data show that they are very promising (see e.g. [3], [4], [5], [7], [8]).

Standard rough set methods (see [44], [45], [52]) are not always sufficient for extracting laws from data. One of the reasons is that these methods are not taking into account the fact that part of the reduct set (see Section 2) is chaotic i.e. is not stable in randomly chosen samples of a given decision table. We propose a method for selection of feature (attribute) sets relevant for extracting laws from data. These sets of attributes are called dynamic reducts (see [4]). Dynamic reducts are in some sense the most stable reducts of a given decision table, i.e. they are the most frequently appearing reducts in subtables created by random samples of a given decision table.

The most popular method for classification algorithms construction is based on learning rules from examples. The methods based on calculation of all reducts allow to compute, for a given data, the descriptions of concepts by means of decision rules (see [45], [44]). Unfortunately, the decision rules constructed in this way can often be not appropriate to classify unseen cases. We propose a method of the decision rule generation on the basis of dynamic reducts. We suggest that the rules calculated by means of dynamic reducts are better predisposed to classify unseen cases (see [3], [4], [5], [6]).

The Chapter is structured as follows. In Section 2 we present rough set preliminaries. Standard rough set algorithms for synthesis of decision rules from decision tables are described in Section 3.

For purpose of data preprocessing we propose methods for intelligent scaling of real value attributes (see Section 4). Our algorithms are especially useful if the input data table contains continues attributes with many different values. Applying discretization to data decreases further processing time of many methods as well as increases quality of results, e.g. decision rules containing scaled attributes are more general.

Our methods for decision rule generation are based on algorithm for the reduct set computation (see Section 2). Unfortunately, the searching problem for reduct of minimal length (minimal number of attributes) is *NP*-hard (see [51]). Therefore we often apply approximation algorithms to obtain some knowledge about the reduct set (see [38], [62]). In Section 5 we propose some heuristic for finding local reducts. In Section 6 we present a genetic algorithm for reduct set computation which is very fast and gives its good approximation.

In Sections 7, 8, 9, and 10 we recall the notion of a dynamic reduct and we give some statistical arguments showing that dynamic reducts offer a good tool for extracting laws from decision table.

Some applications of dynamic reducts, e.g. for decision rules generation and dynamic selection of cuts in discretization, are presented in Sections 4, 11 and 12.

In Section 13 we show how the idea of dynamic reducts can be adapted for new methods of dynamic rules computation.

Sometimes the decision rules generated by application of rough set methods can be not acceptable as laws valid for data encoded in a given decision table. This occurs, e.g. when the number of examples supporting the decision rule (see

Section 3) is relatively small. In Section 14 we introduce *approximate rules* to eliminate this drawback. Different methods (e.g. [1], [35], [46], [67]) are now widely used to generate approximate decision rules.

When a set of decision rules has been computed then it is necessary to decide how to resolve conflicts between rule sets classifying tested objects to different decision classes. In Section 15 we present several measures of *the strength of rule set* matched by a given tested object and classifying this object to decision determined by the rules from this set.

In Section 16 we present a general scheme of classification algorithms based on methods and techniques described in previous sections.

2 Rough Set Preliminaries

In this section we recall some basic notions related to information systems and rough sets.

An *information system* is a pair $\mathbf{A} = (U, A)$, where U is a non-empty, finite set called the *universe* and A – a non-empty, finite set of *attributes*, i.e. $a : U \to V_a$ for $a \in A$, where V_a is called the *value set* of a.

Elements of U are called *objects* and interpreted as, e.g. cases, states, processes, patients, observations. Attributes are interpreted as features, variables, characteristic conditions etc.

We also consider a special case of information systems called decision tables. A *decision table* is an information system of the form $\mathbf{A} = (U, A \cup \{d\})$, where $d \notin A$ is a distinguished attribute called *decision*. The elements of A are called *conditions*.

One can interpret the decision attribute as a kind of classifier on the universe of objects given by an expert, a decision-maker, an operator, a physician, etc. In machine learning decision tables are called training sets of examples (see [28]).

The cardinality of the image $d(U) = \{k : d(s) = k \text{ for some } s \in U\}$ is called the *rank of* d and is denoted by $r(d)$.

We assume that the set V_d of values of the decision d is equal to $\{v_d^1, ..., v_d^{r(d)}\}$.

Let us observe that the decision d determines a partition $CLASS_{\mathbf{A}}(d) = \{X_{\mathbf{A}}^1, \ldots, X_{\mathbf{A}}^{r(d)}\}$ of the universe U, where $X_{\mathbf{A}}^k = \{x \in U : d(x) = v_d^k\}$ for $1 \leq k \leq r(d)$. $CLASS_{\mathbf{A}}(d)$ is called the *classification of objects in* $\mathbf{A}$ *determined by the decision* d. The set $X_{\mathbf{A}}^i$ is called the *i-th decision class of* $\mathbf{A}$. By $X_{\mathbf{A}}(u)$ we denote the decision class $\{x \in U : d(x) = d(u)\}$, for any $u \in U$.

Let $\mathbf{A} = (U, A)$ be an information system. For every set of attributes $B \subseteq A$, an equivalence relation, denoted by $IND_{\mathbf{A}}(B)$ and called the *B-indiscernibility relation*, is defined by

$$IND_{\mathbf{A}}(B) = \{(u, u') \in U^2 : \text{ for every } a \in B, a(u) = a(u')\} \tag{1}$$

Objects u, u' satisfying the relation $IND_{\mathbf{A}}(B)$ are indiscernible by attributes from B.

An attribute $a \in B \subseteq A$ is *dispensable* in B if $IND_{\mathbf{A}}(B) = IND_{\mathbf{A}}(B \setminus \{a\})$, otherwise a is *indispensable* in B. A set $B \subseteq A$ is *independent* in $\mathbf{A}$ if every

attribute from B is indispensable in B, otherwise the set B is *dependent* in $\mathbf{A}$. A set $B \subseteq A$ is called *a reduct* in $\mathbf{A}$ if B is independent in $\mathbf{A}$ and $IND_{\mathbf{A}}(B) = IND_{\mathbf{A}}(A)$. The set of all reducts in $\mathbf{A}$ is denoted by $RED(\mathbf{A})$. This is a classical notion of reduct and it is sometimes referred to as *global reduct*.

Let $\mathbf{A} = (U, A)$ be an information system with n objects. By $M(\mathbf{A})$ (see [51]) we denote an $n \times n$ matrix (c_{ij}), called the *discernibility matrix* of $\mathbf{A}$ such that

$$c_{ij} = \{a \in A : a(x_i) \neq a(x_j)\} \quad \text{for} \quad i, j = 1, \ldots, n\,. \tag{2}$$

A *discernibility function* $f_{\mathbf{A}}$ for an information system $\mathbf{A}$ is a boolean function of m boolean variables $\overline{a}_1, \ldots, \overline{a}_m$ corresponding to the attributes $a_1, \ldots, a_m$ respectively, and defined by

$$f_{\mathbf{A}}(\overline{a}_1, \ldots, \overline{a}_m) = \bigwedge \{ \bigvee \overline{c}_{ij} : 1 \leq j < i \leq n, c_{ij} \neq \emptyset \} \tag{3}$$

where $\overline{c}_{ij} = \{\overline{a} : a \in c_{ij}\}$.

It can be shown (see [51]) that the set of all *prime implicants* of $f_{\mathbf{A}}$ determines the set of all *reducts* of A.

Below we present two deterministic algorithms (see [15]) for computation of the whole reduct set $RED_{\mathbf{A}}(A)$. Both algorithms compute discernibility matrix of $\mathbf{A}$.

Algorithm 1. *Reduct set computation*
Input:
Information system $\mathbf{A} = (U, A)$
Output:
Set $RED_{\mathbf{A}}(A)$ *of all reducts of* $\mathbf{A}$
Method:
Compute indiscernibility matrix $M(\mathbf{A}) = (C_{ij})$
Reduce M *using absorbtion laws*
d *- number of non-empty fields* $C_1, C_2, .., C_d$ *of reduced* M
Build families of sets $R_0, R_1, ..., R_d$ *in the following way:*
begin
$R_0 = \emptyset$
for $i = 1$ **to** d
begin
$R_i = S_i \cup T_i$, *where* $S_i = \{R \in R_{i-1} : R \cap C_i \neq \emptyset\}$
and $T_i = (R \cup \{a\})_{a \in C_i, R \in R_{i-1} : R \cap C_i = \emptyset}$
end
end
Remove dispensable attributes from each element of family R_d
Remove redundant elements from R_d
$RED_{\mathbf{A}}(A) = R_d$
□

It is easy to see that the complexity of this algorithm is exponential. The second algorithm is a modification of Algorithm 1. It allows to stop computation and get a partially computed reduct set.

Algorithm 2. *Reduct set computation with stop possibility*

Input:

Information system $\mathbf{A} = (U, A)$

Output:

Set $RED_{\mathbf{A}}(A)$ *of all reducts of* $\mathbf{A}$

Method:

Compute indiscernibility matrix $M(\mathbf{A}) = (C_{ij})$
Reduce M *using absorbtion laws*
d *- number of non-empty fields of reduced* M
Build a families of sets $R_0, R_1, ..., R_d$ *in the following way:*

```
begin
    Compute R_1 (see Algorithm 1)
    i = 1
    while i > 0 do
    begin
        if stop then return
        if R_i = ∅ then
        begin
            i = i − 1
            continue
        end
        Remove from family R_i the first element
        Compute R_{i+1} (see Algorithm 1)
        i = i + 1
        if i = d then
        begin
            Remove from R_d redundant elements
            RED_A(A) = RED_A(A) ∪ R_d
            i = i − 1
        end
    end
end
```

□

If $\mathbf{A} = (U, A)$ is an information system, $B \subseteq A$ is a set of attributes and $X \subseteq U$ is a set of objects, then the sets: $\{u \in U : [u]_B \subseteq X\}$ and $\{u \in U : [u]_B \cap X \neq \emptyset\}$ are called the *B-lower* and the *B-upper approximation* of X in $\mathbf{A}$, and they are denoted by $\underline{B}X$ and $\overline{B}X$, respectively.

The set $BN_B(X) = \overline{B}X - \underline{B}X$ is called the *B-boundary* of X. When $B = A$ we also write $BN_{\mathbf{A}}(X)$ instead of $BN_A(X)$.

Sets which are unions of some classes of the indiscernibility relation $IND_{\mathbf{A}}(B)$ are called *definable* by B (or, *B-definable*, in short). A set X is thus B-definable iff $\overline{B}X = \underline{B}X$. Some subsets (categories) of objects in an information system cannot be exactly expressed by employing available attributes but they can be defined roughly.

The set $\underline{B}X$ is the set of all elements of U which can be classified with certainty as elements of X, having the knowledge about them represented by attributes from B; the set $BN_B(X)$ is the set of elements which one can classify neither to X nor to $-X$ having knowledge about objects represented by B.

If $X_1, \ldots, X_{r(d)}$ are decision classes of $\mathbf{A}$ then the set $\underline{B}X_1 \cup \ldots \cup \underline{B}X_{r(d)}$ is called *the B-positive region of* $\mathbf{A}$ and is denoted by $POS_B(d)$.

If $\mathbf{A} = (U, A \cup \{d\})$ is a decision table and $B \subseteq A$, then we define a function $\partial_B : U \to \mathbf{P}(V_d\})$, called *the B-generalized decision of* $\mathbf{A}$, by

$$\partial_B(x) = \{v \in V_d : \exists x' \in U \, (x' IND_{\mathbf{A}}(B) x \text{ and } d(x) = v)\} \, . \tag{4}$$

The A-generalized decision ∂_A of $\mathbf{A}$ is called the generalized decision of $\mathbf{A}$.

A decision table $\mathbf{A}$ is called *consistent (deterministic)* if $card(\partial_A(x)) = 1$ for any $x \in U$, otherwise $\mathbf{A}$ is *inconsistent (non-deterministic)*. It is easy to see that a decision table $\mathbf{A}$ is consistent iff $POS_A(d) = U$. Moreover, if $\partial_B = \partial_{B'}$ then $POS_B(d) = POS_{B'}(d)$ for any pair of non-empty sets $B, B' \subseteq A$.

A subset B of the set A of attributes of a decision table $\mathbf{A} = (U, A \cup \{d\})$ is *a relative reduct of* $\mathbf{A}$ iff B is a minimal set with respect to the following property: $\partial_B = \partial_A$. The set of all relative reducts of $\mathbf{A}$ is denoted by $RED(\mathbf{A}, d)$.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a consistent decision table and let $M(\mathbf{A}) = (c_{ij})$ be its discernibility matrix. We construct a new matrix $M'(\mathbf{A}) = (c'_{ij})$ assuming $c'_{ij} = \emptyset$ if $d(x_i) = d(x_j)$ and $c'_{ij} = c_{ij} - \{d\}$, otherwise. The matrix $M'(\mathbf{A})$ is called *the relative discernibility matrix of* $\mathbf{A}$. Now one can construct the *relative discernibility function* $f_{M'(A)}$ of $M'(\mathbf{A})$ in the same way as the discernibility function has been constructed from the discernibility matrix.

It can be shown (see [51]) that the set of all *prime implicants* of $f_{M'(A)}$ determines the set of all *relative reducts* of A.

Another important type of reducts are local reducts. A *local reduct* $r(x_i) \subseteq A$ (or a *reduct relative to decision and object* $x_i \in U$; where x_i is called a *base object*) is a subset of A such that:

a) $\forall \; x_j \in U$, $d(x_i) \neq d(x_j) \Longrightarrow \exists \; a_k \in r(x_i)$: $a_k(x_i) \neq a_k(x_j)$

b) $r(x_i)$ is minimal with respect to inclusion.

If $\mathbf{A} = (U, A \cup \{d\})$ is a decision table then any system $\mathbf{B} = (U', A \cup \{d\})$ such that $U' \subseteq U$ is called a *subtable* of $\mathbf{A}$.

The problem of new cases classification can be described in the following way. Let $\mathbf{W} = (W, A \cup \{d\})$ be a hypothetical *universal decision table* (including known and unknown objects describing an actual considered aspect of reality) and let $\mathbf{A} = (U, A \cup \{d\})$ be a given subtable of the universal decision table. Let $u \in W$ be a so called *tested object*. Our task consists in assigning the value $d(u)$ of the decision d to the tested objects u, knowing only values of condition attributes of u, and relying on a given decision table $\mathbf{A}$. In another words we would like to classify object u to the proper decision class $X_k \in CLASS_{\mathbf{A}}(d)$ (where $k \in 1, ..., r(d)$) on the basis of knowledge included in $\mathbf{A}$. A solution of this problem is a classification algorithm sufficiently approximating the decision function d.

3 Decision Rules

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table and let $V = \bigcup\{V_a : a \in A\} \cup V_d$.

Atomic formulas over $B \subseteq A \cup \{d\}$ and V are expressions of the form $a = v$, called *descriptors* over B and V, where $a \in B$ and $v \in V_a$. The set $\mathbf{F}(B, V)$ of formulas over B and V is the least set containing all atomic formulas over B and V and closed with respect to the classical propositional connectives $\vee$ (disjunction), $\wedge$ (conjunction), and $\neg$ (negation).

Let $\varphi \in \mathbf{F}(B, V)$. Then by $|\varphi|_{\mathbf{A}}$ we denote the meaning of φ in a decision table $\mathbf{A}$, i.e. the set of all objects in U with property φ, defined inductively by

1. if φ is of the form $a = v$ then $|\varphi|_{\mathbf{A}} = \{x \in U : a(x) = v\}$;
2. $|\varphi \wedge \varphi'|_{\mathbf{A}} = |\varphi|_{\mathbf{A}} \cap |\varphi'|_{\mathbf{A}}$; $|\varphi \vee \varphi'|_{\mathbf{A}} = |\varphi|_{\mathbf{A}} \cup |\varphi'|_{\mathbf{A}}$; $|\neg\varphi|_{\mathbf{A}} = U - |\varphi\}_{\mathbf{A}}$

The set $\mathbf{F}(A, V)$ is called the set of *conditional formulas of* $\mathbf{A}$ and is denoted by $\mathbf{C}(A, V)$.

Any formula of the form $(a_1 = v_1) \wedge ... \wedge (a_l = v_l)$, where $v_i \in V_{a_i}$ (for $i = 1, ..., l$) and $P = \{a_1, ..., a_l\} \subseteq A$, is called a *P-basic* formula of $\mathbf{A}$.

If φ is a *P-basic* formula of $\mathbf{A}$ and $Q \subseteq P$, then by φ/Q we mean the Q-basic formula obtained from the formula φ by removing from φ all its elementary subformulas $(a = v_a)$ such that $a \in P \setminus Q$.

A *decision rule* for $\mathbf{A}$ is any expression of the form $\varphi \Rightarrow d = v$ where $\varphi \in \mathbf{C}(A, V)$, $v \in V_d$ and $|\varphi|_{\mathbf{A}} \neq \emptyset$. Formulas φ and $d = v$ are referred to as the *predecessor* and the *successor* of the decision rule $\varphi \Rightarrow d = v$ respectively.

If r is a decision rule in $\mathbf{A}$, then by $Pred(r)$ we denote the predecessor of r and by $Succ(r)$ we denote the successor of r .

An object $u \in U$ is *matched* by a decision rule $\varphi \Rightarrow d = v_d^k$ (where $1 \leq k \leq r(d)$) iff $u \in |\varphi|_{\mathbf{A}}$. If u is matched by $\varphi \Rightarrow d = v_d^k$ then we say that the rule is classifying u to decision class X_k.

The number of objects matched by a decision rule $\varphi \Rightarrow d = v$, denoted by $Match_{\mathbf{A}}(\varphi \Rightarrow d = v)$ is equal $card(|\varphi|_{\mathbf{A}})$.

The number $Supp_{\mathbf{A}}(\varphi \Rightarrow d = v) = card(|\varphi|_{\mathbf{A}} \cap |d = v|_{\mathbf{A}})$ is called *the number of objects supporting a decision rule* $\varphi \Rightarrow d = v$.

A decision rule $\varphi \Rightarrow d = v$ for $\mathbf{A}$ is *true* in $\mathbf{A}$, symbolically $\varphi \Rightarrow_{\mathbf{A}} d = v$, iff $|\varphi|_{\mathbf{A}} \subseteq |d = v|_{\mathbf{A}}$. If a decision rule $\varphi \Rightarrow d = v$ is true in $\mathbf{A}$, we say that the decision rule is *consistent* in $\mathbf{A}$, otherwise the decision rule $\varphi \Rightarrow d = v$ is *inconsistent* or *approximate* in $\mathbf{A}$.

If r is a decision rule in $\mathbf{A}$, then the number $\mu_{\mathbf{A}}(r) = \frac{Supp_{\mathbf{A}}(r)}{Match_{\mathbf{A}}(r)}$ is called *the coefficient of consistency* of the rule r. The coefficient $\mu_{\mathbf{A}}(r)$ can be understood as a degree of consistency of the decision rule r. It is easy to see that a decision rule r for $\mathbf{A}$ is consistent iff $\mu_{\mathbf{A}}(r) = 1$.

The coefficient of consistency of r can be also treated as a degree of inclusion of $|Pred(r)|_{\mathbf{A}}$ in $|Succ(r)|_{\mathbf{A}}$ (see [47]).

If $\varphi \Rightarrow d = v$ is a decision rule for $\mathbf{A}$ and φ is P-basic formula of $\mathbf{A}$ (where $P \subseteq A$), then the decision rule $\varphi \Rightarrow d = v$ is called a *P-basic decision rule for* $\mathbf{A}$, or *basic decision rule* in short.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a consistent decision table with n objects $u_1, ..., u_n$ and with m condition attributes $a_1, ..., a_m$. We present two methods for basic decision rule synthesis for $\mathbf{A}$ (see [52], [53], [45]).

The first method consists of two steps. In the first step we compute the set $RED(\mathbf{A}, d)$ of all relative reducts of decision table $\mathbf{A}$. In the second step, for each reduct $R = \{b_1, ..., b_l\} \in RED(\mathbf{A}, d)$ (where $l \leq m$) and any object $u \in U$ we generate a decision rule in the following way: as the predecessor of the decision rule we take the conjunction $(b_1 = b_1(u)) \wedge ... \wedge (b_l = b_l(u))$ and as the successor of the rule we take the decision attribute d with the value $d(u)$. Hence, the constructed decision rule for the reduct R and the object u is of the form

$$(b_1 = b_1(u)) \wedge ... \wedge (b_l = b_l(u)) \Rightarrow d = d(u).$$

The second method returns basic decision rules with minimal number of descriptors (see [45], [52]).

Let $\varphi \Rightarrow d = v$ be a P-basic decision rule of $\mathbf{A}$ (where $P \subseteq A$) and let $a \in P$. We say that the attribute a is *dispensable* in the rule $\varphi \Rightarrow d = v$ iff $|\varphi \Rightarrow d = v|_{\mathbf{A}} = U$ implies $|\varphi/(P \setminus \{a\}) \Rightarrow d = v|_{\mathbf{A}} = U$, otherwise attribute a is *indispensable* in the rule $\varphi \Rightarrow d = v$. If all attributes $a \in P$ are indispensable in the rule $\varphi \Rightarrow d = v$, then $\varphi \Rightarrow d = v$ is called *independent* in $\mathbf{A}$.

The subset of attributes $R \subseteq P$ is called a *reduct* of a P-basic decision rule $\varphi \Rightarrow d = v$, if $\varphi/R \Rightarrow d = v$ is independent in $\mathbf{A}$ and $|\varphi \Rightarrow d = v|_{\mathbf{A}} = U$ implies $|\varphi/R \Rightarrow d = v|_{\mathbf{A}} = U$. If R is a reduct of the P-basic decision rule $\varphi \Rightarrow d = v$, then $\varphi/R \Rightarrow d = v$ is said to be *reduced*. If R is a reduct of the A-basic decision rule $\varphi \Rightarrow d = v$, then $\varphi/R \Rightarrow d = v$ is said to be *an optimal basic decision rule of* $\mathbf{A}$ (*a basic decision rule with minimal number of descriptors*). The set of all optimal basic decision rules of $\mathbf{A}$ is denoted by $RUL(\mathbf{A})$.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a consistent decision table and $M'(\mathbf{A}) = (c'_{ij})$ be its relative discernibility matrix. We construct a new matrix $M(\mathbf{A}, k) = (c^k_{ij})$ for any $x_k \in U$ assuming $c^k_{ij} = c'_{ij}$ if $d(x_i) \neq d(x_j) \& (i = k \vee j = k)$ and $c^k_{ij} = \emptyset$, otherwise. The matrix $M(\mathbf{A}, k)$ is called *the k-relative discernibility matrix of* $\mathbf{A}$. Now one can construct the *k-relative discernibility function $f_{M(\mathbf{A},k)}$ of* $M(\mathbf{A}, k)$ in the same way as the discernibility function has been constructed from the discernibility matrix (see Section 2).

It can be shown that the set of all *prime implicants* of functions $f_{M(\mathbf{A},k)}$ (for $k = 1, ..., card(U)$) determines the set of all basic optimal rules of $\mathbf{A}$.

Let us assume now that considered decision tables are inconsistent. One can transform an arbitrary inconsistent decision table $\mathbf{A} = (U, A \cup \{d\})$ into a consistent decision table $\mathbf{A}_\partial = (U, A \cup \{\partial_A\})$ where $\partial_A : U \to \mathbf{P}(V_d)$ is the generalized decision of $\mathbf{A}$ defined in Section 2. It is easy to see that $\mathbf{A}_\partial$ is a consistent decision table and one can apply to $\mathbf{A}_\partial$ the described methods to construct decision rules. Hence one can compute decision rules for any inconsistent decision table.

4 Discretization

Suppose we have a decision table $\mathbf{A} = (U, A \cup \{d\})$ where $card(V_a)$ is large for some $a \in A$. Then there is a very low chance that a new object is recognized by rules generated directly from this table, because the attribute value vector of a new object will not match any of these rules. Therefore for decision tables with real value attributes some discretization strategies are built to obtain a higher quality of classification rules. This problem is intensively studied in e.g. [39] and we consider discretization methods presented in [39], [37], [38] and [41]. These methods are based on rough set techniques and boolean reasoning.

4.1 Basic notions

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table where $U = \{x_1, x_2, \ldots, x_n\}$. We assume $V_a = [l_a, r_a) \subset \mathbf{R}$ for any $a \in A$ where $\mathbf{R}$ is the set of real numbers. In the sequel we assume that $\mathbf{A}$ is a consistent decision table.

Any pair (a, c), where $a \in A$ and $c \in \mathbf{R}$, defines a partition of V_a into *left-hand-side* and *right-hand-side interval*. Formally, any attribute-value pair (a, c) is associated with a new binary attribute $f_{(a,c)} : U \to \{0, 1\}$ such that

$$f_{(a,c)}(u) = \begin{cases} 0 & \text{if } a(u) < c \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

In this sense, the pair (a, c) is called a *binary discriminators* or *cut on* V_a. Usually, discretization of real value attributes is determined by cuts.

Let us fix an attribute $a \in A$. Any set of cuts

$$D_a = \{(a, c_1^a), (a, c_2^a), \ldots, (a, c_{k_a}^a)\}$$

where $k_a \in \mathbf{N}$ and $c_0^a = l_a < c_1^a < c_2^a < \ldots < c_{k_a}^a < r_a = c_{k_a+1}^a$, defines a partition on V_a (for $a \in A$) into sub-intervals i.e.

$$V_a = [c_0, c_1^a) \cup [c_1^a, c_2^a) \cup \ldots \cup [c_{k_a}^a, c_{k_a+1}^a).$$

The set of cuts D_a on a defines a discretization of a, i.e. new discreet attribute $a_{D_a} : U \to \{0, .., k_a\}$ such that

$$a_{D_a}(x) = i \quad \Longleftrightarrow \quad a(x) \in [c_i^a, c_{i+1}^a)$$

for any $x \in U$ and $i \in \{0, .., k_a\}$ (see Figure 1).

Analogously, any global discretization is determined by a set of cuts on all real value attributes $\mathbf{D} = \bigcup_{a \in A} D_a$. Any set of cuts

$$\mathbf{D} = \bigcup_{a_i \in A} D_{a_i} = \left\{\left(a_1, c_1^1\right), \ldots, \left(a_1, c_{k_1}^1\right)\right\} \cup \left\{\left(a_2, c_1^2\right), \ldots, \left(a_2, c_{k_2}^2\right)\right\} \cup \ldots$$

transforms the original decision table $\mathbf{A} = (U, A \cup \{d\})$ into new (discreet) decision table $\mathbf{A}|_{\mathbf{D}} = (U, A_{\mathbf{D}} \cup \{d\})$, where $A_{\mathbf{D}} = \{a_{D_a} : a \in A\}$. The table $\mathbf{A}|_{\mathbf{D}}$ is called the $\mathbf{D}$-*discretized table of* $\mathbf{A}$.

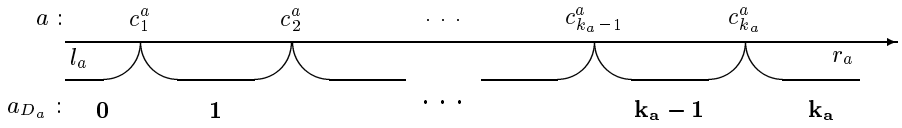

Fig. 1. The discretization of real value attribute $a \in A$ defined by the set of cuts $\{(a, c_1^a), (a, c_2^a), \ldots, (a, c_{k_a}^a)\}$

It is obvious, that discretization process is associated with a loss of information. Usually, the task of discretization is to determine a minimal set of cuts $\mathbf{D}$ (with respect to inclusion) from a given decision table $\mathbf{A}$ such that, in spite of losing information, the $\mathbf{D}$-discretized table $\mathbf{A}|_{\mathbf{D}}$ still keeps some useful properties of $\mathbf{A}$. In the discretization method based on rough set and Boolean reasoning approach, we are trying to keep the discernibility between objects.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a given decision table and $\mathbf{D}$ be a set of cuts. We say that

- $\mathbf{D}$ is *consistent with* $\mathbf{A}$ (or $\mathbf{A}$-*consistent*) if $\partial_{\mathbf{A}} = \partial_{\mathbf{A}|_{\mathbf{D}}}$, where ∂_A and $\partial_{\mathbf{A}|_{\mathbf{D}}}$ are generalized decisions of $\mathbf{A}$ and $\mathbf{A}|_{\mathbf{P}}$. In other words, the set of cuts is $\mathbf{A}$-consistent iff for any two objects $u, v \in U$:

$$\textbf{if} \quad (u, v \text{ are discerned by } A) \quad \textbf{then} \quad (u, v \text{ are discerned by } D).$$

- $\mathbf{D}$ is *irreducible* in $\mathbf{A}$ if $\mathbf{D}$ is $\mathbf{A}$-consistent and $\mathbf{D}'$ is not $\mathbf{A}$-consistent for any proper subset $\mathbf{D}' \subset \mathbf{D}$.
- $\mathbf{D}$ is *optimal* in $\mathbf{A}$ if $card\,(\mathbf{D}) \leq card\,(\mathbf{D}')$ for any $\mathbf{A}$-consistent set of cuts $\mathbf{D}'$.

The problem of searching for optimal set of cuts (Optimal Discretization Problem) has been explored in [39], [37], [38] and [41]. From computational complexity point of view, the Optimal Discretization Problem appears to be hard. We have the following theorem (see [39]):

Theorem 1. *For a given decision table* $\mathbf{A}$ *and an integer* k.

- *The decision problem for checking if there exists an irreducible set of cuts* $\mathbf{P}$ *in* $\mathbf{A}$ *such that* $card(\mathbf{P}) < k$ *(*$k-$**minimal partition problem***) is NP-complete.*
- *The problem of searching for an optimal set of cuts* $\mathbf{P}$ *in* $\mathbf{A}$ *(***optimal partition problem***) is NP-hard.*

Two sets of cuts $\mathbf{D}'$, $\mathbf{D}$ are equivalent (denoted by $\mathbf{D}' \equiv_{\mathbf{A}} \mathbf{D}$, if and only if $\mathbf{A}|_{\mathbf{D}} = \mathbf{A}|_{\mathbf{D}'}$. The equivalence relation $\equiv_{\mathbf{A}}$ has finite number of equivalence classes. In the sequel we do not distinguish between equivalent sets of cuts.

4.2 Maximal Discernibility (MD) Heuristics

We below describe our heuristic for optimal discretization problem.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table. An arbitrary attribute $a \in A$ defines a sequence $v_1^a < v_2^a < \ldots < v_{n_a}^a$, where $\left\{v_1^a, v_2^a, \ldots, v_{n_a}^a\right\} = \{a(x) : x \in U\}$ and $n_a \leq n$. Then the set of all possible cuts on a is denoted by

$$\mathbf{C}_a = \left\{ \left(a, \frac{v_1^a + v_2^a}{2}\right), \left(a, \frac{v_2^a + v_3^a}{2}\right), \ldots, \left(a, \frac{v_{n_a-1}^a + v_{n_a}^a}{2}\right) \right\}$$

The set of possible cuts on all attributes is denoted by

$$\mathbf{C}_A = \bigcup_{a \in A} \mathbf{C}_a$$

In [39] we have shown that any irreducible set of cuts of $\mathbf{A}$ is a relative reduct of another decision table $\mathbf{A_1}$ built from $\mathbf{A}$, where $\mathbf{A_1} = (U_1, A_1 \cup \{d_1\})$ is defined as follows:

- $U_1 = \{(x, y) \in U \times U : d(x) \neq d(y)\} \cup \{new\}$, where $new \notin U \times U$.
- $d_1 : U_1 \to \{0, 1\}$ is defined by $d_1(u) = \begin{cases} 0 & \text{if } u = new \\ 1 & \text{otherwise} \end{cases}$
- $A_1 = \left\{f_{(a,c)} : (a, c) \in \mathbf{C}_A\right\}$ is a set of all discriminators defined by initial set of cuts from $\mathbf{C}_A$ (see Equation (5)).

The algorithm based on Johnson's strategy described in the previous section is searching for a cut $c \in A_1$ which discerns the largest number of pairs of objects. Then we move the cut c from A_1 to the resulting set of cuts $\mathbf{P}$ and remove from U_1 all pairs of objects discerned by c. Our algorithm is continued until $U_1 = \{new\}$. Now we present the details of our algorithm.

Algorithm 3: *MD-discretization*

Input: *The consistent decision table* $\mathbf{A}$.
Output: *The semi-minimal set of cuts* $\mathbf{D}$ *consistent with* $\mathbf{A}$.
Method:
  $\mathbf{D} = \emptyset$; $\mathbf{C}_A$ = *initial set of cuts on* $\mathbf{A}$;
  $\mathbf{L} = \{(x, y) \in U \times U : d(x) \neq d(y)\}$;
  while $(\mathbf{L} \neq \emptyset)$ **do**
  begin
    Choose the cut $c_{\max} \in \mathbf{C}_A$ *which discerns the largest number of pairs of objects in* $\mathbf{L}$. *(i.e. the cut corresponding to the column with the largest number of "1")*
    Insert c_{max} *into* $\mathbf{D}$ *and remove it from* $\mathbf{C}_A$.
    Remove all pairs of objects from $\mathbf{L}$ *discerned by* c_{max}
  end

□

A	a	b	d
u_1	1	2	1
u_2	1.2	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1

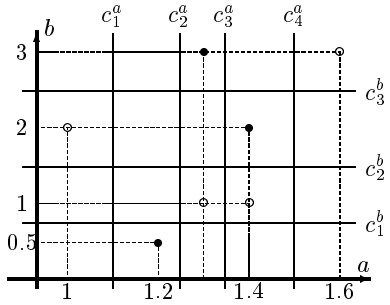

Fig. 2. The decision table with two real value attributes and its geometric interpretation

$\mathbf{A}_1$	$f_{(a,1.1)}$	$f_{(a,1.25)}$	$f_{(a,1.35)}$	$f_{(a,1.5)}$	$f_{(b,0.75)}$	$f_{(b,1.5)}$	$f_{(b,2.5)}$	d_1
(u_1, u_2)	1	0	0	0	1	1	0	1
(u_1, u_3)	1	1	0	0	0	0	1	1
(u_1, u_5)	1	1	1	0	0	0	0	1
(u_4, u_2)	0	1	1	0	1	0	0	1
(u_4, u_3)	0	0	1	0	0	1	1	1
(u_4, u_5)	0	0	0	0	0	1	0	1
(u_6, u_2)	0	1	1	1	1	1	1	1
(u_6, u_3)	0	0	1	1	0	0	0	1
(u_6, u_5)	0	0	0	1	0	0	1	1
(u_7, u_2)	0	1	0	0	1	0	0	1
(u_7, u_3)	0	0	0	0	0	1	1	1
(u_7, u_5)	0	0	1	0	0	1	0	1
new	0	0	0	0	0	0	0	0

Fig. 3. Table $\mathbf{A}_1$ constructed from table $\mathbf{A}$

Example 1. We consider the decision table with two attributes and seven objects (Figure 2).
The set of all cuts consists of four cuts on the attribute a and three cuts on the attribute b.

$$\mathbf{C}_A = \{(a, 1.1), (a, 1.25), (a, 1.35), (a, 1.5)\} \cup \{(b, 0.75), (b, 1.5), (b, 2.5)\}$$

The new decision table $\mathbf{A}_1$ consists of 7 attributes and 13 objects (Figure 3). The MD–discretization algorithm chooses the cut $(b, 1.5)$ first because it discerns 6 pairs of objects, than it chooses the cuts $(a, 1.25)$ and $(a, 1.5)$. The result of the MD–discretization algorithm is the set of cuts $\mathbf{D} = \{(a, 1.25), (a, 1.5), (b, 1.5)\}$.

Let n be a number of objects and k be a number of attributes of decision table $\mathbf{A}$, then $card\,(A_1) \leq (n-1)\,k$ and $card\,(U_1) \leq \frac{n(n-1)}{2}$. It is easy to note that for any cut $c \in A_1$ we need $O\left(n^2\right)$ steps to find the number of all pairs of objects discerned by c. Hence the straightforward realization of this algorithm requires $O\left(kn^2\right)$ of memory space and $O(kn^3)$ steps to determine one *cut*, so it is not useful in practice. The algorithm presented below determines the best cut in $O\,(kn)$ steps using $O\,(kn)$ space only.

At first we show that the number of pairs of objects discerned by a given cut can be computed faster than $O\left(n^2\right)$. For the given cut $(a,c) \in \mathbf{C}_A$ on an attribute $a \in A$ and a given subset of objects $X \subseteq U$ we introduce the following notation:

1. $W^X(a,c)$ = number of pairs of objects from X discerned by (a,c).
2. for $j = 1, ..., r$:
   $l_j^X(a,c) = card\{x \in X : [a(x) < c] \wedge [d(x) = j]\}$ and
   $r_j^X(a,c) = card\{x \in X : [a(x) > c] \wedge [d(x) = j]\}$
   are numbers of objects from X belonging to the j^{th} decision class and being on the left-hand-side and right-hand-side of the cut (a,c), respectively.
3. The number of objects on either side of (a,c) is:
   $l^X(a,c) = \sum_{j=1}^{r} l_j^X(a,c) = card\{x \in X : a(x) < c\}$;
   $r^X(a,c) = \sum_{j=1}^{r} r_j^X(a,c) = card\{x \in X : a(x) > c_m^a\}$.

We obtain the following lemma:

Lemma 2. *For any cut $(a,c) \in \mathbf{C}_A$, and $X \subseteq U$:*

$$W^X(a,c) = l^X(a,c) \cdot r^X(a,c) - \sum_{i=1}^{r} l_i^X(a,c) \cdot r_i^X(a,c)$$

This Lemma shows, that the number of pairs of objects discerned by a cut $(a,c) \in \mathbf{C}_A$ can be computed in $O(n)$ time. The next theorem is showing that if (a,c_m) and (a,c_{m+1}) are consecutive cuts on the attribute a, than the value $W^X(a,c_{m+1})$ can be derived from $W^X(a,c_m)$ in $O(1)$ time (because the attribute a is fixed, we will use the notations: $l_j^X(c)$, $r_j^X(c)$, $l^X(c)$, $r^X(c)$ and $W^X(c)$ for simplification, instead of $l_j^X(a,c)$, $r_j^X(a,c)$, $l^X(a,c)$, $r^X(a,c)$ and $W^X(a,c)$).

Theorem 3. *If there is exactly one object $x \in X \subseteq U$ such that $(c_m < a(x) < c_{m+1}$ and let $t = d(x)$ then:*

$$W^X(c_{m+1}) = W^X(c_m) + \left[r^X(c_m) - l^X(c_m)\right] - \left[r_t^X(c_m) - l_t^X(c_m)\right]$$

Proof: We have:

$$l_j^X(c_{m+1}) = \begin{cases} l_j^X(c_m) & \text{if } j \neq t \\ l_t^X(c_m) + 1 & \text{if } i = t \end{cases} \quad \text{and} \quad r_j^X(c_{m+1}) = \begin{cases} r_j^X(c_m) & \text{if } j \neq t, \\ r_t^X(c_m) - 1 & \text{if } j = t. \end{cases}$$

From those equations it follows that

$$l^X(c_{m+1}) = l^X(c_m) + 1 \text{ and } r^X(c_{m+1}) = r^X(c_m) - 1.$$

From Lemma 2 we have:

$$W^X(c_m) = l^X(c_m)\, r^X(c_m) - \sum_{i=1}^{r} l_i^X(c_m)\, r_i^X(c_m);$$

$$W^X(c_{m+1}) = l^X(c_{m+1}) r^X(c_{m+1}) - \sum_{1}^{r} l_i^X(c_{m+1}) r_i^X(c_{m+1})$$

Thus

$$\begin{aligned} W^X(c_{m+1}) - W^X(c_m) &= l^X(c_{m+1})\, r^X(c_{m+1}) - l^X(c_m)\, r^X(c_m) + \\ &\quad l_t^X(c_m)\, r_t^X(c_m) - l_t^X(c_{m+1})\, r_t^X(c_{m+1}) \\ &= \left[l^X(c_m)+1\right] \cdot \left[r^X(c_m)-1\right] - l^X(c_m)\, r^X(c_m) + \\ &\quad l_t^X(c_m)\, r_t^X(c_m) - \left[l_t^X(c_m)+1\right] \cdot \left[r_t^X(c_m)-1\right] \\ &= [r^X(c_m) - l^X(c_m)] - [r_t^X(c_m) - l_t^X(c_m)]. \end{aligned}$$

□

Let us assume that the number $W^X(a, c_m)$ of pairs of objects from X discerned by the cut (a, c_m) has been determined. Theorem 3 shows that we can compute the value $W^X(a, c_{m+1})$ in time $O(1)$. In the consequence the best cut on any attribute can be determined in time $O(n)$.

We propose two strategies of searching for semi-optimal set of cuts. The first, called *local strategy*, after finding the best cut and dividing the object set into two subsets of objects, repeats this procedure for each object set separately until some stop condition holds. The quality of a cut (i.e. number of objects discerned by cut) in local strategy is computed locally on a subset of objects. In the second strategy, called *global strategy*, quality of cuts are computed on whole set of objects. Usually, the local strategy is easier for realization than the global one, but the set of cuts obtained by the global strategy is smaller.

4.3 Local strategy

The local strategy can be realized by using *decision tree*. A typical algorithm for decision tree generation for a given decision table $\mathbf{A} = (U, A \cup \{d\})$ is described below:

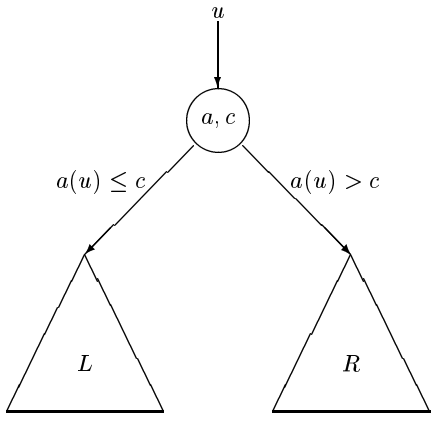

Fig. 4. The decision tree used for local discretization

Algorithm 4: *Local discretization*

Input: *The consistent decision table* $\mathbf{A}$.
Output: *The semi-minimal set of cuts* $\mathbf{D}$ *consistent with* $\mathbf{A}$.
Method:

Initialize the binary tree variable $\mathbf{T}$ *with the empty tree.*
Label the root by the set of all objects U *and fix the status of the root to be* unready.
while *there is a leaf marked by* unready **do**
begin
for *any* unready *leave* N *of the tree* $\mathbf{T}$
begin
if *objects labeling* N *have the same decision value* **then**
begin
Replace the object set at N *by its common decision*
Change the status of N *to* ready.
end
else
begin
Compute the value $W^N(a,c)$ *for all cuts from* $\mathbf{C}_A$ *and search for cut* (a^*,c^*) *maximizing the function* $W^N(.)$, *i.e.*

$$(a^*,c^*) = \arg\max_{(a,c)} W^N(a,c)$$

Replace the label of N *by* (a^*,c^*) *and mark it as* ready;
Create two new nodes N_1 *and* N_2 *with status* unready *as the left and right subtrees of* N, *where:*

$$N_1 = \{u \in N : a^*(u) < c^*\} \quad \text{and} \quad N_{\mathbf{2}} = \{u \in N : a^*(u) \geq c^*\}$$

end
end
end
return T

□

4.4 Global strategy

Now we describe some properties of the $\mathbf{D}$-discretized table $\mathbf{A_D}$, where $\mathbf{D} \subseteq \mathbf{C}_A$ is an arbitrary set of cuts. Assuming $X_1, X_2, ..., X_m$ to be equivalence classes of the discernibility relation $IND(A_\mathbf{D})$ of table $\mathbf{A}|_\mathbf{D}$, one can note that the family $\mathbf{PART(D)} = \{X_1, X_2, ..., X_m\}$ defines a partition of the set of objects U into m disjoint subsets i.e.

$$U = X_1 \cup ... \cup X_m \text{ and } \forall_{i \neq j} X_i \cap X_j = \emptyset$$

In practice the classes $X_1, .., X_m$ are stored in memory instead of $\mathbf{A}|_{\mathbf{D}}$. Observe that objects from X_i $(i = 1, .., m)$ are not discerned by any cut from $\mathbf{D}$. Hence the number of pairs of objects discerned by a cut $c \notin \mathbf{D}$ but not discerned by cuts from $\mathbf{D}$ is equal to

$$W_{\mathbf{D}}(a, c) = W^{X_1}(a, c) + W^{X_2}(a, c) + ... + W^{Xm}(a, c). \tag{6}$$

Let us consider the situation, when we have the set of cuts $\mathbf{D}$ defining the equivalence classes $X_1, .., X_m$ and two consecutive cuts c_j^a, c_{j+1}^a on the attributes a. We can derive the value $W_{\mathbf{D}}\left(a, c_{j+1}^a\right)$ from $W_{\mathbf{D}}\left(a, c_j^a\right)$ in time $O(1)$ applying the following theorem:

Theorem 4. *Let* $\mathbf{D}$ *be a given set of cuts. If there is exactly one object* $x \in U$ *such that* $a(x) \in \left(c_j^a, c_{j+1}^a\right)$ *then:*

$$\begin{aligned} W_{\mathbf{D}}\left(a, c_{j+1}^a\right) - W_{\mathbf{D}}\left(a, c_j^a\right) &= W^{X_i}\left(a, c_{j+1}^a\right) - W^{X_i}\left(a, c_j^a\right) = \\ &= \left(r^{X_i}\left(c_j^a\right) - l^{X_i}\left(c_j^a\right)\right) - \left(r_t^{X_i}\left(c_j^a\right) - l_t^{X_i}\left(c_j^a\right)\right) \end{aligned}$$

where $t = d(x)$ *and* $X_i \in \mathbf{PART}(\mathbf{D})$ *is the equivalence class containing* x.

Proof: This fact follows from Theorem 3 and Equation 6. □

Now we present the details of our algorithm.

Algorithm 5: *Global discretization*

Input: *The consistent decision table* $\mathbf{A}$.
Output: *The semi-minimal set of cuts* $\mathbf{P}$ *consistent with* $\mathbf{A}$.
Data Structure: $\mathbf{D}$ – *the semi-minimal set of cuts;* $\mathbf{L} = \mathbf{PART}(\mathbf{D})$ – *the partition of* U *defined by* $\mathbf{D}$*;* $\mathbf{C}_A$ – *the set of all possible cuts on* $\mathbf{A}$.
Method:

1. $\mathbf{D} = \emptyset; \mathbf{L} = \{U\}$; $A_1 =$ *initial set of cuts on* $\mathbf{A}$
2. *Compute the value* $W_{\mathbf{D}}(a, c)$ *for all cuts from* $\mathbf{C}_A$ *and search for cut* (a^*, c^*) *maximizing the function* $W_{\mathbf{D}}(.)$*, i.e.*

$$(a^*, c^*) = \arg\max_{(a,c)} W_{\mathbf{D}}(a, c)$$

3. *Set*

$$\mathbf{D} = \mathbf{D} \cup \{(a^*, c^*)\}; \mathbf{C}_A = \mathbf{C}_A \setminus \{(a^*, c^*)\}$$

4. **for** $X \in \mathbf{L}$ **do**
   if X *consists of objects from one decision class* **then** *remove* X *from* $\mathbf{L}$;
   if (a^*, c^*) *divides the set* X *into* X_1, X_2 **then**
   – *Remove* X *from* $\mathbf{L}$
   – *Add to* $\mathbf{L}$ *two sets* X_1, X_2
5. **if** $\mathbf{L}$ *is empty* **then** *Stop* **else** *Go to 2.*

□

Theorem 5. *Algorithm 5 needs time of order* $O(kn(|\mathbf{P}| + \log n))$ *and* $O(kn)$ *memory space for computing of the semi-minimal set of cuts* $\mathbf{P}$.

Proof: Step 1 takes $O(kn \log n)$ steps. From Theorem 4 it follows that Loop 2 and Loop 4 require together $O(kn)$ memory space and $O(kn)$ time. □

Example 2. We illustrate the local and global strategy on the decision table presented in the Figure 5.

In both cases our algorithms begin with choosing the best cut $(a_3, 4.0)$ discerning 20 pairs of objects from $\mathbf{A}$. Theorem 3 assures that this cut can be found in linear time.

In the local strategy, the cut $(a_3, 4.0)$ divides the set of objects into two subsets $U_1 = \{u_1, u_3, u_4, u_5, u_9, u_{10}\}$ and $U_2 = \{u_2, u_6, u_7, u_8\}$. Then the local discretization algorithm chooses best cuts locally on U_1 and U_2. The cuts $(a_1, 3.5)$, $(a_2, 3.5)$ and $(a_2, 5.5)$ are best cuts for U_1 (because they are discerning 6 pairs of objects from U_1); the cuts $(a_1, 5.5)$ and $(a_3, 7.0)$ are the best cuts for U_2 (4 pairs of objects from U_2). Assume that the cuts $(a_2, 3.5)$ and $(a_1, 5.5)$ have been chosen for U_1 and U_2, respectively. Hence U_1 is divided into two subsets $X_1 = \{u_1, u_9, u_{10}\}$ and $X_2 = \{u_3, u_4, u_5\}$, but the set U_2 is divided into two sets $X_3 = \{u_2, u_6\}$ and $X_4 = \{u_7, u_8\}$. One can see that the sets X_1, X_3 and X_4 consists of objects from one decision class, then our algorithm continues the searching process for X_2. The result of our algorithm is the set of cuts $\{(a_3, 4.0), (a_2, 3.5), (a_1, 5.5), (a_1, 3.5), (a_2, 6.5)\}$ (see Figure 6).

In the global strategy, the best cut $(a_3, 4.0)$ defines a partition $\mathbf{L} = \{U_1, U_2\}$ of U, where $U_1 = \{u_1, u_3, u_4, u_5, u_9, u_{10}\}$ and $U_2 = \{u_2, u_6, u_7, u_8\}$. Next, the global discretization algorithm chooses the cut $(a_1, 3.5)$ because it discerns 8 pairs of objects (6 pairs in U_1 and 2 pairs in U_2). The set of two cuts $\{(a_3, 4.0), (a_1, 3.5)\}$ defines a new partition $\mathbf{L} = \{\{u_1, u_3, u_4\}, \{u_5, u_9, u_{10}\}, \{u_2\}, \{u_6, u_7, u_8\}\}$. After removing two sets $\{u_5, u_9, u_{10}\}, \{u_2\}$, which consists of objects from one decision class, the algorithm chooses the cut $(a_2, 3.5)$ (3 pairs). The set of cuts $\{(a_3, 4.0), (a_1, 3.5), (a_2, 3.5)\}$ defines a partition $\mathbf{L} = \{\{u_1\}, \{u_3, u_4\}, \{u_6, u_8\}, \{u_7\}\}$. Finally, the algorithm chooses the cut $(a_2, 6.5)$, which discerns all remaining pairs of objects. As the result we have the set of cuts $\{(a_3, 4.0), (a_1, 3.5), (a_2, 3.5), (a_2, 6.5)\}$.

4.5 Application of dynamic reducts to finding set of cuts

Now, we propose another method of searching for an irreducible set of cuts of a given decision table $\mathbf{A}_1 = (U_1, A_1 \cup \{d_1\})$ (see Section 4.2). This method is based on the dynamic reduct notion (see Section 7, 8). We calculate dynamic reducts (or generalized dynamic reducts) for the table $\mathbf{A}_1$ and we choose one with the best stability coefficient. Next, as an irreducible set of cuts we select cuts belonging to the chosen dynamic reduct. Finally, we remove from U_1 and respectively U all objects not belonging to the reduct domain (see Section 9) of the chosen dynamic reduct.

A	a_1	a_2	a_3	d
u_1	1.0	2.0	3.0	0
u_2	2.0	5.0	5.0	1
u_3	3.0	7.0	1.0	2
u_4	3.0	6.0	1.0	1
u_5	4.0	6.0	3.0	0
u_6	5.0	6.0	5.0	1
u_7	6.0	1.0	8.0	2
u_8	7.0	8.0	8.0	2
u_9	7.0	1.0	1.0	0
u_{10}	8.0	1.0	1.0	0

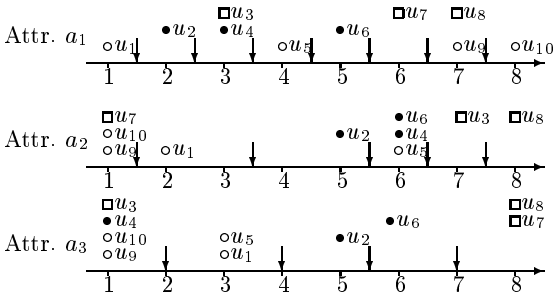

Fig. 5. The exemplary decision table with 10 objects, 3 attributes and 3 decision classes and the illustration of cuts for this table

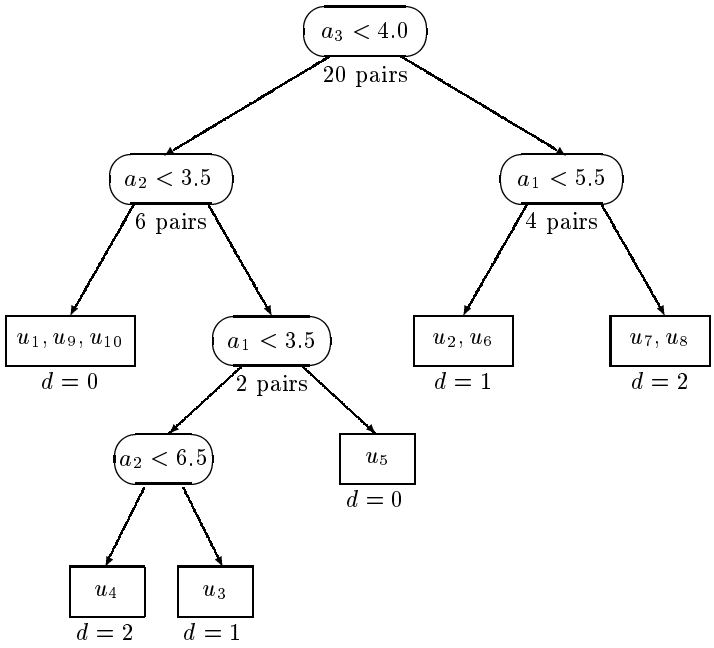

Fig. 6. The construction of decision tree by applying local discretization algorithm

5 Local reducts computation

A notion of local reduct (see Section 2 and [65]) seems to be very useful in classification problems. A set of rules generated basing on these reducts is usually less specific and fits more new (unseen) objects than in a classical case.

Let $\mathbf{A} = (U, A \cup \{d\})$ be an information system, where U - set of objects, A - set of attributes, d - decision. A rule generated by a local reduct is concerned with the base object and may not recognize any other object from U. To assure that a set of rules will recognize (at least) all objects from the training set, we

have to generate a local reduct for every object. A simple algorithm checking whether a subset is a local superreduct works at a time complexity of $O(mn)$: we have to compare our base object with all other objects and check whether condition a) (see local reduct definition) holds. It takes $O(mn^2)$ time to check this property for all objects (in case we are looking for a local reduct for every object), where n = number of objects, m = number of attributes.

An algorithm presented below realizes the following objective: assuming that the information system $\mathbf{A}$ is consistent, find a family of subsets R_1, R_2,... R_k such that for any object x_i from U at least one R_j is a local reduct (we will say, that R_j covers x_i). We look for a possibly small family R_1,... R_k, i.e. we prefer these subsets which cover possibly many objects. We assume, that these subsets reflect regularities in data and generate more general rules and that fact means better classification of new samples and less memory required to store rules.

Algorithm 6 *Optimal covering with local reducts.*
Input: *decision table* $\mathbf{A} = (U, \{a_1, ..., a_m\} \cup \{d\})$
Output: *a set* $\mathbf{R}$ *of local reducts covering* U
Method:

$N = \{N_1, \ldots, N_n\} = 0, \ldots, 0$ *-numbers of local reducts found for each object*
repeat
  $R = \{a_1, \ldots, a_m\}$
  repeat
    $b = \{b_1, \ldots, b_n\} = 0, \ldots, 0$
    $c = \{c_1, \ldots, c_n\} = 0, \ldots, 0$ *-tables of (locally) covered objects*
    $LocRed\,(\mathbf{A}, R, c, N)$ *– check which objects are covered by* R
    for $a_i \in R$ **do**
    begin
      $R = R - a_i$
      $M_i = LocRed\,(\mathbf{A}, R, b, N)$ *– number of objects covered by* R
      $R = R \cup a_i$
    end
    for $i = 0$ **to** n **if** $c_i = 1$ **and** $b_i = 0$ **then**
    begin
      $\mathbf{R} = \mathbf{R} \cup R(u_i)$ *– update set of reducts*
      $N_i = 1$
    end
    $j = \arg\max(M_i)$ *– if there is more than one maximum, select random*
    $R = R - a_j$
  until $R = \emptyset$
until $\forall_{i \in \{1,\ldots,n\}}\ N_i = 1$
return R

□

Lemma. *The algorithm described above generates a covering for all objects in at most* $n = |U|$ *cycles of the outer loop.*

Proof: see [65].

The algorithm is not deterministic, because we use a random selection of maximal M_i. We need to have a method of determining whether a subset is a local superreduct to complete our algorithm. Function *LocRed* checks for which objects a subset R is a local reduct and returns a number of these objects as well as a list of them (in the form of binary table):

Algorithm 7. *Function LocRed*$(\mathbf{A}, R, b, N)$.
Input:
1. Decision table $\mathbf{A} = (U, A \cup \{d\})$
2. Subset $R \subseteq A$
3. Table of already covered objects N
Output:
1. Table b *of objects for which* R *is a local superreduct*
2. Number of newly covered objects
Method:
Sort U *by values of attributes in* R
Create partition $U = U_1 \cup \ldots \cup U_k$ *into indiscernibility classes,*
i.e. $u_1, u_2 \in U_i \Rightarrow (u_1, u_2) \in IND(R)$
$newcover = 0$
for $i = 1$ **to** k **do**
begin
$uniform = 1$
for $j = 2$ **to** $|U_i|$ **do**
begin
if $d(u_{i,j}) \neq d(u_{i,j-1})$ **then** $uniform = 0$
– *where* $U_i = \{u_{i,1}, \ldots, u_{i,|U_i|}\}$
end
if $uniform = 1$ **then**
begin
for $j = 1$ **to** $|U_i|$ **do**
begin
$t =$ *a number of object* $u_{i,j}$ *in table* U
$b_t = 1$
if $N_t = 0$ **then** $newcover = newcover + 1$
end
end
end
return *newcover*
□

The partition of U can be done by n operations if U is properly sorted. Since we may use a fast method of sorting, our algorithm has the complexity of $mn\ log(n)$, where n = number of objects, m = number of attributes.

6 Computing reducts using Genetic Algorithms

Our methods of decision rules generation from decision tables are based on the reduct set computation. The time cost of the reduct set computation can be too high in case the decision table consists of too many: objects or attributes or different values of attributes. The reason is that in general the size of the reduct set can be exponential with respect to the size of the decision table and the problem of computing a minimal reduct is NP-hard (see [51]). Therefore we are often forced to apply approximation algorithms to obtain some knowledge about the reduct set. One way is to use approximation algorithms that do not give the optimal solutions but require short computing time. Among these algorithms are the following ones: Johnson's algorithm, algorithms based on simulated annealing and Boltzmann machines, algorithms using neural networks and algorithms based on genetic algorithms, which we would like to present in this section.

6.1 Genetic and hybrid algorithms

The main idea of genetic algorithms is based on the Darwinian principle of "survival of the fittest" (natural selection). In a case of *classical genetic algorithms* (see [21], [24]) we are given a state space S (finite, but large) and a function: $f : S \to R_+$. Our goal is to find x_o: $f(x_o) = max\{f(x)$: $x \in S\}$. Elements of set S are "*individuals*". We treat a value of the function f as ability to survive in the environment ("*fitness*"), and we simulate the process of evolution as follows:

1. We choose the representation scheme: a mapping from a space of "individuals" into "chromosomes" - usually bit strings.
2. We randomly choose the set of chromosomes as an initial population.
3. We calculate "fitness" $F(c)$ of each chromosome c as a value of $f(s(c))$, where $s(c)$ is the individual encoded by c. Then we create a new population, replacing the chromosomes with low fitness by those with higher fitness.
4. We randomly affect the new population by *genetic operators*, e.g. *mutation* (small, random modifications of chromosomes) and *crossing-over* (exchange of "genetic material" between some pairs of chromosomes).
5. We repeat 3-4 with the new population, until a stopping criterion is satisfied.

The result of evolution is the best individual x_{max} which is usually nearly as good as the global optimum x_o.

The scheme presented above is general and domain-independent. On the other hand, in particular problems we often have some approximation algorithms and heuristics producing maybe not optimal, but good results. To exploit advantages of both genetic and heuristic algorithms, one can use a hybridization strategy [64]. The general scheme of *hybrid algorithm* is as follows:

1. Find a strategy (heuristic algorithm) which gives an approximate result.
2. Modify (parameterize) the strategy using a control sequence, so that the result depends on this sequence (recipe).

3. Encode the control sequence to a chromosome.
4. Use a genetic algorithm to produce control sequences. Proceed with the heuristic algorithm controlled by the sequence. Evaluate an object generated by the algorithm and use its quality measure as a fitness of the control sequence.
5. A result of evolution is the best control sequence, i.e. the sequence producing the best object. Send this object to the output of the hybrid algorithm.

Hybrid algorithms proved to be useful and efficient in many areas, including NP-hard problems of combinatorics. As we will see in the next section, short reduct finding problem also can be solved efficiently by this class of algorithms.

6.2 Finding reducts using GA

An order-based genetic algorithm is one of the most widely used component of various hybrid systems. Theoretical foundations and practical construction of this algorithm are presented in [64]. In this type of genetic algorithm a chromosome is an n-element permutation σ, represented by a sequence of numbers: $\sigma(1)\ \sigma(2)\ \sigma(3)\ \ldots\ \sigma(\mathrm{n})$. Mutation of an order-based individual means one random transposition of its genes (a transposition of random pair of genes). There are various methods of recombination (*crossing-over*) considered in literature. In [21] such methods as PMX (Partially Matched Crossover), CX (Cycle Crossover) and OX (Order Crossover) are described. Another type of crossing-over operator is presented in [63]. After crossing-over, fitness function of every individual is calculated. In the case of a hybrid algorithm a heuristic part is launched under control of individual; a fitness value depends on the result of heuristic algorithm. Then, new population is generated using "roulette wheel" algorithm: the fitness value of every individual is normalized and treated as probability distribution on population; then we randomly choose M new individuals using this distribution. Then all these steps are repeated.

In the hybrid algorithm [62] a simple, deterministic method was used for reduct generation:

Algorithm 8. *Finding a reduct basing on permutation.*
Input:
1. decision table $\mathbf{A} = (U, \{a_1, ..., a_n\} \cup \{d\})$
2. permutation τ *generated by genetic algorithm*
Output: *a reduct* R *generated basing on permutation* τ
Method:
$R = \{a_1, ..., a_n\}$
$(b_1 \ldots b_n) = \tau\,(a_1 \ldots a_n)$
for $i = 1$ **to** n **do**
begin
$R = R - b_i$
if not $Reduct(R, \mathbf{A})$ **then** $R = R \cup b_i$
end

return R

□

A fast algorithm for determining whether R is a superreduct is presented in [40]:

Algorithm 9. *Function Reduct*$(R, \mathbf{A})$.

Input:

1. Decision table $\mathbf{A} = (U, A \cup \{d\})$

2. Subset $R \subseteq A$

3. Binary tree T_{red} *of reducts found so far, list* L_{subred} *of subreducts found so far.*

Output:

1. "True" if R *is a reduct or superreduct of* A*, "False" otherwise.*

2. Updated T_{red} *and* L_{subred} *structures.*

Method:

```
if R ∈ T_red then return True
for s ∈ L_subred do
begin
   if R ⊆ s return False
end
Sort U by values of attributes in R
for i = 2 to m do
begin
   if d(u_i) ≠ d(u_{i-1}) then
   begin
      all_equal = 1
      for a_j ∈ R do
      begin
         if a_j(u_i) ≠ a_j(u_{i-1}) then all_equal = 0
      end
      if all_equal = 1 then
      begin
         Add(L_subred, R)
         return False
      end
   end
end
Add(T_red, R)
return True
```

□

The result of the algorithm will always be a reduct. Every reduct can be found using this algorithm, the result depends on the order of attributes (proof: see [62]). The genetic algorithm is used to generate the proper order. To calculate the function of fitness for a given permutation (order of attributes) we have to

perform one run of the deterministic algorithm and calculate the length of the found reduct. In the selection phase of genetic algorithm we used linear scaling [21] and the following fitness function:

$$F(\tau) = n - L_\tau + 1$$

The hybrid algorithm described above performs much slower that the classical genetic algorithm. On the other hand, the reducts obtained by this algorithm are usually shorter. Moreover, the hybrid algorithm generates from 50 to 500 different reducts in comparison with 5 to 50 reducts generated by the classical GA at the same time.

An algorithm described above generates possibly shortest reducts, i.e. the reducts with as few attributes as possible. On the other hand, our goal is not to calculate reducts, but to construct an efficient system for classification or decision making. Another approach is to select a reduct due to the number of rules it generates rather than to its length. Every reduct generates an indiscernibility relation on the universe and in most cases it identifies some pairs of objects. If a reduct generates less rules, it means, that the rules are more general and they should better recognize new objects.

The number of rules can be easily computed due to the improvement of the reduct generation system described in [40]. The hybrid algorithm described above can be used to find reducts generating the minimal number of rules. The only thing we have to change is the definition of the fitness function:

$$F(\tau) = m - R_\tau + \frac{n - L_\tau + 1}{n}$$

where R_τ denotes the number of rules generated by the reduct. Now, the primary criterion of optimization is the number of rules, while the secondary is the reduct length. The results of experiments [64] show, that the classification system based on the reducts optimized due to the number of rules performs better (or not worse) than the short reduct based one. Moreover, due to the rule set reduction, it occupies less memory and classifies new objects faster.

7 Dynamic Reducts

The methods based on calculation of reducts allow to compute, for a given decision table $\mathbf{A}$, the descriptions of all decision classes of $\mathbf{A}$ in the form of decision rules (see previous sections). Unfortunately, decision rules calculated in this way can often be inappropriate to classify unseen cases. We suggest that the rules calculated by means of dynamic reducts are better predisposed to classify unseen cases, because these reducts are the most stable reducts in a process of random sampling of the original decision table.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table. By $\mathbf{P}(\mathbf{A})$ we denote the set of all subtables of $\mathbf{A}$. If $\mathbf{F} \subseteq \mathbf{P}(\mathbf{A})$ then by $DR(\mathbf{A}, \mathbf{F})$ we denote the set

$$RED(\mathbf{A}, d) \cap \bigcap_{\mathbf{B} \in \mathbf{F}} RED(\mathbf{B}, d).$$

Any element of $DR(\mathbf{A},\mathbf{F})$ is called an $\mathbf{F}$-*dynamic reduct* of $\mathbf{A}$.

From the definition of a dynamic reduct it follows that a relative reduct of $\mathbf{A}$ is dynamic if it is also a reduct of all subtables from a given family $\mathbf{F}$. This notion can be sometimes too much restrictive so we apply also a generalization of dynamic reducts - $(\mathbf{F},\varepsilon)$-dynamic reducts, where $\varepsilon \in [0,1]$. The set $DR_\varepsilon(\mathbf{A},\mathbf{F})$, of all $(\mathbf{F},\varepsilon)$-dynamic reducts is defined by
$DR_\varepsilon(\mathbf{A},\mathbf{F}) = \{C \in RED(\mathbf{A},d) :$

$$\frac{card(\{\mathbf{B} \in \mathbf{F} : C \in RED(\mathbf{B},d)\})}{card(\mathbf{F})} \geq 1-\varepsilon\}$$

If $C \in RED(\mathbf{A},d)$ then the number:

$$\frac{card(\{\mathbf{B} \in \mathbf{F} : C \in RED(\mathbf{B},d)\})}{card(\mathbf{F})}$$

is called the *stability coefficient* of the reduct C (relative to $\mathbf{F}$).

Proposition 6. *Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table.*

1. *If $\mathbf{F} = \{\mathbf{A}\}$ then $DR(\mathbf{A},\mathbf{F}) = RED(\mathbf{A},d)$.*
2. *If $\varepsilon_1 \leq \varepsilon_2$ then $DR_{\varepsilon_1}(\mathbf{A},\mathbf{F}) \subseteq DR_{\varepsilon_2}(\mathbf{A},\mathbf{F})$.*
3. *$DR(\mathbf{A},\mathbf{F}) \subseteq DR_\varepsilon(\mathbf{A},\mathbf{F})$, for any $\varepsilon \in [0,1]$.*
4. *$DR_0(\mathbf{A},\mathbf{F}) = DR(\mathbf{A},\mathbf{F})$.*

8 Generalized Dynamic Reducts

From the definition of a dynamic reduct it follows that a relative reduct of any table from a given family $\mathbf{F}$ of subtables of $\mathbf{A}$ can be dynamic if it is also a reduct of the table $\mathbf{A}$. This notion can be sometimes not convenient because we are interested in useful sets of attributes which are not necessarily reducts of the table $\mathbf{A}$. Therefore we have to generalize the notion of a dynamic reduct.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table and $\mathbf{F} \subseteq \mathbf{P}(\mathbf{A})$. By $GDR(\mathbf{A},\mathbf{F})$ we denote the set

$$\bigcap_{\mathbf{B}\in\mathbf{F}} RED(\mathbf{B},d).$$

Elements of $GDR(\mathbf{A},\mathbf{F})$ are called $\mathbf{F}$-*generalized dynamic reducts* of $\mathbf{A}$.

From the above definition it follows that any subset of $\mathbf{A}$ is a generalized dynamic reduct if it is also a reduct of all subtables from a given family $\mathbf{F}$. Analogously to dynamic reducts we define a more general notion of generalized dynamic reducts - $(\mathbf{F},\varepsilon)$-*generalized dynamic reducts*, where $\varepsilon \in [0,1]$. The set $GDR_\varepsilon(\mathbf{A},\mathbf{F})$ of all $(\mathbf{F},\varepsilon)$-generalized dynamic reducts is defined by
$GDR_\varepsilon(\mathbf{A},\mathbf{F}) = \{C \subseteq A :$

$$\frac{card(\{\mathbf{B} \in \mathbf{F} : C \in RED(\mathbf{B},d)\})}{card(\mathbf{F})} \geq 1-\varepsilon\}.$$

If $C \in RED(\mathbf{B}, d)$ (for any $\mathbf{B} \in \mathbf{F}$) then the number:

$$\frac{card(\{\mathbf{B} \in \mathbf{F} : C \in RED(\mathbf{B}, d)\})}{card(\mathbf{F})}$$

is called the *stability coefficient* of the generalized dynamic reduct C (relative to $\mathbf{F}$).

Proposition 7. *Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table.*

1. $DR(\mathbf{A}, \mathbf{F}) \subseteq GDR(\mathbf{A}, \mathbf{F})$.
2. $DR_\varepsilon(\mathbf{A}, \mathbf{F}) \subseteq GDR_\varepsilon(\mathbf{A}, \mathbf{F})$ *for any* $\varepsilon \in [0, 1]$.
3. *If* $\mathbf{A} \in \mathbf{F}$ *then* $DR(\mathbf{A}, \mathbf{F}) = GDR(\mathbf{A}, \mathbf{F})$.

9 Reduct Domain

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table, $\mathbf{F} \subseteq \mathbf{P}(\mathbf{A})$, and let $GDR_\varepsilon(\mathbf{A}, \mathbf{F})$ (for any $\varepsilon \in [0, 1]$) be the set of $(\mathbf{F}, \varepsilon)$-generalized dynamic reducts.

For any $R \in GDR_\varepsilon(\mathbf{A}, \mathbf{F})$ we define the *reduct domain* (denoted by $RD(\mathbf{A}, \mathbf{F}, R)$) as the set:

$$\bigcup \{U_{\mathbf{B}} : R \in RED(\mathbf{B}, d) \text{ and } \mathbf{B} = (U_{\mathbf{B}}, A \cup \{d\}) \in \mathbf{F}\} \cap POS_R(d).$$

The domain of the reduct R is the set of all objects belonging to decision tables $\mathbf{B} \in \mathbf{F}$ satisfying $R \in RED(\mathbf{B}, d)$ on which the decision can be uniquely determined by attributes from R.

The notion of a reduct domain is very important for decision rules generation (see Section 12).

10 Statistical Inference about Dynamic Reducts

Statistics is frequently defined as the science of collecting and studying numerical data in which deductions are made on the assumption that the relationships between a sufficient sample of numerical data are characteristic of those between all such data (see [12], [18], [25]). The data in most statistical problems relate to a sample drawn from some parent population or universe (as it is called in rough set theory). When a sample is used to make inferences about the population, we generally assume that the sample is random. This usually means (when the population is finite) that any individual in the population has an equal chance of being included in the sample. It is desirable that sampling should be as nearly random as possible, although this is often difficult to be achieved in practice, because sometimes we do not have equal access to all elements of the universe. This problem can be described in language of rough set theory. If we want to calculate the set of decision rules based on a given decision table, we would like to construct rules which are proper not only for objects from this decision

table but also for still unknown examples of objects. Dynamic reducts (defined in the previous sections) are calculated with respect to a family $\mathbf{F}$ of subtables created by random samples of a given decision table. The family $\mathbf{F}$ is, of course, only a subfamily of all subtables of the hypothetical universal decision table (including known and unknown objects describing a currently considered aspect of reality). The family of all subtables of the universal decision table is denoted by $\mathbf{G}$. We are interesting in reducts which most frequently appear in the family $\mathbf{G}$, because we expect that the decision rules generated from these reducts are better predisposed to designate a value of the decision for objects from the universal decision table $\mathbf{W}$. In other words, we are interested in the probability of the event that a given $\mathbf{F}$-generalized dynamic reduct R is a reduct for any subtable from $\mathbf{G}$. This probability is denoted by $P_{\mathbf{G}}(R)$ and defined as the quotient $\frac{card(\mathbf{G}_R)}{card(\mathbf{G})}$, where

$$\mathbf{G}_R = \{\mathbf{B} \in \mathbf{G} : R \in RED(\mathbf{B}, d)\}. \tag{7}$$

We would like to select an $\mathbf{F}$-dynamic reduct R for which the probability $P_{\mathbf{G}}(R)$ is not less than the probability $P_{\mathbf{G}}$ for other dynamic reducts. However we cannot calculate $P_{\mathbf{G}}$ for any $\mathbf{F}$-generalized dynamic reduct R because we do not know the whole family $\mathbf{G}$. In this section we show that the so called stability coefficient of any $(\mathbf{F}, \varepsilon)$-generalized dynamic reduct R is a proper measure of the probability $P_{\mathbf{G}}(R)$. Unfortunately, we usually do not have an access to all subtables from the family $\mathbf{G}$, therefore we construct the family $\mathbf{F}$ based only on subtables of a given decision table $\mathbf{A}$. We have to assume that the decision table $\mathbf{A}$ is a representative sample from the universal decision table $\mathbf{W}$.

Theorem 8. *Stability coefficients as maximum likelihood estimator (see* [6]*) Let us assume that*

- $\mathbf{W} = (W, A \cup \{d\})$ *is a universal decision table,*
- $\mathbf{A} = (U, A \cup \{d\})$ *is a given decision table* ($U \subseteq W$)*,*
- $\mathbf{G} = \mathbf{P}(\mathbf{W})$ *is a family, called the parent population,*
- $\mathbf{F} \subseteq \mathbf{P}(\mathbf{A})$*,*
- R *is an* $(\mathbf{F}, \varepsilon)$*-generalized dynamic reduct for some* $\varepsilon \in [0, 1]$*.*

Then we have: the stability coefficient of the $(\mathbf{F}, \varepsilon)$*-generalized dynamic reduct* R *of the decision table* $\mathbf{A}$ *is the maximum likelihood estimator of the probability* $P_{\mathbf{G}}(R)$ *(see* [25]*).*

The maximum likelihood estimator of the probability $P_{\mathbf{G}}(R)$ is denoted by $MLE(P_{\mathbf{G}}(R))$.

Proof: Let us first introduce the simple binomial distribution $X^R_{\mathbf{G}}(\mathbf{B}) : \mathbf{G} \to \{0, 1\}$ (for the family $\mathbf{G}$ and the reduct R) defined for any $\mathbf{B} \in \mathbf{G}$:

$$X^R_{\mathbf{G}}(\mathbf{B}) = \begin{cases} 1 \text{ for } R \in RED(\mathbf{B}, d) (\text{success}) \\ 0 \text{ for } R \notin RED(\mathbf{B}, d) (\text{defeat}) \end{cases} \tag{8}$$

Let $\mathbf{G}^1 = \{\mathbf{B} \in \mathbf{G} : X^R_{\mathbf{G}}(\mathbf{B}) = 1\}$ and $\mathbf{G}^0 = \{\mathbf{B} \in \mathbf{G} : X^R_{\mathbf{G}}(\mathbf{B}) = 0\}$. Now it is easy to observe that the probability P of the success in our binomial distribution is: $P[X^R_{\mathbf{G}}(\mathbf{B}) = 1] = P_{\mathbf{G}}(R)$ and the probability of the defeat in our binomial distribution is: $P[X^R_{\mathbf{G}}(\mathbf{B}) = 0] = 1 - P_{\mathbf{G}}(R)$.

From [25] we know that the probability of a success in our distribution $X^R_{\mathbf{G}}$ may be estimated by taking a sample of subtables from the family $\mathbf{G}$ (for example $\mathbf{F}$) and next using the method of maximum likelihood estimator. The maximum likelihood estimator of success probability in our distribution is the arithmetic mean of values $X^R_{\mathbf{G}}$ for all subtables from the sample (see for instance [25]).

Hence:

$$MLE(P_{\mathbf{G}}(R)) = \frac{\sum\limits_{\mathbf{B}\in\mathbf{F}} X^R_{\mathbf{G}}(\mathbf{B})}{card(\mathbf{F})} =$$

$$= \frac{\sum\limits_{\mathbf{B}\in\mathbf{F}\cap\mathbf{G}^1} X^R_{\mathbf{G}}(\mathbf{B}) + \sum\limits_{\mathbf{B}\in\mathbf{F}\cap\mathbf{G}^0} X^R_{\mathbf{G}}(\mathbf{B})}{card(\mathbf{F})} = \frac{card(\mathbf{F}\cap\mathbf{G}^1)}{card(\mathbf{F})}. \tag{9}$$

From our definition of $\mathbf{F}$-generalized dynamic reducts (Section 8) we conclude that the number $\frac{card(\mathbf{F}\cap\mathbf{G}^1)}{card(\mathbf{F})}$ is equal to the stability coefficient of the generalized dynamic reduct R.
This completes the proof.

□

Remark 1. *Dynamic reducts as the tools for classification*
Theorem 8 is showing that dynamic reducts with large stability coefficients are "good" candidates for decision rules generation. They allow to construct rules with better classification quality of unseen objects than reducts with smaller stability coefficients.

Remark 2. *Minimal size of family* $\mathbf{F}$
It is easy to observe (see the proof of Theorem 8) that for any dynamic reduct R the problem of calculating the stability coefficient of R is equivalent to the problem of calculating an unknown success probability $P_{\mathbf{G}}(R)$ in binomial distribution $X^R_{\mathbf{G}}$. Therefore, the maximum likelihood estimator of the probability $P_{\mathbf{G}}(R)$ is equal to $\frac{card(\mathbf{F}\cap\mathbf{G}^1)}{card(\mathbf{F})}$. For calculating the necessary size of family $\mathbf{F}$, we need to make some assumption about a confidence coefficient: $1 - \alpha$. The confidence coefficient can be understood as a measure of probability estimation correctness (see for instance [18]). From the Moivre-Laplace theorem (see for instance [18] - Theorem 6.7.1) we know that

$$\frac{MLE(P_{\mathbf{G}}(R)) - P_{\mathbf{G}}(R)}{\sqrt{\frac{MLE(P_{\mathbf{G}}(R))\cdot(1-MLE(P_{\mathbf{G}}(R)))}{card(\mathbf{F})}}} \tag{10}$$

has approximately a standard normal distribution. Hence,

$$P\left[-t_\alpha < \frac{MLE(P_{\mathbf{G}}(R)) - P_{\mathbf{G}}(R)}{\sqrt{\frac{MLE(P_{\mathbf{G}}(R))\cdot(1-MLE(P_{\mathbf{G}}(R)))}{card(\mathbf{F})}}} < t_\alpha\right] = 1 - \alpha, \tag{11}$$

where the number t_α is satisfying the equation:

$$1-\alpha=\sqrt{\frac{2}{\pi}}\int_{-t_\alpha}^{t_\alpha}\exp(-\frac{t^2}{2})dt. \tag{12}$$

From equation (11) we have

$$P\left[MLE(P_{\mathbf{G}}(R)-t_\alpha\cdot\sqrt{\frac{MLE(P_{\mathbf{G}}(R))\cdot(1-MLE(P_{\mathbf{G}}(R)))}{card(\mathbf{F})}}<P_{\mathbf{G}}(R)<\right.$$

$$\left.MLE(P_{\mathbf{G}}(R)+t_\alpha\cdot\sqrt{\frac{MLE(P_{\mathbf{G}}(R))\cdot(1-MLE(P_{\mathbf{G}}(R)))}{card(\mathbf{F})}}\right]=1-\alpha. \tag{13}$$

Hence, the $100\cdot(1-\alpha)\%$ confidence interval for $P_{\mathbf{G}}(R)$ is given by

$$MLE(P_{\mathbf{G}}(R))-t_\alpha\cdot\sqrt{\frac{MLE(P_{\mathbf{G}}(R))\cdot(1-MLE(P_{\mathbf{G}}(R)))}{card(\mathbf{F})}}<P_{\mathbf{G}}(R)<$$

$$MLE(P_{\mathbf{G}}(R))+t_\alpha\cdot\sqrt{\frac{MLE(P_{\mathbf{G}}(R))\cdot(1-MLE(P_{\mathbf{G}}(R)))}{card(\mathbf{F})}}. \tag{14}$$

If $\Delta MLE(P_{\mathbf{G}}(R))$ is a maximal acceptable estimation error of $MLE(P_{\mathbf{G}}(R))$, then we conclude from inequality (14) that

$$t_\alpha\cdot\sqrt{\frac{P_{\mathbf{G}}(R)\cdot(1-P_{\mathbf{G}}(R))}{card(\mathbf{F})}}\leq\Delta MLE(P_{\mathbf{G}}(R)). \tag{15}$$

Hence

$$card(\mathbf{F})\geq\frac{t_\alpha^2\cdot P_{\mathbf{G}}(R)\cdot(1-P_{\mathbf{G}}(R))}{(\Delta MLE(P_{\mathbf{G}}(R)))^2}. \tag{16}$$

It is easy to observe that if $P_{\mathbf{G}}(R)=\frac{1}{2}$ then the product $P_{\mathbf{G}}(R)\cdot(1-P_{\mathbf{G}}(R))$ takes the maximum value of $\frac{1}{4}$. Therefore it is enough to require that the size of $\mathbf{F}$ is no less than $\frac{{t_\alpha}^2}{4\cdot(\Delta MLE(P_{\mathbf{G}}(R)))^2}$.

The value of t_α one can read from the table of standardized normal distribution function (see e.g. [12], [25]).

For example, if we take $1-\alpha=0.9$ and $\Delta MLE(P_{\mathbf{G}}(R))=0.05$ than the value t_α is 1.64 and $card(\mathbf{F})\geq\frac{1.64^2}{4\cdot0.05^2}=268.96$.

11 Techniques for Dynamic Reduct Computation

In this section we present a method for computing generalized dynamic reducts and reduct domains. In our method a random set of subtables $\mathbf{F}$ from a given data table $\mathbf{A} = (U, A \cup \{d\})$ is taken and reducts for all these tables are calculated. The number of samples $card(\mathbf{F})$ can be selected by taking into account the minimal family size (see previous section). The method of the random choice of sub-table consists of the two steps. In the first step we randomly choose some numbers with the probability:

$$P(k) = \frac{\binom{n}{k}}{\sum_{i=1}^{n} \binom{n}{i}} \tag{17}$$

where $k = 1, ..., n$ and $n = card(U)$. Next, we randomly choose a sub-table of $\mathbf{A}$ consisting of k objects.

Thus we receive a family $\mathbf{F}$ of decision tables and for each $\mathbf{A} \in \mathbf{F}$ we compute reducts. Next step is to compute the reduct domain for every reduct (as the positive region of the table constructed from the sum of all objects from proper subtables). In the following step reducts with the stability coefficients higher than a fixed threshold are extracted. The reducts distinguished in such a way are treated as the true generalized dynamic reducts of the table $\mathbf{A}$ together with their reduct domains.

We would like to mention another method of dynamic reduct computation for decision tables. In this case we assume that the reduct set $RED(\mathbf{A})$ is already computed. Instead of computing reduct sets for subtables from $\mathbf{F}$ it is enough to check which reducts from $RED(\mathbf{A})$ are also reducts for subtables from $\mathbf{F}$. This can save computing time because time necessary for checking which reducts from $RED(\mathbf{A})$ are $\mathbf{F}$-dynamic reducts of $\mathbf{A}$ is polynomial with respect to the size of $\mathbf{F}$ and $RED(\mathbf{A})$. However this method can be used only in case we are able to compute the set of all reducts.

12 Decision Rules Computed from Dynamic Reducts

After a set of dynamic reducts with their reduct domains has been computed it is necessary to decide how to compute the set of decision rules. We discuss methods based on the decision rule set calculation for any chosen dynamic reduct. The rules for each reduct are calculated separately. For example one can calculate decision rules for any chosen dynamic reduct R. Let us consider the two following methods.

In the first method for any object from the reduct domain R we take the value vector of conditional attributes from R and the corresponding decision of the object. Unfortunately, decision rules generated by this method have poor performance, because the number of objects supporting such rules is usually very small.

In the second method we generate decision rules with minimal number of descriptors (see Section 3) for the table consisting of the object set equal to the reduct domain of R, the conditional attribute set equal to R, and assuming the decision attribute to be the same as in the original decision table. The final decision rule set is equal to the union of all these sets.

When a new object is to be classified, it is first matched against all decision rules from the constructed decision rule set. Next, the final decision is predicted by applying some strategy constructing the final decision from all "votes" of decision rules (see Section 15).

13 Dynamic Rules

From [3] and [4] we know that the quality of unseen object classification based on dynamic reducts (see Section 12) is usually better than the quality of classification based on the whole set of attributes. Therefore one can adopt an idea of dynamic reducts as a method for decision rules computation. By analogy with dynamic reducts, we propose the following method for dynamic rules computation. At the beginning, from a given data table a random set of subtables is chosen. Next the optimal decision rule sets for all these tables are calculated. In the following step the rule memory is constructed where all rule sets are stored. Intuitively, any dynamic rule is appearing in all (or almost all) of experimental subtables. The decision rules can be computed from the so called *k-relative discernibility matrix* used to generate decision rules with the minimal number of descriptors (see Section 3).

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table and $\mathbf{F} \subseteq \mathbf{P}(\mathbf{A})$. A decision rule $r \in \bigcup_{\mathbf{B} \in \mathbf{F}} RUL(\mathbf{B})$ is called an $\mathbf{F}$-*dynamic rule* of $\mathbf{A}$ iff

$$Supp_{\mathbf{B}}(r) \neq 0 \Rightarrow r \in RUL(\mathbf{B}), \text{ for any } \mathbf{B} \in \mathbf{F}.$$

By $DRUL(\mathbf{A}, \mathbf{F})$ we denote the set of all $\mathbf{F}$-dynamic rules of $\mathbf{A}$.

From the definition of dynamic rules it follows that any optimal decision rule for $\mathbf{B} \in \mathbf{F}$ is an $\mathbf{F}$-dynamic rule of $\mathbf{A}$ if it is also an optimal rule for all subtables from a given family $\mathbf{F}$ (having some objects matched by the considered decision rule). This notion can be sometimes too much restrictive, so we apply also a more general notion of a dynamic rule - $(\mathbf{F}, \varepsilon)$-*dynamic rules*, where $\varepsilon \in [0, 1]$. The set $DRUL_{\varepsilon}(\mathbf{A}, \mathbf{F})$ of all $(\mathbf{F}, \varepsilon)$-dynamic rules is defined as

$$DRUL_{\varepsilon}(\mathbf{A}, \mathbf{F}) = \{r \in \bigcup_{\mathbf{B} \in \mathbf{F}} RUL(\mathbf{B}) : \frac{card(\{\mathbf{B} \in \mathbf{F} : r \in RUL(\mathbf{B})\})}{card(\{\mathbf{B} \in \mathbf{F} : Supp_{\mathbf{B}}(r) \neq 0\})} \geq 1 - \varepsilon\}.$$

If $r \in RUL(\mathbf{B})$ (for any $\mathbf{B} \in \mathbf{F}$) then the number:

$$\frac{card(\{\mathbf{B} \in \mathbf{F} : r \in RUL(\mathbf{B})\})}{card(\{\mathbf{B} \in \mathbf{F} : Supp_{\mathbf{B}}(r) \neq 0\})}$$

is called the *stability coefficient* of the dynamic rule r (relatively to $\mathbf{F}$) and it is denoted $SC^{\mathbf{F}}_{\mathbf{A}}(r)$.

Proposition 9. *Let* $\mathbf{A} = (U, A \cup \{d\})$ *be a decision table.*

1. *If* $\mathbf{F} = \{\mathbf{A}\}$ *then* $DRUL(\mathbf{A}, \mathbf{F}) = RUL(\mathbf{A})$.
2. *If* $\varepsilon_1 \leq \varepsilon_2$ *then* $DRUL_{\varepsilon_1}(\mathbf{A}, \mathbf{F}) \subseteq DRUL_{\varepsilon_2}(\mathbf{A}, \mathbf{F})$.
3. $DRUL(\mathbf{A}, \mathbf{F}) \subseteq DRUL_{\varepsilon}(\mathbf{A}, \mathbf{F})$, *for any* $\varepsilon \geq 0$.
4. $DRUL_0(\mathbf{A}, \mathbf{F}) = DRUL(\mathbf{A}, \mathbf{F})$.

Our methods for dynamic rules generation from decision tables are based on the reduct set computation. A random set of subtables from a given data table is taken (see Section 11) and the optimal rules for all these tables are calculated (see Section 3). The time cost of the reduct set computation can be very high when the decision table has too many: objects or attributes or different values of attributes (see Section 3). Therefore we often apply some approximate algorithms to obtain knowledge about optimal rule sets (see Section 12, 15, [38], [62]) and next we use the following proposition to compute stability coefficient of calculated decision rules.

Proposition 10. *Let us assume that*

- $\mathbf{A} = (U, A \cup \{d\})$ *is a decision table, where* $card(U) = n$ *and* $card(A) = m$,
- $r = ((a_1 = v_1) \wedge ... \wedge (a_l = v_l) \Rightarrow (d = v_d)) \in DRUL_{\varepsilon}(\mathbf{A}, \mathbf{P}(\mathbf{A}))$ *for some* $\varepsilon \in [0, 1]$ *, where* $a_i \in A$, $v_i \in V_{a_i}$ *for* $i = 1, ..., l$ $(l \leq m)$ *and* $v_d \in V_d$,
- $h_{-1} = card(H_{-1})$, *where* $H_{-1} = \{u \in U : \forall i \in \{1, ..., l\} : a_i(u) = v_i \wedge d(u) \neq v_d\}$,
- $h_0 = card(H_0)$, *where* $H_0 = \{u \in U : \forall i \in \{1, ..., l\} : a_i(u) = v_i \wedge d(u) = v_d\}$,
- $h_i = card(H_i)$, *where* $H_i = \{u \in U : \forall j \in \{1, ..., l\} \setminus \{i\} : a_j(u) = v_j \wedge a_i(u) \neq v_i \wedge d(u) \neq v_d\}$ *for* $i = 1, ..., l$.

Then we have: the stability coefficient of the $(\mathbf{P}(\mathbf{A}), \varepsilon)$*-dynamic rule* r *of the decision table* $\mathbf{A}$ *can be computed using the following equation:*

$$SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r) = \frac{1}{2^{h_{-1}}} \text{ for } l = 1, \tag{18}$$

and by

$$SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r) = \frac{\prod_{i=1}^{l} \left(2^{h_i} - 1\right)}{2^{h_{-1} + \sum_{i=1}^{l} h_i}} \text{ for } l > 1. \tag{19}$$

One can prove the above proposition using some basic facts from combinatorics and rough set theory.

It is easy to construct an algorithm, based on the equation (18) and (19), calculating the stability coefficient $SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r)$ for the rule r in time $O(m \cdot n)$ and space $O(C)$, where C is a constant.

14 Approximate Rules

One can use approximate decision rules instead of optimal decision rules to construct the classification algorithm for a decision table $\mathbf{A}$ (see Section 3). We have implemented a method for computing approximate rules. We begin with algorithm for synthesis of optimal decision rules from a given decision table (see Section 3). Next, we compute approximate rules from already calculated optimal decision rules. Our method is based on the notion of consistency of a decision rule. The original optimal rule is reduced to an approximate rule with the coefficient of consistency exceeding a fixed threshold.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table and $r_0 \in RUL(\mathbf{A})$. The approximate rule (based on rule r_0) is computed using the following algorithm.

Algorithm 10 *Approximate rule synthesis (by descriptor dropping)*
Input:
1. decision table $\mathbf{A} = (U, A \cup \{d\})$,
2. decision rule $r_0 \in RUL(\mathbf{A})$,
3. threshold of consistency μ_0 *(e.g.* $\mu_0 = 0.9$*).*
Output: the approximate rule r_{app} (based on rule r_0).
Method:
Calculate the coefficient of consistency $\mu_{\mathbf{A}}(r_0)$
if $\mu_{\mathbf{A}}(r_0) < \mu_0$ **then** *STOP (in this case no approximate rule).*
$\mu_{max} = \mu_{\mathbf{A}}(r_0)$ and $r_{app} = r_0$.
while $\mu_{max} > \mu_0$ **do**
begin
$\mu_{max} = 0$
for $i = 1$ **to** *the number of descriptors from* $Pred(r_{app})$ **do**
begin
$r = r_{app}$.
Remove i-th descriptor from $Pred(r)$.
Calculate the coefficient of consistency $\mu_{\mathbf{A}}(r)$ *and* $\mu = \mu_{\mathbf{A}}(r)$.
if $\mu > \mu_{max}$ **then** $\mu_{max} = \mu$ *and* $i_{max} = i$.
end
if $\mu_{max} > \mu_0$ *then remove* i_{max} *-th conditional descriptor from* r_{app}.
end
return r_{app}.
□

It is easy to see that the time and space complexity of Algorithm 10 are of order $O(l^2 \cdot m \cdot n)$ and $O(C)$, respectively (where l is the number of conditional descriptors in the original optimal decision rule r_0 and C is a constant).

The approximate rules, generated by the above method, can help to extract interesting laws from decision table. By applying approximate rules instead of optimal rules one can slightly decrease the quality of classification of objects from the training set but we expect in return to receive more general rules with the higher quality of classification for new objects (see [9]).

15 Negotiations among Rules

Suppose we have a set of decision rules. In most cases it is necessary to decide how to resolve conflicts between sets of rules classifying tested objects to different decision classes. In this section we present several methods for constructing the measure called *the strength of rule set*. The strength of rule set is a rational number belonging to [0,1] representing the importance of the sets of decision rules relative to the considered tested object.

Let us assume that:

- $\mathbf{W} = (W, A \cup \{d\})$ is a universal decision table,
- $\mathbf{A} = (U, A \cup \{d\})$ is a given decision table ($U \subseteq W$),
- $u_t \in W$ is a tested object,
- $Rul(X_j)$ is a set of all calculated basic decision rules for $\mathbf{A}$, classifying objects to the decision class X_j (where $v_d^j \in V_d$)
- $MRul(X_j, u_t) \subseteq Rul(X_j)$ is a set of all decision rules from $Rul(X_j)$ matching tested objects u_t.

We define several measures for the rule set $MRul(X_j, u_t)$ depending on the number of rules from this set matching tested object, the number of objects supporting decision rules from this set and the stability coefficient of rules.

1. *The simple strength* of a decision rule set is defined by

$$SimpleStrength(X_j, u_t) = \frac{card(MRul(X_j, u_t))}{card(Rul(X_j))}. \tag{20}$$

2. *The maximal strength* of a decision rule set is defined by

$$MaximalStrength(X_j, u_t) = max_{r \in MRul(X_j, u_t)} \left\{ \frac{Supp_{\mathbf{A}}(r)}{card(|d = v_d^j|_{\mathbf{A}})} \right\}. \tag{21}$$

3. *The basic strength* of a decision rule set is defined by

$$BasicStrength(X_j, u_t) = \frac{\sum\limits_{r \in MRul(X_j, u_t)} Supp_{\mathbf{A}}(r)}{\sum\limits_{r \in Rul(X_j)} Supp_{\mathbf{A}}(r)}. \tag{22}$$

4. *The global strength* of a decision rule set is defined by

$$GlobalStrength(X_j, u_t) = \frac{card\left(\bigcup\limits_{r \in MRul(X_j, u_t)} |Pred(r)|_{\mathbf{A}} \cap |d = v_d^j|_{\mathbf{A}} \right)}{card(|d = v_d^j|_{\mathbf{A}})}. \tag{23}$$

5. *The stability strength* of a decision rule set is defined by

$$StabilityStrength(X_j, u_t) = max_{r \in MRul(X_j, u_t)} \{SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r)\}. \tag{24}$$

6. *The maximal stability strength* of a decision rule set is defined by

$$MaximalStabStrength(X_j, u_t) = max_{r \in MRul(X_j, u_t)} \frac{Supp_{\mathbf{A}}(r)}{card(|d = v_d^j|_{\mathbf{A}})} \cdot SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r). \tag{25}$$

7. *The basic stability strength* of a decision rule set is defined by

$$BasicStabStrength(X_j, u_t) = \frac{\sum\limits_{r \in MRul(X_j, u_t)} Supp_{\mathbf{A}}(r) \cdot SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r)}{\sum\limits_{r \in Rul(X_j)} Supp_{\mathbf{A}}(r) \cdot SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r)}. \tag{26}$$

8. *The global stability strength* of a decision rule set (denoted *GlobStabS*) is defined inductively by

$$\begin{cases} GlobStabS(\{r\}) = SC_{\mathbf{A}}^{\mathbf{P}(\mathbf{A})}(r), \text{ for } r \in MRul(X_j, u_t) \\ GlobStabS(R \setminus \{r\} \cup \{r\}) = \\ \quad = GlobStabS(\{r\}) + GlobStabS(R \setminus \{r\}) - \\ \quad\quad GlobStabS(\{r\}) \cdot GlobStabS(R \setminus \{r\}), \\ \quad\quad\quad \text{for } R \neq \emptyset \text{ and } R \subseteq MRul(X_j, u_t). \end{cases} \tag{27}$$

The maximal strength of a decision rule set is similar to the strength of rule presented in [31] and [57]. The basic strength of a decision rule set is similar to the strength of rule presented in [22] and [23]. The global strength of a decision rule set is similar to the strength of rule presented in [31], [22] and [23]. Measures of strengths of rules defined above can be applied in constructing classification algorithms (see next section).

16 General Scheme of Classification Algorithm

In this section we present general scheme of classification algorithms based on methods and techniques described in previous sections. One can choose options presented in the general scheme (see Figure 7) for the construction of a particular classification algorithm.

17 Summary

In this Chapter we discussed some methods for extracting laws from data. We presented some techniques based on standard rough set methods (see [44], [45], [52]) like reduct set and rule set computation. We also described dynamic techniques e.g. dynamic reducts and dynamic rules computation (see [3], [4], [5], [6]), that give potentially more general decision rules more capable to new cases classification. For the case of larger data tables we proposed a genetic algorithm based method for computation of a approximate reduct set. We also proposed some discretization algorithms of real value attributes that can be used in the

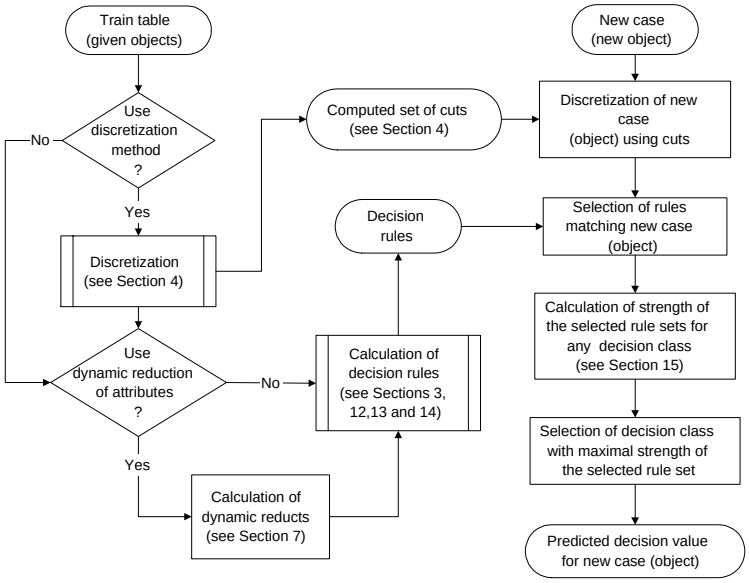

Fig. 7. General scheme of classification algorithm

process of data preprocessing. All presented methods can be successfully used together in order to construct a good classification algorithms.

Acknowledgments: This work was partially supported by the grants of Polish National Committee for Scientific Research (KBN) No. $8T11C01011$ and No. $8T11C02417$ and also by the *ESPRIT* project 20288 CRIT-2.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, I.: Fast Discovery of Association Rules, *Proceedings of the Advances in Knowledge Discovery and Data Mining.* AAAI Press/The MIT Press, CA (1996) 307–328.
2. Almuallim, H., Dietterich, T. G.: Learning with many irrelevant features, *Proceedings of the Ninth National Conference on Artificial Intelligence* (1991) 547–552.
3. Bazan, J., Skowron, A., Synak, P.: Discovery of Decision Rules from Experimental Data, *Proceedings of the Third International Workshop on Rough Sets and Soft Computing.* San Jose, California (1994) 526–533.
4. Bazan, J., Skowron, A., Synak, P.: Dynamic reducts as a tool for extracting laws from decision tables, *Proceedings of the Eighth International Symposium on Methodologies for Intelligent Systems (ISMIS'94), Lecture Notes in Artificial Intelligence 869.* Berlin: Springer-Verlag (1994) 346–355,.
5. Bazan, J., Skowron, A., Synak, P.: Market data analysis: A rough set approach. *ICS Research Report 6/94,* Warsaw University of Technology (1994).

6. Bazan, J.: Dynamic reducts and statistical inference, *Proceedings of Information Processing and Management of Uncertainty on Knowledge Based Systems (IPMU-96)*, July 1-5, Granada, Spain, Universidad de Granada, vol. III, (1996) 1147–1152.
7. Bazan, J., Nguyen, H. S., Nguyen, T. T., Skowron, A., Stepaniuk, J.: Synthesis of Decision Rules for Object Classification, Orowska E. (ed.): *Incomplete Information: Rough Set Analysis*. Heidelberg: Physica-Verlag (1998) 23-57.
8. Bazan, J.: Discovery of Decision Rules by Matching New Objects Against Data Tables. *Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC-98)*, Warsaw, June 22-26 (1998), *Lecture Notes in Artificial Intelligence 1424*. Berlin: Springer-Verlag (1998) 521-528.
9. Bazan, J.: A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Table, Polkowski L., Skowron A. (eds.): *Rough Sets in Knowledge Discovery*. Heidelberg: Physica-Verlag (1998) 321-365.
10. Bloedorn, E., Michalski, R. S.: Data Driven Constructive Induction in AQ17-PRE: A Method and Experiments, *Proceedings of the Third International Conference on Tools for AI*. San Jose, CA (1991)
11. Brown, E. M.: *Boolean reasoning*. Dordrecht: Kluwer (1990).
12. Brownlee, K. A.: *Statistical theory and methodology in science and engineering*. New York: John Wiley&Sons (1965).
13. Cestnik, B., Kononenko, I., Bratko, I.: ASSISTANT 86: A Knowledge Elicitation Tool for Sophisticated Users, *Proceedings of EWSL-87*. Bled, Yugoslavia (1987) 31–47.
14. Clark, P., Niblett, T.: The CN2 Induction Algorithm, *Machine Learning 3*. Kluwer Academic, Boston, MA (1989) 261–284.
15. Cykier, A.: Prime Implicants of Boolean Functions - Applications and Methods of Computations (in Polish), MSc Thesis, University of Warsaw, Warsaw, Poland (1997).
16. Downton, A. C., Tregidgo, R. W. S., Leedham, C. G.: Recognition of handwritten British postal addresses. From Pixels to Features III. Frontiers in Handwriting Recognition North-Holland (1992) 129–144.
17. Dzeroski, S.: *Handling Noise in Inductive Logic Programming*. MS Thesis, Dept. of EE and CS, University of Ljubljana, Slovenia (1991).
18. Fisz, M.: *Probability theory and mathematical statistics*, New York (1961).
19. Fahlman, S. E., Lebiere, C.: The Cascade-Correlation Learning Architecture, in *Advances in Neural Information Processing Systems,* vol. II. Morgan Kaufmann, San Mateo, CA (1990).
20. Friedman, J.: Smart user's guide. *Technical Report 1.* Laboratory of Computational Statistics, Department of Statistics, Stanford University (1984).
21. Goldberg, D. E.: GA in Search, Optimisation, and Machine Learning. Addison-Wesley (1989).
22. Grzymała-Busse, J. W.: LERS - a system for learning from examples based on rough sets. In R. Słowiński, (ed.) *Intelligent Decision Support,* Kluwer Academic Publishers, Dordrecht, Boston, London (1992) 3–18.
23. Grzymała-Busse, J. W.: A new version of the rule induction system LERS. Fundamenta Informaticae **31** (1997) 27–39.
24. Holland, J. H.: Adaptation in Natural and Artificial Systems. The MIT Press, Cambridge (1992).
25. Keeping, E. S.: *Introduction to statistical inference*. Prinston, New Jersey: D.Van Nostrand Company, Inc. (1962).

26. Kira, K., Rendell, L. A.: A practical approach to feature selection, In D.Sleeman (ed.), *Proceedings of the Ninth International Workshop on Machine Learning (ML92)*, Morgan Kaufmann (1992) 249–256.
27. Kittler, J.: Feature selection and extraction, In Young and Fu (ed.), *Handbook of pattern recognition and image processing*. New York: Academic Press (1996).
28. Kodratoff, Y., Michalski, M. (ed.): *Machine Learning vol. III*. Morgan Kaufmann, San Mateo, CA (1990).
29. Michalski, R., Carbonell, J. G. and Mitchel, T. M. (ed): *Machine Learning vol. I*. Tioga/Morgan Kaufmann, Los Altos, CA (1983).
30. Michalski, R., Carbonell, J. G. and Mitchel, T. M. (ed): *Machine Learning vol. II*. Morgan Kaufmann, Los Altos, CA (1986).
31. Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N.: The Multi-Purpose Incremental Learning System AQ15 and its Testing to Three Medical Domains, *Proceedings of AAAI-86*. Morgan Kaufmann, San Mateo, CA (1986) 1041–1045.
32. Michalski, R., Wnęk, J.: Constructive Induction: An Automated Improvement of Knowledge Representation Spaces for Machine Learning, *Proceedings of a Workshop on Intelligent Information Systems, Practical Aspect of AI II*, Augustów (Poland) (1993) 188–236..
33. Michie, D., Spiegelhalter, D. J., Taylor, C. C.: *Machine learning, neural and statistical classification*. England: Ellis Horwood Limited (1994).
34. Mienko, R., Słowiński, R., Stefanowski, J., Susmaga, R.: RoughFamily - software implementation of rough set based data analysis and rule discovery techniques, Tsumoto S. (ed.), *Proceedings of the Fourth International Workshop on Rough Sets, Fuzzy Sets and Machine Discovery*, Tokyo, November 6-8 (1996), 437–440.
35. Mollestad, T.: A rough set approach to default rules data mining. PhD Thesis, supervisor J. Komorowski, Norvegian Institute of Technology, Trondheim, Norway (1996)
36. Muggelton, S. (ed.): *Inductive logic programming*. Academic Press (1992).
37. Nguyen, H. S., Nguyen, S. H., Skowron, A.: Searching for features defined by hyperplanes, Z.W. Ras, M. Michalewicz (ed.), *Proceedings of Ninth International Symposium on Methodologies for Intelligent Systems (ISMIS-96)*, Zakopane, Poland, June 10-13, (1996). Lecture Notes in Artificial Intelligence vol. 1079, Springer, Berlin (1996) 366–375; see also: *ICS Research Report 16/95*, Warsaw University of Technology.
38. Nguyen, S. H., Nguyen, H. S.: Some efficient algorithms for rough set methods, *Proceedings Information Processing and Management of Uncertainty on Knowledge Based Systems (IPMU-96)*, July 1-5, Granada, Spain, Universidad de Granada, vol. III, (1996) 1451–1456.
39. Nguyen, H. S., Skowron, A.: Quantization of real value attributes, *Proceedings of Second Joint Annual Conf. on Information Sciences*, Wrightsville Beach, North Carolina, September 28 - October 1, USA (1995) 34–37.
40. Nguyen, S. H., Skowron, A., Synak, P., Wróblewski, J.: Knowledge Discovery in Databases: Rough Set Approach. *Proc. of The Seventh International Fuzzy Systems Association World Congress*, vol. II, pp. 204-209, IFSA97, Prague, Czech Republic (1997).
41. Nguyen, H. S.: Discretization of Real Value Attributes: Boolean reasoning Approach. Ph.D. Thesis, Warsaw University, Warsaw, Poland (1997).
42. Nguyen, T., Świniarski, R., Skowron, A., Bazan, J., Thyagarajan, K.: Application of Rough Sets, Neural Networks and Maximum Likelihood for Texture Classification Based on Singular Value Decomposition, *Proceedings of the Third Interna-*

tional Workshop on Rough Sets and Soft Computing. San Jose, California (1994) 332–339.

43. Oehrn, A., Komorowski, J.: ROSETTA - A rough set tool kit for analysis of data. *Proceedings of the Fifth International Workshop on Rough Sets and Soft Computing (RSSC'97) at the Third Joint Conference on Information Sciences (JCIS'97),* Research Triangle Park, NC, March 2-5 (1997) 403–407.
44. Pawlak, Z.: *Rough sets: Theoretical aspects of reasoning about data.* Dordrecht: Kluwer 1991.
45. Pawlak, Z., Skowron, A.: A rough set approach for decision rules generation, *ICS Research Report 23/93*, Warsaw University of Technology, *Proceedings of the IJCAI'93 Workshop W12: The Management of Uncertainty in AI*, France 1993.
46. Piasta, Z., Lenarcik, A., Tsumoto S.: Machine discovery in databases with probabilistic rough classifiers. In: S. Tsumoto, S. Kobayashi, T. Yokomori, H. Tanaka and A. Nakamura (eds.), *Proceedings of The fourth International Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery (RS96FD), November 6–8,* The University of Tokyo (1996) 353–359
47. Polkowski, L., Skowron, A.: Synthesis of decision systems from data tables. In: T. Y. Lin and N. Cecerone (eds.), Rough Sets and Data Mining. Analysis for Imprecise Data, Kluwer Academic Publishers, Boston, London, Dordrecht (1997) 259–299.
48. Quinlan, J. R.: Induction of Decision Trees, *Machine Learning 1.* Kluwer Academic, Boston, MA (1986) 81–106.
49. Quinlan, J. R.: *C4.5: Programs for Machine Learning.* San Mateo, California: Morgan Kaufmann (1993).
50. De Raedt, L.: *Interactive Theory Revision. An Inductive Logic Programming.* Academic Press (1992).
51. Skowron, A., Rauszer, C.: The Discernibility Matrices and Functions in Information Systems. In R. Słowiński (ed.), *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory.* Dordrecht: Kluwer (1992) 331–362.
52. Skowron, A.: Boolean reasoning for decision rules generation, *Proceedings of the 7-th International Symposium ISMIS'93*, Trondheim, Norway 1993, In Komorowski J. and Ras Z. (ed.), *Lecture Notes in Artificial Intelligence, vol. 689.* Springer-Verlag (1993) 295–305.
53. Skowron, A.: A synthesis of decision rules: Applications of discernibility matrix properties, *Proceedings of the Workshop Intelligent Information Systems, Augustów (Poland),* 7–11 June, 1993.
54. Słowiński, R. (ed.): Intelligent Decision Support. *Handbook of Applications and Advances of the Rough Sets Theory.* Dordrecht: Kluwer (1992).
55. Słowiński, R., Stefanowski, J.:'RoughDAS' and 'RoughClass' software implementations of the rough set approach, Słowiński R. (ed.): *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory.* Dordrecht: Kluwer (1992) 445–456.
56. Thrun, S. B, Bala, J., Bloedorn, E., Bratko, I., Cestnink, B., Cheng, J., DeJong, K. A., Dzeroski, S., Fahlman, S. E., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R., S., Mitchell, T., Pachowicz, P., Vafaie, H., Van de Velde, W., Wenzel, W., Wnęk, J., and Zhang, J.: The MONK's Problems: A Performance Comparison of Different Learning Algorithms. *Technical Report,* Carnegie Mellon University (1991).

57. Tsumoto, S., Tanaka H.: Incremental learning of probabilistic rules from clinical databases. *Proceedings Information Processing and Management of Uncertainty on Knowledge Based Systems (IPMU-96)*, July 1–5, Granada, Spain, Universidad de Granada, vol. II, (1996) 1457–1462.
58. Utgoff, P. E.: Incremental Learning of Decision Trees, *Machine Learning, vol. IV*. Kluwer Academic, Boston, MA (1990) 161–186.
59. Vafaie, L. G., De Jong, K.: Improving the Performance of a Rule Induction System Using Genetic Algorithm, *Proceedings of the First International Workshop on Multistrategy Learning*. Harpers Ferry WV, George Mason University, Center for Artificial Intelligence (1991) 305–315.
60. Van De Velde, W.: IDL, or Taming the Multiplexer, *Proceedings of the 4th European Working Session on Learning*. Pitman, London (1989).
61. Wnęk, J., Michalski, R. S.: Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments, *Proceedings of the IJCAI-91 Workshop on Evaluating and Changing Representation,* K. Morik, F. Bergadano and W. Buntine (ed.). Sydney, Australia (1991) 13–22.
62. Wróblewski, J.: Finding minimal reducts using genetic algorithm (extended version), *Proceedings of Second Joint Annual Conference on Information Sciences*, Wrightsville Beach, North Carolina, 28 September - 1 October, USA, (1995) 186–189; see also: *ICS Research Report 16/95*, Warsaw University of Technology.
63. Wróblewski, J.: Theoretical Foundations of Order-Based Genetic Algorithms. *Fundamenta Informaticae*, vol. 28 (3, 4), pp: 423-430. IOS Press, (1996).
64. Wróblewski, J.: Genetic algorithms in decomposition and classification problem. In: L. Polkowski, A. Skowron (eds.). Rough Sets in Knowledge Discovery. Physica Verlag, (1998).
65. Wróblewski, J.: Covering with reducts – a fast algorithm for rule generation, *Proceedings of RSCTC'98*, Springer-Verlag (LNAI 1424), Berlin Heidelberg (1998) 402 – 407.
66. Ziarko, W., Shan, N.: An incremental learning algorithm for constructing decision rules, *Proceedings of the International Workshop on Rough Sets and Knowledge Discovery*. Banff. (1993) 335–346.
67. Ziarko, W.: Variable Precision Rough Set Model. Journal of Computer and System Sciences **40** (1993) 39–59.