Zeszyty Naukowe Politechniki Białostockiej

INFORMATYKA

Zeszyt 1

Wydawnictwo Politechniki Białostockiej
Białystok 2002

Redaktor naukowy:
dr hab. Jarosław Stepaniuk, prof. PB

Recenzenci:
dr hab. inż. Teodor Breczko, prof. UW-M – 3, 4
prof. dr hab. inż. Adam Grzech – 12
prof. dr hab. Mieczysław A. Kłopotek – 10
prof. dr hab. Antoni Mazurkiewicz – 8
prof. dr hab. Mirosława Grażyna Mirkowska-Salwicka – 1, 7
prof. dr hab. Wojciech Mokrzycki – 13, 14
prof. dr hab. Sergiej Nowikow – 5, 6
prof. dr hab. Andrzej Skowron – 9, 16
dr Zenon Sosnowski – 2, 15
prof. dr hab. inż. Aleksander Zgrzywa – 11

ISSN 1644-0331

Opracowanie, redakcja techniczna i druk:
Dział Wydawnictw i Poligrafii Politechniki Białostockiej

SPIS TREŚCI

CONTENTS

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Anna Borowska[1]

DETERMINING TRANSITION PROBABILITIES IN PROBABILISTIC ALGORITHMS

Abstract: The main problem of the paper is related to the algebraic method for determining transition probabilities in probabilistic algorithms interpreted in finite structures. The correctness of this method is based on a lemma stating that the determinant of a matrix (being of a special form) is different from zero. The paper contains two proofs of this lemma, formulated without a proof in [3].

Key words: probabilistic algorithm, probabilistic program, random generation, final state, looping state, not looping state

1. Introduction

Statistics plays an important part both in our everyday life and in the science. It permits "deriving knowledge" included in huge sets of data and studying different correlations and relationships. Statistical technologies support different kinds of researches and diagnosis. Results and analyses of results obtained in this way can be used in solutions of different kind of problems in the future. This motivates us to concentrate our interests on the so-called "probabilistic algorithms", which take into consideration dynamically changing situations (of analyzed occurrence), and also different possibilities of developing of the actual situation. The algorithms consider not only state in a given moment, but also sequences of successive changes. The analysis of changes occurring one after another (e.g., analysis of changes of the weather in a given area) causes most problems in detecting regularities in them. We replace determinism with probabilistic solution because determinism does not take random factors into consideration and it does not account for

[1] Department of Computer Science Technical University of Bialystok;
15-351 Bialystok; Wiejska 45 A; Poland, e-mail: anbor@ii.pb.bialystok.pl

uncertainty. However, uncertainty is related with every researched occurrence in the real world.

Summing up, probabilistic algorithms are useful in analysis of incomplete information. We use probabilistic technologies in simulation of real processes, for which empirical collected of data is very expensive, time-consuming or even completely impossible.

In the present paper iterative probabilistic algorithms are understood as iterative programs using typical program constructions:

$x := a$,

begin ... end,

if ... then ... else ...,

while ... do ...,

and two probabilistic constructions:

$x := ?$,

either$_{(\rho)}$ ... or ...

interpreted as follows: the first construction corresponds to a random generation of a value of the variable x and the first part of the second construction is chosen with the probability ρ (the second one is chosen with the probability $1-\rho$).

For a finite interpretation of an algorithm $K(x_1,\ldots,x_h)$ in a finite set $A=(a_1,\ldots,a_t)$ we have $n=t^h$ possible valuations of variables from $X^h=(x_1,\ldots,x_h)$. Let us denote them by $v_1,\ldots,v_n$.

The main fact, which uses essentially the aforementioned LEMMA, concerns the following procedure of algebraization (cf. Lemma 3.2 in [3]) consisting in associating with every program $\boldsymbol{K}$ a $n\times n$ matrix K where k_{ij} denotes the probability of passing from an initial valuation v_i to a final valuation v_j.

For each probabilistic program $\boldsymbol{K}(x_1,\ldots,x_h)$ interpreted in a finite universe $A=(a_1,\ldots,a_t)$ we can construct, in an effective way, a $n\times n$ matrix $K=\left[k_{ij}\right]_{i,j=1,\ldots,n}$ (where $n=t^h$), such as if the probabilities of appearing $v_1,\ldots,v_n$ as the input valuations are equal $p_1,\ldots,p_n$, respectively, then the probabilities $q_1,\ldots,q_n$ of appearing $v_1,\ldots,v_n$ as output valuations satisfy:

$$[q_1,\ldots,q_n]=[p_1,\ldots,p_n]\circ K.$$

This fact can be illustrated as follows:

$$v_i \xrightarrow{p_i} \boldsymbol{K} \xrightarrow{q_j} v_j,\ i,j=1,\ldots,n.$$

The construction of the matrix K for a given program $\boldsymbol{K}$ is inductive with respect to the number of program constructions used in $\boldsymbol{K}$. The most difficult case is when $\boldsymbol{K}$ is of the form while γ do $\boldsymbol{M}$.

Suppose (inductively) that we have given the matrix for the program $\boldsymbol{M}$. If denote by $[\gamma?]$ subprogram while $\neg\gamma$ do $x := x$; then the matrix K for the program $\boldsymbol{K}$ is defined as follows (cf. [3]):

$$K = \sum_{i=0}^{\infty} K_i ,$$

where K_i denotes the matrix corresponding to the program:

begin $\underbrace{[\gamma?] \text{ if } \gamma \text{ then } \boldsymbol{M}; \ \ldots \ ; [\gamma?] \text{ if } \gamma \text{ then } \boldsymbol{M};}_{\text{i-times}}$ $[\neg\gamma?]$; end;

Now, we describe an effective method which enables us to determine the matrix K in a finite number of steps is described in [3]. The starting point is the equivalence of the program $\boldsymbol{K}$ of the form while γ do $\boldsymbol{M}$ and the following program:

if γ then begin $\boldsymbol{M}$; $\underbrace{\text{while } \gamma \text{ do } \boldsymbol{M};}_{\boldsymbol{K}}$ end;.

This equivalence motivates the following equation:

$$K = I_{\neg\gamma} + I_{\gamma} \circ M \circ K ,$$

where

$$I_{\gamma[i,j]} = \begin{cases} 1 & \text{for } i = j \text{ and the valuation } v_i \text{ satisfies the condition } \gamma \\ 0 & \text{otherwise} \end{cases}$$

This equation may be written in the equivalent way as

$$(I - I_{\gamma} \circ M) \circ K = I_{\neg\gamma} .$$

Since the determinant of the matrix $I - I_{\gamma} \circ M$ (denoted by $\det(I - I_{\gamma} \circ M)$) may be equal to 0, to determine the matrix K, we need a more subtle analysis described in details in [3].

Speaking informally, an effective classification is realized and suitable positions of matrix M are determined in the following way:

- First, we determine positions equal to 1 (corresponding to valuations which are not satisfying γ (final states)).

- Next, we put the 0's at positions corresponding to initial valuations, for that all computations are infinite (looping states).
- The remaining part of the matrix M (not looping states) is of the form mentioned in LEMMA, which enables to determine its positions.

1.1. Example

Let $\boldsymbol{K}$ denote the program

while $x \neq 3$ do

if $x = 1$ then $x := ?$;

including one variable interpreted in finite 3-element universe. Let $\left[\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3}\right]$ be distribution of probability of variable x for instruction $x := ?$.

If denote by $\boldsymbol{M}$ the subprogram if $x = 1$ then $x := ?$ then it is easy to determine, that M is of the form $\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (cf. [3])).

Since the determinant of the matrix $I\text{-}I_\gamma \circ M = \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is equal to zero, thus we determine the matrix K using mentioned procedure. In the case of the program $\boldsymbol{K}$ we have the following states:

- $v_0(x) = 1$ (not looping),
- $v_1(x) = 2$ (looping),
- $v_2(x) = 3$ (not looping),

thus the matrix K is of the form

$$K = \begin{bmatrix} 0 & 0 & ? \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The remaining part of K denoted by "?" is of the form mentioned in LEMMA, which enables to determine this position.

□

Thus, the analysis of probabilistic algorithms essentially depends on the LEMMA. It is formulated in [3] without a proof. This paper contains two proofs of the LEMMA: algebraic and analytical. The sketch of analytical proof has been known since 1993. The question concerning the elementary proof remains open until 1999 (this proof is contained in [1]).

1.2. LEMMA

The determinant of the matrix:

$$M = \begin{vmatrix} 1-m_{11} & -m_{12} & \dots & -m_{1n} \\ -m_{21} & 1-m_{22} & \dots & -m_{2n} \\ \dots & \dots & \dots & \dots \\ -m_{n1} & -m_{n2} & \dots & 1-m_{nn} \end{vmatrix}, \text{ where}$$

$$m_{ij} \geq 0, \qquad i,j = 1,\dots,n; \tag{1.1}$$

$$\sum_{j=1}^{n} m_{ij} < 1, \qquad i = 1,\dots,n \tag{1.2}$$

is positive.

□

Element m_{ij} in the LEMMA means the probability of the transition from the state i to the state j.

2. An algebraic proof of the LEMMA

For more readable notation of the proof, we can formulate the LEMMA as follows:

Let M_n be a $n \times n$ matrix M. Denote by $IA(M,n)$ (inductive assumption) the properties denoted above by (1.1) and (1.2), i.e., $m_{ij} \geq 0$ for $i,j = 1,\dots,n$ and $\sum_{j=1}^{n} m_{ij} < 1$ for $i = 1,\dots,n$.

If the matrix M_n has the properties $IA(M,n)$ then $\det M_n > 0$.

Let us denote elements of the matrix M_n by t_{ij}, where $i,j = 1,\dots,n$.

We get the matrix:

$$M_n = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2n} \\ \dots & \dots & \dots & \dots \\ t_{n1} & t_{n2} & \dots & t_{nn} \end{bmatrix} = \begin{vmatrix} 1-m_{11} & -m_{12} & \dots & -m_{1n} \\ -m_{21} & 1-m_{22} & \dots & -m_{2n} \\ \dots & \dots & \dots & \dots \\ -m_{n1} & -m_{n2} & \dots & 1-m_{nn} \end{vmatrix},$$

where properties $IA(M,n)$ may be written:

- $\sum_{j=1}^{n} t_{ij} > 0$, for $i = 1,\dots,n$; (because $1 - \sum_{j=1}^{n} m_{ij} > 0$, $i = 1,\dots,n$) (2.1)
- $t_{ii} > 0$, for $i = 1,\dots,n$; (from (1.1) and (1.2)); (2.2)
- $t_{ij} \le 0$, for $i \neq j$, $i,j = 1,\dots,n$; (from (1.1)). (2.3)

The proof proceeds by induction on the number n – the common length of rows and of columns. Let us assume that the thesis of the LEMMA is valid for each $n \times n$ matrix satisfying condition $IA(M,n)$.

The fact is obvious for $n = 1$.

Let us consider a $(n+1)\times(n+1)$ matrix M_{n+1} with following properties $IA(M,n+1)$:

- $\sum_{j=1}^{n+1} t_{ij} > 0$, for $i = 1,\dots,n+1$;
- $t_{ii} > 0$, for $i = 1,\dots,n+1$;
- $t_{ij} \le 0$, for $i \neq j$, $i,j = 1,\dots,n+1$.

After the first step of the Gauss Elimination Method we get the following matrix:

$$M'_{n+1} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \dots & t_{1j} & \dots & t_{1n+1} \\ 0 & t'_{22} & t'_{23} & \dots & t'_{2j} & \dots & t'_{2n+1} \\ 0 & t'_{32} & t'_{33} & \dots & t'_{3j} & \dots & t'_{3n+1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & t'_{i2} & t'_{i3} & \dots & t'_{ij} & \dots & t'_{in+1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & t'_{n+12} & t'_{n+13} & \dots & t'_{n+1j} & \dots & t'_{n+1n+1} \end{bmatrix}$$

where $t'_{ij} = t_{ij} - t_{1j} \cdot \frac{t_{i1}}{t_{11}}$ for $i = 2,\ldots,n+1$, $j = 1,\ldots,n+1$.

We shall prove that the $n \times n$ matrix:

$$S = \begin{bmatrix} s_{11} & \ldots & s_{1n} \\ \ldots & \ldots & \ldots \\ s_{n1} & \ldots & s_{nn} \end{bmatrix} = \begin{bmatrix} t'_{22} & t'_{23} & \ldots & t'_{2n+1} \\ t'_{32} & t'_{33} & \ldots & t'_{3n+1} \\ \ldots & \ldots & \ldots & \ldots \\ t'_{n+12} & t'_{n+13} & \ldots & t'_{n+1n+1} \end{bmatrix}$$

satisfies the conditions $IA(M,n)$.

Consider the first row of the matrix S.

Observe that $-\frac{t_{i1}}{t_{11}} \geq 0$ for $i = 2,\ldots,n+1$ and let $\alpha_i = -\frac{t_{i1}}{t_{11}}$ $i = 2,\ldots,n+1$.

This means that (2.1) holds for the first row of S:

$$s_{11} + \ldots + s_{1n} = t'_{22} + \ldots + t'_{2n+1} = t_{22} + \ldots + t_{2n+1} + \alpha_2(t_{12} + \ldots + t_{1n+1}) > 0.$$

Similarly, we can prove (2.1) for $i = 2,\ldots,n$.

Now, consider the first element of the diagonal in S:

$$s_{11} = t'_{22} = t_{22} - t_{12} \cdot \frac{t_{21}}{t_{11}}$$

Denote by $\varepsilon_i = 1 - \sum_{j=1}^{n} m_{ij}$, where $i = 1,\ldots,n$. By virtue of (1.2) we have $\varepsilon_i > 0$.

Then

- $t_{11} = 1 - m_{11} = m_{12} + m_{13} + \ldots + m_{1n+1} + \varepsilon_1$,
- $t_{12} = -m_{12}$,
- $t_{21} = -m_{21}$,
- $t_{22} = 1 - m_{22} = m_{21} + m_{23} + \ldots + m_{2n+1} + \varepsilon_2$.

Thus

$$s_{11} = t'_{22} = m_{23} + \ldots + m_{2n+1} + \varepsilon_2 + m_{21}\left(1 - \frac{m_{12}}{m_{12} + m_{13} + \ldots + m_{1n+1} + \varepsilon_1}\right) > 0$$

because

$$1 \geq 1 - \frac{m_{12}}{m_{12} + m_{13} + \ldots + m_{1n+1} + \varepsilon_1} \geq 0$$

This means that the first element of the diagonal of S is positive.

Analogously, we can prove that all elements on the diagonal of S are positive, i.e., $s_{ii} > 0$ for $i = 2,\ldots,n$. Thus S satisfies (2.2).

In an analogous way we can show (2.3) (the remaining elements of the matrix are less than or equal to zero, because we subtract positive values from negative values):

$$s_{ij} = t'_{i+1\,j+1} \le 0 \quad \text{for } i \ne j,\ i,j = 1,\ldots,n;$$

Thus, we can apply the inductive assumption to the above matrix S, being part of the matrix M_{n+1}, because the matrix satisfies conditions $IA(M,n)$.

Thus $\det S > 0$. Therefore,

$\det M_{n+1} = t_{11} \cdot \det S > 0$.

By virtue of the principle of induction, the Lemma is valid for each $n \times n$ matrix satisfying condition $IA(M,n)$.

□

3. The analytical proof of the LEMMA

We start with an illustration for $n = 2$.

Let $M_2 = \begin{bmatrix} 1-m_{11} & -m_{12} \\ -m_{21} & 1-m_{22} \end{bmatrix}$ be a matrix.

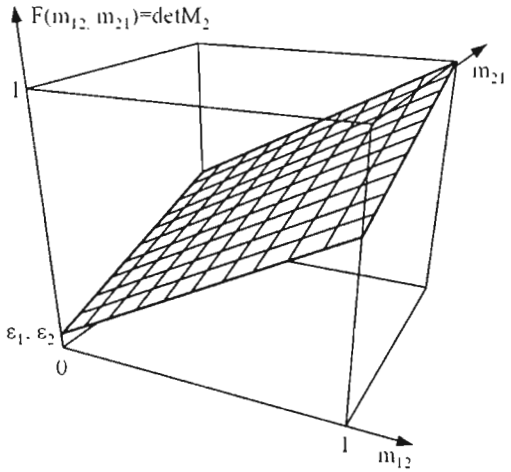
Denote by $\begin{cases} \varepsilon_1 = 1-(m_{11}+m_{12}) \\ \varepsilon_2 = 1-(m_{21}+m_{22}) \end{cases}$. By virtue of (1.2) we have $\varepsilon_1, \varepsilon_2 > 0$.

Consequently, $\det M_2$ can be written as:

$$\det M_2 = \begin{vmatrix} m_{12}+\varepsilon_1 & -m_{12} \\ -m_{21} & m_{21}+\varepsilon_2 \end{vmatrix} = m_{12}\varepsilon_2 + m_{21}\varepsilon_1 + \varepsilon_1\varepsilon_2 .$$

Since $m_{12}, m_{21} \ge 0$, then $\det M_2 > 0$.

We illustrate this fact on the figure below. $\det M_2$ is treated as a function of m_{12}, m_{21}, $F(m_{12},m_{21}) = \det M_2$ $(0 \le m_{12}, m_{21} < 1)$.



Ill. 1. Graph of function $F(m_{12},m_{21}) = \det M_2$ for $m_{12},m_{21} \geq 0$ and $\varepsilon_1,\varepsilon_2 > 0$

It is easy to observe, that F has the minimum at $(0,0)$, moreover, the minimum is positive. Thus all values of F are positive. This means that $\det M_2 > 0$ provided that the elements of M_2 satisfy (1.1) and (1.2)

Let M_n be a $n \times n$ matrix.

The proof proceeds by induction on n. Assume that:

$$\begin{vmatrix} 1-m_{11} & -m_{12} & \dots & -m_{1k} \\ -m_{21} & 1-m_{22} & \dots & -m_{2k} \\ \dots & \dots & \dots & \dots \\ -m_{k1} & -m_{k2} & \dots & 1-m_{kk} \end{vmatrix} > 0$$

for all the determinants of this form (satisfying the assumptions of LEMMA) for $k < n$.

Denote by $\varepsilon_i = 1 - \sum_{j=1}^{n} m_{ij}$, where $i = 1,\dots,n$.

Thus $1 - m_{ii} = m_{i1} + m_{i2} + \dots + m_{ii-1} + m_{ii+1} + \dots + m_{in} + \varepsilon_i$.

By virtue of (1.2) we have $\varepsilon_i > 0$.

Thus the determinant of the matrix M_n can be written as:

$$\det M_n = \begin{vmatrix} m_{12}+\ldots+m_{1n}+\varepsilon_1 & -m_{12} & \ldots & -m_{1n} \\ -m_{21} & m_{21}+m_{23}+\ldots+m_{2n}+\varepsilon_2 & \ldots & -m_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ -m_{n1} & -m_{n2} & \ldots & m_{n1}+\ldots+m_{n,n-1}+\varepsilon_n \end{vmatrix},$$

where $\varepsilon_i > 0$, $i = 1,\ldots,n$.

Let us denote the vector by
$\vec{m} = (\overline{m_{11}},\ldots,m_{1n},m_{21},\overline{m_{22}},\ldots,m_{2n},\ldots,m_{n1},\ldots,\overline{m_{nn}})$, where $\overline{m_{ii}}$ $(i = 1,\ldots,n)$ means, that $\vec{m}$ does not contain the element m_{ii}.

Let us treat the determinant of M_n as the function of $\vec{m}$:
$\det M_n = F(\vec{m}) = F(\overline{m_{11}},\ldots,m_{1n},m_{21},\overline{m_{22}},\ldots,m_{2n},\ldots,m_{n1},\ldots,\overline{m_{nn}})$.

Let us determine the derivative $\dfrac{\partial \det M_n}{\partial m_{12}}$.

Since element m_{12} appears in the first row only, we obtain:

$$\frac{\partial \det M_n}{\partial m_{12}} = \begin{vmatrix} 1 & -1 & \ldots & 0 \\ -m_{21} & m_{21}+m_{23}+\ldots+m_{2n}+\varepsilon_2 & \ldots & -m_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ -m_{n1} & -m_{n2} & \ldots & m_{n1}+\ldots+m_{n,n-1}+\varepsilon_n \end{vmatrix}$$

After adding the first and second columns we obtain:

$$\frac{\partial \det M_n}{\partial m_{12}} = \begin{vmatrix} 1 & 0 & \ldots & 0 \\ -m_{21} & m_{23}+\ldots+m_{2n}+\varepsilon_2 & \ldots & -m_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ -m_{n1} & -(m_{n1}+m_{n2}) & \ldots & m_{n1}+\ldots+m_{n,n-1}+\varepsilon_n \end{vmatrix}$$

After applying the Laplace's development to the first row we obtain:

$$\frac{\partial \det M_n}{\partial m_{12}} = \begin{vmatrix} m_{23}+\ldots+m_{2n}+\varepsilon_2 & -m_{23} & \ldots & -m_{2n} \\ -(m_{31}+m_{32}) & m_{31}+\ldots+m_{3n}+\varepsilon_3 & \ldots & -m_{3n} \\ \ldots & \ldots & \ldots & \ldots \\ -(m_{n1}+m_{n2}) & -m_{n3} & \ldots & (m_{n1}+m_{n2})+\ldots+m_{n,n-1}+\varepsilon_n \end{vmatrix}$$

If we denote:

$$\begin{array}{llll} m'_{22}=1-(m_{23}+\ldots+m_{2n}+\varepsilon_2) & m'_{23}=m_{23} & \ldots & m'_{2n}=m_{2n} \\ m'_{32}=m_{31}+m_{32} & m'_{33}=1-(m_{31}+\ldots+m_{3n}+\varepsilon_3) & \ldots & m'_{3n}=m_{3n} \\ \ldots & \ldots & \ldots & \ldots \\ m'_{n2}=m_{n1}+m_{n2} & m'_{n3}=m_{n3} & \ldots & m'_{nn}=1-(m_{n1}+\ldots+m_{n,n-1}+\varepsilon_n) \end{array}$$

then we can write:

$$\frac{\partial \det M_n}{\partial m_{12}} = \begin{vmatrix} 1-m'_{22} & -m'_{23} & \ldots & -m'_{2n} \\ -m'_{32} & 1-m'_{33} & \ldots & -m'_{3n} \\ \ldots & \ldots & \ldots & \ldots \\ -m'_{n2} & -m'_{n3} & \ldots & 1-m'_{nn} \end{vmatrix}.$$

It's easy to verify that this determinant satisfies the assumptions of LEMMA, e.g., for $i=2$ we have:

$$m'_{22}+m'_{23}+\ldots+m'_{2n}=1-(m_{23}+\ldots+m_{2n}+\varepsilon_2)+m_{23}+\ldots+m_{2n}=1-\varepsilon_2<1.$$

Analogously, we can show this fact for $i=3,\ldots,n$. In this way we have obtained the $(n-1)\times(n-1)$ determinant of a $(n-1)\times(n-1)$ matrix being in the form investigated in Lemma, therefore by the inductive assumption we obtain $\frac{\partial \det M_n}{\partial m_{12}}>0$.

Analogously, we can prove that:

$\frac{\partial \det M_n}{\partial m_{ij}}>0$, where $i,j=1,\ldots,n$, and $i\neq j$.

Similarly to the case of two variables we shall argue that the function $F(\vec{m})$ has its minimum value at the point $(0,\ldots,0)$, i.e. $F(\vec{m})>F(0,\ldots,0)$.

By the definition of F we have:

$F(0,\ldots,0) = \varepsilon_1 \cdot \ldots \cdot \varepsilon_n$;

Now, let us consider Taylor's formula for the function F around the point $(0,\ldots,0)$ for $n = 1$:

$$F(\vec{m}) = F(0,\ldots,0) + \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \frac{\partial F(\theta\vec{m})}{\partial m_{ij}} \cdot m_{ij} = \varepsilon_1 \cdot \ldots \cdot \varepsilon_n + \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \frac{\partial F(\theta\vec{m})}{\partial m_{ij}} \cdot m_{ij}, \quad (3.1)$$

Since $\sum_{j=1}^{n} m_{ij} < 1$, $i = 1,\ldots,n$, and $0 < \theta < 1$, then $\sum_{j=1}^{n} \theta m_{ij} < 1$, for $i = 1,\ldots,n$.

Since $\frac{\partial \det M_n}{\partial m_{ij}} = \frac{\partial F(\vec{m})}{\partial m_{ij}} > 0$, $i,j = 1,\ldots,n$ and $i \neq j$, we have that $\frac{\partial F(\theta\vec{m})}{\partial m_{ij}} > 0$, where $i,j = 1,\ldots,n$ and $i \neq j$. Then, according to (3.1):

$F(\vec{m}) > \varepsilon_1 \cdot \ldots \cdot \varepsilon_n > 0$.

Since at the point $(0,\ldots,0)$ the function has the minimum (in its domain), with positive value at the point $(0,\ldots,0)$, then for all $\vec{m}$ belonging to the domain of the function F $F(\vec{m}) > 0$ and therefore $\det M_n = F(\vec{m}) > 0$.

Thus, we have proved the LEMMA for the $n \times n$ matrix M_n, i.e., $\det M_n > 0$.

□

References

[1] Borowska A., *Wyznaczanie prawdopodobieństw przejść w algorytmach probabilistycznych.*, master thesis, University of Bialystok 1999, (in Polish).

[2] Busłowski A., *Determining transition probabilities in finite state Marcov Chains.*, master thesis, Department of Computer Science Technical University of Bialystok, 1995, (in Polish).

[3] Dańko W., *The set of probabilistic algorithmic formulas valid in a finite structure is decidable with respect to its diagram.*, Foundamenta Informaticae, vol. 19, 3-4, 1993, (417-431).

[4] Dańko W., *Rought set methods and probabilistic technics are complementary (in analysing changes in large data bases)*, to appear in Foundamenta Informaticae.

[5] Jankowski T., *Linear algebra*, Gdańsk: Politechnika Gdańska, 1997.

[6] Maurin K., *Analiza*, Warszawa: Wydawnictwo Naukowe PWN, 1991, (in Polish).

WYZNACZANIE PRAWDOPODOBIEŃSTW PRZEJŚĆ W ALGORYTMACH PROBABILISTYCZNYCH

Streszczenie: Poniższa praca zawiera dwa dowody lematu opublikowanego w pracy [3] bez dowodu. Algebraiczny fakt rozważany w lemacie jest punktem wyjściowym dla metody wyznaczania prawdopodobieństw przejść w iteracyjnych algorytmach probabilistycznych interpretowanych w skończonych dziedzinach. Dotyczy on niezerowości wyznacznika macierzy o pewnej specyficznej postaci.

Słowa kluczowe: algorytm probabilistyczny, program probabilistyczny, stan końcowy, stan pętlący, stan nie pętlący

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Teodora Dimitrova-Grekow, Walery Sołowiew[1]

PROBLEMY MINIMALIZACJI MOCY POBIERANEJ PRZEZ UKŁADY LOGIKI PROGRAMOWALNEJ

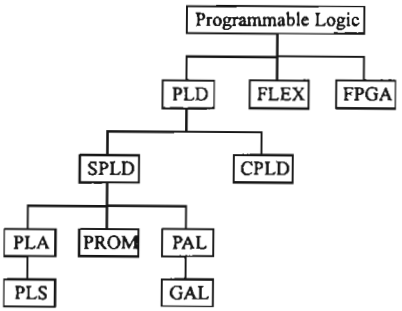
Streszczenie: Przedstawiona praca prezentuje problematykę energii w schematach logiki programowalnej. Pokazane są najpopularniejsze aspekty nowoczesnych metod minimalizacji mocy pobieranej przez schematy sekwencyjne. Analiza różnych metod pokazuje w najszerszej perspektywie możliwości badań dotyczących zarządzania energią pobieraną, konsumowaną i traconą.

Słowa kluczowe: energia konsumowana, utraty energii, ocena mocy, logika programowalna, kodowanie stanów, architektury energooszczędne

1. Wprowadzenie

Logika Programowalna znajduje coraz szersze zastosowanie w projektowaniu systemów elektronicznych i automatycznych. Przestrzeń jej zastosowania wzrasta w bardzo szybkim tempie. Trudno można odnaleźć współczesny system sterowania, w którym nie jest użyty chociażby jeden układ scalony typu PLD (Programmable Logic Device) albo FPGA (Field Programmable Gate Array).

[1] Wydział Informatyki, Politechnika Białostocka ul. Wiejska 45A, PL 15-300 Białystok, e-mail: teodora@ii.pb.bialystok.pl



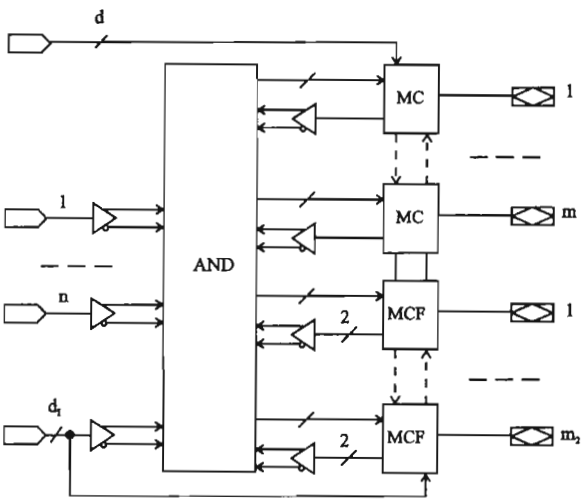
Rys. 1. Klasyfikacja układów logiki programowalnej

W rozwoju mikroelektroniki energia konsumowana spełniała zawsze bardzo istotną rolę. Równolegle z logiką programowalną rozwinęły się bardzo mocno metody, odwzorowujące pobieraną przez te układy energię. Specyfika większości opisanych w tym artykule metod jest ściśle związana ze strukturami układów.

Aby pokazać różnorodność typów układów logiki programowalnej, na rysunku 1 przedstawiono ich obowiązującą klasyfikację.

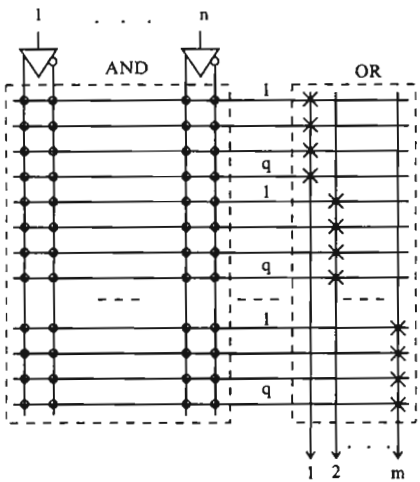
Rysunek 2 prezentuje przykładową strukturę układu PAL. Chociaż ten schemat nie odwzorowuje specyficznych charakterystyk innych struktur i podstruktur (patrz rys.1), można dzięki niemu zrozumieć niektóre przedstawione poniżej metody optymalizacji poboru i straty energii.

Dokładna struktura wewnętrzna układów PLA i PAL (na poziome łącz elektrycznych) pokazana jest na rysunku 3.

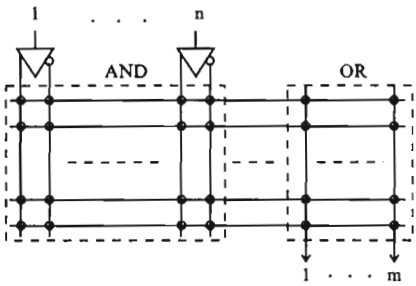


Rys. 2. Schemat blokowy struktury układu PAL

a)



b)



⊕ – łącze programowalne

* – łącze nie programowalne

Rys. 3. a) Struktura układów PAL; b) Struktura układów PLA

W niniejszym artykule rozpatrywane są metody z różnych poziomów technologicznych. Naszym celem jest usystematyzowanie najpopularniejszych znanych metod, aby pokazać perspektywy rozwoju dalszych badań naukowych w kierunku projektowania energooszczędnych i niezawodnych układów i urządzeń. Do opisania układów logiki programowalnej użyto, ze względu na przejrzystość, modelu automatu skończonego.

2. Analiza mocy na podstawie entropii [14], [22]

Metodą tą śledzi się procesy na poziomie bramek elektronicznych, by ocenić moc i wcześnie ostrzec o ewentualnych problemach związanych z energią. Jako miara średniej aktywności danego zaprogramowanego układu wykorzystywana jest entropia. Metoda ta została zaimplementowana i przetestowana na różnych standartowych przykładach.

Entropia sieci automatów skończonych równa się ich własnej informacji o komponentach sieci:

$$H(N)=H(A^1,A^2,..., A^k)= I(A^1, A^2,..., A^k)=$$

$$=\Sigma I(A^i)-\Sigma I(A^i;A^j)+ \Sigma I(A^i;A^j)+...+(-1)^{k-1}\Sigma(A^i;A^j, ..,A^k) \quad (1)$$

Gdzie A^i jest i-tym komponentem automatu skończonego A, charakteryzującego się sieciami wejścia X^i, wyjścia Y^i, stanów wewnętrznych S^i, następnych stanów δ^i i funkcji wyjściowych λ^i; przyjęte jest także i<j<l...<k.

Jest to granica dolna entropii. Granica górna to:

$$H(N)= \Sigma_i I(A^i) = \Sigma_i H(A^i) \quad (2)$$

Na tej podstawie można rozróżnić dwa podejścia: pierwsze z nich bierze pod uwagę zbiorczą entropię osobnych komponentów przyjętych modeli. Drugie podejście uwzględnia również informację wymienioną między komponentami automatu skończonego i jest ono podejściem bardziej precyzyjnym. Ocena ta nie jest jednak pełna i wystarczająco dokładna, ponieważ nie odwzorowuje specyfiki technologicznego wykonania różnych układów scalonych. Nie jest brana również pod uwagę lokalizacja poszczególnych mikrokomórek, co ma duży wpływ na wydzielaną energię.

3. Metody oparte na strukturze logicznej

3.1. Kodowanie stanów

Metoda ta jest jednym z najpopularniejszych podejść. Celem jej jest zmniejszanie aktywności przełączeń poprzez zmniejszanie liczby przełączających się bramek. Definiowane są kodowania, żeby przedzielić kod binarny do każdego stanu symbolicznego.

Ze względu na różnorodność kodowania stanów można je podzielić według grup:

- Metody oparte na teorii grafów [3], [5], [11], [36]. Teoria grafów jest wykorzystywana do opisania stanów i ich relacji. Na tej podstawie jest robiona ocena najkorzystniejszego kodowania stanów. Niektóre z pokazanych wyników są bardzo obiecujące, ale żadna z metod nie jest uniwersalna. Istnieją modyfikacje tego podejścia [17], [24], [25] które biorą pod uwagę grafy prawdopodobieństw przejścia przez dany stan, dzięki czemu osiągają lepsze wyniki. Mimo to można powiedzieć, że problem z wyborem kryterium oceny nie jest rozwiązany.
- Metody oparte na modelach Markova [4], [12], [21]. Autorzy próbują uzyskać optymalne kryteria oceny. W pracach tych pokazane są teoretyczne granice dla uśrednionej odległości Hamminga.
- Metody oparte na sieciach neuronowych [5]. Jest to propozycja kombinacji charakterystyk symulacji i sieci neuronowych Hopfielda. Graf automatu skończonego jest mapowany na sieć neuronową i sformułowana jest funkcja podziału energii. Wyniki eksperymentów są bardzo dobre.

- Metody oparte na algorytmach genetycznych [18]. Tu pojawia się równoległość algorytmów genetycznych, co znacznie wzbogaca rozwiązania zadań w wieloprocesorowym środowisku. Bardzo efektowne są także eksperymenty, korzystające ze specyficznych dla algorytmów genetycznych reprodukcji, skrzyżowań i mutacji.
- Multikodowanie [1], [2]. Każdemu stanowi może zostać przypisany więcej niż jeden kod. Zwiększa to możliwość wyboru liczby przełączających się równocześnie bramek, w zależności od kodu poprzedniego stanu. Takie podejście jednak prowadzi do zwiększenia liczby bitów, uczestniczących w każdym kodzie. Oprócz tego wymagana jest dodatkowa logika do zarządzania multikodami.

3.2. Mapowanie [6], [16], [23]

Często może się zdarzyć, że równocześnie aktywnych jest kilka stanów, zlokalizowanych w tym samym obszarze układu scalonego. Podobna sytuacja powoduje intensywne wydzielanie energii i może doprowadzić do spalenia układu. Mapowanie polega na znalezieniu optymalnego usytuowania każdego stanu albo podschematu automatu skończonego na układzie scalonym. Dodatkowym utrudnieniem pracy układu scalonego na poziome sygnałów elektrycznych są niewykorzystane węzły (patrz rys. 2). Dodają one dodatkową pojemność pasożytniczą. Można to ominąć puszczając a priori sygnał aktywujący; potem optymalnie rekonfigurują się dane [16], [26]. Jednak pozostaje problem z wyborem sygnału wejściowego.

4. Metody oparte na strukturze fizycznej

4.1. Energooszczędne architektury [8], [9], [10], [15], [23], [26]

Metody te mają na celu zapewnienie niskiego pobierania mocy w VLSI. Wykorzystane są techniki z pipeline-architektury, pracujące nawet z obniżonym napędem zasilania. Zaprojektowane są także architektury giętkie, projektowane do niskoenergetycznej pracy; opracowane są efektywne sposoby adresacji pamięci, aby uniknąć użycia tradycyjnych bramek, których konsumpcja energii jest bardzo duża. Pojawiły się nowe FPGA (Field Programmable Gate Array) – architektury z zaimplementowaną online arytmetyką Field Programmable On-line oPerators (FFOP). Ich przeznaczeniem jest jednoukładowa implementacja numerycznych algorytmów dla energooszczędnych procesów i aplikacji cyfrowego sterowania.

4.2. Wybór komponentów [7], [20]

Używane w strukturach bramki i tranzystory mają duży wpływ na pobór mocy. Niestety, wybór elementów jest zależny również od wielu faktorów. Dlatego też w tej dziedzinie jest jeszcze dużo do zrobienia.

4.3. Niskie zasilanie [8], [10], [19]

Niektóre projekty eksperymentują prawie krytyczne niskie zasilanie układów. Jest niewiele osiągnięć. Wyniki badań chociaż w niewielkim stopniu przemawiają pozytywnie za kontynuacją pracy.

5. Metody oparte na bezpośredniej kontroli układów w trakcie pracy

5.1. Selektywne taktowanie [1], [2]

Ponieważ w automatach skończonych oddzielne podstruktury nie zawsze pracują w tym samym czasie, często zdarzają się sytuacje, w których dany podschemat nie jest aktywny, ale jest cały czas taktowany. Te przypadki mają uregulować opisane w tym podpunkcie metody: włączanie i wyłączanie poszczególnych podschematów układu scalonego w zależności od tego, czy jest dany schemat w tej chwili potrzebny (czy ma być aktywny). Rozłączenie lokalnej linii taktowania oszczędza niemało energię. Jest przy tym, oczywiście, niezbędna dodatkowa logika sterowania, co obciąża dodatkowo system.

5.2. Selekcja sygnałów

Metoda ta jest podobna do poprzedniej, z tym że szczególną uwagę zwraca się na formę sygnału. Taktuje się nie typowym, ale dopasowanym do funkcji danego układu sygnałem. Analogicznie do poprzedniej metody potrzebna jest dodatkowa logika sterowania.

6. Zakończenie

Spróbowaliśmy pokazać współczesny stan techniki pod kątem oszczędzania energii i omijania problemów akumulowania niepotrzebnej mocy w układzie. Chociaż artykuł jest krótkim przeglądem zagadnienia, mamy nadzieję, że zainicjuje nowe pomysły i przyczyni się do rozwoju dziedzin zajmujących się zmniejszeniem

mocy pobieranej w układach LP. Jest to bardzo istotny punkt dla energooszczędności, jak również dla bezpiecznej pracy schematów przy podwyższonej prędkości taktowania.

Literatura

[1] Xunwei Wu, Pedram M. Low power sequential circuit design using priority encoding and clock gating. Proceedings of ISLPED'00: ACM. 2000, pp.143-8

[2] Wu X, Pedram M, Wang L. Multi-code state assignment for low power design. JP, IEE Proc: CDS, vol.147, no.5, Oct. 2000, pp.271-5.

[3] Bacchetta P, Daldoss L, Sciuto D, Silvano C. Low-power state assignment techniques for finite state machines. IEEE ISCS. Proc. vol.2, 2000, pp.641-4

[4] Marculescu D, Marculescu R, Pedram M. Theoretical bounds for switching activity analysis in finite-state machines.IEEE Trans. on VLSI Sys. June 2000, pp.335-9

[5] Ahmad I, Dhodhi MK. State assignment of finite-state machines. JP, IEE Proc.-E CDT Jan.2000, pp.15-22

[6] Zhi-Hong W. En-Cheng L. Jianbang L. Ting-Chi W. Power minimization in LUT-based FPGA technology mapping. Proc of the ASP-DAC IEEE 2001, pp.635-40. Piscataway, NJ, USA.

[7] Wolff FG. Knieser MJ. Weyer DJ. Papachristou CA. High-level low power FPGA design methodology. Proc of the IEEE 2000 NAECON 2000 pp.554-9. Piscataway, NJ, USA.

[8] Garcia A. Burleson W. Danger JL. Low power digital design in FPGAs: a study of pipeline architectures implemented in a FPGA using a low supply voltage to reduce power consumption. 2000 IEEE ISET pp.561-4

[9] Yonghee Im. Kaushik Roy. A novel high-performance predictable circuit architecture for the deep sub-micron era. Proc of the IEEE 2000 Custom Integrated Circuits Conference.

[10] Takeuchi K. Satoh S. Imamiya K. Sakui K. A source-line programming scheme for low-voltage operation NAND flash memories. IEEE J of SSC May 2000, pp.672-81.

[11] Ranjan RK, Singhal V, Somenzi F, Brayton RK. On the optimization power of retiming and resynthesis transformations. CP, 1998 IEEE/ACM IC on CAD. Digest of Technical Papers., pp.402-7. New York, NY, USA.

[12] Marculescu D, Marculescu R, Pedram M. Theoretical bounds for switching activity analysis in finite-state machines.CP, Proc. 1998 ISLPED, pp.36-41

[13] Koegst M, Franke G, Rulke S, Feske K. Multi-criterial state assignment for low power FSM design. PC, Proc.24th EUROMICRO Conference IEEE CS, 1998, pp.261-8 vol.1,USA

[14] Kashirova L, Tveretina O. Entropy-based design of low power FSMs. CP, Proc.24thEUROMICRO Conf IEEE, 1998, pp.188-91 vol.1

[15] Takahashi H. Mizushima S. A 1.2 V, 30 MIPS, 0.3 mA/MIPS and 200 MIPS, 0.58 mA/MIPS digital signal processors. IEICE Tran on Electronics, vol.E83-C, no.2, Feb. 2000, pp.179-85. Japan.

[16] Jan-Min H. Feng-Yi C. TingTing H. A re-engineering approach to low power FPGA design using SPFD. Proc 1998 35th DAC. IEEE. 1998, pp.722-5. New York, NY, USA.

[17] Tisserand A. Marchal P. Piguet C., An on-line arithmetic based FPGA for lowpower custom computing., Field Programmable Logic and Applications. 9th International Workshop,FPL'99. pp.264-73. Germany.

[18] Chattopadhyay S, Chaudhuri PP. Genetic algorithm based approach for integrated state assignment and flipflop selection in finite state machine synthesis. CP Proc 11th IC on VLSI Design,IEEE CS 1997, pp.522-7.

[19] Bloch M. Lauterbach C. Weber W. High efficiency charge pump circuit for negative high voltage generation at 2V supply voltage. ESSCIRC '98. Proc of the 24th ESSCIRC

[20] Katkoori S. Vemuri R. Simulation based architectural power estimation for-PLA-based controllers. 1996 IS on L PED. IEEE. 1996, pp.121-4. New York, NY, USA

[21] Surti P, Chao L-F. Controller power estimation using information from behavioral description. [Conference Paper] 1996 IEEE, ISCAS. Part vol.4, 1996, pp.679-82

[22] Tuagi A. Entropic bounds on FSM switching. IEEE Trans. on VLSISys, vol.5, no.4, Dec. 1997, pp.456-64.

[23] Hartenstein R. Herz M. Hoffmann T. Nageldinger U., Using the KressArray for reconfigurable computing. SPIE-Int. SOE Proc of SI SOE, vol.3526, 1998, pp.150-61.

[24] Ming-Der Shieh, Wann-Shyang Ju, Ming-Hwa Sheu. Low-power state assignment for asynchronous finite state machines. CP, Proc of the 39th MS on CS IEEE vol.3, 1996, pp.1325-8 vol

[25] Koegst M, Franke G, Feske K. State assignment for FSM low power design. [Conference Paper] Proc EURO-DAC '96. EURO-VHDL '96 Soc. Press. 1996, pp.28-33.

[26] Eisenring M. Teich J. Interfacing hardware and software. Field-Programmable Logic and Applications. From FPGAs to Computing Paradigm. 8th IW, FPL'98, pp.520-4, Germany.

PROBLEMS OF MINIMIZATION OF POWER CONSUMED BY PROGRAMMABLE LOGIC DEVICES

Summary: The problems of the consumed in the sequential logic power also deferent methods of their solving are addressed. Also the low-power techniques, used in different levels of the technology are considered. The aim of the work is to systematize the most popular, known methods and to show the ways of development it at the branch of the programmable logic.

Key words: power consumption, power dissipation, power estimation, low power, programmable logic, state assignment, clock gating, low power architectures

Artykuł zrealizowano w ramach pracy badawczej własnej, W/II/4/2001.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Wiktor Jakowluk, Michał Czech[1]

ADEKWATNOŚĆ I EFEKTYWNOŚĆ PLANU KOMPLETNEGO ORAZ HARTLEYA W ZASTOSOWANIU DO PROGNOZOWANIA NATĘŻENIA OŚWIETLENIA

Streszczenie: Celem pracy jest przeprowadzenie analizy adekwatności i efektywności planu trójwartościowego kompletnego oraz Hartleya do badania natężenia oświetlenia, w dowolnym punkcie, na płaszczyźnie roboczej modelu pomieszczenia. W tym celu porównano średnie błędy kwadratowe z średnimi odchyłkami kwadratowymi.

Słowa kluczowe: wielokrotne odbicia, planowanie doświadczeń, średni błąd kwadratowy, średnia odchyłka kwadratowa

1. Wprowadzenie

W pracy przeprowadzono analizę porównawczą planów trójwartościowych, kompletnego oznaczanego jako PS/DK – 3^i oraz planu Hartleya oznaczanego jako PS/DS-P: Ha_i, ze względu na ich adekwatność oraz efektywność, w zastosowaniu do wyznaczania natężenia oświetlenia. Zmienne wejściowe oznaczono w następujący sposób: $x_1 = \rho_1$ – współczynnik odbicia powierzchni sufitu, $x_2 = \rho_2$ – współczynnik odbicia powierzchni ścian, $x_3 = h_m$ – wysokość usytuowania opraw oświetleniowych nad płaszczyzną roboczą, x_4 – odcięta położenia punktu na płaszczyźnie roboczej modelu pomieszczenia, x_5 – rzędna położenia punktu na płaszczyźnie roboczej modelu pomieszczenia.

Założono $r \geq 3$ – krotne powtórzenie pomiarów wartości wielkości wyjściowej (z) dla każdego układu planu doświadczenia, oprócz centralnego układu planu, dla którego r = 10. Na tej podstawie wykonane zostały pomiary wielkości wyj-

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok

ściowej $z^{(u)}{}_j = E_{3j}^{(u)}$ – natężenie oświetlenia na płaszczyźnie roboczej modelu pomieszczenia, w punkcie opisanym przez zmienne wejściowe x_4, x_5, gdzie: $u = 1,2,\ldots,n$, dla u – tego układu planu doświadczenia, natomiast n zależy od przyjętego planu doświadczenia; $j = 1,\ldots,r$.

Funkcję obiektu badań przyjęto w postaci wielomianu algebraicznego stopnia drugiego zawierającego składniki liniowe, kwadratowe oraz interakcje o postaci:

$$z = a_0 + a_1x_1 + \ldots + a_ix_i + a_{11}x_1^2 + \ldots + a_{ii}x_i^2 + a_{12}x_1x_2 + \ldots + a_{i-1,i}x_{i-1}x_i \qquad (1)$$

$$N_b = \frac{1}{2}(i+1)(i+2)$$

gdzie:

a_i – współczynniki funkcji aproksymującej z,

N_b – liczba współczynników wielomianu aproksymującego,

i – liczba wielkości wejściowych x_i, gdzie $i = 5$.

Opis modelu pomieszczenia oraz stanowiska pomiarowego można znaleźć w pracy współautora [2].

2. Plan doświadczenia

Charakterystykę obiektu badań, schemat rozmieszczenia punktów pomiarowych na płaszczyźnie roboczej makiety pomieszczenia oraz ocenę normalności rozkładu zmiennej wyjściowej (z) zamieszczono w p. 3 oraz 4 [1].

Do wyznaczania rozkładu natężenia oświetlenia na płaszczyźnie roboczej makiety pomieszczenia wykorzystano plan statyczny zdeterminowany kompletny PS/DK – 3^i oraz plan polisekcyjny specjalny Hartleya PS/DS-P: Ha_i. Liczba wielkości wejściowych $i = 5$, [3].

2.1. Plan kompletny trójwartościowy

Funkcję obiektu badań przyjęto jako wielomian algebraiczny stopnia drugiego (1). Współczynniki wielomianu aproksymującego ($N_b = 21$) wyznaczono metodą najmniejszych kwadratów. Założono poziom istotności $\alpha = 0,05$, natomiast liczbę układów planu $n = 3^i = 243$ (plan trójwartościowy dla $i = 5$). Na podstawie otrzymanych wyników obliczeń przeprowadzono testy istotności współczynników re-

gresji. Liczbę współczynników wielomianu aproksymującego ustalono na $N_b = 12$. Dokładniejszą analizę planu kompletnego przeprowadzono w p. 5.1 [1].

Tabela 1

Przedziały ufności i testy istotności współczynników regresji wielokrotnej dla ($N_b = 12$)

współczynniki regresji zmiennej (z); $R^2 = 0{,}95796$						
wielkość wejściowa	współczynnik regresji a_i	błąd standardowy S_{ai}	$t_i = a_i/S_{ai}$	poziom istotności α	-95,% granica ufności	+95,% granica ufności
średn./stała	**692,99**	**32,30**	**21,45**	**0,0000**	**629,35**	**756,63**
(1)X1 (L)	**-107,97**	**16,78**	**-6,43**	**0,0000**	**-141,04**	**-74,91**
(2)X2 (L)	**-684,33**	**55,45**	**-12,34**	**0,0000**	**-793,58**	**-575,08**
X2 (Q)	**726,28**	**47,62**	**15,25**	**0,0000**	**632,46**	**820,10**
(3)X3 (L)	**-1098,17**	**175,51**	**-6,26**	**0,0000**	**-1443,97**	**-752,38**
X3 (Q)	**797,02**	**262,59**	**3,04**	**0,0027**	**279,64**	**1314,40**
(4)X4 (L)	**15,73**	**2,90**	**5,43**	**0,0000**	**10,02**	**21,43**
X4 (Q)	**-239,57**	**11,83**	**-20,25**	**0,0000**	**-262,88**	**-216,26**
(5)X5 (L)	**-58,77**	**6,53**	**-9,00**	**0,0000**	**-71,64**	**-45,90**
X5 (Q)	**-1192,56**	**60,18**	**-19,82**	**0,0000**	**-1311,13**	**-1073,99**
1L wz.2L	**392,64**	**32,47**	**12,09**	**0,0000**	**328,66**	**456,61**
2L wz.3L	**276,58**	**79,05**	**3,50**	**0,0006**	**120,83**	**432,34**

Dla poziomu ufności $1 - \alpha = 0{,}95$ i $\nu = n - N_b$ stopni swobody odczytujemy z [4] kwantyl t(0,95; 231) = 1,651.

Jeżeli zweryfikujemy hipotezę H: $a_i = 0$ na poziomie istotności α, przeciw hipotezie alternatywnej k, to otrzymamy:

$k : a_i > 0$, to zbiorem krytycznym jest $I = <1{,}651; +\infty)$,

$k : a_i < 0$, to zbiorem krytycznym jest $I = (-\infty; -1{,}651>$. (2)

Z tabeli 1 oraz zależności (2) widać, że hipotezę H: $a_i = 0$ należy odrzucić dla wszystkich a_i.

2.2. Plan Hartleya

Funkcję obiektu badań przyjęto w postaci wielomianu algebraicznego stopnia drugiego, natomiast liczbę współczynników wielomianu aproksymującego ustalono na $N_b = 12$. Założono poziom istotności $\alpha = 0{,}05$, natomiast liczbę układów planu $n = 27$ wyznaczono z zależności ($2^{i-1} + 2i + n_0$, gdzie: 2^{i-1} – liczba układów

planu frakcyjnego przy pojedynczym kontraście – jądro planu, $2i$ – układy stanowiące odpowiednik punktów gwiezdnych, $n_0 = 1$ – centrum planu [3]).

W celu przeprowadzenia statystycznej weryfikacji adekwatności planu kompletnego PS/DK – 3^i oraz Hartleya PS/DS-P: Ha_i przyjęto wielomian aproksymujący (1) po odrzuceniu nieistotnych współczynników zgodnie z p. 5.1 [1].

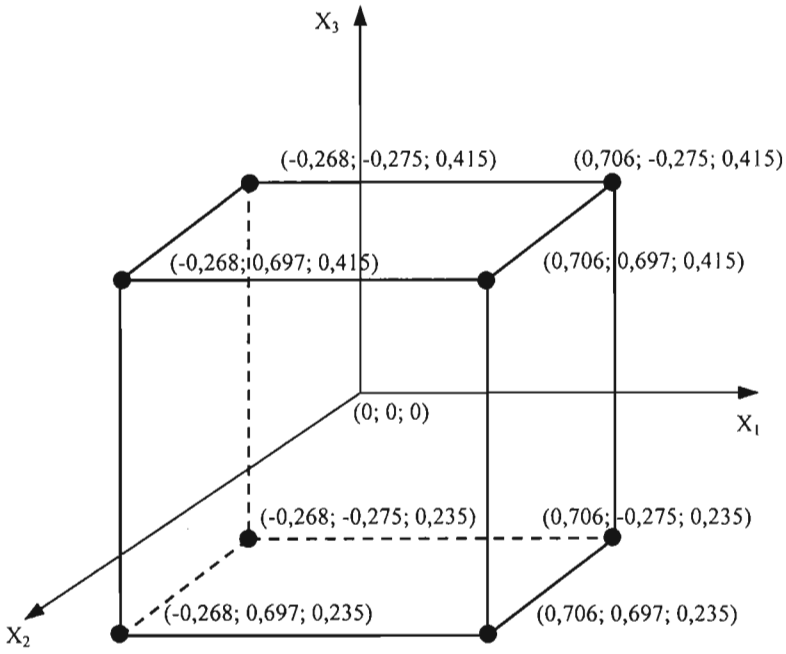
Tabela 2

Przedziały ufności i testy istotności współczynników regresji wielokrotnej dla ($N_b = 12$)

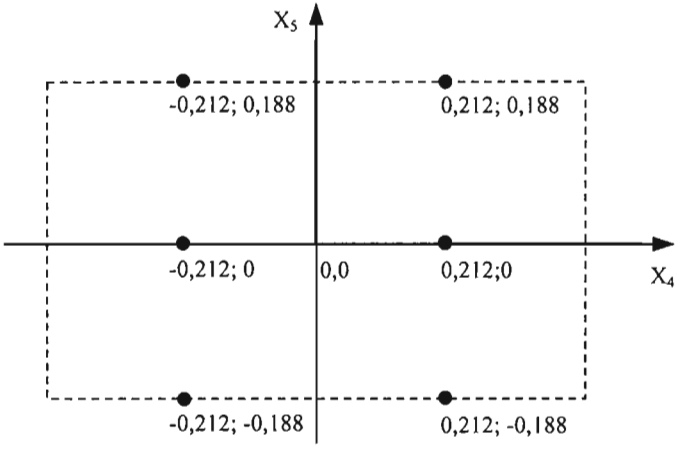
współczynniki regresji zmiennej (z); $R^2 = 0{,}97035$							
wielkość wejściowa		współczynnik regresji a_i	błąd standardowy S_{ai}	$t_i = a_i/S_{ai}$	poziom istotności α	-95,% granica ufności	+95,% granica ufności
średn./stała		**540,11**	**149,99**	**3,60**	**0,0026**	**220,41**	**859,81**
(1)X1	(L)	**-122,44**	**51,73**	**-2,37**	**0,0318**	**-232,70**	**-12,18**
(2)X2	(L)	-415,85	264,26	-1,57	0,1364	-979,11	147,40
X2	(Q)	**533,21**	**254,48**	**2,17**	**0,0461**	**10,80**	**1095,62**
(3)X3	(L)	-544,99	920,86	-0,59	0,5628	-2507,73	1417,74
X3	(Q)	127,11	1403,38	0,09	0,9290	-2864,14	3118,35
(4)X4	(L)	1,95	10,13	0,19	0,8502	-19,65	23,55
X4	(Q)	**-243,67**	**63,23**	**-3,85**	**0,0016**	**-378,47**	**-108,92**
(5)X5	(L)	**-61,20**	**22,85**	**-2,68**	**0,0172**	**-109,91**	**-12,49**
X5	(Q)	**-920,49**	**321,62**	**-2,86**	**0,0118**	**-1606,02**	**-234,97**
1L wz.2L		**447,95**	**98,42**	**4,55**	**0,0004**	**238,16**	**657,74**
2L wz.3L		42,53	239,53	0,18	0,8614	-468,00	553,07

3. Statystyczna weryfikacja adekwatności planu kompletnego i Hartleya

Weryfikację adekwatności planu kompletnego oraz Hartleya wykonano z wykorzystaniem wybranych punktów pomiarowych zmiennej wyjściowej ($z_j^{(u)} = E_{3j}^{(u)}$ – natężenie oświetlenia na płaszczyźnie roboczej modelu pomieszczenia w wybranym punkcie), które nie były wcześniej wykorzystywane do wyznaczania współczynników regresji. Rysunki 1 i 2 przedstawiają wybrane punkty pomiarowe zmiennej wyjściowej (z).



Rys. 1. Schemat rozmieszczania współrzędnych, ($x_1 = \rho_1, x_2 = \rho_2, x_3 = h_m$) punktów pomiarowych w przestrzeni trójwymiarowej



Rys. 2. Schemat rozmieszczania współrzędnych, (x_4, x_5) punktów pomiarowych na płaszczyźnie roboczej modelu pomieszczenia

W celu porównania planu kompletnego i planu Hartleya oraz dokonania oceny ich poprawności obliczamy średni kwadratowy błąd bezwzględny oraz względny ze wzorów:

$$r_1 = \left\{ \left[\sum_{l}^{s} \sum_{m}^{t} \sum_{o}^{w} \sum_{p}^{x} \left(\overline{E}^{e}_{lmop} - E^{t}_{lmop} \right)^2 \right] / stwx \right\}^{1/2} \tag{3}$$

$$r_2^w = \left\{ \left[\sum_{l}^{s} \sum_{m}^{t} \sum_{o}^{w} \sum_{p}^{x} \left(\frac{\overline{E}^{e}_{lmop} - E^{t}_{lmop}}{\overline{E}^{e}_{lmop}} \right)^2 \right] / stwx \right\}^{1/2}$$

gdzie:

$\overline{E}^{e}_{lmop}$ – średnia eksperymentalna wartość natężenia oświetlenia w luksach przy l – tym współczynniku odbicia sufitu, m – tym współczynniku odbicia ścian, o – tej wysokości usytuowania opraw oświetleniowych, p – tym punkcie pomiarowym na płaszczyźnie x_4Ox_5,

E^{t}_{lmop} – wartość teoretyczna natężenia oświetlenia w luksach przy l – tym współczynniku odbicia sufitu, m – tym współczynniku odbicia ścian, o – tej wysokości usytuowania opraw oświetleniowych, p – tym punkcie pomiarowym na płaszczyźnie x_4Ox_5,

$s = 2$ – liczba wartości wielkości wejściowej $x_1 = \rho_1$,

$t = 2$ – liczba wartości wielkości wejściowej $x_2 = \rho_2$,

$w = 2$ – liczba wartości wielkości wejściowej $x_3 = h_m$,

$x = 6$ – liczba punktów pomiarowych na płaszczyźnie x_4Ox_5.

Wartości błędów obliczone według wzorów (3) porównano ze średnimi odchyłkami bezwzględnymi i względnymi. Odchyłki te obliczono ze wzorów:

$$\Delta_1 = t_{\alpha,\nu} \left\{ \left[\sum_{j}^{r} \sum_{l}^{s} \sum_{m}^{t} \sum_{o}^{w} \sum_{p}^{x} \left(E^{e}_{jlmop} - \overline{E}^{e}_{lmop} \right)^2 \right] / \nu \right\}^{1/2} \tag{4}$$

$$\Delta_2^w = t_{\alpha,\nu} \left\{ \left[\sum_{j}^{r} \sum_{l}^{s} \sum_{m}^{t} \sum_{o}^{w} \sum_{p}^{x} \left(\frac{E^{e}_{jlmop} - \overline{E}^{e}_{lmop}}{\overline{E}^{e}_{lmop}} \right)^2 \right] / \nu \right\}^{1/2}$$

gdzie:

E^{e}_{jlmop} – wartość eksperymentalna natężenia oświetlenia w luksach przy j – tym powtórzeniu pomiaru natężenia oświetlenia na płaszczyźnie roboczej modelu pomieszczenia, l – tym współczynniku odbicia sufitu, m – tym współczynniku odbicia ścian, o – tej wysokości usytuowania opraw oświetleniowych, p – tym punkcie pomiarowym na płaszczyźnie x_4Ox_5,

$t_{\alpha,\nu}$ – wartość krytyczna rozkładu t – Studenta dla poziomu istotności $\alpha = 0{,}05$ lub $\alpha = 0{,}01$ i o $\nu = (r-1)\cdot s\cdot t\cdot w\cdot x$ stopniach swobody,

$r = 3$ – liczba powtórzeń pomiarów natężenia oświetlenia w określonym punkcie.

Przed obliczeniem odchyłek sprawdzono jednorodności wariancji testem Cochrana (jednakowa liczba powtórzeń). Wyniki obliczeń statystycznych zestawiono w tabeli 3 oraz 4.

Tabela 3

Zestawienie obliczeń statystycznych dla planu kompletnego

zestawienie	współczynnik punktu na płaszczyźnie x_4Ox_5	wielkości statystyczne					
				poziom istotności			
		r_l [lx]	r^w_2 [%]	$\alpha = 0{,}05$		$\alpha = 0{,}01$	
				Δ_l [lx]	Δ^w_2 [%]	Δ_l [lx]	Δ^w_2 [%]
lokalnie	(-0,212; 0,188)	16,86	4,56	24,09	5,83	33,19	8,03
	(0,212; 0,188)	15,73	4,79	18,51	5,47	25,49	7,54
	(-0,212; 0)	16,34	3,94	23,19	5,02	31,95	6,92
	(0,212; 0)	18,14	5,27	19,49	5,18	26,84	7,14
	(-0,212; -0,188)	13,60	3,59	18,61	4,60	25,63	6,33
	(0,212; -0,188)	9,36	2,42	16,16	4,32	22,27	5,96
globalnie		15,00	4,10	18,69	4,73	24,82	6,29

Tabela 4

Zestawienie obliczeń statystycznych dla planu Hartleya

zestawienie	współczynnik punktu na płaszczyźnie x_4Ox_5	wielkości statystyczne					
				poziom istotności			
		r_l [lx]	r^w_2 [%]	$\alpha = 0{,}05$		$\alpha = 0{,}01$	
				Δ_l [lx]	Δ^w_2 [%]	Δ_l [lx]	Δ^w_2 [%]
lokalnie	(-0,212; 0,188)	6,40	1,70	24,09	5,83	33,19	8,03
	(0,212; 0,188)	24,60	6,50	18,51	5,47	25,49	7,54
	(-0,212; 0)	14,33	4,11	23,19	5,02	31,95	6,92
	(0,212; 0)	27,36	6,90	19,49	5,18	26,84	7,14
	(-0,212; -0,188)	11,82	3,20	18,61	4,60	25,63	6,33
	(0,212; -0,188)	20,24	4,80	16,16	4,32	22,27	5,96
globalnie		17,46	4,53	18,69	4,73	24,82	6,29

4. Statystystyczna weryfikacja istotności współczynnika korelacji wielowymiarowej R

Wartość współczynnika korelacji wielowymiarowej planu kompletnego odczytano z tabeli 1, $R^2 = 0{,}958$, a odpowiadającą mu statystykę F obliczono z poniższego wzoru:

$$F = \frac{(n - N_b)R^2}{(N_b - 1)(1 - R^2)} = 479{,}00$$
$$f_2 = n - N_b = 243 - 12 = 231 \qquad (5)$$
$$f_1 = N_b - 1 = 12 - 1 = 11$$

Wartość krytyczna dla powyższej statystyki wynosi $F^* = 2{,}43$ [4] dla przyjętej wartości poziomu istotności $\alpha = 0{,}05$. Ponieważ $F > F^*$, to współczynnik korelacji wielorakiej R jest istotny. Wartość współczynnika korelacji wielokrotnej planu Hartleya odczytano z tabeli 2, $R^2 = 0{,}970$, a odpowiadającą mu statystykę $F = 44{,}09$ obliczono na podstawie (5), dla $f_2 = n - N_b = 27 - 12 = 15$ stopni swobody w liczniku i $f_1 = N_b - 1 = 12 - 1 = 11$ stopni swobody w mianowniku. Wartość krytyczna powyższej statystyki wynosi $F^* = 2{,}72$ [4]. Ponieważ $F > F^*$ – to współczynnik korelacji wielorakiej R jest istotny.

5. Wnioski

Z tabeli 3 i 4 wynika adekwatność globalna obu planów dla poziomu istotności $\alpha = 0{,}05$. Można również stwierdzić adekwatność lokalną obu planów dla $\alpha = 0{,}01$, a planu kompletnego dla $\alpha = 0{,}05$ (bardziej adekwatny).

Z weryfikacji współczynników korelacji wielokrotnej przeprowadzonej w p. 4 wynika także większa istotność planu kompletnego.

Literatura

[1] Jakowluk W, Czech M.: Analiza porównawcza planu kompletnego i Pesočinskiego w zastosowaniu do badania rozkładu natężenia oświetlenia. Praca w druku.

[2] Jakowluk W.: Weryfikacja modelu matematycznego sprawności oświetlenia z wykorzystaniem wielokrotnych odbić we wnętrzu. Elektryka 14. Białystok 1996, str. 115-124.

[3] Polański Z.: Planowanie doświadczeń w technice. PWN. Warszawa, 1984.
[4] Volk W.: Statystyka stosowana dla inżynierów. WNT. Warszawa, 1973.

ADEQUACY AND EFFECTIVENESS OF BOTH COMPLETE AND HARTLEY DESIGNS TO PREDICT INTENSITY OF ILLUMINATION

Summary: The purpose of this work is to carry out an adequate and effective analysis of trivalent complete and Hartley designs to determine the intensity of illumination at an optional point on the working plane of the room. For this purpose a comparison of a probable average error and mean square deviation was performed.

Key words: multiple reflections, experimental designs, probable average error, mean square deviation

Artykuł zrealizowano w ramach pracy badawczej własnej W/II/3/00.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Wiktor Jakowluk, Michał Czech[1]

PORÓWNANIE PLANU KOMPLETNEGO I PLANU PESOČINSKIEGO W ZASTOSOWANIU DO WYZNACZANIA ROZKŁADU NATĘŻENIA OŚWIETLENIA

Streszczenie: W pracy przeprowadzono analizę porównawczą planów trójwartościowych kompletnego oraz Pesočinskiego w zastosowaniu do predykcji rozkładu natężenia oświetlenia we wnętrzach. Funkcję obiektu badań przyjęto w postaci wielomianu algebraicznego drugiego stopnia ze współdziałaniami pierwszego rzędu.

Słowa kluczowe: wielokrotne odbicia, planowanie doświadczeń, statystyka F-Snedecora

1. Wprowadzenie

Analizę rozkładu natężenia oświetlenia we wnętrzu pomieszczenia wykonano z wykorzystaniem planu trójwartościowego kompletnego, oznaczanego jako PS/DK-3^i oraz planu trójwartościowego Pesočinskiego, oznaczanego jako PS/DS-P: Pe_i, dla pięciu zmiennych wejściowych: $x_1 = \rho_1$ – współczynnik odbicia powierzchni sufitu, $x_2 = \rho_2$ – współczynnik odbicia powierzchni ścian, $x_3 = h_m$ – – wysokość usytuowania opraw oświetleniowych nad płaszczyzną roboczą, x_4 – odcięta położenia punktu na płaszczyźnie roboczej modelu pomieszczenia, x_5 – rzędna położenia punktu na płaszczyźnie roboczej modelu pomieszczenia.

Założono $r \geq 3$ – krotne powtórzenie pomiarów wartości wielkości wyjściowej (z) dla każdego układu planu doświadczenia, oprócz centralnego układu planu, dla którego $r = 10$. Na tej podstawie wykonane zostały pomiary wielkości wyjściowej $z^{(u)}{}_j = E_{3j}^{(u)}$ – natężenie oświetlenia na płaszczyźnie roboczej modelu po-

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok

mieszczenia, w punkcie opisanym przez zmienne wejściowe x_4, x_5, gdzie: $u = 1,2,\ldots,n$, dla u – tego układu planu doświadczenia, natomiast n zależy od przyjętego planu doświadczenia; $j = 1,\ldots,r$.

Funkcję obiektu badań przyjęto w postaci wielomianu algebraicznego stopnia drugiego, zawierającego składniki liniowe, kwadratowe oraz interakcje o postaci:

$$z = a_0 + a_1x_1 + \ldots + a_ix_i + a_{11}x_1^2 + \ldots + a_{ii}x_i^2 + a_{12}x_1x_2 + \ldots + a_{i-1,i}x_{i-1}x_i \qquad (1)$$

$$N_b = \frac{1}{2}(i+1)(i+2)$$

gdzie:

a_i – współczynniki funkcji aproksymującej z,

N_b – liczba współczynników wielomianu aproksymującego,

i – liczba wielkości wejściowych x_i, gdzie $i = 5$.

2. Model pomieszczenia

Stanowisko pomiarowe składa się z modelu pomieszczenia o wymiarach 11,65x5,65x5,35 [m], wykonanego w skali 1:10. W pomieszczeniu można zastosować trzy rodzaje oświetlenia sztucznego klasy pierwszej, trzeciej i piątej [1]. W skład modelu pomieszczenia wchodzą także cztery komplety ścian o różnych współczynnikach odbicia. W analizie statystycznej zostaną wykorzystane pomiary natężenia oświetlenia (z) na płaszczyźnie roboczej makiety pomieszczenia dla klasy I oświetlenia. Dokładny opis stanowiska pomiarowego można znaleźć w pracy współautora [2].

3. Charakterystyka obiektu badań

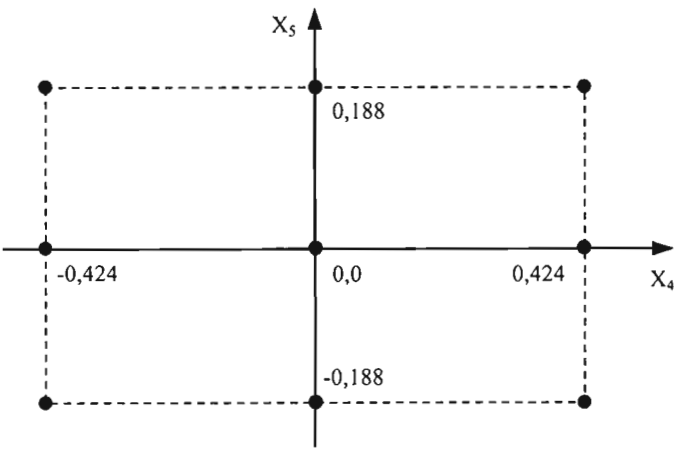
Wartości wielkości wejściowych dla $i = 5$, pochodzące z badań eksperymentalnych, zestawiono w tabeli 1.

Tabela 1

Zakresy wartości wielkości wejściowych

i	$x_1 = \rho_1$	$x_2 = \rho_2$	$x_3 = h_m$ [m]	x_4 [m]	x_5 [m]
x_{min}	0,268	0,275	0,235	- 0,424	- 0,188
$x_{śr}$	0,474	0,490	0,325	0,000	0,000
x_{max}	0,706	0,697	0,415	0,424	0,188

Wartości współczynników odbicia, zamieszczone w tabeli 1, są wynikiem badań eksperymentalnych, dlatego $x_{śr} \approx \frac{x_{min} + x_{max}}{2}$. Położenie punktów pomiarowych wartości wielkości wyjściowej (z), na płaszczyźnie roboczej makiety pomieszczenia, ilustruje rysunek1.

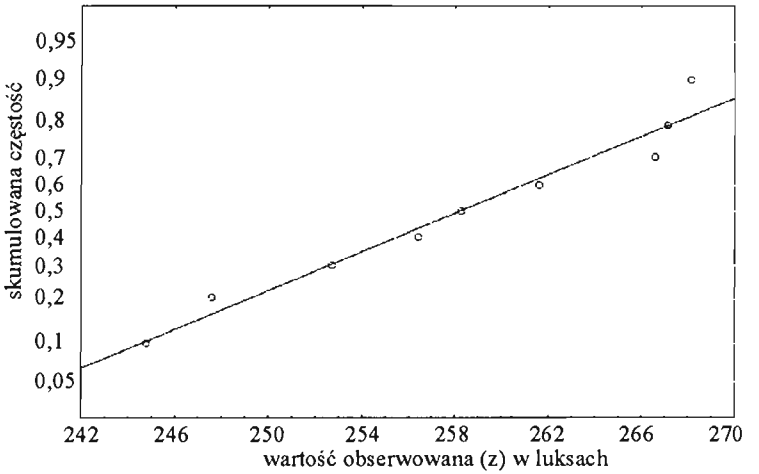


Rys. 1. Schemat rozmieszczenia punktów pomiarowych na płaszczyźnie roboczej makiety pomieszczenia

Przyjęto trójkrotne powtórzenie pomiarów wartości wielkości wyjściowej (z) dla każdego układu planu, z wyjątkiem centralnego układu planu, dla którego pomiary wielkości wyjściowej powtórzono dziesięciokrotnie.

4. Ocena normalności rozkładu zmiennej wyjściowej (z)

Sprawdzenie normalności rozkładu wykonano z wykorzystaniem centralnego układu planu, tj. ($x_1 = 0{,}474$; $x_2 = 0{,}490$; $x_3 = 0{,}325$), dla każdego punktu pomiarowego, (rysunek 1). Przykładowy wykres skumulowanej częstości zmiennej wyjściowej (z), w punkcie o współrzędnych (-0,424;0,188), przedstawia rysunek 2.



Rys. 2. Wykres skumulowanej częstości zmiennej wyjściowej (*z*) o rozkładzie normalnym

Na rysunku 2 jest 9 punktów pomiarowych; brak punktu, dla którego wartość skumulowana wynosi 1. Wynika to stąd, że na osi rzędnych nie istnieje taka skończona wartość.

Chcąc stwierdzić, czy korelacja zmiennej wyjściowej (*z*) jest rzeczywiście istotna, można zweryfikować hipotezę zerową o postaci $H_0: R = 0$ wobec hipotezy alternatywnej $H_1: R \neq 0$.

Tabela 2

Prawdopodobieństwo wartości większej lub równej *R*

stopnie swobody N - 2	współczynnik korelacji liniowej R dla punktów o współrzędnych:									wartość R_k na poziomie 0,001
	0; 0	0,424; 0	0,424; 0,188	0; 0,188	-0,424; 0,188	-0,424; 0	-0,424; -0,188	0; -0,188	0,424; -0,188	
7	0,939	0,958	0,944	0,983	0,979	0,972	0,957	0,926	0,950	0,898

Z tabeli 2 widać, że jeśli przy obliczaniu współczynnika korelacji *R* z 9 obserwacji otrzyma się wartość większą niż wartość krytyczna $R_k = 0{,}898$ [4], wtedy hipotezę, że nie ma korelacji $H_0: R = 0$ można odrzucić z szansą popełnienia błędu wynoszącą jedynie 0,001.

5. Plan doświadczenia

Do wyznaczenia rozkładu natężenia oświetlenia na płaszczyźnie roboczej makiety pomieszczenia wykorzystano plan statyczny zdeterminowany kompletny PS/DK–3^i oraz plan polisekcyjny specjalny Pesočinskiego PS/DS-P: Pe_i. Liczba wielkości wejściowych $i = 5$ [3].

5.1. Plan kompletny trójwartościowy – model 1

Funkcję obiektu badań przyjęto w postaci wielomianu algebraicznego stopnia drugiego (1). Współczynniki wielomianu aproksymującego ($N_b = 1$) wyznaczono metodą najmniejszych kwadratów. Założono poziom istotności $\alpha = 0{,}05$, a liczbę układów planu $n = 3^i = 243$ (plan trójwartościowy dla $i = 5$).

Tabela 3

Przedziały ufności i testy istotności współczynników regresji wielokrotnej ($N_b = 21$)

współczynniki regresji zmiennej (z); $R^2 = 0{,}9593$						
wielkość wejściowa	współczynnik regresji a_i	błąd standardowy S_{ai}	$t_i = a_i/S_{ai}$	poziom istotności α	-95,% granica ufności	+95,% granica ufności
średn./stała	**708,33**	**35,78**	**19,79**	**0,0000**	**637,81**	**778,85**
(1)X1 (L)	**-155,70**	**53,02**	**-2,94**	**0,0037**	**-260,19**	**-51,21**
X1 (Q)	29,03	44,70	0,65	0,5167	-59,06	117,12
(2)X2 (L)	**-684,33**	**55,65**	**-12,30**	**0,0000**	**-794,00**	**-574,65**
X2 (Q)	**726,28**	**47,79**	**15,20**	**0,0000**	**632,10**	**820,47**
(3)X3 (L)	**-1126,89**	**179,98**	**-6,26**	**0,0000**	**-1481,57**	**-772,21**
X3 (Q)	**797,02**	**263,55**	**3,02**	**0,0028**	**277,63**	**1316,40**
(4)X4 (L)	**39,09**	**17,39**	**2,25**	**0,0256**	**4,81**	**73,36**
X4 (Q)	**-239,57**	**11,87**	**-20,17**	**0,0000**	**-262,97**	**-216,17**
(5)X5 (L)	**-101,50**	**39,22**	**-2,59**	**0,0103**	**-178,80**	**-24,20**
X5 (Q)	**-1192,56**	**·60,40**	**-19,74**	**0,0000**	**-1311,59**	**-1073,53**
1L wz.2L	**392,64**	**32,59**	**12,05**	**0,0000**	**328,41**	**456,86**
1L wz.3L	59,55	76,55	0,78	0,4375	-91,31	210,40
1L wz.4L	-11,72	16,25	-0,72	0,4716	-43,74	20,30
1L wz.5L	-15,13	36,65	-0,41	0,6801	-87,35	57,09
2L wz.3L	**276,58**	**79,34**	**3,49**	**0,0006**	**120,23**	**432,94**
2L wz.4L	-24,42	16,84	-1,45	0,1485	-57,61	8,77
2L wz.5L	20,41	37,98	0,54	0,5916	-54,44	95,26
3L wz.4L	-17,88	39,56	-0,45	0,6516	-95,84	60,07
3L wz.5L	123,33	89,22	1,38	0,1683	-52,49	299,14
4L wz.5L	-19,78	18,94	-1,04	0,2973	-57,10	17,54

gdzie:
średn./stała – oznacza wyraz wolny,
X1 (L) – oznacza efekt liniowy x_1,
X1 (Q) – oznacza efekt kwadratowy x_1^2,
1L wz. 2L – oznacza liniową interakcję $x_1 * x_2$,
wartości wytłuszczone w tabeli są wartościami istotnymi.

Dla poziomu ufności $1-\alpha = 0{,}95$ i $\nu = n - N_b = 243-21 = 222$ stopni swobody odczytujemy z [4] kwantyl $t_{(0,95;222)} = 1{,}652$.

Jeżeli zweryfikujemy hipotezę H: $a_i = 0$ na poziomie istotności α, przeciw hipotezie alternatywnej k, to otrzymamy:

$k: a_i > 0$, to zbiorem krytycznym jest $I = < 1{,}652; +\infty)$,

$k: a_i < 0$, to zbiorem krytycznym jest $I = (-\infty; -1{,}652 >$. (2)

Z tabeli 2 oraz zależności (2) widać, że hipotezę H: $a_i = 0$ należy odrzucić dla wszystkich a_i wytłuszczonych.

5.2. Plan kompletny trójwartościowy – model 2

Funkcję obiektu badań przyjęto w postaci wielomianu algebraicznego stopnia drugiego, natomiast liczbę współczynników wielomianu aproksymującego ustalono zgodnie z punktem 5.1, $N_b = 12$. Założono poziom istotności $\alpha = 0{,}05$, natomiast liczbę układów planu $n = 3^i = 243$.

Tabela 4
Przedziały ufności i testy istotności współczynników regresji wielokrotnej dla ($N_b = 12$)

współczynniki regresji zmiennej (z); $R^2 = 0{,}95796$						
wielkość wejściowa	współczynnik regresji a_i	błąd standardowy S_{ai}	$t_i = a_i/S_{ai}$	poziom istotności α	-95,% granica ufności	+95,% granica ufności
1	2	3	4	5	6	7
średn./stała	**692,99**	**32,30**	**21,45**	**0,0000**	**629,35**	**756,63**
(1)X1 (L)	**-107,97**	**16,78**	**-6,43**	**0,0000**	**-141,04**	**-74,91**
(2)X2 (L)	**-684,33**	**55,45**	**-12,34**	**0,0000**	**-793,58**	**-575,08**
X2 (Q)	**726,28**	**47,62**	**15,25**	**0,0000**	**632,46**	**820,10**
(3)X3 (L)	**-1098,17**	**175,51**	**-6,26**	**0,0000**	**-1443,97**	**-752,38**
X3 (Q)	**797,02**	**262,59**	**3,04**	**0,0027**	**279,64**	**1314,40**
(4)X4 (L)	**15,73**	**2,90**	**5,43**	**0,0000**	**10,02**	**21,43**
X4 (Q)	**-239,57**	**11,83**	**-20,25**	**0,0000**	**-262,88**	**-216,26**

cd. tabeli 4

1	2	3	4	5	6	7
(5)X5 (L)	**-58,77**	**6,53**	**-9,00**	**0,0000**	**-71,64**	**-45,90**
X5 (Q)	**-1192,56**	**60,18**	**-19,82**	**0,0000**	**-1311,13**	**-1073,99**
1L wz.2L	**392,64**	**32,47**	**12,09**	**0,0000**	**328,66**	**456,61**
2L wz.3L	**276,58**	**79,05**	**3,50**	**0,0006**	**120,83**	**432,34**

Dla poziomu ufności $1-\alpha = 0{,}95$ i $\nu = n - N_b = 243-12=231$ stopni swobody odczytujemy z [4] kwantyl $t_{(0,95;231)} = 1{,}651$.

Jeżeli zweryfikujemy hipotezę H: $a_i = 0$ na poziomie istotności α, przeciw hipotezie alternatywnej k, to otrzymamy:

$k: a_i > 0$, to zbiorem krytycznym jest $I = <1{,}651;+\infty)$,

$k: a_i < 0$, to zbiorem krytycznym jest $I = (-\infty;-1{,}651>$. (3)

Z tabeli 3 oraz zależności (3) widać, że hipotezę H: $a_i = 0$ należy odrzucić dla wszystkich a_i.

5.3. Plan Pesočinskiego

Funkcję obiektu badań aproksymowano wielomianem algebraicznym stopnia drugiego, natomiast liczbę współczynników wielomianu aproksymującego ustalono zgodnie z punktem 5.1, $N_b = 12$. Założono poziom istotności $\alpha = 0{,}05$, natomiast liczbę układów planu $n = 50$ wyznaczono z zależności $(2^{i-2} + 5 \cdot 2^{i-1-1} + n_0$, gdzie: 2^{i-2} – liczba układów planu frakcyjnego, przy podwójnym kontraście – jądro planu, 2^{i-1-1} – liczba układów planu frakcyjnego przy odpowiednim $x_i = 0$ i odpowiednim kontraście, $n_0 = 2$ – centrum planu) [3].

W celu porównania planu kompletnego PS/DK–3^i oraz planu Pesočinskiego PS/DS-P: Pe_i, przyjęto wielomian aproksymujący z punktu 5.1, po odrzuceniu nieistotnych współczynników. Wyniki obliczeń dla planu Pesočinskiego zestawiono w tabeli 5.

Tabela 5

Przedziały ufności i testy istotności współczynników regresji wielokrotnej dla ($N_b = 12$)

współczynniki regresji zmiennej (z); $R^2 = 0{,}9741$							
wielkość wejściowa	współczynnik regresji a_i	błąd standardowy	$t_i = a_i/S_{ai}$	poziom istotności α	-95,% granica ufności	+95,% granica ufności
średn./stała	**550,03**	**75,77**	**7,26**	**0,0000**	**396,64**	**703,42**
(1)X1 (L)	**-96,37**	**30,91**	**-3,12**	**0,0035**	**-158,95**	**-33,79**
(2)X2 (L)	**-524,54**	**129,24**	**-4,06**	**0,0002**	**-786,18**	**-262,90**
X2 (Q)	**603,16**	**120,29**	**5,01**	**0,0000**	**359,64**	**846,67**
(3)X3 (L)	-418,63	435,85	-0,96	0,3429	-1300,95	463,69
X3 (Q)	-156,81	660,90	-0,24	0,8137	-1494,74	1181,12
(4)X4 (L)	**13,14**	**5,65**	**2,33**	**0,0254**	**1,71**	**24,57**
X4 (Q)	**-264,51**	**29,78**	**-8,88**	**0,0000**	**-324,79**	**-204,23**
(5)X5 (L)	**-64,11**	**13,07**	**-4,91**	**0,0000**	**-90,56**	**-37,66**
X5 (Q)	**-1460,51**	**151,46**	**-9,64**	**0,0000**	**-1767,14**	**-1153,89**
1L wz.2L	**398,07**	**59,41**	**6,71**	**0,0000**	**277,81**	**518,33**
2L wz.3L	212,56	140,95	1,51	0,1398	-72,77	497,89

6. Statystyczna weryfikacja adekwatności: test F – Snedecora

Dalsze rozważania dotyczyć będą tylko modelu drugiego planu kompletnego oraz planu Pesočinskiego dla $N_b = 12$. Weryfikację adekwatności z zastosowaniem testu F – Snedecora przeprowadzono w następujący sposób:

– Obliczanie wariancji adekwatności $s^2(z)_a$:

$$s^2(z)_a = \frac{1}{f_2} \sum_{u=1}^{n} \left[\bar{z}^{(u)} - \check{z}^{(u)}\right]^2 \tag{4}$$

$$f_2 = n - N_b \tag{5}$$

gdzie:

$\bar{z}^{(u)}$ – średnia arytmetyczna wartości wielkości wyjściowej dla u – układu planu doświadczenia ($u = 1,2,...,n$),

$\check{z}^{(u)}$ – aproksymowana wartość wielkości wyjściowej dla u – układu planu doświadczenia, stanowiąca wartość funkcji aproksymującej.

Wariancja adekwatności pomiarów dla modelu drugiego planu kompletnego wynosi $s^2(z)_a = 244{,}30 \ \ (lx)^2$ dla $f_2 = 243 - 12 = 231$ stopni swobody. Nato-

miast wariancja adekwatności pomiarów dla planu Pesočinskiego wynosi $s^2(z)_a = 229{,}26\ (lx)^2$ dla $f_2 = 50 - 12 = 38$ stopni swobody.

– Obliczanie wariancji niedokładności pomiarów $s^2(z)$ z powtórzeń $(r_u > 1)$ w układach planu doświadczenia:

$$s^2(z) = \frac{1}{f_1} \sum_{u=1}^{n} (r_u - 1) s^2(z)^{(u)} = 137{,}14\left[(lx)^2\right] \tag{6}$$

$$f_1 = \sum_{u=1}^{n} (r_u - 1) = 9(3-1) = 18 \tag{7}$$

$$s^2(z)^{(u)} = \frac{1}{r_u - 1}\left\{ \sum_{j=1}^{r_u} \left[z_j{}^{(u)}\right]^2 - \bar{z}^{(u)} \sum_{j=1}^{u} z_j{}^{(u)} \right\} \tag{8}$$

$$\bar{z}^{(u)} = \frac{1}{r_u} \sum_{j=1}^{r_u} z_j{}^{(u)} \tag{9}$$

gdzie:

r_u – liczba powtórzeń pomiarów dla każdego układu planu doświadczenia,

$z_j{}^{(u)}$ – wartości wielkości wyjściowej w identycznych układach planu.

Wariancja niedokładności pomiarów $s^2(z)$, obliczona dla modelu drugiego planu kompletnego oraz planu Pesočinskiego jest identyczna.

– Obliczenie ilorazu F:

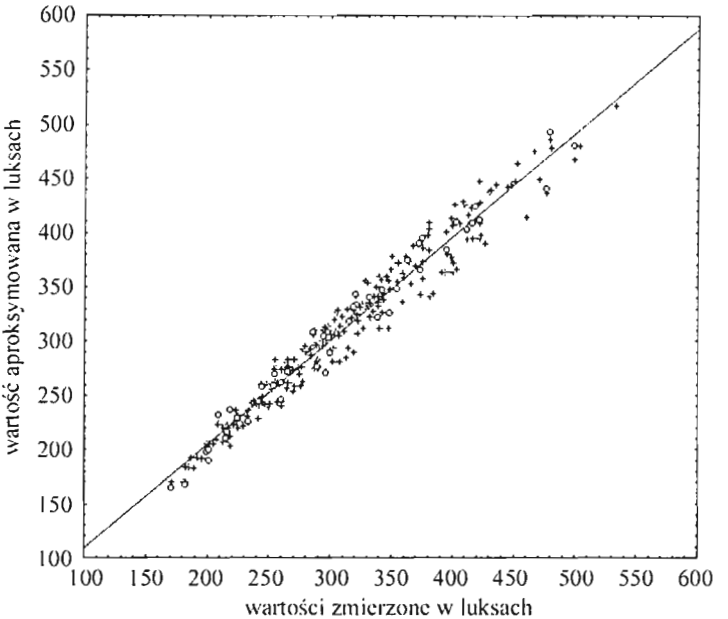
$$F = \frac{s^2(z)_\alpha}{s^2(z)} \tag{10}$$

Wartość ilorazu dla modelu drugiego planu kompletnego wynosi $F = 1{,}78$, natomiast dla planu Pesočinskiego $F = 1{,}67$.

Wartość krytyczna ilorazu dla modelu drugiego planu kompletnego wynosi $F^* = 1{,}94$ [4], dla przyjętej wartości poziomu istotności $\alpha = 0{,}05$. Ponieważ $F < F^*$, to przyjęta funkcja aproksymująca jest adekwatna.

Wartość krytyczna ilorazu dla planu Pesočinskiego wynosi $F^* = 2{,}07$ [4], dla przyjętej wartości poziomu istotności $\alpha = 0{,}05$. Ponieważ $F < F^*$, to przyjęta funkcja aproksymująca jest adekwatna.

Graficzną weryfikację adekwatności planu kompletnego oraz Pesočinskiego przedstawiono na rysunku 3.



Rys. 3. Wykres wartości przewidywanych i aproksymowanych planu kompletnego oraz Pesočinskiego, ↖ – plan kompletny, o – plan Pesočinskiego

7. Statystyczna weryfikacja istotności współczynnika korelacji wielowymiarowej R

Wartość współczynnika korelacji wielowymiarowej modelu drugiego planu kompletnego odczytano z tabeli 4, $R^2 = 0{,}958$, a odpowiadającą mu statystykę F obliczono z poniższego wzoru:

$$F = \frac{(n - N_b)R^2}{(N_b - 1)(1 - R^2)} = 479{,}00$$
$$f_2 = n - N_b = 243 - 12 = 231 \qquad (11)$$
$$f_1 = N_b - 1 = 12 - 1 = 11$$

Wartość krytyczna dla powyższej statystyki wynosi $F^* = 2{,}43$ [4], dla przyjętej wartości poziomu istotności $\alpha = 0{,}05$. Ponieważ $F > F^*$, współczynnik korelacji wielorakiej R – jest istotny. Wartość współczynnika korelacji wielokrotnej planu Pesočinskiego odczytano z tabeli 5, $R^2 = 0{,}974$, a odpowiadającą mu statystykę

F = 129,41 obliczono na podstawie (11), dla $f_2 = n - N_b$ = 50 – 12 = 38 stopni swobody w liczniku i $f_1 = N_b - 1 = 12 - 1 = 11$ stopni swobody w mianowniku. Wartość krytyczna powyższej statystyki wynosi F^* = 2,53 [4]. Ponieważ $F > F^*$, współczynnik korelacji wielorakiej R – jest istotny.

8. Otrzymane wyniki – błędy aproksymacji

Ustalono relacje między wartościami wielkości wyjściowej (z), stanowiącymi wyniki pomiarów, a odpowiednimi aproksymowanymi wartościami wielkości wyjściowej $(\tilde{z})$ z wyznaczonej funkcji aproksymującej. Maksymalny błąd względny dla modelu drugiego planu kompletnego wyliczono z zależności:

$$e^{(u)}{}_{\max} = \frac{\max\left|z - \tilde{z}\right|^{(u)}}{\left|z\right|^{(u)}} \cdot 100\% = 11\% \qquad (12)$$

$$z^{(u)} \neq 0$$

gdzie:

$e^{(u)}$ – błąd względny dla u – układu planu doświadczenia (u = 1,2,...,n).

Średni błąd względny obliczono z następującej zależności:

$$e_{śr} = \frac{\sum_{n}^{n} e^{(u)}}{n} = 3{,}81\% \qquad (13)$$

Maksymalny błąd względny $e^{(u)}{}_{\max}$ = 10,60% dla planu Pesočinskiego obliczono z zależności (12), natomiast średni błąd względny obliczony na podstawie (13) wynosi $e_{śr} = 3{,}61\%$. Powyższe błędy dotyczą wszystkich układów danego planu, (plan kompletny n = 243, plan Pesočinskiego n = 50).

9. Wnioski

Z zależności (10) i (11) widać, że dla poziomu istotności α = 0,05 przyjęta funkcja aproksymująca jest adekwatna, a wartość współczynnika korelacji wielowymiarowej jest wysoce istotna dla obu rozpatrywanych planów. Wyznaczony z zależności (13) średni błąd względny oszacowania jest zadowalający. Z rysunku

3 widać, że przewidywane wartości skupiają się wzdłuż przekątnej wykresu wskazując na dobre dopasowanie modelu do wyników pomiaru.

Literatura

[1] Banach M.: Podstawy techniki oświetlania. PWN. Warszawa, 1982.
[2] Jakowluk W.: Weryfikacja modelu matematycznego sprawności oświetlenia z wykorzystaniem wielokrotnych odbić we wnętrzu. Elektryka 14. Białystok, 1996, str. 115-124.
[3] Polański Z.: Planowanie doświadczeń w technice. PWN. Warszawa, 1984.
[4] Volk W.: Statystyka stosowana dla inżynierów. WNT. Warszawa, 1973.

A COMPARATIVE OF BOTH COMPLETE AND PESOČINSKI DESIGNS TO DETERMINE THE INTENSITY OF ILLUMINATION DISTRIBUTION

Summary: A comparative analysis of both complete and Pesočinski designs for forecasting the intensity of illumination in interiors was carried out in this paper. The function of the study object is considered in a form of second order polynomial and first order interactions.

Key words: multiple reflections, experimental designs, F-test

Artykuł zrealizowano w ramach pracy badawczej własnej W/II/3/00.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Wiaczesław Jarmolik, Marek Gruszewski[1]

CYFROWY GENERATOR SYGNAŁÓW SINUSOIDALNYCH DO TESTOWANIA I WERYFIKACJI UKŁADÓW ANALOGOWO-CYFROWYCH

Streszczenie: W artykule przedstawiony jest cyfrowy generator sygnałów sinusoidalnych, który wchodzić może w skład uniwersalnego modułu do samotestowania układów analogowo-cyfrowych (mixed-signal). Sygnał sinusoidalny okazuje się najbardziej złożonym sygnałem (z punktu widzenia jego kształtowania) w porównaniu z innymi sygnałami (np. trójkątnym, piłokształtnym, czy prostokątnym). Tradycyjnie generator sygnału sinusoidalnego zbudowany jest z układu sterowania i bloku pamięci, w której znajdują się dyskretne próbki 1/4 części okresu tego sygnału. Przedstawione w artykule podejście oparte jest na wykorzystaniu stochastycznego integratora, który jest podstawowym elementem generatora. Praca generatora sprawdzona została przy pomocy symulacji komputerowej.

Słowa kluczowe: samotestowanie, układy analogowo-cyfrowe, integrator stochastyczny, sygnał sinusoidalny

1. Wprowadzenie

W ostatnich czasach dużo uwagi poświęca się problemowi kontroli poprawności pracy układów hybrydowych (tj. układów, które zawierają w swojej strukturze obwody analogowe oraz cyfrowe), zrealizowanych w postaci pojedynczych układów scalonych. Podczas testowania takich układów należy rozwiązać problemy związane z brakiem dostępu do wewnętrznych punktów układu, które potrzebne są do podania sygnałów testowych i zebrania odpowiedzi układu na te sygnały, a także z ograniczeniem ilości zewnętrznych wyprowadzeń.

W ostatnich latach pojawiły się publikacje, które poświęcone są problemom syntezy układów hybrydowych, dogodnych do testowania [1-4]. Jednym z najbardziej perspektywicznych kierunków w tej dziedzinie okazuje się sposób przedsta-

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok

wiony w [1], który oparty jest na wykorzystaniu uniwersalnego urządzenia do testowania zarówno układów cyfrowych, jak i analogowych. Takie podejście otrzymało nazwę Hybrydowe Wbudowane Samotestowanie (Hybrid Built-In Self-Test, HBIST). Przy wykorzystywaniu HBIST osiągane są następujące podstawowe cele:

- zastosowanie samotestowania dla analogowych obwodów w zgodzie z metodami samotestowania dla obwodów cyfrowych;
- do organizacji samotestowania konieczne są minimalne nakłady sprzętowe, ponieważ formowanie testowych oddziaływań i analiza rekcji układu na te oddziaływania dla obwodów cyfrowych i analogowych realizowane są praktycznie przy pomocy tego samego urządzenia.

Na początku takie podejście do organizacji samotestowania układów hybrydowych było sformułowane dla uniwersalnego modułu BILBO [5]. W tym wypadku w charakterze sygnałów testowych dla układów analogowych i cyfrowych wykorzystuje się ciągi pseudolosowe, a do analizy odpowiedzi układu stosuje się analizator sygnaturowy.

O ile do organizacji samotestowania układów cyfrowych już tradycyjnie wykorzystuje się pseudolosowe zestawy testowe [5], o tyle do testowania układów analogowych mogą być potrzebne różne sygnały, takie jak sygnały sinusoidalne, prostokątne, piłokształtne, trójkątne i pseudolosowe [6, 7].

Jak już wspomniano, HBIST pozwala formować tylko pseudolosowe sygnały testowe dla części analogowej i cyfrowej, co może okazać się niewystarczające dla sprawdzenia działania analogowego podukładu.

W danej pracy przedstawiony jest uniwersalny moduł, który pozwala wytwarzać szeroki zestaw sygnałów testowych, zarówno dla obwodów cyfrowych, jak i analogowych oraz otrzymywać wyniki testowania.

2. Opis sygnałów testowych

2.1. Piłokształtny sygnał testowy

Istnieją dwa rodzaje sygnału piłokształtnego: narastający i opadający [8]. Sygnał piłokształtny w postaci cyfrowej przedstawia sobą ciąg stanów licznika (inkrementacja dla narastającego i dekrementacja dla opadającego sygnału piłokształtnego).

2.2. Trójkątny sygnał testowy

Sygnał trójkątny przedstawia sobą narastający piłokształtny sygnał w pierwszej połowie swojego okresu i opadający w drugiej połowie okresu.

2.3. Prostokątny sygnał testowy

Prostokątny sygnał można przedstawić jako kolejne występowanie maksymalnego U_{max} i minimalnego U_{min} poziomu napięcia. Do ukształtowania takiego sygnału można wykorzystać urządzenie zbudowane z licznika rewersyjnego i układu sterowania.

2.4. Pseudolosowy sygnał testowy

Tradycyjnie, ciąg pseudolosowy wytwarzany jest przy pomocy rejestru przesuwnego ze sprzężeniem zwrotnym (Linear Feedbac Shift Register).

2.5. Sinusoidalny sygnał testowy

Sygnał sinusoidalny okazuje się najbardziej złożonym sygnałem (z punktu widzenia jego kształtowania) w porównaniu z rozpatrzonymi wyżej sygnałami. Tradycyjnie generator sygnału sinusoidalnego zbudowany jest z układu sterowania i bloku pamięci, w której znajdują się dyskretne próbki ¼ części okresu tego sygnału. Taka realizacja generatora sygnału sinusoidalnego nie jest racjonalna ze względu na nakłady sprzętowe. Sygnał sinusoidalny należy więc kształtować przy pomocy sprzętowego rozwiązania równania różniczkowego drugiego rzędu:

$$y''(t)+\omega^2 y(t)=0 \qquad (1)$$

gdzie:

$y(t)$ – funkcja sinusa

Wyrażenie (1) w postaci cyfrowej rozwiązuje się przy pomocy urządzenia, które zbudowane jest z dwóch połączonych integratorów.

2.6. Wykładniczy sygnał testowy

Oprócz przedstawionych wyżej przebiegów testowych istnieje możliwość formowania wykładniczych przebiegów testowych przy wykorzystaniu integratora stochastycznego, który pracuje w trybie śledzącego integratora stochastycznego.

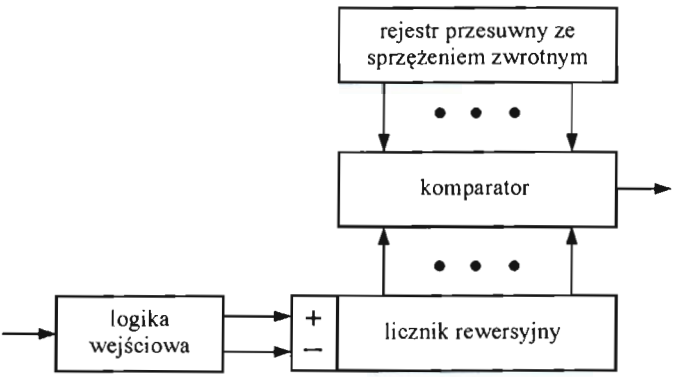
3. Uniwersalny moduł do samotestowania układów hybrydowych

3.1. Realizacja uniwersalnego modułu

Biorąc pod uwagę przedstawione wyżej sytuacje, dochodzimy do wniosku, że urządzenie do samotestowania układów hybrydowych musi odpowiadać następującym wymaganiom:

- wytwarzanie szerokiego zakresu sygnałów testowych, zarówno dla układów cyfrowych, jak i analogowych;
- wykonywanie analizy odpowiedzi układów cyfrowych i analogowych;
- zastosowanie metod samotestowania dla układów analogowych nie może nakładać ograniczeń na metody samotestowania układów cyfrowych;
- minimalne dodatkowe nakłady sprzętowe.

Podstawowym podzespołem proponowanego uniwersalnego modułu jest integrator stochastyczny (rys. 1.), który pozwala wytwarzać (z minimalnymi dodatkowymi nakładami sprzętowymi) szeroki zakres sygnałów testowych oraz analizować odpowiedzi testowanych układów przy pomocy analizy sygnaturowej i/lub analizy ilości jedynek.

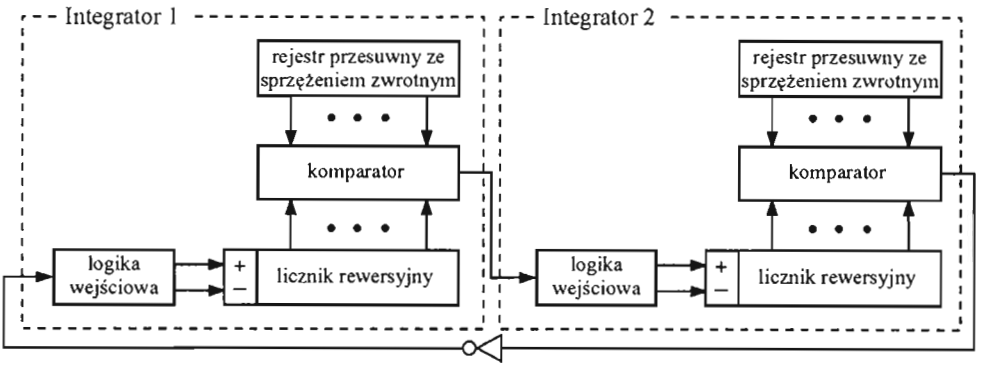


Rys. 1. Integrator stochastyczny

Tak jak w [1] proponowany moduł uniwersalny wytwarza sygnały testowe i analizuje odpowiedzi testowanego układu w obszarze cyfrowym. Analogowe sygnały testowe otrzymywane są drogą przetworzenia sygnału cyfrowego w analogowy, a analogowe odpowiedzi testowanego układu są przed analizą poddawane cyfryzacji.

3.2. Wytwarzanie sinusoidalnego sygnału testowego

W celu uformowania sinusoidalnego sygnału testowego wykorzystuje się przedstawiony na rys. 2 układ, który złożony jest z dwóch stochastycznych integratorów i jednego inwertera. W tym przypadku sygnał sinusoidalny wytwarzany jest na pozycjach rewersyjnego licznika drugiego integratora.



Rys. 2. Genarator sygnałów sinusoidalnych

3.3. Analiza poprawności działania układu generatora

Wykażemy poprawność zbudowanego układu. Wartość oczekiwana stanu stochastycznego integratora $y(t)$ w chwili czasu t opisana jest równaniem [9]:

$$y(t) = y(t-1) + x \tag{2}$$

gdzie:

$x = 1 \cdot p(t)$, $p(t)$ – prawdopodobieństwo pojawienia się sygnału „1" na wejściu integratora w chwili czasu t.

Prawdopodobieństwo pojawienia się sygnału „1" na wyjściu układu porównawczego $z(t)$ w chwili czasu t wynosi:

$$z(t) = \frac{1}{2^n} y(t) \tag{3}$$

gdzie:

n – liczba pozycji licznika w integratorze

Załóżmy, że sygnał sinusoidalny formowany jest na pozycjach licznika drugiego integratora. Wtedy na pozycjach licznika pierwszego integratora tworzony jest sygnał kosinusoidalny. Dlatego sinusoidalny sygnał w chwili czasu t będzie opisany wyrażeniem:

$$y_2(t) = y_2(t-1) + x_2 \tag{4}$$

gdzie:

$y_2(t)$ – wartość oczekiwana stanu drugiego integratora i $x_2 = 1 \cdot p_2(t)$,

$p_2(t)$ – prawdopodobieństwo pojawienia się sygnału „1" na wejściu drugiego integratora w chwili czasu t.

Ponieważ wartość $p_2(t)$ jest równa z_1, to wyrażenie (4) przyjmuje postać:

$$y_2(t) = y_2(t-1) + \frac{1}{2^n} y_1(t) \tag{5}$$

gdzie:
$y_1(t)$ – wartość oczekiwana stanu pierwszego integratora

Biorąc pod uwagę, że:

$$x_1(t) = -\frac{1}{2^n} y_2(t) \tag{6}$$

kosinusoidalny sygnał opisany jest równaniem:

$$y_1(t) = y_1(t-1) - \frac{1}{2^n} y_2(t) \tag{7}$$

Wiadomo, że:

$$\cos^2 t + \sin^2 t = 1 \tag{8}$$

To wyrażenie z uwzględnieniem wyrażeń (14) i (15) może być przedstawione w postaci:

$$\left(1 + \frac{1}{2^{2n}}\right) y_1{}^2(t-1) = 1 - \left(1 + \frac{1}{2^{2n}}\right) y_2{}^2(t-1) \tag{9}$$

Ponieważ $y_2{}''(t) = -y_1(t)$, to powyższe wyrażenie po obliczeniu pochodnej przekształca się do postaci równania:

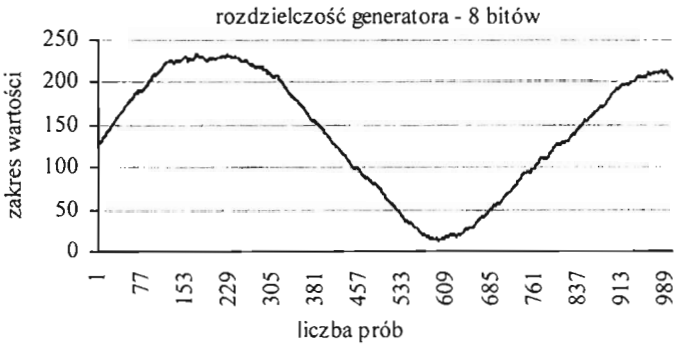
$$y_2{}''(t) = -y_2(t-1) \tag{10}$$

Jak wynika z (10), przedstawiony na rysunku 2 układ realizuje wyrażenie (1) z $\omega = 1$.

4. Rezultaty eksperymentów

Działanie układu generatora zbadane zostało przy pomocy symulacji komputerowej. Napisany został program, który symuluje działanie układu i przesyła wyniki do pliku tekstowego. Generator był symulowany przy założeniu rozdzielczości wynoszącej 8 bitów (ilość pozycji licznika rewersyjnego) i przy liczbie prób równej 1000. Wykonano serię pięciu symulacji i stwierdzono całkowitą powtarzalność

wyników. Rysunek 3 przedstawia rezultat symulacji pracy generatora sygnałów sinusoidalnych. Otrzymany wykres jest bardzo zbliżony do przebiegu sinusoidalnego.



Rys. 3. Rezultaty symulacji pracy generatora sygnałów sinusoidalnych

5. Zakończenie

Przedstawiony w pracy sposób projektowania uniwersalnego modułu do samotestowania układów hybrydowych pozwala wytwarzać szeroki zakres sygnałów testowych i analizować odpowiedź badanego układu z dużym stopniem prawdopodobieństwa. Przy wykorzystaniu takiego podejścia do syntezy urządzeń testujących, metody samotestowania układów analogowych nie nakładają ograniczeń na metody samotestowania układów cyfrowych. Dlatego podczas realizacji urządzeń testujących uzyskuje się minimalne nakłady sprzętowe. Wyniki symulacji (kształt przebiegu i jego powtarzalność) analizowanego generatora sinusoidalnego pozwalają wnioskować o poprawności jego działania i przydatności do samotestowania układów analogowo-cyfrowych.

Literatura

[1] M. Ohletz: *Hybrid Built-In Self_Test (HBIST) for Mixed Analogue/Digital Circuits*; IEEE European Test Conference, 1991, 307-316.

[2] C. L.Wey: *Built-In Self_Test (BIST) Strukture for Analog Circuit Fault Diagnosis*; IEEE Trans. On Instrumentation and Measurement Vol. 39 N3, 1990, 517-521.

[3] S. Khaled, B. Kaminska, B. Courtois, M. Lubaszewski: *Freguency-based BIST for analog circuit testing*; 13th VLSI Test Symposium, 1995, 54-59.
[4] M. Renovell, F. Azais, Y. Bertrand, The Multi-Configuration: *A DFT Technique for Analog Circuits*; 14th VLSI Test Symposium, 1996, 54-59.
[5] W. N. Jarmolik: *Kontrolowanie i diagnostyka cyfrowych podzespołów komputerowych*, Mińsk, 1988.
[6] M. Soma: *Fault Modeling and Test Generation for Sample-and-Hold Circuits*; International Symposium on Circuits and Systems, 1991, 2072-2075.
[7] C. Y. Pan, K. T. Cheng: *Pseudo-Random Testing and Signature Analysis for Mixed-Signal Circuits*; ICCAD'95, 102-107.
[8] W. I. Goroszkow: *Elementy urządzeń radioelektronicznych*; Mińsk, 1988.
[9] W. W. Jakowlew, R. F. Fiedorow: *Komputery stochastyczne*; Leningrad, 1974.

DZIGITAL GENERATOR OF SINUSOIDAL SIGNAL FOR TESTING AND VERYFICATION OF ANALOG-DIGITAL CIRCUIT

Summary: The article describes the digital generator of sinusoidal signals which can be a part of a universal module designed for a self-testing of mixed-signal systems. The sinusoidal signal appears to be the most complex (considering its construction) in comparison with the others (e.g. triangular, pentagonal, rectangular signals). A standard generator of a sinusoidal signal is built by a control unit and a block of memory. A memory contains discreet samples of 1/4 part of a period of a sinusoidal signal. The approach presented in the article is based on idea of a stochastic integrator- a fundamental constituent of a generator. The work of a generator was tested by a computer simulation.

Key words: Self-Testing, mixed signal circuits, stochastic integrator, sinusoidal signal

Artykuł zrealizowano w ramach pracy własnej W/II/4/00

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Wiaczesław Jarmolik, Marek Gruszewski[1]

ZASTOSOWANIE INTEGRATORA STOCHASTYCZNEGO W PROCESIE OTRZYMYWANIA WYNIKÓW SAMOTESTOWANIA UKŁADÓW HYBRYDOWYCH

Streszczenie: W pracy przeprowadzono badanie odwzorowania niesprawności testowanego układu w błędy różnego stopnia. Na podstawie tych badań przedstawiony został taki sposób otrzymywania wyników testowania, który jest najbardziej odpowiedni (pozwala zwiększyć wiarygodność wyników) podczas wbudowanego samotestowania układów hybrydowych (analogowo-cyfrowych).

Słowa kluczowe: samotestowanie, układy analogowo-cyfrowe, integrator stochastyczny, analizator sygnaturowy

1. Wprowadzenie

W ostatnich czasach szerokie zastosowanie znajdują układy hybrydowe, czyli takie układy, które w swojej strukturze zawierają obwody analogowe oraz cyfrowe [1]. Jednak sprawdzanie właściwego funkcjonowania procesu wytwarzania układów analogowo-cyfrowych w postaci mikroukładów nastręcza wiele trudności. Są one (jak w przypadku układów cyfrowych o dużym stopniu scalenia) wywołane brakiem dostępu do wewnętrznych punktów układu. Dostęp taki potrzebny jest do podania tam sygnałów testowych oraz pobrania odpowiedzi testowanego układu. Rozwiązaniem tego problemu jest stosowanie (przy projektowaniu układów cyfrowych) specjalnych, dogodnych do testowania metod syntezy. Metody te, to: skanowanie i Wbudowane Samotestowanie (Built-In Self-Test, BIST). Jedną z najbardziej rozpowszechnionych metod organizacji WST układów cyfrowych jest me-

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok

toda oparta na wykorzystaniu sekwencji pseudolosowych jako źródła sygnałów testowych oraz analizatora sygnaturowego (AS) do otrzymania wyników testu [2].

W pracy [3] była przedstawiona po raz pierwszy organizacja BIST w układach analogowo-cyfrowych z wykorzystaniem sekwencji pseudolosowych jako źródła sygnałów testowych i AS do oceny wyników. Rozpatruje się tam odwzorowanie niesprawności układów analogowo-cyfrowych w błędy różnego stopnia i proponuje się metodę, która oparta jest na wykorzystaniu cyfrowego integratora stochastycznego (IS) i która pozwala poprawić zdolność analizatora do wykrywania błędów.

2. Odwzorowanie niesprawności w błędy różnego stopnia dla układów analogowo-cyfrowych

Metodyka przedstawiona w [3] pokazuje, że samotestowanie dla układów analogowo-cyfrowych odbywa się w dwóch etapach: w pierwszym etapie testowana jest część cyfrowa urządzenia, a w drugim etapie testowana jest część analogowa. Należy więc oddzielnie rozpatrywać odwzorowanie niesprawności w błędy dla podukładu cyfrowego i analogowego.

Analiza powstających niesprawności w układach cyfrowych i ich odwzorowanie w błędy różnego stopnia przedstawiona jest w pracy [4]. Analiza ta pokazuje, że zależność zmian prawdopodobieństwa powstania błędu stopnia $\mu = \overline{1,k}$, gdzie k – ilość wyjść układu, może być dowolna w wielu obiektach diagnozowania.

Rozpatrzymy teraz odwzorowanie niesprawności w błędy różnego stopnia w układzie analogowym. Niesprawności w układach analogowych klasyfikowane są jako katastroficzne (hard) i odchylenia (soft) [3]. Ponieważ interesuje nas sygnał na wyjściu podukładu analogowego, to dla ułatwienia będziemy uważać, że niesprawność katastroficzna objawia się na wyjściu jako stała wartość napięcia [5], a niesprawność odchylenia objawia się jako odchyłka sygnału od jego wartości wzorcowej (odniesienia). Analiza odpowiedzi testowanego podukładu przeprowadzana jest w dziedzinie cyfrowej i dlatego sygnał z wyjścia tego układu podawany jest do przetwornika analogowo-cyfrowego (AC), przy pomocy którego otrzymuje się cyfrową wartość sygnału. Zniekształcenie sygnału analogowego prowadzi do zniekształcenia sygnału cyfrowego, który otrzymujemy na wyjściu przetwornika AC, tzn. innymi słowy doprowadzi do pojawienia się błędu określonego stopnia. Tu i wszędzie dalej stopień błędu oznacza ilość niezgodnych bitów kodu cyfrowego dla wartości wzorcowej i rzeczywistej.

Rozpatrzymy (na przykładzie 3-bitowego przetwornika AC z wagą najmłodszego bitu wynoszącą 1V – rys. 1a) odwzorowanie niesprawności katastroficznej i niesprawności odchylenia w błędy różnego stopnia. Na rysunku 1b pokazane

zostały wartości stopnia błędu w funkcji odpowiednich sygnałów wzorcowych przy niesprawnościach katastroficznych – „stała wartość napięcia na wyjściu – 2V" i „stała wartość napięcia na wyjściu – 5V". W ogólnym przypadku przy powstawaniu niesprawności katastroficznej liczbę błędów W_{Hard} stopnia μ dla wszystkich możliwych kodów można znaleźć z zależności:

$$W_{Hard}(\mu) = C_n^{\mu} \tag{1}$$

gdzie:

n – liczba pozycji przetwornika AC.

Teraz rozpatrzymy wpływ niesprawności odchylenia na powstanie błędów różnego stopnia. Niech w testowanym układzie analogowym istnieje niesprawność odchylenia (lub kilka niesprawności odchylenia), pod wpływem których wypadkowy sygnał analogowy na wyjściu układu odchyla się od wartości wzorcowej o – 4V (+5V). Na rysunku 1b przedstawione są wartości stopni błędów dla poszczególnych sygnałów wzorcowych.

Jak widać z rysunku 1 każdej wartości wzorcowej odpowiada własny rozkład stopnia błędu w zależności od wielkości odchyłki. Ogólną liczbę błędów W_{Soft} stopnia μ dla wszystkich wzorcowych kodów i dla wszystkich możliwych odchyłek można wyrazić zależnością:

$$W_{Soft}(\mu) = W_{Soft}^{1} + W_{Soft}^{2} + W_{Soft}^{3} \tag{2}$$

gdzie:

W_{Soft}^{i}, $i = \overline{1,3}$, wyraża się wzorami:

$$W_{Soft}^{1} = \sum_{i=1}^{V} (2^n - NUM_i) \qquad \mu \le n \tag{3}$$

gdzie:

n – liczba bitów przetwornika AC,

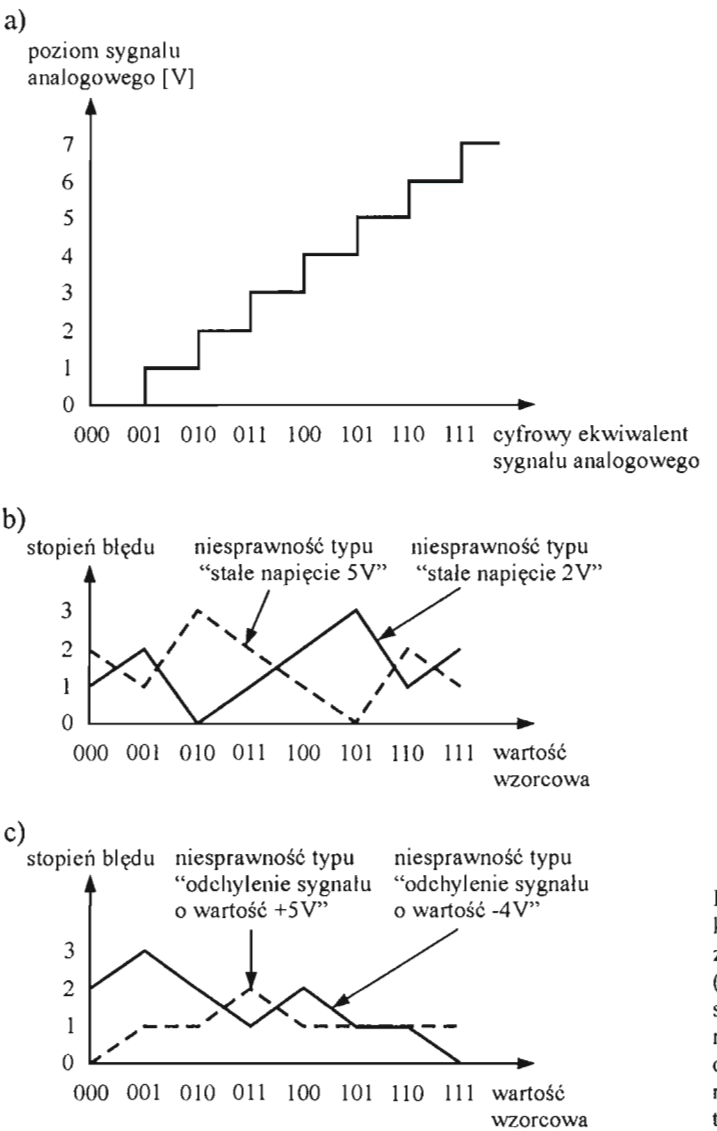
$V = C_n^{\mu}$, NUM_i – wartość n-bitowej liczby dwójkowej o μ jedynkach, przy czym $NUM_i \neq NUM_j$ dla $i \neq j$.

$$W_{Soft}^{2} = C_{n-1}^{\mu-1}(2^{n-1} - 1) \qquad \mu \le n \tag{4}$$

$$W_{Soft}^{3} = \begin{cases} C_{n-1}^{\mu-1}(2^{n-1} - 1) & \mu < n \\ 0 & \mu = n \end{cases} \tag{5}$$

W przypadku czterobitowego przetwornika AC rozkład błędów dla wszystkich wzorcowych kodów i dla wszystkich możliwych odchyłek wygląda następująco: $W_{Soft}(1) = 77$; $W_{Soft}(2) = 90$; $W_{Soft}(3) = 48$; $W_{Soft}(4) = 8$.

W ten sposób urządzenie analizujące odpowiedź układu musi zapewnić równomierne prawdopodobieństwo niewykrywania błędów różnego stopnia.



Rys. 1. Przyporządkowanie kodu cyfrowego AC poziom sygnału analogowego (a), odwzorowanie niesprawności katastroficznych (b) i niesprawności odchylenia (c) w błędy różnego stopnia dla trzybitowego przetwornika AC.

3. Otrzymywanie wyników testowania

Przeanalizujemy teraz sposoby otrzymywania wyników testowania. Najbardziej znane są następujące podejścia do problemu otrzymywania wyników.

1. Otrzymywanie sygnatury przy pomocy generatora pseudolosowych kodów testowych GPKT, który funkcjonuje jak jednokanałowy analizator sygnaturowy (JAS). W tym przypadku zwarta sekwencja binarna podawana jest na wejście GPKT. Stan GPKT po k taktach można przedstawić w następujący sposób [2]:

$$a_i(0) = 0, \qquad i = \overline{1,n}$$

$$a_1(k) = y(k) \oplus \sum_{i=1}^{n} \oplus \alpha_i a_i(k-1) \tag{6}$$

$$a_j(k) = a_{j-1}(k-1), \qquad j = \overline{2,n}$$

gdzie:

n – ilość pozycji GPKT,
$\alpha_i \in \{0,1\}$ – i-ty współczynnik zbudowany na podstawie wielomianu GPKT,
$y(i)$ – i-ty symbol sekwencji.

2. Otrzymywanie sumy kontrolnej metodą zliczania jedynek w sekwencji wejściowej przy pomocy licznika sumującego. Zawartość n – bitowego licznika po k taktach opisany jest następującym wyrażeniem:

$$M(0) = 0 \tag{7}$$

$$M(k) = \left[\sum_{i=1}^{k} y(i) \right] \bmod 2^n$$

3. Jednoczesne otrzymywanie sygnatury na JAS i sumy kontrolnej w liczniku. W rezultacie otrzymamy dwa ścisłe wyniki: sygnatura w JAS i suma kontrolna w liczniku [6].

Oprócz tego rozpatrzymy i porównamy z przedstawionymi wyżej następujące podejścia do otrzymywania wyników testowania.

4. Otrzymywanie wyników testowania przy pomocy stochastycznego integratora – IS (szczegółowy opis IS znajduje się w pracy [7]). Rezultatem kompresji sekwencji wejściowej przez IS jest wartość, która powstała na pozycjach licznika rewersyjnego. Rezultat testu opisać można przy pomocy następującego wyrażenia:

$$M(0) = 0 \tag{8}$$

$$M(k) = \sum_{i=1}^{k} (y(i) - z(i))$$

gdzie:

$y(i)$ – i-ty symbol kompresowanej sekwencji,

$z(i)$ – wartość na wyjściu komparatora w i-tym takcie, która otrzymywana jest na podstawie wyrażenia:

$$z(i) = \begin{cases} 1, & M(i-1) \geq G(i-1) \\ 0, & M(i-1) < G(i-1) \end{cases} \tag{9}$$

gdzie:

$G(i-1)$ – stan GPTN w $i-1$ chwili czasu,

$M(i-1)$ – stan licznika w $i-1$ chwili czasu.

Przy czym $G(0) \neq 0$. Wyrażenie do określenia stanu GPTN w *k*–tej chwili czasu przedstawione jest w [2].

5. Jednoczesne otrzymywanie sygnatury przy pomocy JAS i wyniku w IS. Tryb ten wyróżnia się tym, że funkcjonowanie GPKT opisane jest wyrażeniem (6) i wynikiem jego pracy będą sygnatury, otrzymane w JAS i liczniku.

Rozpatrzymy każdą z wymienionych metod z punktu widzenia wiarygodności analizy kompresowanych sekwencji. Do oceny efektywności metod otrzymywania wyników testu wykorzystywany jest rozkład prawdopodobieństwa p_e^{μ} niewykrycia błędu w zależności od jego stopnia $\mu \in \{1, l\}$, gdzie: l długość kompresowanej sekwencji [2].

Problemy wiarygodności analizy sygnaturowej i metody zliczania jedynek rozpatrywane były dostatecznie szeroko w pracach [2] i [6]. Wyrażenie określające prawdopodobieństwo niewykrycia błędu stopnia μ w JAS przy długości sekwencji $l = 2^n - 1$, gdzie: n – liczba pozycji JAS, przedstawione jest w [2].

Prawdopodobieństwo niewykrycia błędu stopnia μ w metodzie zliczania jedynek określone jest przez wyrażenie [2]:

$$p_e^{\mu} = 0, \qquad \mu = 2k+1 \tag{10}$$

$$p_e^{\mu} = \frac{C_r^{\mu/2} C_{l-r}^{\mu/2}}{C_l^{\mu}}, \qquad \mu = 2k$$

gdzie:

$l = 2^n - 1$ – długość sekwencji,

r – liczba jedynek w sekwencji.

Ponieważ, jak wynika z (10), prawdopodobieństwo niewykrycia błędu stopnia μ w metodzie zliczania jedynek zależy od liczby symboli jedynkowych w sekwencji wejściowej, należy znać średnie prawdopodobieństwo $p^{\mu}_{e\_mid}$ niewykrycia błędu stopnia μ:

$$p^{\mu}_{e\_mid} = \frac{\sum_{r=1}^{l} p^{\mu}_{e}}{l} \tag{11}$$

gdzie:

p^{μ}_{e} – prawdopodobieństwo niewykrycia błędu stopnia μ w metodzie zliczania jedynek w sekwencjach z r jedynkami,

r – liczba jedynek w sekwencji o długości l.

Przy jednoczesnym wykorzystaniu do kompresji sekwencji wejściowej JAS i licznika błąd okazuje się niewykrywalny w takim przypadku, gdy sygnatura i suma kontrolna sekwencji, mającej błąd, jest zgodna z sygnaturą i sumą kontrolną sekwencji wzorcowej [6]. Liczby niewykrywalnych błędów V^{μ}_{e} stopnia μ należy szukać dla każdej wzorcowej sekwencji, tak jak dla każdej sekwencji z r jedynkami istnieje rozkład pary sygnatura plus ilościowa ocena. Średnie prawdopodobieństwo niewykrycia błędu stopnia μ można przedstawić tak:

$$p^{\mu}_{e\_mid} = \frac{\sum_{i=0}^{N-1} V^{\mu}_{e}(i)}{NC^{\mu}_{l}} \tag{12}$$

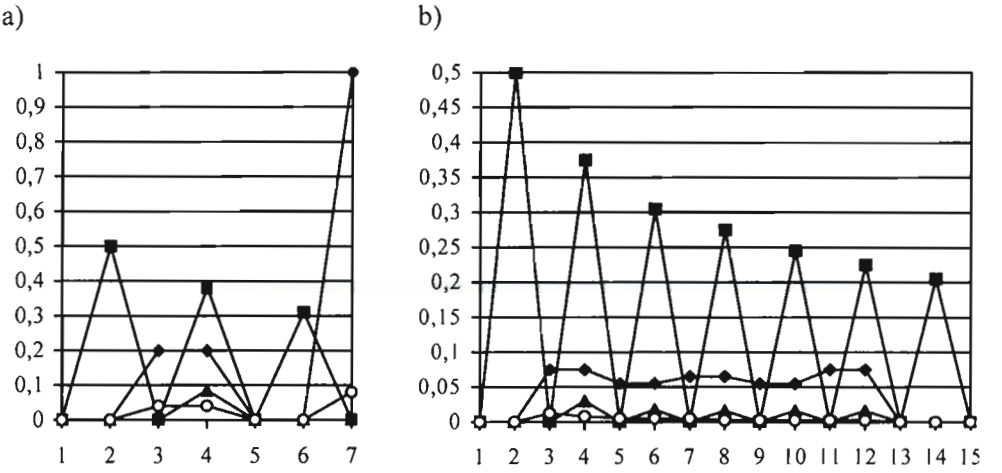
gdzie:

$V^{\mu}_{e}(i)$ – ilość niewykrywalnych błędów stopnia μ dla sekwencji i,

$N = 2^n$, n – ilość pozycji licznika GPKT.

Średnie prawdopodobieństwo $p^{\mu}_{e\_mid}$ niewykrycia błędu stopnia μ jest równe zero w tych przypadkach, gdy ilość niewykrywalnych błędów wynosi zero dla analizy sygnaturowej lub dla metody zliczania jedynek.

W przypadku wykorzystania integratora stochastycznego lub równolegle IS i JAS, to jak widać z (8), sygnatura rezultatu znajduje się w złożonej funkcjonalnej zależności od wartości sekwencji wejściowej, stanu początkowego GPKT i jego zadanego wielomianu. Dana zależność utrudnia analityczne określenie prawdopodobieństwa niewykrycia błędu stopnia μ. W ogólnym przypadku średnie prawdopodobieństwo niewykrywania błędu można określić zgodnie z (12), gdzie $V_e^{\mu}(i)$ – ilość niewykrywalnych błędów stopnia μ dla IS lub IS i JAS.



Rys. 2. Prawdopodobieństwa niewykrycia błędu stopnia μ w sekwencji o długości 8 bitów dla trzybitowego JAS, licznika i integratora (a), dla sekwencji o długości 15 bitów oraz czterobitowego JAS, licznika i integratora (b), gdzie: –◆– – JAS, –■– – licznik, –▲– – licznik i JAS, –○– – integrator i JAS.

Na rysunku 2 przedstawiona jest zależność prawdopodobieństwa niewykrywania błędu stopnia μ dla różnych ilości pozycji GPKT i długości sekwencji wejściowej. Na rysunku 2b nie przedstawiono wartości prawdopodobieństwa niewykrywania błędu stopnia $\mu = 15$ dla JAS. Wartość tego prawdopodobieństwa wynosi 1. Jak widać z rysunku 2 i wynika z wyżej przytoczonego wymagania na równomierność rozkładu prawdopodobieństwa niewykrywania błędów różnego stopnia, można wywnioskować, że tryb otrzymywania oceny jednocześnie na IS i JAS jest najbardziej odpowiedni do tego celu.

4. Zakończenie

W pracy przeprowadzono badanie odwzorowania niesprawności układów analogowo-cyfrowych w błędy różnego stopnia i przedstawiono sposób otrzymywania wyników, który pozwala zwiększyć wiarygodność testowego eksperymentu przy pomocy wbudowanego samotestowania układów analogowo-cyfrowych.

Literatura

[1] R. I. Gruszwicki, A. Ch. Mursajew, W. B. Smołow: *Analogowo-cyfrowe urządzenia peryferyjne w systemach mikroprocesorowych*; Leningrad 1989.
[2] W. N. Yarmolik: *Kontrolowanie i diagnostyka cyfrowych podzespołów komputerowych*; Nauka i technika, Mińsk 1988.
[3] M. Ohletz: *Hybrid Built-In Self-Test (HBIST) for Mixed Analogue/Digital Circuits*; IEEE European Test Conference, 1991, 307-316.
[4] W. N. Yarmolik, I, V. Kaczan: *Określenie stopnia błędów, wywoływanych niesprawnością układów cyfrowych; Automatyka i technika komputerowa nr 18*, 1989, 101-106.
[5] M. Soma: *Fault Modeling and Test Generation for Sample-and-Hold Circ*uits; International Symposium on Circuits and Systems, 1991, 2072-2075.
[6] J. P. Robinson, N. R. Saxena: *Simultaneous Signature and Syndrome Compression*; IEEE Trans. on CAD Vol. 7 N5, 1988, 584-589.
[7] W. W. Jakowlew, R. F. Fiedorow: *Komputery stochastyczne*; Leningrad 1974.

MIXED SIGNAL TESTING BASED ON COMPACT ESTIMATES BY STOCHASTIC INTEGRATOR

Summary: The work proposes the analysis of an inefficiency of a tested system in different degree errors. On the ground of this examination the most applicable method of receiving test results is presented. It allows to increase the credibility of results during hybrid built-in self-tests.

Key words: Self-Testing, mixed signal circuits, stochastic integrator, signature analyser

Artykuł zrealizowano w ramach pracy własnej W/II/4/00

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Joanna Karbowska[1]

AN ALGORITHM FOR SOLVING THE TAUTOLOGY PROBLEM

Abstract: In the present paper we propose an algorithm deciding, whether a given formula α is a tautology of the propositional logic. The worst case complexity of the algorithm is of the order $O((1,62)^{|\alpha|})$.

Key words: decidability procedure, propositional tautology, worst case complexity of the algorithms

1. Preliminaries

We will present and analyze new algorithm for solving the tautology problem. There are many different algorithms solving the tautology and the satisfability problems for formulas of the propositional logic, (cf. for example [1], [2], [3]). In this paper we decide to use only one logical connective of implication to study influence of its asymmetry on the quality of analysis of the formula.

First test of the algorithm seems to us advantageous. We plan further experiments and their statistic estimation. The present paper does not contain theoretical analysis of the average case complexity of the algorithm.

1.1. Formulas

We first define the language of propositional logic. Its semantics is assumed to be standard; the language is interpreted in the two-element Boolean algebra $\boldsymbol{B_0}$ with the universe $\{0, 1\}$. The logical connectives are interpreted as usual.

[1] Department of Computer Science, Bialystok University of Technology, Wiejska 45A, 15-351 Bialystok, Poland, e-mail: asia@chilan.com

1.1.1. Syntax

The alphabet A of the language L contains countable set of propositional variables denoted by p, q, r... and the logical connectives:

$$A = \{\neg, \wedge, \vee, \rightarrow, (,), 0, 1, p, q, r...\}$$

The set F of formulas is the least subset of A^* *such* that:

1. the symbols $0, 1, p, q, r...$ belong to F,
2. if α, β belong to F then $\neg\alpha$, $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta)$ belong to F.

The subset of F, consisting of all formulas not containing propositional connectives different from the implication symbol will be denoted by $F_{\{\rightarrow\}}$. The subset of F, consisting of all formulas not containing propositional connectives different from the implication symbol and containing the constant 0 will be denoted by $F_{\{\rightarrow,0\}}$.

By $Var(\alpha)$ we shall denote the set of all variables occurring in α. The length of the formula α will be denoted by $|\alpha|$. The number of (different) variables occurring in α will be denoted by $|Var(\alpha)|$. The number of occurrences of the implication symbol in α will be denoted by $NI(\alpha)$.

Let $\alpha, \beta \in F$ be the formulas and let p be a variable. By $\alpha[p/\beta]$ we shall denote the formula obtained from α after simultaneous replacing all occurrences of the variable p by the formula β.

1.1.2. Semantics

We shall denote by ν a valuation of variables $\nu : \{p, q, r...\} \rightarrow \{0,1\}$. The value of a formula $\alpha \in L$ for valuation ν is defined in the standard way (using usual interpretation of the logical connectives in two-element Boolean algebra $\boldsymbol{B_0}$) and will be denoted by $\alpha[\nu]$ (cf. [5]). In the sequel we shall use the following lemma.

Lemma 1. *For every formula* $\alpha \in F$ *one can construct effectively an equivalent formula* $\alpha^* \in F_{\{\rightarrow,0\}}$ *containing only the implication symbol, the constant* 0 *and variables from the set* $Var(\alpha)$ *and the parentheses. Moreover, the length of the formula* α^* *is of the order* $O(|\alpha|)$.

Remark 1. *It is easy to propose simple algorithms of the order* $O(|\alpha|)$ *realizing transformations the mentioned above.*

2. An algorithm solving the tautology problem

In this section we shall describe the idea of an algorithm deciding for a formula $\alpha \in F_{\{\rightarrow,0\}}$, whether α is a tautology or not. The algorithm will transform sequences of formulas of F starting from one element sequence. The steps of transformation are described below in an informal way as the transition rules. The algorithm halts if there is no suitable rule for a given sequence of formulas. If the final sequence is empty then the formula α is a tautology, otherwise α is not tautology. Formally, the rules of transformation realized by the algorithm will be written in the form

$$\langle \beta, \gamma_1, ..., \gamma_m \rangle \rightarrow \langle \beta_1, ..., \beta_k, \gamma_1, ..., \gamma_m \rangle .$$

Speaking informally $\beta_1, ..., \beta_k$ will denote the result of the transformation of the initial formula β.

We shall use the abbreviate form of such a rule if it does not lead to misunderstanding:

$$\langle \beta \rangle \rightarrow \langle \beta_1, ..., \beta_k \rangle .$$

In the case $m = 0$, $\gamma_1, ..., \gamma_m$ is the empty sequence and will be denoted by $\langle\ \rangle$.

2.1. Transition rules of the algorithm

We present a set of the transformation rules, that describe the main steps of the algorithm. The formulas occurring in these rules belong to the language $F_{\{\rightarrow,0\}}$ (i.e. each formula α_i occurring in a transition rule contains only parenthesis, the implication sign, variables, and the constant 0 only). Transformation rules are based on following equivalences:

$$(\gamma \vee \delta) \equiv (\neg\gamma \rightarrow \delta),$$
$$(\gamma \wedge \delta) \equiv ((\gamma \rightarrow \neg\delta) \rightarrow 0),$$
$$\neg\gamma \equiv (\gamma \rightarrow 0),$$
$$1 \equiv (0 \rightarrow 0)$$

and consist of analyzing the subformula after ''the last implication sign''.

1. $\langle\ (\ \alpha_1 \rightarrow (\alpha_2 \rightarrow ... (\alpha_{k-2} \rightarrow (\alpha_{k-1} \rightarrow (0 \rightarrow 0))) ...\ ))\ \rangle \rightarrow \langle\ \rangle,$
2. $\langle\ (\ \alpha_1 \rightarrow (\alpha_2 \rightarrow ... (\alpha_{k-2} \rightarrow ((\delta' \rightarrow \delta'')\rightarrow 0)) ..\ ))\ \rangle$
$\rightarrow \langle\ (\ \alpha_1 \rightarrow (\alpha_2 \rightarrow ... (\alpha_{k-2} \rightarrow \delta') ..\ )),\ (\ \alpha_1 \rightarrow (\alpha_2 \rightarrow ... (\alpha_{k-2} \rightarrow (\delta'' \rightarrow 0)) ..)\ )\ \rangle,$

3. $\langle\ (\ \alpha_1 \to (\alpha_2 \to ...(\alpha_{k-2} \to (q \to p))...)\ \rangle \to$
$\langle\ (\ \alpha_1[p/0, q/(0 \to 0)] \to (\alpha_2[p/0, q/(0 \to 0)] \to ...(\alpha_{k-2}[p/0, q/(0 \to 0)] \to 0)...)\ \rangle$,

4. $\langle\ (\ \alpha_1 \to (\alpha_2 \to ...(\alpha_{k-2} \to ((0 \to 0) \to p))...)\ \rangle \to$
$\langle\ (\ \alpha_1[p/0] \to (\alpha_2[p/0] \to ...(\alpha_{k-2}[p/0] \to 0)...)\ \rangle$,

5. $\langle\ (\ \alpha_1 \to (\alpha_2 \to ...(\alpha_{k-2} \to (0 \to p))...)\ \rangle \to \langle\ \rangle$.

Lemma 1. *In the above set of rules, their left parts exhaust all possible form of a formula of* $F_{\{\to,0\}}$.

□

We shall argue this as follows: the symbols of formula α are analyzed. If α contains the implication sign then we enumerate the occurrence of implication sign:

$$\alpha : \left(\alpha_1 \underset{1}{\to} \alpha' \right)$$

$$\alpha : \left(\alpha_1 \to \left(\alpha_2 \underset{2}{\to} \alpha'' \right) \right)$$

...

$$\alpha : \left(\alpha_1 \to \left(\alpha_2 \to ... \left(\alpha_k \underset{k}{\to} \alpha^{(k)} \right) \right) \right)$$

until the last formula $\alpha^{(k)}$ does not contain the implication sign. The rule 1 corresponds to the case where α_k and $\alpha^{(k)}$ are the constant 0. The case where α_k is the constant 0 and α_{k-1} is the form $\delta' \to \delta''$ described rule number 2. If α_k is a variable are possible three forms of α_{k-1}. The rule 3 corresponds to the case where α_{k-1} is a variable. The case where α_{k-1} is $(0 \to 0)$ we considered in rule 4 and case where α_{k-1} is constant 0 described rule 5.

Remark 1. *Note, that each of the rules given below leads to a part of the algorithm. For the first rule an informal description of this algorithm can be formulated as follows:*
the symbols of the formula α *are analyzed until we get the sequence* "$\to (0 \to 0)$" *of six symbols.*

We shall write $s \mapsto \ldots \mapsto s'$ if and only if s' can be sequence of formulas obtained from sequence s by use of rules of inference.

The idea of transforming sequences of formulas is to preserve tautologies. The following lemmas explain this fact precisely.

Lemma 2. *If formula* β *is a tautology and* $\langle\beta\rangle \to \langle\beta_1,\ldots,\beta_k\rangle$ *is a transition rule then the formulas* $\beta_1,\ldots,\beta_k$ *are tautologies.*
If formula β *is not a tautology and* $\langle\beta\rangle \to \langle\beta_1,\ldots,\beta_k\rangle$ *is a transition rule then at least one of formula* $\beta_1,\ldots,\beta_k$ *is not a tautology.*

As a corollary we obtain:

Lemma 3. If $s \mapsto \ldots \mapsto s'$ *then all formulas from* s *are tautologies if and only if all formulas from* s' *are tautologies.*

2.2. The algorithm

The informal description of the algorithm is the following. Let α be an arbitrary formula of $F_{\{\to,0\}}$. If α is a variable or a constant 0 then there is no admissible transition. Otherwise, α contains at least one implication sign, i.e., α is of the form $(\alpha_1 \to \alpha')$. If α' contains at least one implication symbol then α is of the form $(\alpha_1 \to (\alpha_2 \to \alpha''))$ and so on. So, after a number k steps we determine that α is of the form $(\alpha_1 \to (\alpha_2 \to \ldots(\alpha_{k-1} \to \alpha_k)\ldots)))$ where α_k does not contain the implication symbol.

This means that implication sign before α_k is the last implication sign in the whole formula α and that α_k is a variable or the constant.

In the first case, where α_k is a variable, we need to analyze the all possible forms of α_{k-1} (cf. the rules 3-5). If α_{k-1} is the constant 0 or the formula $(0 \to 0)$ then the rule 5 or 4 is respectively used.

If α_{k-1} contains an implication sign (i.e., α_{k-1} is of the form $\delta' \to \delta''$) then the rule 2 is used.

Note, that in the rules 1-5 all possible forms α_{k-1} were considered (cf. Lemma 1).

We present the algorithm in a more formal manner as follows:
Let $s = (\alpha_1\ \alpha_2\ \ldots\alpha_k)$ be a sequence of formulas.

Table 1

```
input (s)

DT{ while not[(s contains a variable or a constant) or ( s is empty sequence)] do
    {begin
       α := first formula of s ;
       s' := sequence remaining formulas of s ;
       determine the rule applicable to the formula α ;
       s'' := the sequence of formulas appearing on the right side of the rule
              (provided that α is on the left side)
       s := concatenate s' and s''
    end;

if s is empty sequence then answer: = all formulas of s are tautologies
else answer: = then at least one of formula from s is not a tautology;

output (answer).
```

Example 1

We have formula α': $(p \to (q \to ((p \to (p \to 0)) \to 0)))$. The following table present steps of the algorithm for formula α.

Table 2

step of the algorithm	sequence s	used rule	sequence s''
1	2	3	4
1	$\langle\ (p \to (q \to ((p \to (p \to 0)) \to 0)))\ \rangle$	2	$\langle\ (p \to (q \to p)),$ $(p \to (q \to ((p \to 0) \to 0)))\ \rangle$
2	$\langle\ (p \to (q \to p)),$ $(p \to (q \to ((p \to 0) \to 0)))\ \rangle$	3	$\langle\ (0 \to ((0 \to 0) \to 0))\ \rangle$
3	$\langle\ (0 \to ((0 \to 0) \to 0)),$ $(p \to (q \to ((p \to 0) \to 0)))\ \rangle$	2	$\langle\ (0 \to 0), (0 \to (0 \to 0))\ \rangle$
4	$\langle\ (0 \to 0), (0 \to (0 \to 0)),$ $(p \to (q \to ((p \to 0) \to 0)))\ \rangle$	1	$\langle\ \rangle$

etc. tabele 2

1	2	3	4
5	$\langle\ (0\to(0\to 0)),$ $(p\to(q\to((p\to 0)\to 0)))\ \rangle$	1	$\langle\ \rangle$
6	$\langle\ (p\to(q\to((p\to 0)\to 0)))\ \rangle$	2	$\langle\ (p\to(q\to p)),(p\to(q\to(0\to 0)))\ \rangle$
7	$\langle\ (p\to(q\to p)),(p\to(q\to(0\to 0)))\ \rangle$	3	$\langle\ (0\to((0\to 0)\to 0))\ \rangle$
8	$\langle\ (0\to((0\to 0)\to 0)),$ $(p\to(q\to(0\to 0)))\ \rangle$	2	$\langle\ (0\to 0),(0\to(0\to 0))\ \rangle$
9	$\langle\ (0\to 0),(0\to(0\to 0))$ $(p\to(q\to(0\to 0)))\ \rangle$	1	$\langle\ \rangle$
10	$\langle\ (0\to(0\to 0))$ $(p\to(q\to(0\to 0)))\ \rangle$	1	$\langle\ \rangle$
11	$\langle\ (p\to(q\to(0\to 0)))\ \rangle$	1	$\langle\ \rangle$

Answer: formula α' is a tautology.

Example 2

We have following formula $\alpha' : (p\to(q\to((p\to q)\to 0)))$.

Table 3

step of the algorithm	sequence s	used rule	sequence s''
1	2	3	4
1	$\langle\ (p\to(q\to((p\to q)\to 0)))\ \rangle$	2	$\langle\ (p\to(q\to p)),(p\to(q\to(q\to 0)))\ \rangle$
2	$\langle\ (p\to(q\to p)),(p\to(q\to(q\to 0)))\ \rangle$	3	$\langle\ (0\to((0\to 0)\to 0))\ \rangle$
3	$\langle\ (0\to((0\to 0)\to 0)),$ $(p\to(q\to(q\to 0)))\ \rangle$	2	$\langle\ (0\to 0),(0\to(0\to 0))\ \rangle$
4	$\langle\ (0\to 0),(0\to(0\to 0)),$ $(p\to(q\to(q\to 0)))\ \rangle$	1	$\langle\ \rangle$
5	$\langle\ (0\to(0\to 0)),$ $(p\to(q\to(q\to 0)))\ \rangle$	1	$\langle\ \rangle$
6	$\langle\ (p\to(q\to(q\to 0)))\ \rangle$	3	$\langle\ (p\to((0\to 0)\to((0\to 0)\to 0)))\ \rangle$

etc. tabele 3

1	2	3	4
7	$\langle\ (p \to ((0 \to 0) \to ((0 \to 0) \to 0)))\ \rangle$	2	$\langle\ (p \to ((0 \to 0) \to 0)),$ $(p \to ((0 \to 0) \to (0 \to 0)))\ \rangle$
8	$\langle\ (p \to ((0 \to 0) \to 0)),$ $(p \to ((0 \to 0) \to (0 \to 0)))\ \rangle$	2	$\langle\ (p \to 0),(p \to (0 \to 0))\ \rangle$
9	$\langle\ (p \to 0),(p \to (0 \to 0)),$ $(p \to ((0 \to 0) \to (0 \to 0)))\ \rangle\ \rangle$	3	$\langle\ ((0 \to 0) \to 0)\ \rangle$
10	$\langle\ ((0 \to 0) \to 0),(p \to (0 \to 0)),$ $(p \to ((0 \to 0) \to (0 \to 0)))\ \rangle\ \rangle$	2	$\langle\ 0,(0 \to 0)\ \rangle$

Answer: formula α' is not a tautology.

3. Correctness of the algorithm

In this section we shall consider the problem of correctness of the algorithm.

We shall prove that the main part $\boldsymbol{T}$ of the algorithms satisfies the following partial correctness formulas of Hoare logic (cf. [6]) (written in an informal way)

$\{s$ is the sequence of tautologies$\}$ $\boldsymbol{T}$ $\{s$ is the sequence of tautologies$\}$

$\{s$ contains a formula not being a tautology$\}$ $\boldsymbol{T}$ $\{s$ is not the sequence of tautologies$\}$

This is an easy consequence of Lemma 2 from the previous chapter.

The algorithm consists of transforming sequences of formulas according to the rules. This means, that if the initial sequence only include tautologies, then after each step of the algorithm (the realization of a transition rule) the actual sequence will contain tautology only. Conversely, if the initial sequence contains a formula not being tautology, then during the computation each sequence will contain a formula that is not a tautology. Therefore the algorithm can accept (i.e., the output sequence is empty) only sequence consisting of tautologies. If an input sequence of formulas contains components, that are not tautologies, then the output sequence will not be empty. Note that, in this case, the output sequence will contain at least one variable.

3.1. Partial correctness

Let s be a initial sequence $(s = (\alpha_1\ \alpha_2 \ldots \alpha_k))$ for that algorithm stops. $s_1, s_2, \ldots, s_n$ are sequences of formulas corresponding to the realization of the transition rules the computation of the algorithm. Then s_1 consists of tautologies if

sition rules the computation of the algorithm. Then s_1 consists of tautologies if and only if s_n contains tautology only. s_1 consists a formula not being tautology if and only if s_n contains a formula that is not a tautology.

Thus using the Hoare rule $\dfrac{\{\delta \wedge \gamma\} T \{\delta\}}{\{\delta\} \textit{while}\ \gamma\ \textit{do}\ T\ \{\delta \wedge \neg\gamma\}}$ we obtain the formulas for ***D***:

{ s is a sequence of tautologies} ***D*** {[s is a sequence of tautologies] $\wedge\neg$ [$\neg$ ((s contains a variable or a constant) or (s is empty sequence))] }

with the following meaning of δ and γ:

δ: s is a sequence of tautologies,

γ: $\neg$ ((s contains a variable or a constant) or (s is empty sequence)).

Similar, we can repeat the reasoning for the formula.

{ s contains a formula not being a tautology} ***D*** {[s contains a formula not being a tautology] $\wedge\neg$ [$\neg$ ((s contains a variable or a constant) or (s is empty sequence))] }

with δ: s contains a formula not being a tautology,

γ: $\neg$ ((s contains a variable or a constant) or (s is empty sequence)).

3.2. Halting property

Let us notice that the algorithm is constructed in this way that always stops. It is sufficient to note that formulas appearing on the right side of a rule are shorter than formula on the left side.

Consequently, last sequence s_n is empty or contain variable or constant (otherwise a rule of transition may be applied). In the first case this means that the initial sequence consists of the tautologies. In the second case, this means that initial sequence contains at least one formula not being tautology.

4. Complexity of the algorithm

Let us denote by $t(\alpha)$ the number of rules used by the algorithm during its execution with the initial sequence α. By $T(n)$ we denote the maximum of all $t(\alpha)$, where the length of α is not greater than n, $T(n) = \max\{t(\alpha) : |\alpha| \leq n\}$.

Before we estimate the complexity of the algorithm, we analyze the number of implications in each rule. Let α be a formula and let n denote number of implications in α, i.e., $NI(\alpha) = n$. It is easy to observe, that the number of implications on the right hand side of the rule does not exceed $n-1$ (i.e. rules 3,4) and $(n-1)+(n+2)$ (the rule 2).

Remark 1. *Let us notice that the parts of the algorithms related to the elimination of propositional connectives different from implication sign and choice the suitable rule are the order* $O(n)$

After analysis for all rules we can conclude:
$T(n) \leq \max\{t(\alpha)\ where\ |\alpha| \leq n-1, t(\alpha)+t(\beta)\ where\ |\alpha| \leq n-2, |\beta| \leq n-1\} + C \times n$
where $C \leq 4$. The first expression is related to the rules 3 and 4, the rule 2 leads to the second expression.

Finally $T(n) \leq T(n-1)+T(n-2)+C \times n$, where $C \leq 4$. To estimate the worst case complexity of the algorithm we solve the following recursive equation:
$T(n) = T(n-1)+T(n-2)+C \times n$ where $C = 4$ and $T(0) = 1$, $T(1) = 1$.

The part $T(n-1)$ corresponds to the number of application of transition rules during the realization of the algorithm for formula β (one element sequence consisting of the formula β). Similarly $T(n-2)$ corresponds to the formula α. If we replace n by $n-1$ then we obtained: $T(n-1) = T(n-2)+T(n-3)+C \times (n-1)$ and successively $T(n) = 2T(n-1)-T(n-3)+C$. Similarly, we can eliminate constant C from previous equation. Finally we obtained a linear recursive equation:
$T(n) = 3T(n-1)-2T(n-2)-T(n-3)+T(n-4)$.
It is known (cf. [1]), that every solution of this equality is of the form:
$T(n) = C_1 \times (r_1)^n + C_2 \times (r_2)^n + C_3 \times (r_3)^n + C_4 \times (r_4)^n$,
where $C_1, C_2, C_3, C_4 \in R$ and r_1, r_2, r_3, r_4 are zeros of the polynomial $x^4 - 3x^3 + 2x^2 + x - 1 = 0$. Note that:
$x^4 - 3x^3 + 2x^2 + x - 1 = (x-1)^2 (x - \frac{1-\sqrt{5}}{2})(x - \frac{1+\sqrt{5}}{2})$. It is easy to verify that $|r_1|, |r_2|, |r_3|, |r_4| \leq 1{,}62$. Thus $T(n) \leq K(1{,}62)^n$, where K is a constant.

The average case complexity will be discussed in a separate paper.

References

[1] A. V. Aho, J. E. Hopcroft , J. D. Ullman: *Projektowanie i analiza algorytmów komputerowych*, PWN, Warszawa 1983.

[2] B. Monien, E.: *Speckenmeyer*, Solving satisfability in less than 2^n steps, Discrete Appl. Math. 10 (1985), no 3, 287-295.

[3] C. Papadimitriou: *Computational Complexity*, Addison – Wesley (1994).

[4] N. Deo, J. Nievergelt, E. M. Reingold: *Algorytmy kombinatoryczne*, PWN, Warszawa 1985.

[5] H. Rasiowa, R. Sikorski: *The matematics of metamathematics*, PWN, Warszawa 1970.

[6] S. Alagić, M. A. Arbib: P*rojektowanie programów poprawnych i dobrze zbudowanych*, WNT, Warszawa 1982.

O PEWNYM ALGORYTMIE BADANIA TAUTOLOGII RACHUNKU ZDAŃ

Streszczenie: W pracy został przedstawiony algorytm sprawdzania czy dana formuła rachunku zdań należąca do języka $F_{\{\rightarrow,0\}}$, zbudowana ze zmiennych, stałej zero, nawiasów oraz znaku implikacji, jest tautologią. Idea algorytmu jest oparta na regułach transformacji. Reguły te zachowują tautologiczność formuły, tzn. formuła α, która jest tautologią po zastosowaniu odpowiedniej reguły w dalszym ciągu jest tautologią. Jeśli dana formuła $\alpha \in F_{\{\rightarrow,0\}}$ jest różna od stałej lub zmiennej, to w formule α znajdujemy ostanie wystąpienie znaku implikacji i stosujemy odpowiednią regułę otrzymując nowy ciąg formuł. Formuły przekształcamy do momentu, kiedy otrzymamy ciąg pusty (wtedy jest ona tautologią), bądź zmienną bądź stałą. Złożoność tego algorytmu, w najgorszym przypadku, wynosi $O((1{,}62)^{|\alpha|})$, gdzie $|\alpha|$ oznacza długość formuły.

Słowa kluczowe: metoda rozstrzygania, tautologia rachunku zdań, złożoność pesymistyczna algorytmu

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Mieczysław A. Kłopotek[1, 3, 4], Sławomir T. Wierzchoń[1, 2]

DISTRIBUTED ENUMERATION PROTOCOL FOR VALUATION BASED SYSTEMS

Abstract: The paper presents a new algorithm for the problem of an enumeration protocol for nodes in a network. The new algorithm, contrary to previous ones, is local both in information access (neighbourhood only) and information stored (proportional to the number of neighbours). This property is achieved at the expense of the type of connectivity the network is assumed to exhibit.

Key words: enumeration, local computation, triangulated graphs, join trees

1. Introduction

Mazurkiewicz [9], [10] poses the problem of enumeration of nodes of a graph by local transformations. He points to three major arguments for the need of considering such a problem:

- Local transformations of graph labelings are good models of distributed computation
- Enumeration itself is a good model for resolving global conflicts by local means
- The theoretical problem is by itself interesting

In the paper [10] he provides with a solution to this general problem. The proposed solution is general in that it holds most types of undirected graphs connecting the nodes in the network, except for some pecular locally highly symmetric networks, where the decision cannot be made. The solution appears to have a certain disadvantage, however, as each node has to keep itself separately information

[1] Institute of Computer Science, Polish Academy of Sciences
[2] Dept of Computer Science, Białystok University of Technology
[3] Dept. of Computer Science, University of Podlasie
[4] Institute of Mathematics, Warsaw University of Technology
e-mail: {klopotek,stw}@ipipan.waw.pl

received from all the nodes of the network, which it eventually receives. That is, actually, each node reflects the structure of the whole network. Also a single node has to be activated many times.

In this paper we are asking whether it is possible to invent a protocol that would assure that each node has a totally local behavior, that is:

- Each node collects information only from its neighbors
- Each node holds only an amount of information proportional to its neighbourhood and not to the entire network.
- Each node is activated only at most a fixed number of times during the process.

We shall call this constrained setting as "strictly local computation problem".

As Mazurkiewicz demonstrates the necessity of his solution in the general case, it is obvious that we have to look for solutions for this more constraining local computation by turning to a special set of graphs.

In the past, strictly local computations have been investigated intensely in the context of computations of marginal distributions of joint probability distributions in graphical expert systems (called also Bayesian Networks). Bayesian Networks were transformed to a special representation of so-called Markov trees (hypertrees) e.g. via the so-called the trangularization process (consult [13] for details). A triangulated graph is one in which any loop with more than three nodes has a direct connection in the graph between a pair of non-neighbors in the loop. After the triangularization the computational problem, that was otherwise exponential in time and space in the total number of variables, turned to be bounded by local neighbourhood size so that computations could run in both reasonable time and reasonable space consumption.

As it turned out, triangulated graphs are applicable not only to decomposition of Bayesian networks, but also to Dempster-Shafer networks, to necessity calculus and to Spohn's disbelief function and to some reasoning problems in the fuzzy set theory [14]. So the original local reasoning engine was extended from Bayesian networks to the more general concept of "valuation based systems" (VBS). Many types of reasoning processes like conditional probabilities or belief functions computations may be carried out efficiently in the VBS framework. Also such hard tasks like finding most probable explanation or k-most-probable explanations for a hypothesis, some decision making tasks etc. may be solved efficiently using the formalism of the valuation based systems.

The idea of graphical expert systems, called also Bayesian networks (BN) was initiated by Pearl, [11], and worked-out by Lauritzen and Spiegelhalter, [8]. Its generalization to different uncertainty formalisms was proposed by Shenoy [15] under the name valuation based systems (VBS in short).

Since the VBS proved to be very efficient in diverse local computation task, we posed the question whether it can also be applied to the enumeration by local computation problem.

2. Problem formulation

Let X = $\{X_1,...,X_n\}$ be a set of variables with discrete domains $D_1,..., D_n$, respectively. Each domain is a set of integers $0,1...n$. 0 meaning the node does not possess any enumeration value assigned, otherwise it has assigned a value under enumeration. Obviously our goal is to assign each X a different natural number value. Let D stands for the Cartesian product of these domains, $D = D_1 \times ... \times D_n$; it represents the space of total configurations. Furthermore let $E=\{\{ X_i, X_j\} \mid X_i, X_j \in X, X_i \neq X_j \}$ be a set of undirected edges over X. Let us call (X,E) a network.

We are obviously interested in finding a configuration of the network in which all the values of all the variables are distinct and ranging from 1 to n.

For this purpose we want to elaborate a protocol for local computations that is with each node acquiring its "identity number" only by communicating its direct neighbors. For a given node X_i its neighborhood $Nh(X_i)$ be defined as a set $Nh(X_i)=\{ X_j \mid \{ X_i, X_j\} \in E\}$.We say that any subset of $Nh(X_i)$ such that for any X_k, $X_j \in Nh(X_i)$ we have $\{X_i, X_j\} \in E$ is a completely connected subneighborhood of X_i. Let us say in passing that a set consisting of all sets comprising of nodes from X joint with their completely connected subneighborhoods constitute a hypergraph H over X such that it is equivalent to E.

We want now to have enumeration of nodes based on some local criterion, with limited information flow. In particular we want to distinguish one node that has the highest number assigned, and then starting from it assign numbers to all the other nodes. The problem is of course that locally no node knows how many nodes are there in the network.

One idea for choosing the starting node would be that of graph trangulation. It consists essentially in removal steps until only one node remains. This would be the winner. Then the sequential numbers should be assigned corresponding to the removal order. We want to elaborate this idea in more detail in the subsequent section.

3. Problem solution

3.1. The general idea

Imagine we create a "spanning tree" over the network such that given a node with highest value all the subtrees in the tree after removing it have values only within some range each. The same happening in all the subtrees. To have a valuation like that we would have to pick a node from the network, give it the entire range of values 1..n and to pass to it the control. The node in control would assign itself the highest rank available and give its neighbors disjoint subranges of values and pass to each of them the control as to separate subtrees. In this case the node with the highest rank node of all and in each subtree would have to possess knowledge only of the range it has been assigned to (two integers) and the number of nodes within each of subtrees originating from it (one integer per neighbor).

The problem is now to find the spanning tree based only on local properties (the actual neighborhood) of each node.

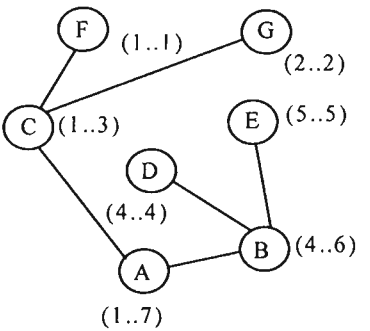


Fig. 1. The idea of range passing

In Figure 1 you see the idea of message passing over the tree. A node A receives the whole range of values (1..7), retains the trop value (6) and passes the range (1..3) to C and (4..6) to B and so forth. Of course, A would have to be informed about the cardinality of nodes in the tree sub-branch starting at B and in that starting at C.

3.2. Triangulation and junction trees

The task to find a spanning tree may be understood as identical with finding a removal sequence of nodes such that the graph is connected all the time. How to do a thing like this based on local properties? If we remove a node only if its actual

neighborhood is completely connected then we have a guarantee that the whole graph will remain connected. But is this procedure granted a success?

To find good deletion ordering Shenoy [15] proposed a metaphor being in a sense a redefinition of the graphical approach of Bertele and Brioschi [2]. Suppose H = $\{H_1,..., H_k\}$ is a family of subsets of X (to be thought of as nodes with their completely connected neighborhoods), and $D(H_i)$ is the Cartesian product of the domains of variables in H_i.

For a given undirected graph G (being our network) consider the set H=$\{H_1,..., H_k\}$ of all its maximal (that is not containing one another) completely connected subgraphs. H_i can be treated as a set of nodes.

Among all such sets of special importance are those obtained from triangulated graphs. For a triangulated graph G, let us denote the set of its completely connected subgraphs with N They admit two attractive properties. First, the elements of N, denoted N_j, $j = 1,...,q$, (q is the cardinality of N) can always be ordered in such a way that the *running intersection property*[1], or RIP for short, holds:

$$(\forall k \geq 2)(\exists j<k)\colon N_j \supseteq N_k \cap (N_1 \cup ... \cup N_{k-1})$$

This property means that the variables in node N_k also contained in previous nodes ($N_1,..., N_{k-1}$) are all members of one previous node, N_j. Second, the elements of N can be organized into a tree, called *junction tree*. To introduce this tree we need further definitions. The set

$$S_k = N_k \cap (N_1 \cup ... \cup N_{k-1})$$

is said to be *separator*: it separates the *residual*

$$R_k = N_k \backslash S_k$$

from $(N_1 \cup ... \cup N_{k-1})\backslash S_k$. Particularly $S_1 = \varnothing$ and $R_1 = N_1$. Any node N_j containing S_k with $j < k$ is called a *parent* of N_k in the tree; similarly the node N_k is called a *child* of N_j. Hence N_1 is the root of the junction tree and N_q is a leaf node in this tree. Lemma 1 below characterizes some useful properties of the set of residuals.

Lemma 1. Let N be a a set of complete subgraphs of a triangulated graph with nodes ordered such that the RIP holds, and let T be its junction tree. Denote $Ch(N_i)$ the set of indices of the children of the node N_i, $T(N_i)$ the set of indices of the nodes belonging to the subtree rooted at the node N_i. Let $De(N_i)$ be the set of indices of the descendants of the node N_i, i.e. $De(N_i) = T(N_i)\backslash\{i\}$ Then

a) if $\cup De(N_i)$ stands for the set theoretical union of all the subsets being descendants of the node N_i, then $\cup\{R_j | j \in De(N_i)\} = \cup De(N_i)\backslash N_i$,
b) the residuals $R_2,...,R_q$ form a partition of X/N_1.

[1] Precise description of all, used here, notions concerning hypergraphs can be found e.g. in [1]

Proof: (a) We prove this identity by induction. It trivially holds if N_i is a leaf in the junction tree. Suppose it holds for any $i > 1$, and let N_p be the parent of N_i. Then

$$\cup\{R_w | w \in De(N_p)\} = \bigcup_{k \in Ch(N_p)} [R_k \cup (\cup De(N_k) \backslash N_k)]$$

Consider a single child, say N_k of N_p. Since $R_k \subseteq N_k$ then $R_k \cup (\cup De(N_k)\backslash N_k) = \cup De(N_k)\backslash S_k = \cup De(N_k)\backslash(N_k \cap N_p) = \cup De(N_k)\backslash N_p$. By the identity $A_1\backslash B \cup ... \cup A_n\backslash B = (A_1 \cup ... \cup A_n)\backslash B$ we obtain the thesis.

(b) To prove this statement we must show that: (i) the residuals are mutually exclusive, and (ii) their set theoretical union equals X/N_1. Part (ii) is just the case (a) when i=1, i.e. when N_i is the root of the junction tree. Hence we must prove only part (i). Let A^c be the complement of A in X. Then $R_i = N_i\backslash S_i = N_i \cap (N_i \cap (N_1 \cup ... \cup N_{i-1}))^c = N_i \cap (N_1^c \cap ... \cap N_{i-1}^c)$. Now, let R_j, $j > i$, be another separator. Then $R_i \cap R_j = (N_1^c \cap ... \cap N_{i-1}^c \cap N_i) \cap (N_1^c \cap ... \cap N_i^c \cap ... \cap N_{j-1}^c \cap N_j) = \varnothing$.

Graham test allows verifying if a graph G is a triangulated one. Consider its set of maximal completely connected subgraphs. H. Graham test consists of two, performed in any order, rules: (i) if a variable X belongs to exactly one H, remove it from X, i.e. $H \leftarrow H - \{X\}$, and (ii) if H_1, H_2 are two hypernodes such that $H_1 \subset H_2$, and $H_1 \neq H_2$ then delete H_1 from H. If any of two rules cannot be applied further, and H = $\varnothing$, it means that G is triangulated.

Example 1. Consider the set H consisting of the sets of variables (nodes) $H_1 = \{X_1, X_3, X_5\}$, $H_2 = \{X_1, X_2\}$, and $H_3 = \{X_2, X_4, X_5\}$. Hence X = $\{X_1, X_2, X_3, X_4, X_5\}$ Variable X_3 belongs to H_1 only and the variable X_4 belongs to H_3 only. Hence, applying twice the rule (i) we obtain H = $\{\{X_1, X_5\}, \{X_1, X_2\}, \{X_2, X_5\}\}$ and we can do nothing more. Observe, however that adding the element $H' = \{X_1, X_2, X_5\}$ to H we make the graph triangulated. Indeed, both the modified, by rule (i), H_1 and H_3 as well as H_2 are subsets of H', and - by rule (ii) - they can be deleted from H which now takes the form H = $\{\{X_1, X_2, X_5\}\}$. Adding some new sets of nodes to H is a general method of transforming a non-triangulated graph to a triangulated one. □

Note that completely connected subgraphs are also called cliques. This explains also another name of the junction tree: *tree of cliques*.

Example 2. Let us illustrate the notions of separator, residual, and tree of cliques. Suppose the modified set of sets of nodes N consists of $N_1 = \{X_1, X_2\}$, $N_2 = \{X_1, X_2, X_5\}$, $N_3 = \{X_2, X_4, X_5\}$, $N_4 = \{X_1, X_3, X_5\}$. It is easy to check that such ordered sets admit the RIP. Separators, residuals and potential parents are given in the table below. □

Node	Separator	Residual	Potential parent
$N_1=\{X_1,X_2\}$	$S_1=\varnothing$	$R_1=\{X_1,X_2\}$	–
$N_2=\{X_1,X_2,X_5\}$	$S_2=\{X_1,X_2\}$	$R_2=\{X_5\}$	N_1
$N_3=\{X_2,X_4,X_5\}$	$S_3=\{X_2,X_5\}$	$R_3=\{X_4\}$	N_2
$N_4=\{X_1,X_3,X_5\}$	$S_4=\{X_1,X_5\}$	$R_4=\{X_3\}$	N_2

3.3. The informal algorithm specification

Let us exploit the good deletion properties of a hyper-tree for the construction of enumeration. This would be our enumeration protocol. Let us first describe the algorithm informally.

Each node should be characterized by a state (green, blue, red), the blue link to a distinguished neighbor (present or absent), the sum of claims to it, and the assigned enumeration value as well as the assigned range of values. The Figure 2 illustrates the graphical denotation used in the sequel.

The initial state is green, no distinguished neighbor and the sum of claims equal zero, and there is no assigned enumeration value (Figure 3).

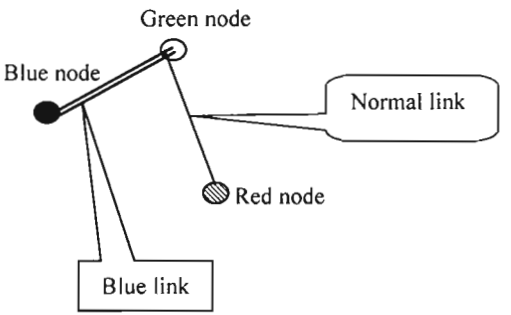


Fig. 2. The denotation

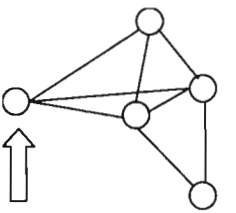


Fig. 3. The initial state

First a green node asks if all its neighbors in state "green" are interconnected.

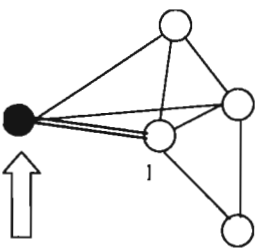


Fig. 4. One node turned blue

If so, then it changes its state to blue and sends a message to one of the green neighbors that it requests its own sum of claims plus one and it remembers this neighbor as its "when-blue neighbor".

The when-blue neighbor receives the message and increases its sum of claims by the received number.

This procedure is continued as long as there are green nodes that have more than zero green neighbors.

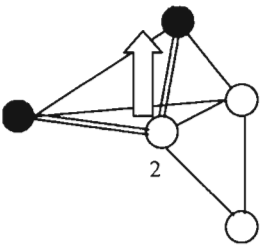


Fig. 5. The second node turned blue

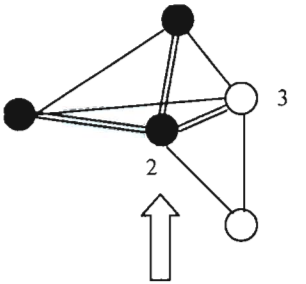


Fig. 6. Now a node turns blue that could't so before some of its neighbors turned blue

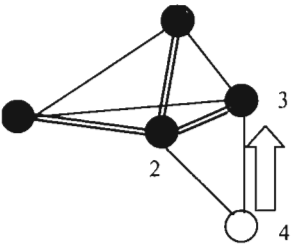


Fig. 7. Now the forth node turns blue

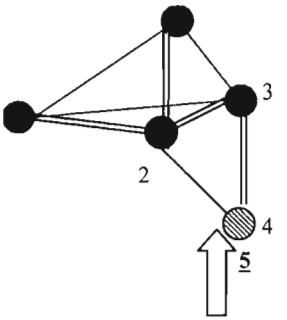


Fig. 8. The remaining green node turns red

Now the remaining (guaranteed only one) green node has no green neighbors.

Then it assigns itself enumeration value equal to its sum of claims plus one and changes its state to red.

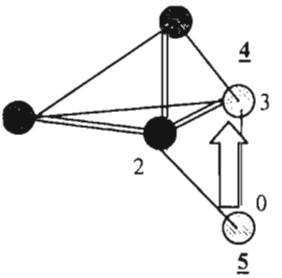


Fig. 9. The blue-link neighbor turns red

Now the neighbor of red along the blue line requests over the blue link the sum of its claims plus one. The receiver node replies by sending back its actual sum of claims, thereafter it diminishes the sum of claims by the number actually received. The sender receives the number, sets it as its enumeration value and sets its sum of claims to the newly received number minus one. It changes the color to red. The procedure continues until all nodes turn red.

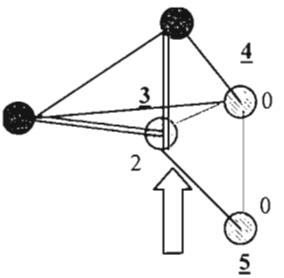


Fig. 9. Another blue-link neighbor of a red node turns red

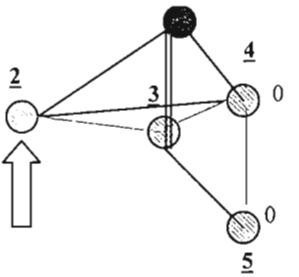


Fig. 9. Another blue-link neighbor of a red node turns red

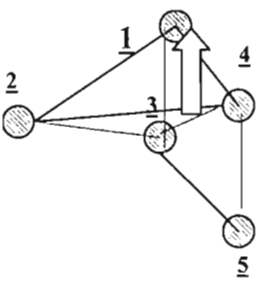


Fig. 10. Another blue-link neighbor of a red node turns red

3.4. The formal protocol specification

In this subsection we want to specify the protocol formally. The protocol tells not how the whole network behaves, but rather how a local node behaves. Let in this protocol the node speak of itself as "me".

Each node can send/receive the following messages:

- Request_message – the count of nodes in the subtree
- Rejection-message – refusing to accept the request-message
- Accepting-message – accepting the request message
- Range-request-message – inquiring about the assigned range of numbers in the enumeration; this message cannot be rejected
- Range-assigment-message – the actual assignment of enumeration range

The request-message and a range-request-message are targeted at some particular node. The other ones are replies back to the sending node.
Each node has local variables as described in step 0 of the protocol.

The Protocol

```
0. Let me initialize: my_state:=green, my_valuation:= unknown(0),
   my_blue_link:=unknown, my_request_count:=0, my_range_upper_bound:=-1,
   my_range_lower_bound:=0
1. If my_state=red and my_range_upper_bound < my_range_lower_bound then
   STOP
2. While my_state=red and incoming range_request_message queue is not
   empty repeat: x:= pop_the_range_request_message_queue(),
   send_range_assigment_message_back(my_range_upper_bound),
   my_range_upper_bound:= my_range_upper_bound-x
3. While my_state=green and incoming request_message queue is not empty
   repeat: my_request_count:= my_request_count+pop_the_message_queue(),
   send_accepting_message_back(pop_the_message_queue())
4. While my_state is not green and incoming request_message queue is not
   empty repeat: send_rejection_message_back(pop_the_message_queue())
5. If my_state=green and all my green neighbors are interconnected then
   my_blue_link:=any of the green neighbors,
   send_request_message(my_blue_link,my_request_count+1),
   wait_till_accepting_or_rejection_message
6. If rejection_message received then my_blue_link:=unknown, goto step 1
7. If my_state=green then my_state:=blue
8. If my_state=blue and my_blue_link.my_state=red then
   send_range_request_message(my_blue_link, my_request_count+1), wa-
   it_till_range_assignment_message, my_valuation:=pop_range_message(),
```

```
   my_range_upper_bound:= my_valuation - 1, my_range_lower_bound:=
   my_valuation - my_request_count, my_state=red
9. Go to step 1
```

4. Final remarks

Mazurkiewicz [10] studies the problem of "fairness" of the assignment of the enumeration values. He proves that under his settings the enumeration is fair that is all permissible configurations are actually possible.

This cannot be claimed about the algorithm of our invention. Our protocols use at a node only limited knowledge about the whole network: some cumulative statistics. In particular, the value assignments are restricted by permissible node removals in a trangularized network.

Mazurkiewicz investigates also the question of termination of the network protocol. He shows that the network can identify whether or not the protocol may successfully terminate (that is whether or not the network matches the conditions for network structure). Our protocol, running on a network with presumably simpler structure, can only determine whether or not it can be completed successfully by setting a time-out condition. Determining whether or not the graph has a hypertree is linear in the number of nodes in the graph. Setting a reasonable high number of potential nodes in the network will do the job.

Finally, the most important thing we gain from the VBS approach (and just from the assumptions about the network structure) is that for the enumeration task we need only space linear in the number of neighbors and execution time linear in the number of variables. If you think of real applications of network protocols like those for the Internet, with a large and unpredictable number of nodes, then it may be of value to construct the network spending the money for a structure suitable for triangulation with potential revenue of linear time local network protocols.

Notice that in the past algorithms for networks with special structure have been studied. Mazurkiewicz [21] investigates planar decomposable graphs. The algorithm developed there may be adjusted to serve as an enumeration protocol. Our approach is more general because it can be easily shown that networks with triangulated structure are in general non-planar.

Recently Dembinski [22] proposed an enumeration protocol based on random number generation. Though applicable to general types of networks, the protocol has other disadvantages compared to our model. The node identities stem from a set with a much larger cardinality than set than the set of nodes. Furthermore each node has to store the identities of all nodes in the network that is its memory requirements depend on global settings. In our approach the node identities stem

from a set with cardinality equal to the cardinality of the set of nodes and the memory requirements depend only on local settings (the number of direct neighbors).

References

[1] Beeri, C., *et.al.* (1983) On the desirability of acyclic database schemes. *J. ACM*, 30, 1983, 479-513.

[2] Bertele, U., and Brioschi, F., *Nonserial Dynamic Programming*, Academic Press, New York, 1972.

[3] Cooper G. The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.* 42, 1990, 395-405.

[4] Dechter, R. Bucket elimination: A unifying framework for probabilistic inference algorithms. In Uncertainty in AI (*UAI-96*), pp. 211-219.

[5] Henrion M. An introduction to algorithm for inference in belief nets, *Uncertainty in Artificial Intelligence* 5 (1990).

[6] Jensen, F. V. *An Introduction to Bayesian Networks*. University College Press, London, 1996.

[7] Lauritzen, S. L., and Spiegelhalter, D.J. Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Statist. Soc.* B-50 (1988) 157-224.

[8] Mazurkiewicz A.: Solvability of the asynchronous ranking problem, *Inform. Process. Lett.* 28 (1988), 221-224.

[9] Mazurkiewicz A.: Distributed enumeration. *Inform.Process.Lett.* 61 (1997) 233-239.

[10] Pal, N., Bezdek, J., and Hemanisha, R. Uncertainty measures for evidential reasoning I: a review. *Int. J.Approx. Reas.* 7 (1992) 162-183.

[11] Pearl, J. Fusion, propagation and structuring in Bayesian networks. *Artif. Intel.* 28 (1986), 241-288.

[12] Pearl, J. *Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, 1988.

[13] Shafer, G. *Probabilistic Expert Systems*, SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, 1996, vol. 67.

[14] Shenoy, P.P. A valuation-based language for expert systems. *Int. J. Approx. Reas.*, 3 (1989) 383-411.

[15] Shenoy, P.P. Conditional independence in valuation-based systems. *Int. J. Approx. Reas.*, 10 (1993) 203-234.

[16] Shimony, S. E., and Charniak, E. A new algorithm for finding MAP assignments to belief networks. *Proc. Conference on Uncertainty in AI*, Cambridge, MA, 1990, 98-103.

[17] Wierzchoń, S. T. Constraint propagation over restricted space of configurations, and its use in optimization. In: R.R. Yager, J. Kacprzyk and M. Fedrizzi (eds.) *Advances in the Dempster-Shafer Theory of Evidence*, J. Wiley, New York, 1994, 375-394.

[18] Wierzchoń, S. T., Kłopotek, M.A., Michalewicz, M. Reasoning and facts explanation in valuation based systems, *Fundamenta Informaticae*, 30 (1997) 359-371.

[19] Wu, T. A problem decomposition method for efficient diagnosis and interpretation of multiple disorders, *Proc. 114th. Symp.Computer Appl. in Medical Care*, 1990, 86-92.

[20] Yen J. Generalizing Dempster-Shafer theory to fuzzy sets. *IEEE Trans. Syst.Man.Cybern.* 20 (3), 1990, 559-570.

[21] A. W. Mazurkiewicz: Distributed Disassembly of Mosaics. *Information Processing Letters*, Volume 46, Number 1, 29 April 1993, 173-178.

[22] P. Dembiński, Distributed and Randomized Enumeration, *Proc. of the 8th Euromicro Workshop on Parallel and Distributed Processing* (PDP'2000), IEEE Society Press, 2000.

ROZPROSZONY PROTOKÓŁ ENUMERACJI DLA SYSTEMÓW Z WARTOŚCIOWANIAMI

Streszczenie: W pracy przedstawiono nowy algorytm enumeracji węzłów sieci. W odróżnieniu od dotychczasowych algorytmów jest on lokalny zarówno w sensie dostępu do informacji (uwzględnia się wyłącznie informacje pochodzące od sąsiadów aktualnie przetwarzanego węzła) jak i przechowywania informacji (ilość informacji jest proporcjonalna do liczby sąsiadów danego węzła). Cechę lokalności uzyskano zawężając rozważania do rodziny grafów triangulowanych, które odgrywają podstawową rolę w teorii sieci bayesowskich. Uogólnieniem tych ostatnich są systemy z wartościowaniami, nazywane też grafowymi systemami ekspertowymi, czyli struktury grafowe służące do reprezentacji niedeterministycznych zależności między zmiennymi (odpowiadają im węzły grafu).

Słowa kluczowe: enumeracja, obliczenia lokalne, grafy triangulowane, drzewa złączeń

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Katarzyna Kościuk[1], Jarosław Stepaniuk[2]

ROZSTRZYGANIE KONFLIKTÓW MIĘDZY REGUŁAMI

Streszczenie: W pracy przedstawiono metody klasyfikacji obiektów oparte na teorii zbiorów przybliżonych i sztucznych sieci neuronowych. Omówiono wyniki eksperymentów dotyczących klasyfikatorów zespołowych używających reguł decyzyjnych i sieci neuronowej do rozstrzygania konfliktów między nimi.

Słowa kluczowe: zbiory przybliżone, sieci neuronowe, klasyfikacja, reguły decyzyjne

1. Wstęp

Powiększający się rozmiar i poziom złożoności informacji powoduje rosnące zapotrzebowanie na systemy przetwarzania danych, zdolne do klasyfikacji prezentowanych im obiektów. Najważniejsze stało się konstruowanie systemów, które działałyby efektywnie wtedy, kiedy człowiek nie będzie w stanie samodzielnie podjąć decyzji. Wpływać na to może rozmiar i stopień skomplikowania zadania lub wymagania dotyczące szybkości i precyzji otrzymywanego rezultatu. Doprowadziło to do powstania i rozwoju wielu dziedzin badań dotyczących systemów wspomagających klasyfikacje: sztucznych sieci neuronowych, systemów maszynowego uczenia [3], nowoczesnych metod uczenia statystycznego, a także niektórych z metod stosowanych w odkrywaniu wiedzy, eksploracji danych [2] i zbiorów przybliżonych [6, 7, 5]. W badaniach tych można wyróżnić pewne wspólne elementy, takie jak: korzystanie ze zbioru dostępnych przykładów w celu stworzenia lub nauczenia się modelu działania, dążenie do odwzorowania zależności występujących w dostępnym fragmencie przestrzeni danych (w postaci przykładów), uzyskanie modelu stosowalnego do jak największego obszaru przestrzeni, także do nowo pojawiających się przypadków, uodpornienie modelu na możliwe zakłócenia

[1] Wydział Zarządzania, Politechnika Białostocka, ul. Ojca Stefana Tarasiuka 2, 16-001 Kleosin – Białystok
[2] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok

w postaci zaburzonych lub niepasujących przykładów (mogących powstać w wyniku błędu pomiarowego).

Cele konstruowania klasyfikatorów są zbieżne, ale metody ich osiągania najczęściej zasadniczo się różnią, czego skutkiem są różnice występujące między rezultatami uzyskiwanymi dla tych samych przykładów. Osiągnięcie poziomu jakości, czytelności lub elastyczności rozwiązania właściwego jednemu z podejść wiąże się z bardzo dużym wzrostem kosztów obliczeniowych przy użyciu innych metod. Postępujące zrozumienie zjawisk w działaniu różnego rodzaju klasyfikatorów pozwoliło wyróżnić wiele współzależności występujących pomiędzy różnymi metodami. Zaowocowało to rozważaniami nad łączeniem elementów różnych podejść w systemy, zwane hybrydowymi i zespołowymi. Wykorzystują one różne metody w celu stworzenia klasyfikatora, przejmując najlepsze cechy systemów wchodzących w jego skład i przynajmniej w części uwalniają konstrukcję hybrydową od ich ograniczeń.

Elementem systemów hybrydowych są często sztuczne sieci neuronowe. Ich zalety to duże potencjalne możliwości aproksymacyjne, zdolność adaptacyjna i odporność na zakłócenia. Podstawową wadą sieci jest problem wyboru struktury dla zadanego problemu jak również trudności przy wyjaśnieniu przyczyn podjęcia takiej czy innej decyzji. Od tych niedostatków można się uwolnić korzystając przy konstrukcji sieci z innych metod, na przykład: drzew decyzyjnych, zbiorów przybliżonych, zbiorów rozmytych.

Pokrewnym rodzajem systemów są klasyfikatory zespołowe, w których informacja z różnych systemów trafia na wejście modułu odpowiedzialnego za podjęcie ostatecznej decyzji. Jednym z podejść jest wykorzystywanie różnego rodzaju metod do ostatecznej klasyfikacji na podstawie odpowiedzi uzyskanych za pomocą reguł decyzyjnych. Przy takiej metodzie różne reguły pasujące do konkretnego przypadku mogą dawać sprzeczne odpowiedzi lub moża też nie znaleźć reguły pasującej do nowego obiektu. Zjawisko to nosi nazwę konfliktu lub częściowego pokrycia. Aby rozwiązać taką sytuację, wykorzystuje się klasyfikator wielopoziomowy. Odpowiedzi udzielane przez reguły stanowią wtedy wejścia do następnego poziomu klasyfikatora, przykładowo do sieci neuronowej.

2. Klasyfikacja za pomocą reguł decyzyjnych

System klasyfikacyjny oparty na regułach decyzyjnych jest jednym z najczęściej używanych systemów do klasyfikacji określonych obiektów na podstawie symbolicznej reprezentacji wiedzy. W tym rozdziale przedstawimy sposoby klasyfikacji przy użyciu reguł decyzyjnych korzystające z miar jakości reguł oraz meto-

dy głosowania reguł. Na zakończenie rozdziału opisano pojęcie konfliktu mogącego powstać między regułami przy klasyfikacji nowych obiektów.

2.1. Miary jakości reguł

W literaturze (np. [1, 12]) można znaleźć różne metody wyznaczania miar jakości reguł, które wywodzą się często z rozważań intuicyjnych popartych eksperymentami.

Niech $DT = (U, A \cup \{d\})$ będzie tablicą danych, taką że U jest zbiorem obiektów, A jest zbiorem atrybutów warunkowych, natomiast d jest atrybutem decyzyjnym.

W przedstawionych poniżej miarach [11, 12] wykorzystano następujące współczynniki reguł:

- dokładność reguły r (ang. accuracy): $\mu_{DT}(r) = \frac{card(Supp_{DT}(r))}{card(Match_{DT}(r))}$ (1)

- pokrycie reguły r (ang. coverage): $\nu_{DT}(r) = \frac{card(Supp_{DT}(r))}{card(B_i)}$ (2)

gdzie $Supp_{DT}(r)$ jest zbiorem obiektów spełniających część *if* i część *then* reguły r, $Match_{DT}(r)$ jest zbiorem obiektów spełniających część *if* reguły r, natomiast $B_i = \{x \in U : d(x) = v_i\}$.

Miara Michalskiego: $q_M = \lambda\mu_{DT}(r) + (1-\lambda)\nu_{DT}(r)$ (3)

Współczynnik $\lambda \in [0,1]$, zwany często poziomem dokładności, jest wyznaczany eksperymentalnie.

Miara Torgo: $q_T(r) = \lambda_T\mu_{DT}(r) + (1-\lambda_T)\nu_{DT}(r)$ gdzie $\lambda_T = \frac{1}{2} + \frac{1}{4}\mu_{DT}(r)$ (4)

Miara Brazdila: $q_B(r) = \mu_{DT}(r) \cdot e^{\nu_{DT}(r)-1}$ (5)

2.2. Metoda standardowego głosowania

Metoda głosowania reguł (ang. Standard Voting) wykorzystuje podzbiory zbioru reguł w celu zaklasyfikowania obiektu. Dla każdego zbioru reguł wskazującego na daną wartość decyzji przypisujemy wagę, to znaczy liczbę głosów. Największa liczba głosów wskazuje na wybraną decyzję.

Niech $B(DT)$ będzie zbiorem reguł tablicy decyzyjnej $DT = (U, A \cup \{d\})$, a S oznacza zbiór wszystkich możliwych obiektów $U \subset S$. Wagą rodziny reguł $B \subset B_i$ $(i = 1,...,n_d)$ względem obiektu $u \in S$ nazywamy funkcję

$W : 2^{B_i} \times S \rightarrow [0,1]$ taką, że $W(B,u) = 0$, gdy wartości atrybutów dla obiektu u nie spełniają części if żadnej z reguł w podzbiorze B.

W systemie Rosetta [8] zaimplementowano poniżej prezentowane wagi zbioru reguł, które zostały wykorzystane do klasyfikacji obiektów i porównania wyników z klasyfikatorem zespołowym.

- **Waga Support/Firing** wykorzystująca wsparcie reguły, czyli głos każdej reguły należącej do zbioru reguł danej decyzji B jest liczbą obiektów popierającą daną regułę.

$$W_{Support/Firing}(B,u) = \frac{\sum_{r \in Match(B,u)} card(Supp_{DT}(r))}{\sum_{r \in Match(B(DT),u)} card(Supp_{DT}(r))} \tag{11}$$

Powyższa waga obrazuje procent głosów jakie otrzymała każda klasa decyzyjna względem wszystkich reguł rozpoznających obiekt u.

Istnieje również waga **Support/All,** która przedstawia stosunek głosów danej klasy decyzyjnej względem wszystkich reguł w bazie $B(DT)$. Wówczas mianownik powyższej wagi przyjmuje postać:

$$\sum_{r \in B(DT)} card(Supp_{DT}(r)) \tag{12}$$

- **Waga Simple/Firing,** która przyjmuje, że głos każdej reguły należącej do zbioru rodziny reguł B jest jednakowy i wynosi jeden.

$$W_{Simple/Firing}(B,u) = \frac{card(Match(B,u))}{card(Match(B(DT),u))} \tag{13}$$

Ta waga obrazuje liczbę reguł z B do liczby reguł rozpoznających dany obiekt.

Podobnie jak w poprzedniej wadze, istnieje podobna waga **Simple/All,** reprezentująca liczbę reguł z B do liczby reguł w całej bazie reguł. Mianownik takiej wagi wygląda następująco: $card(B(DT))$

Przykład. Zbiór $B(DT)$ składa się z reguł zamieszczonych poniżej, zakładamy, że reguły $r1$, $r2$, $r3$ rozpoznają obiekt u.

Tabela 1

Przykładowy zbiór reguł

reguła	przesłanka	decyzja	card(supp(r))
r1	t(1) AND p(m)	d(1)	5
r2	t(1) AND w(0)	d(0)	3
r3	t(1) AND w(1)	d(1)	4
r4	t(2) AND p(z)	d(1)	2
r5	t(2) AND p(m)	d(0)	6

$Match(B(d=0),u)=\{r2\}$, $Match(B(d=1),u)=\{r1,r3\}$

Jak widać, wsparcie dla reguł rozpoznających obiekt u wynosi 12, natomiast wsparcie wszystkich reguł wynosi 20.

Poniżej zostały obliczone wagi dla poszczególnych zbiorów reguł:

$$W_{Support/Firing}(B(d=0),u)=\frac{3}{12}=0.25,\quad W_{Support/Firing}(B(d=1),u)=\frac{9}{12}=0.75$$

$$W_{Support/All}(B(d=0),u)=\frac{3}{20}=0.15,\quad W_{Support/All}(B(d=1),u)=\frac{9}{20}=0.45$$

$$W_{Simple/Firing}(B(d=0),u)=\frac{1}{3}\approx 0.33,\quad W_{Simple/Firing}(B(d=1),u)=\frac{2}{3}\approx 0.66$$

$$W_{Simple/All}(B(d=0),u)=\frac{1}{5}=0.2,\quad W_{Simple/All}(B(d=1),u)=\frac{2}{5}=0.1$$

Istnieje możliwość wykorzystania miar jakości reguł przytoczonych w poprzednim rozdziale do obliczania wag zbioru reguł. Dla danej miary jakości q możemy zdefiniować następującą wagę:

$$W_q(B,u)=\frac{\sum_{r\in Match(B,u)} q(r)}{\sum_{r\in Match(B(DT),u)} q(r)} \tag{14}$$

Przytoczone wyżej wzory na obliczanie wartości wag dla zbioru reguł opierają się na wyznaczaniu współczynnika dokładności i pokrycia, które intuicyjnie są powiązane z prawdopodobieństwem. Ponadto należy zauważyć, że wartość wag (liczba głosów) wynosi zero dla zbioru reguł B, w którym obiekt $u\in S$ nie wspiera żadnej z reguł w B.

2.3. Pojęcie konfliktu między regułami

Jeśli mamy wygenerowane reguły decyzyjne, proces klasyfikacji pojedynczego obiektu polega na znalezieniu reguł, których poprzednik (część *if*) jest spełniony dla rozpatrywanego obiektu. Zbiór takich reguł może, niestety, posiadać różne następniki. Opisana sytuacja nosi nazwę konfliktu między regułami. Istnieją metody klasyfikacji obiektów rozstrzygające konflikty między regułami na podstawie różnych strategii, na przykład przedstawione wcześniej wyznaczanie miar jakości reguł (brana jest tylko jedna reguła o najwyższej jakości) lub też głosowanie, czyli wyznaczanie wag dla zbioru reguł (wygrywa ta decyzja, która ma najwięcej głosów). Jedną z metod rozstrzygania konfliktów jest klasyfikacja przy użyciu sieci neuronowej. Taka metoda stała się przedmiotem badań w dalszej części pracy. Poniżej zostanie przedstawione szczegółowo pojęcie konfliktu między regułami [11].

Niech $B(DT) \subset RUL(DT)$ oznacza bazę reguł dla tablicy decyzyjnej $DT = (U, A \cup \{d\})$ oraz $r_1,...,r_n$ są kolejnymi regułami z bazy, gdzie $n = card(B(DT))$. Reguła r_i ma następującą postać $\phi_i \Rightarrow \psi_i$.

Rozważa się dwie rodziny podzbiorów $B(DT)$

- ze względu na wartość decyzji: $B_i = \{r_i \in B(DT) : \psi_j \equiv (d = v_i)\}$ gdzie $v_i \in V_d$,
- podzbiór reguł względem obiektów: $B_i(u) = B_i \cap Match(B(DT),u)$ dla $u \in U$

Konflikt reguł w bazie $B(DT)$ określa się następująco:

$$\bigcup_{i \neq j} (Mach_{DT}(B_i) \cap Mach_{DT}(B_j)) \neq \emptyset \tag{15}$$

Jeśli zbiór wszystkich obiektów z DT rozpoznawalnych przez zbiór reguł B_i mających decyzję $d = v_i$ przecina się ze zbiorem obiektów rozpoznawalnych przez zbiór reguł $B_j(d = v_j)$, oznacza to, że istnieją przynajmniej dwie reguły klasyfikujące dany obiekt do różnych klas.

Dla pojedynczego obiektu u konflikt zachodzi gdy:

$$B_i(u) \neq \emptyset \wedge B_j(u) \neq \emptyset \text{ dla } i \neq j \tag{16}$$

Oznacza to, że reguły spełniające wartość przesłanki reguły dla obiektu u przyjmują różne decyzje $d = v_i$ i $d = v_j$.

Przykład. Tablica decyzyjna (DT_C) opisuje dziesięciu przykładowych pacjentów chorych na cukrzycę.

Tabela 2

Przykładowe dane dzieci chorych na cukrzycę

nr	płeć	wiek	czas	występowanie	rodzaj	infekcje	mikroalbuminuria
u1	z	12	5	nie	KIT_IIT	tak	tak
u2	m	1	4	nie	KIT	tak	nie
u3	m	15	5	tak	KIT	tak	nie
u4	z	13	4	nie	KIT_IIT	tak	tak
u5	z	11	5	tak	KIT	tak	tak
u6	m	11	5	nie	KIT	nie	tak
u7	z	17	3	tak	KIT_IIT	nie	nie
u8	z	8	4	nie	KIT	tak	nie
u9	m	9	6	nie	KIT	nie	tak
u10	z	12	4	nie	KIT	tak	tak

Po dyskretyzacji i przefiltrowaniu otrzymano następujący zbiór reguł $B(DT_C)$:

Tabela 3

Przykładowe reguły

nr	postać reguły
r1	wiek([9, 14)) => mikroalbuminuria(tak)
r2	wiek([*, 9)) => mikroalbuminuria(nie)
r3	wiek([14, *)) => mikroalbuminuria(nie)
r4	płeć(z) AND infekcje(tak) => mikroalbuminuria(tak)
r5	płeć(m) AND infekcje(tak) => mikroalbuminuria(nie)
r6	płeć(z) AND występ(nie) AND infekcje(tak) => mikroalbuminuria(tak)
r7	płeć(z) AND występ(nie) AND infekcje(tak) => mikroalbuminuria(nie)
r8	płeć(m) AND występ(tak) AND rodzaj(KIT) => mikroalbuminuria(nie)

W regule $r2$: Wiek([*,9)) oznacza wiek dziecka mniejszy niż 9 lat, zaś w regule $r4$: Wiek([14,*)) oznacza wiek nie mniejszy niż 14 lat.

Używając poprzednich oznaczeń, podzielono powyższy zbiór reguł na podzbiory ze względu na decyzję $v_1 = tak$, $v_2 = nie$: $B_1 = \{r1, r4, r6\}$, $B_2 = \{r2, r3, r5, r7, r8\}$

Podział reguł ze względu na rozpoznawalne obiekty:

$B_1(u1) = \{r1, r4, r6\}$

$B_2(u2) = \{r2, r5\}$

$B_2(u3) = \{r3, r5, r8\}$

$B_1(u4) = \{r1, r4, r6\}$

$B_1(u5) = \{r1, r4\}$

$B_1(u6) = \{r1\}$

$B_2(u7) = \{r3, r8\}$

$B_2(u8) = \{r2, r7\}$

$B_1(u9) = \{r1\}$

$B_1(u10) = \{r1, r4, r6\}$.

Wykorzystując wzór (15) na konflikt reguł w bazie $B(DT_C)$, mamy jeden składnik dla $i = 1$ oraz $j = 2$. Poniżej wyznaczono zbiory obiektów rozpoznawalnych przez reguły mające jednakowe konkluzje.

$Match_{DT_c}(B_1) = Match_{DT_c}(\{r1, r4, r6\}) = \{u1, u4, u5, u6, u8, u9, u10\}$

$Match_{DT_c}(B_2) = Match_{DT_c}(\{r2, r3, r5, r7, r8\}) = \{u2, u3, u4, u7, u8, u10\}$

Jak widać, iloczyn obu zbiorów jest niepusty, co dowodzi istnienia konfliktu w bazie reguł dla obiektów $u4$, $u8$ i $u10$:

$Match_{DT_c}(B_1) \cap Match_{DT_c}(B_2) = \{u4, u8, u10\}$

Odnosząc się również do wzoru (16) na konflikt pojedynczego obiektu widzimy, że tylko dla obiektu $u4$, $u8$ i $u10$ zbiory $B_1(u)$ i $B_2(u)$ są w obu przypadkach niepuste, co potwierdza istnienie konfliktu.

3. Rozstrzyganie konfliktów między regułami za pomocą sieci neuronowej

Pomysł zastosowania sieci neuronowej do rozstrzygnięcia konfliktów między regułami [11, 12] zrodził się z pojawieniem się problemu wyboru strategii eliminacji konfliktów. Znalezienie najlepszej strategii postępowania w konkretnym przypadku wymaga kosztownych eksperymentów związanych z doborem metody rozstrzygania konfliktów. Na przykład w procedurze doboru wag (głosowanie) trzeba wybrać z „góry" najlepszą metodę obliczania wag dla zbioru. W przypadku sieci neuronowej oblicza się adaptacyjnie wagi, których wartości zapewniają dobrą klasyfikację nowych obiektów. Ponadto wykorzystując prostą sieć neuronową można dokonać interpretacji układu wag w wyuczonej sieci. Interpretuje się w ten sposób wpływ poszczególnych reguł na wartość podejmowanej decyzji.

System rozstrzygający konflikt między regułami można podzielić na dwie zasadnicze części: transformację tablicy i zastosowanie sieci neuronowej.

3.1. Transformacja tablicy

Załóżmy, że mamy daną tablicę danych $(U, A \cup \{d\})$, taką że U jest zbiorem obiektów, A jest zbiorem atrybutów warunkowych, natomiast d jest atrybutem decyzyjnym. Tablica danych $(U, A \cup \{d\})$ dzielona jest na trzy rozłączne części: $(U_1, A \cup \{d\})$, $(U_2, A \cup \{d\})$, (U_3, A), to znaczy zbiór obiektów $U = U_1 \cup U_2 \cup U_3$ oraz $U_1 \cap U_2 = \emptyset$, $U_1 \cap U_3 = \emptyset$ i $U_2 \cap U_3 = \emptyset$.

Po dyskretyzacji pierwszej tablicy $(U_1, A \cup \{d\})$ generuje się reguły, tworząc zbiór reguł $Rule\_Set$.

Reguły wygenerowane są przez system Rosetta [8]. Mają one następującą postać:

Atrybut1(wartość1) AND Atrybut2(wartość2) AND .... => Decyzja (wartość decyzji)

Wartości atrybutów po dyskretyzacji mogą mieć postać przedziałów.

Pozostałe dwie tablice: $(U_2, A \cup \{d\})$, (U_3, A) także zostają poddane dyskretyzacji za pomocą cięć utworzonych przy dyskretyzacji pierwszej tablicy.

Program ConRes przekształca tablice $(U_2, A \cup \{d\})$, (U_3, A) odpowiednio na nowe tablice $(U_2, A_{Rule\_Set} \cup \{d\})$, (U_3, A_{Rule_Set}).

gdzie :

U – zbiór obiektów (pozostaje bez zmian)

A_{Rule_Set} – zbiór atrybutów odpowiadających wcześniej uzyskanym regułom, a wartości dla poszczególnych obiektów są wyznaczane podczas konwersji

d – decyzja

Dla każdej reguły $R \in Rule\_Set$ postaci $R: if\ \lambda\ then\ \beta$

Wyznacza się atrybut $a_R \in A_{Rule\_Set}$ przyjmujący w zależności od zastosowanej konwersji i rozpatrywanego obiektu różne możliwe wartości.

Konwersja zwykła

W przypadku zbioru uczącego wartości nowych atrybutów a_R dla każdego obiektu u zależeć będą od tego, czy i jak stosuje się do niego dana reguła :

$a_R(u) = 1$ – gdy dany obiekt spełnia część *then* i część *if* reguły,

$a_R(u) = 0$ – gdy obiekt nie spełnia części *if* reguły

$a_R(u) = -1$ – gdy obiekt spełnia tylko część *if*, a nie spełnia części *then*

Reguła zadziała tylko przy całkowitym spełnieniu części *if*.

Dla zbioru testowego wartości atrybutów wyliczane są następująco:
$a_R(u) = 1$ – gdy dany obiekt spełnia część *if* reguły,
$a_R(u) = 0$ – gdy obiekt nie spełnia części *if* reguły

Konwersja częściowa

Postać przesłanki λ reguły R składa się dla większości reguł z kilku warunków zawierających określone zakresy czy wartości danego atrybutu. Można więc regułę R przedstawić następująco: *if* $(\lambda_1$ and $\lambda_2 \ldots \lambda_M)$ *then* β, gdzie M jest liczbą atrybutów występujących w regule R.
Oznaczając dalej :
L – liczba atrybutów przyjmujących taką samą wartość jak atrybuty obiektu (częściowe spełnienie przesłanki).

W zależności od prawidłowego spełnienia części ***then*** przyjmuje się dla zbioru uczącego U_2:

$$\omega(u) = \begin{cases} 1 & \text{gdy } \beta \text{ jest poprawną dezyzją dla } u \\ -1 & \text{w przeciwnym przypadku} \end{cases} \tag{17}$$

W przypadku obiektu u ze zbioru testowego $\omega(u) = 1$ (18)

Wartość atrybutu a_R dla danego obiektu u oblicza się następująco:

$$a_R(u) = \omega(u)\frac{L}{M} \tag{19}$$

Zatem wartości nowych atrybutów dla każdego obiektu i danej reguły będą zależeć od tego, ile atrybutów w części ***if*** reguły przyjmuje taką samą wartość jak atrybuty obiektu.

Przykłady konwersji na podstawie wybranych danych dzieci chorych na cukrzycę dla zbioru uczącego.

Dane są następujące cztery reguły:
r1: *if* Płeć(m) and Infekcje(tak) *then* Mikroalbuminuria(tak)
r2: *if* Wiek([8, 11)) and Remisja(tak) *then* Mikroalbuminuria(nie)
r3: *if* Płeć(z) and Wiek ([*, 6)) and Infekcje(nie) *then* Mikroalbuminuria(tak)
r4: *if* Wiek([13, *)) and Infekcje(tak) and Remisja(nie) *then* Mikroalbuminuria(tak)
oraz tablica danych opisująca pięć obiektów.

Tabela 4

Przykładowe dane

nr obiektu	płeć	wiek	infekcje	remisja	mikroalbuminuria
u1	m	[8, 11)	nie	tak	tak
u2	z	[*, 6)	tak	nie	nie
u3	m	[8, 11)	tak	tak	tak
u4	z	[11, 13)	nie	tak	nie
u5	m	[13, *)	nie	nie	tak

Po konwersji danych powstanie tablica, gdzie obiekty pozostają te same, zaś atrybutami staną się reguły.

Tabela 5

Tablica konwersji zwykłej

nr reguły / nr obiektu	r1	r2	r3	r4
u1	0	-1	0	0
u2	0	0	0	0
u3	1	-1	0	0
u4	0	0	0	0
u5	0	0	0	0

Na przykład konwersja zwykła dla obiektu **u1**:

- część *if* reguły **r1** nie jest spełniona, więc atrybut ma wartość 0;
- część *if* reguły **r2** jest spełniona, lecz decyzja reguły jest „nie", a obiekt należy do klasy decyzyjnej „tak", w związku z tym atrybut ma wartość –1;
- część *if* reguły **r3** nie jest spełniona, więc atrybut ma wartość 0;
- część *if* reguły **r4** nie jest spełniona, więc atrybut ma wartość 0;

Tabela 6

Tablica konwersji częściowej

nr reguły / nr obiektu	r1	r2	r3	r4
u1	0,5	-1	0,33	0
u2	-0,5	0	-0,66	-0,66
u3	1	-1	0	0,33
u4	0	0,5	-0,66	0
u5	0,5	0	0,33	0,66

Konwersja częściowa dla obiektu **u1** wygląda następująco:

- część *if* reguły **r1** jest spełniona w stopniu 0.5: chory jest płci męskiej, ale infekcje nie występują; reguła prawidłowo przypisuje do obiektu klasę decyzyjną „tak" więc atrybut ma wartość 0.5;
- część *if* reguły **r2** jest spełniona całkowicie, lecz decyzja reguły jest „nie", a obiekt należy do klasy decyzyjnej „tak", w związku z tym atrybut ma wartość –1;
- część *if* reguły **r3** jest częściowo spełniona: infekcje nie występują, ale pacjent nie jest płci żeńskiej i jego wiek jest nie mniejszy niż 6; reguła prawidłowo przypisuje do obiektu klasę decyzyjną „tak" więc atrybut ma wartość 0.33;
- część *if* reguły **r4** nie jest spełniona, więc atrybut przyjmuje wartość 0;

Przy konwersji zwykłej w powstałej tablicy są prawie same zera, na przykład reguła **r4** oraz reguła **r3** nie będą brane pod uwagę, gdyż żaden obiekt nie spełnia części *if* tych reguł. Konwersja częściowa jest dokładniejsza. Można zauważyć, że reguły **r4** i **r3** mają także wpływ na obiekty.

3.2. Zastosowanie sieci neuronowej

Rozważana sieć neuronowa będzie miała tylko dwie warstwy: wejściową i wyjściową (bez warstwy ukrytej). W warstwie wejściowej liczba neuronów będzie odpowiadać liczbie reguł +1 (bias). W warstwie wyjściowej będzie tyle neuronów, ile jest klas decyzyjnych.

Wyznaczona tablica $(U_2, A_{Rule\_Set} \cup \{d\})$ określa dane treningowe sieci neuronowej. Do warstwy wejściowej podaje się wartości otrzymane podczas konwersji odpowiednio dla każdego neuronu odpowiadającego każdej regule. Warstwa wejściowa neuronów nie zmienia wartości podawanych na wejściu sieci; wyjścia neuronów będą takie same jak wejścia. W trakcie uczenia ulegają modyfikacjom wagi między warstwą wejściową a wyjściową.

Testowanie wyuczonej sieci neuronowej przeprowadza się korzystając z tablicy (U_3, A_{Rule_Set}). W wyniku uczenia sieci wyznaczane są wagi na połączeniach między neuronami wejściowymi (z których każdy odpowiada kolejnej regule) a wyjściowymi (które odpowiadają za wybór decyzji). Wskazują one na to w jakim stopniu poszczególne reguły wyznaczają dla danego obiektu decyzję.

4. Eksperymenty

Do przeprowadzenia badań eksperymentalnych wytworzono oprogramowanie o nazwie ConRes. W tym rozdziale przedstawione zostały wyniki klasyfikacji wykonanej przez program ConRes na trzech zbiorach obiektów: „Iris", „Cukrzyca"

i „Australian". Przed przystąpieniem do korzystania z programu ConRes w systemie Rosettta [8] w większości przypadków dokonuje się podziału każdej tablicy danych na dwa podzbiory oraz dyskretyzacji algorytmem Boolean reasoning. Następnie z jednego zbioru generowane są reguły. W tym celu wykorzystane zostały metody generacji reduktów: algorytm genetyczny (GeneticReducer), algorytm Johnsona (JohnsonReducer) oraz dynamiczne redukty (RSESDynamicReducer). Ten zbiór również posłużył do uczenia sieci. Drugi zbiór posłużył do sprawdzenia jakości klasyfikatora. Do doświadczeń zastosowane zostały następujące parametry uczenia sieci: maksymalna liczba iteracji: 1000, maksymalny dopuszczalny błąd sieci: 0.6, współczynnik uczenia: 0.01. Dla zbioru uczącego zastosowano opisane wcześniej metody konwersji:

- **konwersja –1, 0, 1** gdzie znak nowego atrybutu w tablicy konwersji zależy od decyzji, dla konwersji zwykłej wartości przyjmowane przez nowe atrybuty wynoszą –1, 0, 1, w przypadku zaś konwersji częściowej wartości te należą do przedziału [-1,1];
- **konwersja 0, 1** taka jak dla zbioru testowego (bez wpływu decyzji), dla konwersji zwykłej nowe atrybuty przyjmują wartości 0, 1, a dla konwersji częściowej są one z przedziału [0,1]

4.1. Zbiór „Iris"

Twórcą zbioru danych o irysach [4] jest R. A. Fisher. Zbiór zawiera 150 elementów, podzielonych na trzy klasy, po 50 obiektów w każdej. Każda klasa odpowiada innemu podgatunkowi kwiatu irysa: Iris Setosa, Iris Versicolor, Iris Virginica. Każdy obiekt ma cztery numeryczne atrybuty: długość działki kielicha i szerokość działki kielicha oraz długość płatka i szerokość płatka kwiatu.

W systemie Rosetta dokonany został podział całej tablicy danych na dwa równoliczne podzbiory (po 75 elementów). Z pierwszego zbioru, po dyskretyzacji, wygenerowane zostały reguły; ten zbiór posłużył również do uczenia sieci neuronowej. Drugi po przeprowadzeniu dyskretyzacji został zbiorem testowym. Metoda dynamicznych reduktów przy zastosowaniu algorytmu Johnsona, po filtracji stosującej miarę jakości Torgo, stworzyła podzbiór ośmiu reguł. Poniżej zamieszczono wyniki klasyfikacji przy podwójnej walidacji krzyżowej.

Tabela 7

Porównanie wyników klasyfikacji zbioru „Iris" dla 8 reguł

				uczenie	testowanie
	metoda		typ konwersji	%	%
Rosetta	Standard Voting		–	88	79
Rosetta + ConRes	przy użyciu reguł konwersja 0, 1	perceptron	zwykła	85	61
			częściowa	55	55
		propagacja wsteczna	zwykła	97	95
			częściowa	98	94
	przy użyciu reguł konwersja -1, 0, 1	perceptron	zwykła	63	65
			częściowa	65	37
		propagacja wsteczna	zwykła	99	95
			częściowa	100	80

Przy propagacji wstecznej jakość klasyfikacji okazała się lepsza niż przy perceptronie. Konwersja zwykła dała w tym przypadku lepsze wyniki niż konwersja częściowa. Może to wynikać stąd, że przy konwersji częściowej sieć neuronowa w zbyt dużym stopniu dopasowuje się do zbioru uczącego i w związku z tym gorzej klasyfikuje zbiór testowy. Jak widać po wynikach, przy konwersji częściowej prawidłowo zaklasyfikowanych obiektów w zbiorze uczącym jest nawet 100%. Oznacza to doskonałe dopasowanie sieci do tego zbioru, czego konsekwencją jest spadek jakości klasyfikacji zbioru testowego. Zastosowanie na zbiorze uczącym konwersji 0, 1 (średnio 76% prawidłowo zaklasyfikowanych obiektów) dało ogólnie lepsze rezultaty niż konwersji –1, 0, 1, (średnio 69% prawidłowo zaklasyfikowanych obiektów). Na zbiorze testowym zawsze stosuje się konwersję 0, 1, więc efektywniejsze uczenie sieci jest wtedy, gdy tą samą metodę konwersji zastosuje się również na zbiorze uczącym. Przy konwersji –1, 0, 1 sieć w większym stopniu dopasowuje się do zbioru uczącego – daje lepsze wyniki klasyfikacji tego zbioru (niż konwersja 0, 1), ale gorzej klasyfikuje zbiór testowy. Dobrą jakość klasyfikacji uzyskano również przy użyciu metody zaimplementowanej w systemie Rosetta – standardowego głosowania. Wynik okazał się lepszy niż dla klasyfikatora zespołowego przy perceptronie.

Interpretacja wag sieci neuronowej

Poniżej zamieszczono przykładowe wagi sieci neuronowej. Uzyskano je przy użyciu tablicy konwersji –1, 0, 1 dla zbioru uczącego i 8 reguł.

Tabela 8

Przykładowe wagi sieci neuronowej zbioru „Iris" dla 8 reguł

perceptron		propagacja wsteczna	
konwersja zwykła	konwersja częściowa	konwersja zwykła	konwersja częściowa
Klasa: Iris-virginica r1) 0,02 r2) -0,0199 r3) 0 r4) 0,02 r5) 0,02 r6) 0,0199 r7) 0 r8) 0	Klasa: Iris-virginica r1) 0,01 r2) -0,01 r3) 0,01 r4) -0,03 r5) 0,0099 r6) 0,0001 r7) 0,01 r8) -0,0199	Klasa: Iris-virginica r1) 0,6327 r2) -5,1332 r3) -3,4274 r4) -0,7615 r5) 1,9142 r6) -1,1252 r7) 1,5818 r8) -2,0092	Klasa: Iris-virginica r1) 1,2496 r2) -5,1572 r3) -1,8831 r4) -1,0177 r5) 0,6312 r6) -1,1074 r7) 0,9428 r8) -3,5559
Klasa: Iris-versicolor r1) 0 r2) 0,02 r3) 0 r4) -0,02 r5) -0,02 r6) 0 r7) 0 r8) 0	Klasa: Iris-versicolor r1) 0,0001 r2) 0,02 r3) 0 r4) -0,01 r5) 0,0001 r6) -0,01 r7) -0,01 r8) 0	Klasa: Iris-versicolor r1) -0,5063 r2) 5,3546 r3) -0,5792 r4) 0,3084 r5) -0,8368 r6) -2,3755 r7) -1,4323 r8) -1,7429	Klasa: Iris-versicolor r1) -1,076 r2) 5,3317 r3) -2,4239 r4) 0,6776 r5) -0,5263 r6) 0,0392 r7) -1,7565 r8) -0,2057
Klasa: Iris-setosa r1) 0 r2) 0 r3) 0,02 r4) 0,02 r5) 0 r6) 0 r7) 0 r8) 0,02	Klasa: Iris-setosa r1) 0 r2) 0,0001 r3) 0 r4) -0,0001 r5) -0,0099 r6) 0,0099 r7) 0,01 r8) 0,04	Klasa: Iris-setosa r1) -1,6176 r2) -1,0209 r3) 2,5733 r4) 2,2802 r5) -0,9308 r6) 0,011 r7) 0,1758 r8) 4,1733	Klasa: Iris-setosa r1) -1,9372 r2) -1,5039 r3) 4,5628 r4) 2,2468 r5) -1,9883 r6) -0,211 r7) -0,0841 r8) 2,0901

Otrzymane wagi sieci neuronowej na połączeniach między warstwą wejściową a wyjściową obrazują wpływ reguł na klasyfikację obiektów. Wartości wag dla perceptronu i konwersji zwykłej przyjmują tylko trzy wartości 0, 0.02 i –0.02. W tym przypadku widać, że nie każda reguła ma wkład w przypisaniu obiektu do danej klasy (zerowe wartości wag). Tylko w jednej klasie waga konkretnej reguły przyjmuje wartość różną od zera. W tablicy konwersji jedynie sporadycznie występują zerowe wiersze. Przy konwersji częściowej tablica wypełniona jest 0, 0.5, 1, -0.5, w zależności od spełnienia (niespełnienia) reguły. Wagi w takim przypadku są bardziej zróżnicowane. Niektóre reguły dla poszczególnych klasy mają margi-

nalne znaczenie. Takimi przypadkami są wagi dla Iris-virginica - r6; Iris-versicolor – r1, r3, r5, r8; Iris-setosa – r1, r2, r3, r4. Przy uczeniu sieci wsteczną propagacją błędu sieć generuje większe wartości wag niż w przypadku perceptronu. Zarówno przy konwersji zwykłej jak i częściowej wagi przyjmują podobne wartości. Można powiedzieć nawet więcej - niektóre z nich są prawie identyczne (tak jak wagi r2). Większość wag także pokrywa się co do znaku w obu metodach konwersji. Dość wyraźnie widać przewagę niektórych reguł w procesie podejmowania decyzji. Są to reguły: r2, r3, r8, którym sieć utworzyła duże wagi dla wszystkich neuronów wyjściowych. Należy zaznaczyć, że wagi otrzymane przy zastosowaniu wstecznej propagacji błędu są inne przy każdym nowym uczeniu sieci, ponieważ wartości początkowe wag są generowane losowo.

4.2. Zbiór „Australian"

Zbiór dotyczy kredytów, wszystkie nazwy atrybutów zostały zmienione [4]. Dane są interesujące ze względu na zestaw typów atrybutów: ciągłych, nominalnych z małym zbiorem wartości oraz nominalnych przyjmujących wiele wartości. Jest tu także kilka brakujących danych – w 37 przypadkach brakuje jednej lub więcej wartości atrybutów. Luki te zostały zastąpione wartością średnią lub modalną danego atrybutu (czyli wartością najczęściej występującą). Liczba obiektów wynosi 690, liczba atrybutów 14 (w tym 6 numerycznych) plus atrybut decyzyjny: 307 obiektów należy do klasy 0, zaś 383 do klasy 1.

W systemie Rosetta dokonany został podział całej tablicy danych na dwa równe podzbiory, po 345 elementów. Z jednego zbioru wygenerowane zostały reguły i posłużył on do uczenia sieci neuronowej. Drugi zbiór został zbiorem testowym. W tym przypadku wykorzystano zbiór reguł generowanych metodą dynamicznych reduktów algorytmem Johnsona. Ten zbiór okazał się także dobry w przypadku metody głosowania. Reguły otrzymano z utworzonych 30 reduktów składających się z kombinacji 5, 6 lub 7 atrybutów. Po filtracji powstały reguły o długości 5 i 6 składników w części *if*.

Poniżej zamieszczono wyniki klasyfikacji przy podwójnej walidacji krzyżowej.

Tabela 9

Porównanie wyników klasyfikacji zbioru „Australian" dla 17 regul.

				uczenie	testowanie
	metoda		typ konwersji	%	%
Rosetta	Standard Voting		–	15	7
Rosetta + ConRes	przy użyciu reguł konwersja 0, 1	perceptron	zwykła	52	51
			częściowa	59	62
		propagacja wsteczna	zwykła	62	59
			częściowa	84	79
	przy użyciu reguł konwersja -1, 0, 1	perceptron	zwykła	52	51
			częściowa	78	52
		propagacja wsteczna	zwykła	62	59
			częściowa	100	76

Wyniki otrzymane przy użyciu klasyfikatora zespołowego są zadowalające. Najefektywniejszą metodą uczenia sieci okazała się znowu wsteczna propagacja błędu. Przy konwersji zwykłej wyniki są gorsze niż przy konwersji częściowej, czyli odwrotnie niż dla zbioru irysów. Może to wynikać z większego podobieństwa obiektów w zbiorze uczącym i testowym w tym przypadku, jak również z tego, że konwersja częściowa lepiej obrazuje powiązanie reguły z obiektem, w związku z tym sieć ma więcej informacji na wejściu. Ułamkowe spełnienie (niespełnienie) reguły jest dużą podpowiedzią dla sieci. Tak jak w przypadku poprzedniego zbioru danych konwersja 0, 1 dała lepsze rezultaty niż konwersja –1, 0, 1. Nie jest to jednak różnica znacząca. Najgorszą metodą okazało się standardowe głosowanie. Spowodowała to zdecydowanie zbyt mała liczba reguł dla tak dużego, 345 elementowego, zbioru danych.

4.3. Zbiór „Cukrzyca"

Zbiór zawiera dane 107 dzieci chorych na cukrzycę insulinozależną [9, 10]. Opisuje je 12 atrybutów, w tym trzy liczbowe, oraz atrybut decyzyjny, przyjmujący dwie wartości (tak, nie). Zbiór 107 elementowy w pierwszym doświadczeniu został podzielony na dwa podzbiory po 54 i 53 elementów. Ze zbioru 54 elementów, po dyskretyzacji, wydobyte zostały reguły. Na tym zbiorze uczono również

sieć neuronową. Do testowania posłużył drugi 53 elementowy zbiór. W przeprowadzonym drugim doświadczeniu podzielono zbiór w następujący sposób: 18 obiektów posłużyło jako zbiór, z którego po dyskretyzacji wygenerowaliśmy reguły; na innych 36 obiektach uczona była sieć neuronową; pozostałe zaś 53 obiekty stały się przedmiotem weryfikacji.

Eksperyment 1 (50% - zbiór uczący, 50% - zbiór testowy)

Najlepszym zbiorem reguł użytych do konstrukcji klasyfikatora zespołowego okazał się zbiór utworzony na podstawie algorytmu genetycznego. Po filtracji wykorzystującej miarę jakości Torgo otrzymano 15 reguł. Poniżej zamieszczono wyniki otrzymane po podwójnej walidacji krzyżowej.

Tabela 10

Porównanie wyników klasyfikacji zbioru „Cukrzyca" przy podziale na dwa zbiory

				uczenie	testowanie
	metoda		typ konwersji	%	%
Rosetta	Standard Voting		–	72	11
Rosetta + ConRes	przy użyciu reguł konwersja 0, 1	perceptron	zwykła	52	53
			częściowa	57	58
		propagacja wsteczna	zwykła	74	53
			częściowa	80	66
	przy użyciu reguł konwersja -1, 0, 1	perceptron	zwykła	52	53
			częściowa	91	60
		propagacja wsteczna	zwykła	74	53
			częściowa	100	53

Najlepsze rezultaty – 66% prawidłowo zaklasyfikowanych obiektów – uzyskano stosując klasyfikator zespołowy przy konwersji częściowej 0, 1 i uczeniu sieci metodą wstecznej propagacji błędu. Znów lepsza okazała się konwersja 0, 1 niż konwersja –1, 0, 1, z jednym wyjątkiem: perceptron osiągnął najlepszy wynik przy konwersji częściowej –1, 0, 1. Dla zbioru „Cukrzyca" jakość klasyfikacji przy konwersji częściowej jest w większości przypadków wyższa niż przy konwersji zwykłej (tylko przy konwersji –1, 0, 1 wsteczna propagacja dała jednakowe wyni-

ki, zarówno dla konwersji zwykłej jak i częściowej). Najgorszym klasyfikatorem okazała się standardowa metoda głosowania dająca zaledwie 11% prawidłowej klasyfikacji na zbiorze testowym (większość obiektów pozostała nierozpoznana). Wynika z tego, że wygenerowane ze zbioru uczącego reguły nie bardzo pasowały jednak do obiektów ze zbioru testowego.

Eksperyment 2 (17% – zbiór do generowania reguł, 33% – zbiór uczący, 50% – zbiór testowy)

Najlepsze wyniki otrzymano dla 5 reguł uzyskanych metodą dynamicznych reduktów algorytmem genetycznym. Poniżej zamieszczono wyniki klasyfikacji przy podwójnej walidacji krzyżowej.

Tabela 11

Porównanie wyników klasyfikacji zbioru „Cukrzyca" przy podziale na trzy zbiory

				uczenie	testowanie
	metoda		typ konwersji	%	%
Rosetta	Standard Voting		–	72	43
Rosetta + ConRes	przy użyciu reguł konwersja 0, 1	perceptron	zwykła	53	53
			częściowa	53	53
		propagacja wsteczna	zwykła	69	43
			częściowa	69	62
	przy użyciu reguł konwersja -1, 0, 1	perceptron	zwykła	79	62
			częściowa	51	49
		propagacja wsteczna	zwykła	91	57
			częściowa	100	45

W przypadku tego eksperymentu wsteczna propagacja błędu nie jest efektywniejsza niż perceptron. Najlepszą metodą okazuje się klasyfikator zespołowy złożony, z 5 reguł wygenerowanych dynamicznie algorytmem genetycznym i sieci uczonej metodą wstecznej propagacji błędu przy zastosowaniu konwersji częściowej 0, 1 lub też sieci uczonej perceptronem przy konwersji zwykłej –1, 0, 1. Perceptron przy konwersji 0, 1 dał jednakowe wyniki dla konwersji zwykłej i częściowej. Przy konwersji –1, 0, 1 konwersja zwykła okazała się lepsza niż częścio-

wa. Standardowe głosowanie reguł klasyfikuje zdecydowanie gorzej niż klasyfikator zespołowy. Należy zauważyć w tym przypadku dużą różnicę jakości klasyfikacji występującą między zbiorem uczącym a testowym. Wpływ na to ma stosunkowo mały rozmiar zbioru uczącego (17%) w porównaniu ze zbiorem testowym (50%).

Wnioski

Po przeprowadzeniu serii eksperymentów można zauważyć następujące prawidłowości:

1. W większości przypadków konwersja 0, 1 okazuje się lepsza niż konwersja –1, 0, 1. Wynika to stąd, że na zbiorze testowym stosowana jest zawsze konwersja 0, 1 (ponieważ nie jest znana decyzja, do jakiej klasy należy dany obiekt, więc porównuje się jedynie wartości atrybutów w części *if* reguły z wartościami odpowiednich atrybutów dla obiektu). Zastosowanie wtedy tego samego typu konwersji na zbiorze uczącym daje lepsze rezultaty klasyfikacji zbioru testowego. Sam zbiór uczący jest efektywniej klasyfikowany przy konwersji –1, 0, 1, która w większym stopniu odzwierciedla dopasowanie reguły do obiektu, ponieważ bierze pod uwagę decyzję. Zastosowanie jej jednak powoduje zbytnie dopasowanie sieci do zbioru uczącego i tym samym spadek jakości klasyfikacji zbioru testowego.
2. Konwersja częściowa okazuje się lepsza niż zwykła w eksperymentach ze zbiorami „Australian" i „Cukrzyca" przy podziale na dwa zbiory, ale nieco gorsza dla zbiorów „Iris" i „Cukrzyca przy podziale na trzy zbiory (przy czym dla zbioru Iris jest to bardziej widoczna różnica).
3. Podział zbioru na trzy podzbiory nie wpływa na klasyfikację w przypadku korzystania z klasyfikatora zespołowego.
4. Czasami lepsza klasyfikacja zbioru testowego niż uczącego wypływa z konstrukcji klasyfikatora zespołowego – obiekty mogą lepiej dopasować się do danej bazy reguł, co ma odzwierciedlenie w tablicy konwersji podawanej na wejście sieci.
5. Prosta sieć jednokierunkowa pozwala na łatwą interpretacją wag - widać zależności między regułami.
6. W większości eksperymentów lepszą metodą generowania reguł okazała się metoda reduktów dynamicznych, wykorzystująca zarówno algorytm genetyczny jak i algorytm Johnsona.
7. Wartości wag otrzymane po uczeniu sieci pokazują na ile dana reguła jest ważna dla klasyfikacji obiektów. Wagi otrzymane po uczeniu sieci metodą propagacji wstecznej różnią się, zależnie od metody konwersji. Przy konwersji zwy-

kłej mają zazwyczaj większą wartość bezwzględną niż przy konwersji częściowej; znak w większości przypadków pozostaje taki sam. Można to wytłumaczyć tym, że przy konwersji zwykłej reguła odpalana jest tylko przy całkowitym spełnieniu części *if*, a więc rzadziej niż przy konwersji częściowej, gdzie reguły są odpalane już przy częściowym spełnieniu części *if*. Przy konwersji częściowej wartości wag są bardziej zróżnicowane niż przy konwersji zwykłej.

Zastosowanie sztucznych sieci neuronowych pozwala uzyskać lepszą optymalizację całego układu klasyfikującego, tak ze względu na liczbę reguł jak i jakość rozwiązania. Ponadto charakterystyczna dla sieci zdolność adaptacji i dość duża elastyczność pozwoliła na skonstruowanie klasyfikatora zespołowego cechującego się prostą strukturą i mającego dobrą odporność na zakłócenia w zbiorze danych.

Pracę wykonano w ramach realizacji grantu Komitetu Badań Naukowych.

Literatura

[1] An A., Cercone N.: Rule Quality Measures for Rule Induction Systems: Description and Evaluation, Computational Intelligence: An International Journal, Vol. 17, No 3, 2001, 409-424.

[2] Fayyad U.M., Piatetsky-Shapiro G., Smyth P., Uthurusamy R. (Eds.): Advances in Knowledge Discovery and Data Mining, The AAAI Press/The MIT Press 1996.

[3] Mitchell T.M.: Machine Learning. Mc Graw-Hill, Portland, 1997.

[4] Michie D., Spiegelhalter D., Taylor C.C.: Machine Learning, Neural and Statistical Classification, Ellis Horwood Limited, England, 1994.

[5] Pal S.K., Skowron A. (Eds.): Rough Fuzzy Hybridization: A New Trend in Decision-Making. Springer-Verlag, Singapore, 1999.

[6] Pawlak Z.: Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, 1991.

[7] Polkowski L., Skowron A. (Eds.): Rough Sets in Knowledge Discovery 1 and 2, Physica-Verlag, Heidelberg, 1998.

[8] Rosetta: Home Page, http://www.idi.ntnu.no/~aleks/rosetta/.

[9] Stepaniuk J.: Rough Set Data Mining of Diabetes Data. Proceedings of the Eleventh International Symposium on Methodologies for Intelligent Systems, Foundations of Intelligent Systems (ISMIS'99), June 8-11 1999, Warsaw, Lecture Notes in Artificial Intelligence 1609, Springer - Verlag, Berlin, 457-465.

[10] Stepaniuk J.: Knowledge discovery by application of rough set models, L. Polkowski, S. Tsumoto, T.Y. Lin, (Eds.), Rough Sets: New Developments, Physica-Verlag, Heidelberg, 2000, 137-233.

[11] Szczuka M. S.: Symbolic methods and artificial neural networks in classifier construction, Ph. D. thesis, supervisor A. Skowron, Warsaw University, 1999.
[12] Szczuka M. S.: Refining Classifiers with Neural Networks, International Journal of Intelligent Systems, Vol. 16 No 1, 2001, 39-55.

CONFLICT RESOLUTION BETWEEN RULES

Summary: In this paper the methods of objects classification based on rough set theory and artificial neural networks are presented. The results of the experiments were discussed in relation to coupled classifier using decision rules and neural network to resolve the conflicts between them.

Key words: rough sets, neural networks, classification, decision rules

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Marek Krętowski[1], Leon Bobrowski[1,2]

GENEROWANIE WIELOWYMIAROWYCH DRZEW DECYZYJNYCH NA PODSTAWIE ZBIORÓW DANYCH

Streszczenie: W pracy przedstawiono nowe metody konstruowania klasyfikatorów o postaci wielowymiarowych (skośnych) drzew decyzyjnych na podstawie zbiorów uczących. Generowane drzewa zawierają w każdym węźle liniową regułę decyzyjną (hiperpłaszczyznę), która dzieli wektory cech docierające do tego węzła na dwa podzbiory. Strategie poszukiwania hiperpłaszczyzny decyzyjnej opierają się na minimalizacji dipolowych funkcji kryterialnych, a jako procedury optymalizujące wykorzystywane są algorytmy wymiany rozwiązań bazowych lub algorytmy ewolucyjne. Aby wyeliminować nadmiarowe cechy, w proces poszukiwania optymalnej kombinacji liniowej w węźle wbudowana została selekcja cech. Dla uniknięcia zbytniego dopasowania do danych treningowych i zwiększenia zdolności do generalizacji, uzyskane drzewo jest przycinane. Przedstawione metody zostały zweryfikowane eksperymentalnie na publicznie dostępnych zbiorach danych i porównane z innymi algorytmami indukującymi drzewa decyzyjne.

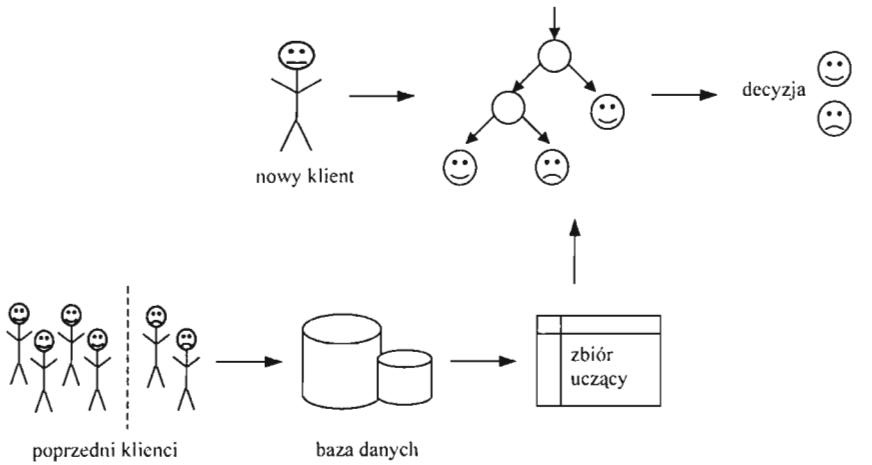
Słowa kluczowe: wielowymiarowe drzewa decyzyjne, liniowa reguła decyzyjna, kryteria dipolowe

1. Wstęp

Z problemami rozpoznawania i klasyfikacji człowiek ma do czynienia w wielu dziedzinach życia, chociaż rzadko je bezpośrednio dostrzega i nazywa. Klasyfikację możemy nieformalnie określić jako czynność polegającą na przypisaniu zjawiskom bądź rzeczom wcześniej sprecyzowanych etykiet (klas, kategorii). W wyniku klasyfikacji dany zbiór obiektów zostaje podzielony na rozłączne grupy. Niekiedy nie zdajemy sobie z tego sprawy, że aby poprawnie klasyfikować obiekty najpierw musimy się nauczyć je rozpoznawać. Czasami dysponujemy wiedzą zdobytą i przekazaną nam przez poprzedników. W takich wypadkach, wystarczy sto-

[1] Wydział Informatyki, Politechnika Białostocka, Białystok
[2] Instytut Biocybernetyki i Inżynierii Biomedycznej, PAN, Warszawa

sować ją w praktyce. Gdy tej wiedzy nie mamy, musimy sami stworzyć reguły postępowania, opierając się na zdobywanym doświadczeniu. Można przyjąć, że uczenie się na podstawie uprzednio zgromadzonej informacji jest niejako częścią naszego sposobu podejmowania decyzji.



Rys. 1. Schemat wykorzystania metod klasyfikacji (drzew decyzyjnych) w procesie podejmowania decyzji w przykładowej instytucji

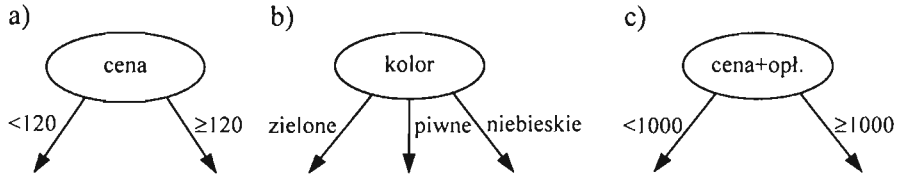
Wraz z nadejściem ery komputerów i ich ciągłym doskonaleniem pojawiła się szansa wykorzystania coraz większej mocy obliczeniowej maszyn we wspomaganiu procesu podejmowania decyzji przez człowieka (rys. 1). Dzięki systemom baz danych mamy możliwość gromadzenia i bezpośredniego dostępu do coraz większej liczby informacji. Powstaje specjalistyczne oprogramowanie, które pozwala na prawie automatyczne przetwarzanie i analizę nawet bardzo dużych zbiorów danych. Uzyskiwane w taki sposób klasyfikatory charakteryzują się wysoką jakością, i same mogą stać się źródłem pożytecznej wiedzy na temat rozpatrywanego zjawiska. Zastosowań jest bardzo wiele i są one niezwykle zróżnicowane: począwszy od ustalenia prawdopodobieństwa zwrócenia pożyczki przez potencjalnego klienta banku, na podstawie wcześniej zrealizowanych pożyczek, po rozpoznawanie komórek nowotworowych na obrazach biomedycznych, czy też wykrywanie nieuprawnionych rozmów w sieciach telefonii komórkowej. Z punktu widzenia metod tworzenia klasyfikatorów wszystkie te problemy są bardzo zbliżone. Zagadnienia tego typu, ze względu na ich bardzo duże znaczenie praktyczne, są przedmiotem intensywnych badań w takich dziedzinach jak statystyka, rozpoznawanie wzorców (ang. *pattern recognition*) czy uczenie maszyn (ang. *machine learning*). Badania zaowocowały m. in. opracowaniem różnego typu klasyfikatorów (np.: w postaci

reguł i drzew decyzyjnych czy sieci neuropodobnych) oraz znacznej liczby algorytmów automatycznego tworzenia takich struktur decyzyjnych na podstawie zbiorów danych. Chociaż każda z tych metod ma swoje zalety, wiemy, że nie można zbudować klasyfikatora, optymalnego dla wszystkich sytuacji („no free lunch theorem" [12]). W niniejszej pracy koncentrujemy się na jednym typie klasyfikatora, a mianowicie na drzewach decyzyjnych. Do głównych zalet drzew decyzyjnych możemy zaliczyć przede wszystkim szybkość ich tworzenia i prostotę korzystania z nich oraz łatwość uzasadnienia proponowanych decyzji. W pracy pokazujemy jak wykorzystując metody dipolowe można efektywnie generować wielowymiarowe drzewa decyzyjne na podstawie zbiorów uczących.

Dalsza część pracy jest zorganizowana w następujący sposób. W kolejnym punkcie przedstawione zostanie krótkie wprowadzenie do metody drzew decyzyjnych. Omówiony zostanie podstawowy algorytm generowania drzewa, a także scharakteryzowane różne typy drzew decyzyjnych. Punkt 3 zawiera przegląd istniejących systemów służących do indukcji (głównie wielowymiarowych) drzew decyzyjnych. W punkcie 4 przedstawiono pojęcie dipola i sformułowano postulaty projektowe. Na tej podstawie zostaną zaproponowane metody poszukiwania optymalnych kombinacji liniowych cech w węzłach drzewa decyzyjnego. Metody te obejmują zarówno zdefiniowanie funkcji kryterialnych (służących do oceny jakości proponowanych podziałów) jak i algorytmy optymalizacji tych funkcji. Inne zagadnienia związane z tworzeniem wielowymiarowych drzew decyzyjnych, takie jak np. selekcja cech czy też problem unikania zbytniego dopasowania do danych uczących, zostaną omówione w punkcie 5. Eksperymentalna weryfikacja zaproponowanych algorytmów indukcji drzew znajduje się w punkcie 6. W ostatnim punkcie podsumowujemy oraz przedstawiamy możliwe kierunki prac badawczych.

2. Wprowadzenie do drzew decyzyjnych

Drzewo decyzyjne jest strukturą hierarchiczną, zbudowaną z węzłów i łączących je krawędzi (gałęzi). W strukturze tej jeden węzeł jest wyróżniony i nazywany *korzeniem* (ang. *root node*), a każdy z pozostałych węzłów ma swojego poprzednika w drzewie. Węzły terminalne, czyli takie, które nie posiadają potomków, nazywane są liśćmi (ang. *leaves*). Każdemu liściowi przypisana jest decyzja klasyfikacyjna. W węzłach niebędących liśćmi, zapisane są testy (zwane też regułami podziału). Z każdym możliwym wynikiem testu związana jest gałąź prowadząca do węzła potomnego. W najprostszej sytuacji, teście opartym na pojedynczym atrybucie numerycznym, mamy dwa możliwe wyniki – dwie gałęzie. W przypadku testów opartych na atrybutach nominalnych liczba poddrzew może być większa. Na rysunku 2 przedstawiono przykłady różnych testów w węzłach drzewa.



Rys. 2. Przykładowe testy w węźle drzewa decyzyjnego: (a) jednowymiarowy, binarny, na wartościach rzeczywistych (b) jednowymiarowy, na wartościach nominalnych (c) binarny, wielowymiarowy

Zakładając, że znamy wartości atrybutów nowego obiektu, a nie wiemy, do której klasy należy, możemy przy użyciu drzewa decyzyjnego spróbować przewidzieć jego przynależność do jednej z klas. W tym celu należy wykonać serię testów w kolejnych węzłach drzewa, rozpoczynając od testu zapisanego w korzeniu drzewa, a kończąc na osiągnięciu liścia. Wybór potomka rozpatrywanego węzła drzewa, do którego następuje przejście, dokonywany jest na podstawie wyniku testu zastosowanego do klasyfikowanego obiektu. W momencie, gdy obiekt ten dochodzi do liścia, przypisujemy mu decyzję związaną z tym liściem. Warto zauważyć, że ścieżka od korzenia do liścia odpowiada pojedynczej regule decyzyjnej, z testami połączonymi koniunkcją po jej lewej stronie i decyzją związaną z liściem po stronie prawej.

W zasadzie wszystkie algorytmy generowania drzew decyzyjnych na podstawie zbioru danych opierają się na powszechnie stosowanej w informatyce metodzie znajdującej rozwiązanie problemu poprzez podział go na podproblemy (ang. *divide and conquer*). W odniesieniu do drzew, rozpoczynając od korzenia, znajdowany jest test i na podstawie jego wyników tworzone są węzły potomne oraz rozdzielany do nich jest zbiór uczący. Następnie ta sama procedura podziału stosowana jest rekurencyjnie w węzłach potomnych, aż do momentu, kiedy osiągnięty zostaje warunek stopu i tworzony jest liść. Poniżej przedstawiono podstawowy algorytm indukcji drzewa w postaci funkcji w pseudokodzie:

```
void InduceDecTree(DTNode *pN)
{
   ...                  // sprawdzany jest warunek stopu
   if (!Stop(pN))       // np. czy w ogóle warto dzielić

   if (FindTest(pN))    // poszukiwanie testu i o ile zakończone
                        // pomyślnie zwracane true
      {
      nOutcome=SplitNode(pN);     //liczba możliwych wyników testu
                                  //a tym samym gałęzi (podrzew)
```

```
            for (int i=0; i<nOutcome; i++)
                InduceDecTree(pN->GetOutcome(i));
        }
        ...
        // o ile nie znaleziono testu w pN oznaczamy jako liść
        // w przeciwnym przypadku jako węzeł
    }
```

Należy zwrócić uwagę, że opisana powyżej zachłanna (ang. *greedy*) strategia indukcji drzewa od korzenia do liści (ang. *top-down*), jak każda metoda heurystyczna, nie gwarantuje otrzymania optymalnego drzewa. Tym niemniej jest prosta i szybka, a co najważniejsze, uzyskiwane dzięki niej klasyfikatory charakteryzują się dobrą jakością klasyfikacji i małymi rozmiarami.

Podstawowe problemy, które musimy rozwiązać proponując metodę generowania drzewa decyzyjnego [9], to przede wszystkim:

a) sposób wyboru testu w węźle drzewa – `FindTest(pN)`,
b) określenie kiedy przerwać rekurencyjny podział – `Stop(pN)`,
c) przypisanie decyzji liściowi (zwykle stosowana jest reguła większościowa).

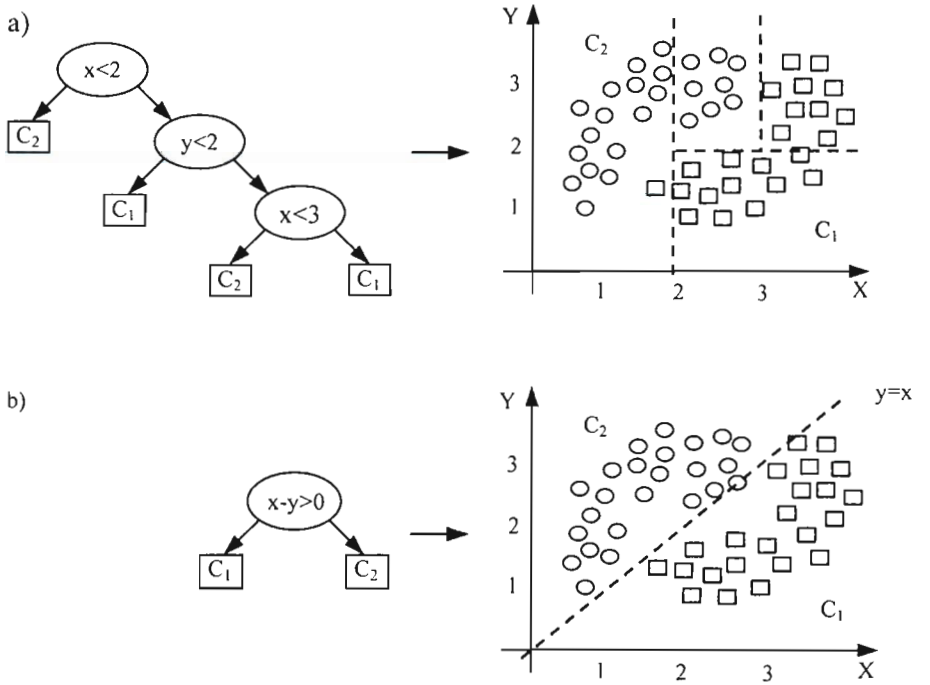
Ad a) Najczęściej stosowana jest metoda wyboru optymalnego testu, polegająca na maksymalnej redukcji zanieczyszczenia (ang. *impurity*) węzła w jego potencjalnych potomkach. W przypadku najprostszych postaci testów możliwe jest nawet sprawdzanie wszystkich znaczących podziałów. Jako miary zanieczyszczenia wykorzystywane są najczęściej: *indeks Gini* [9] lub *entropia* [21].

Ad b) Najbardziej naturalną przesłanką do zatrzymania algorytmu jest otrzymanie węzła zawierającego tylko obiekty z jednej klasy. Zastosowanie takiego kryterium pozwala uzyskać, o ile zbiór uczący nie jest sprzeczny, drzewo poprawnie klasyfikujące wszystkie obiekty ze zbioru uczącego. Niestety, powoduje to często nadmierne dopasowanie się do danych treningowych. Do tego zagadnienia powrócimy w dalszej części pracy.

W zależności od liczby cech wykorzystywanych przez regułę decyzyjną w węźle możemy wyróżnić dwie rodziny drzew decyzyjnych: jednowymiarowe lub wielowymiarowe. Drzewa jednowymiarowe wykorzystują w każdym teście tylko pojedynczą cechę, natomiast wielowymiarowe mogą opierać się na większej liczbie zmiennych w poszczególnych węzłach. Szczególnym przypadkiem drzew wielowymiarowych są tak zwane drzewa skośne (ang. *oblique*), w których reguła decyzyjna ma postać liniowej kombinacji cech.

Z geometrycznego punktu widzenia jednowymiarowe testy związane są z hiperpłaszczyznami równoległymi do osi (ang. *axis-parallel*) w przypadku liniowych testów wielowymiarowych, orientacja hiperpłaszczyzn nie jest praktycznie ograniczona. Drzewa jednowymiarowe znalazły wiele praktycznych zastosowań, głównie

dzięki dostępności komercyjnych systemów służących do ich generowania oraz dzięki szybkości tworzenie i łatwości interpretacji uzyskiwanych klasyfikatorów. Drzewa wielowymiarowe oferują znacznie większe możliwości, choć – niestety – ich indukcja może wymagać więcej czasu (algorytmy poszukiwania testów w węzłach są zwykle bardziej skomplikowane) interpretacja uzyskanego drzewa może być nieco trudniejsza. Na rysunku 3 przedstawiona została sytuacja, w której klasyfikator w postaci skośnego drzewa decyzyjnego jest naturalnym i optymalnym rozwiązaniem.



Rys. 3. Przykładowe drzewa wygenerowane na podstawie tego samego dwuklasowego zbioru danych: (a) jednowymiarowe, (b) wielowymiarowe (skośne)

3. Przegląd metod konstrukcji drzew decyzyjnych

Zaproponowano już wiele systemów generujących drzewa decyzyjne na podstawie zbiorów uczących [19]. Początkowo interesowano się jedynie drzewami jednowymiarowymi. Do najbardziej znanych systemów należy z pewnością zaliczyć ID3 [21], zaproponowany przez Quinlana. System ten ewoluował przez lata i jego publicznie dostępna wersja, pod nazwą C4.5 [22], jest powszechnie wyko-

rzystywana we wszelkiego typu porównaniach. Spośród innych algorytmów warto wymienić CHAID (*Chi-squared Automatic Interaction Detection*) [15], który może być stosowany tylko w przypadku danych nominalnych; do wyboru podziału w węźle wykorzystuje się statystykę chi-kwadrat.

Algorytmy stosowane w przypadku wielowymiarowych drzew decyzyjnych są ze swej natury bardziej złożone niż wykorzystywane w drzewach jednowymiarowych. Dlatego też w początkowym okresie ich rozwój i praktyczne zastosowanie było hamowane przede wszystkim z powodu ograniczonych możliwości komputerów. W ostatnim jednak czasie, dzięki wzrostowi mocy obliczeniowej, drzewa wielowymiarowe zasługują na coraz większą uwagę. W tej chwili mogą być z powodzeniem stosowane nawet w przypadku dużych zbiorów danych (zawierających tysiące obiektów).

Po raz pierwszy możliwość wykorzystania wielowymiarowych testów podczas konstrukcji drzewa została opisana w powszechnie znanej książce Breimana i in. [9]. Autorzy opisują w niej system CART (*Classification and Regression Tree*), który, oprócz standardowych podziałów wykorzystujących pojedyncze zmienne, jest w stanie m. in. szukać testów w postaci liniowej kombinacji cech. Do poszukiwania hiperpłaszczyzny użyte są tylko cechy numeryczne, a wagi są dodatkowo znormalizowane (suma kwadratów wag jest równa 1). Zaproponowany algorytm w kolejnych krokach próbuje modyfikować wagi pojedynczych atrybutów, optymalizując miarę zanieczyszczenia. Następnie w celu uproszczenia tekstu atrybuty są usuwane z kombinacji liniowej, o ile strata przez to spowodowana nie przekracza zadanego procentowego progu. Po znalezieniu optymalnej hiperpłaszczyzny, miara zanieczyszczenia jest porównywana z miarą odpowiadającą najlepszemu podziałowi jednowymiarowemu. Na tej podstawie wybierany jest lepszy test, przy czym preferowane są testy oparte na pojedynczej zmiennej.

Metoda zaproponowana przez Breimana *i in.* jest w pełni deterministyczna i może wpadać w lokalne minima. Murthy *i in.* [18] zaproponowali system OC1 (*Oblique Classifier 1*), rozszerzający metodę poszukiwania liniowych kombinacji systemu CART poprzez wprowadzenie randomizacji. W momencie gdy algorytm zatrzymuje się (być może w lokalnym minimum), system próbuje dodawać losowe wektory do aktualnie optymalnej hiperpłaszczyzny i, jeżeli uzyskana kombinacja liniowa jest lepsza, kontynuuje poszukiwania. Inną możliwością jest kilkukrotny start z losowo wybranych miejsc, różnych od użytego w pierwszej próbie optymalnego testu jednowymiarowego. Wyniki uzyskiwane przy użyciu tak rozszerzonej metody uległy znaczącej poprawie i system został wykorzystany z powodzeniem m. in. w diagnostyce raka piersi.

Chai *et al.* [11] zaproponowali metodę tworzenia liniowo-odcinkowego klasyfikatora w postaci binarnego drzewa (BTGA, *Binary Tree-Genetic Algorithm*) z liniową regułą decyzyjną w każdym węźle. Do poszukiwania optymalnej hiper-

płaszczyzny, maksymalnie redukującej zanieczyszczenie, wykorzystany został standardowy algorytm genetyczny. Zmodyfikowana nieznacznie metoda została wykorzystana z powodzeniem w diagnostyce raka szyjki macicy do klasyfikacji komórek na wymazach cytologicznych.

W pracy [14] Gama i Brazdil przedstawili system *Ltree*, który rozszerza algorytm generowania drzewa decyzyjnego poprzez wykorzystanie konstruktywnej indukcji. W każdym węźle nowe atrybuty tworzone są poprzez rzutowanie wektorów cech na hiperpłaszczyznę wyznaczoną przez liniową regułę dyskryminacyjną. W ten sposób utworzone cechy są następnie propagowane w dół drzewa i mogą być wykorzystane w tworzeniu kolejnych cech.

Większość algorytmów indukcji drzew decyzyjnych wybiera podział w węźle optymalizując, w ten czy inny sposób zdefiniowaną, miarę zanieczyszczenia. Nie jest to jednak jedyna możliwość. Przykładowo w pracy [20] autorzy argumentują, że w przypadku drzew skośnych lepsze rezultaty można osiągnąć stosując do oceny potencjalnych podziałów w węźle miarę bazującą na liniowej separowalności danych. Ponieważ zaproponowana miara wykorzystuje liniową separowalność podwęzłów, może być traktowana jako rozszerzenie tradycyjnej zachłannej strategii konstrukcji drzewa o sprawdzenie „krok do przodu". Podejście to może być zastosowane tylko do problemów dwuklasowych.

Przedstawione powyżej systemy wykorzystują pojedynczą hiperpłaszczyznę w węźle drzewa. Istnieją również drzewa wielowymiarowe oparte na *maszynie liniowej* (ang. *linear machine*) [12], która jest zbiorem działających wspólnie liniowych funkcji dyskryminacyjnych. Najbardziej znanym takim drzewem jest LMDT (*Linear Machine Decision Tree*), zaproponowany przez Utgoff i Brodley [23]. W przeciwieństwie do wcześniej omówionych binarnych drzew, w tym przypadku liczba poddrzew zależna jest od liczby klas. Do uczenia maszyny liniowej w każdym węźle stosowany jest algorytm *thermal training* zbliżony do algorytmu symulowanego wyżarzania (ang. *simulated annealing*). LMDT wyposażony jest również w mechanizmy selekcji cech.

4. Poszukiwanie hiperpłaszczyzny w węźle drzewa

W tym paragrafie zostanie przedstawiony sposób poszukiwania liniowej reguły decyzyjnej w pojedynczym węźle drzewa na podstawie danych ze zbioru uczącego, które dotarły do rozpatrywanego węzła. Omówione zostaną miary jakości podziału (funkcje kryterialne) oparte na pojęciu dipoli oraz opisane metody wykorzystywane do ich optymalizacji.

Zbiór uczący C składa się z M obiektów, którym dla porządku przypisujemy numery w zbiorze. Każdy obiekt O^i opisany jest przez N – wymiarowy wektor cech $\mathbf{x}^i = \left[x_1^i,...,x_N^i\right]^T$, ($x_j^i \in R^1$). W przypadku cech nominalnych (takich jak przykładowo *kolor oczu* czy *kraj zamieszkania*) niezbędne jest wcześniejsze zakodowanie tego typu informacji[3]. Dodatkowo każdy obiekt należy do (jednej z K klas) klasy ω_k ($k = 1,..., K$). W podzbiorze uczącym C_k zgrupowane są wszystkie wektory cech $\mathrm{x}^i(k)$, odpowiadające obiektowi O^i, należące do klasy ω_k (C_k zawiera m_k wektorów cech $\mathbf{x}^i(k)$).

Przestrzeń cech może być podzielona na dwie podprzestrzenie przy użyciu hiperpłaszczyzny $H(\mathbf{w},\theta)$ będącej kombinacją liniową cech:

$$H(\mathbf{w},\theta) = \{\mathbf{x}:\langle \mathbf{w},\mathbf{x}\rangle = w_1x_1 + w_2x_2 + ... + w_Nx_N = \theta\}, \tag{1}$$

gdzie $\mathbf{w}$ jest N-wymiarowym wektorem wag, θ – progiem, a $\langle \mathbf{w},\mathbf{x}\rangle$ jest iloczynem skalarnym. Mówimy, że wektor cech $\mathbf{x}^i$ leży po *dodatniej* (*ujemnej*) stronie hiperpłaszczyzny $H(\mathbf{w},\theta)$ jeżeli:

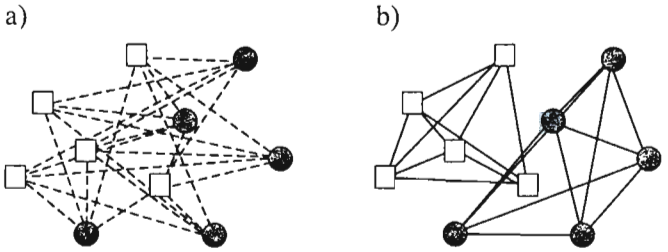
$$\langle \mathbf{w},\mathbf{x}^i\rangle > \theta \qquad (\langle \mathbf{w},\mathbf{x}^i\rangle < \theta). \tag{2}$$

Aby uprościć zapis, wprowadźmy tak zwaną rozszerzoną przestrzeń cech, w której wektor cech ma postać $\mathbf{y} = [1, x_1,..., x_N]^T$, rozszerzony wektor wag przedstawia się następująco $\mathbf{v} = [-\theta, w_1,..., w_N]^T$, natomiast $H(\mathbf{v}) = \{\mathbf{y} : \langle \mathbf{v},\mathbf{y}\rangle = 0\}$ jest hiperpłaszczyzną.

4.1. Ocena jakości hiperpłaszczyzny – funkcja kryterialna

Podstawowym pojęciem, od którego przedstawienia rozpoczniemy, jest *dipol*, czyli para $(\mathbf{y}^i,\mathbf{y}^j)$ różnych wektorów cech [3]. Wyróżniamy dwa typy dipoli: *czyste* i *mieszane*. Mówimy, że dipol jest czysty, jeżeli wektory cech tworzące go należą do tego samego podzbioru uczącego C_k. Z dipolem mieszanym mamy natomiast do czynienia w sytuacji, gdy obiekty tworzące dipol mają przypisane różne decyzje. Na rysunku 4 zilustrowano typy dipoli na przykładzie 10-elementowego zbioru danych, natomiast w tabeli 1 przedstawione zostały liczności poszczególnych typów dipoli.

[3] Istnieje wiele metod kodowania informacji nominalnych (symbolicznych) w postaci numerycznej. Jedną z najprostszych jest stworzenie dodatkowych binarnych atrybutów odpowiadających poszczególnym kategoriom.



Rys. 4. Wszystkie dipole mieszane (a) oraz czyste (b), utworzone z wektorów cech (punktów) należących do przykładowego 10-elementowego zbioru danych

Tabela 1

Liczby dipoli poszczególnych typów

Liczba czystych	$D_c = \frac{1}{2} \cdot \sum_{i=1}^{K} m_i \cdot (m_i - 1)$
Liczba mieszanych	$D_m = \frac{1}{2} \cdot \sum_{i=1}^{K} m_i \cdot (M - m_i)$
Razem (wszystkich) dipoli	$\frac{1}{2} \cdot M \cdot (M - 1)$

Hiperpłaszczyzna $H(\mathbf{v})$ przecina dipol $(\mathbf{y}^i, \mathbf{y}^j)$ jeżeli wektory cech go tworzące znajdują się po jej przeciwnych stronach, czyli:

$$\langle \mathbf{v}, \mathbf{y}^i \rangle \cdot \langle \mathbf{v}, \mathbf{y}^j \rangle < 0 \tag{3}$$

Z punktu widzenia wykorzystania liniowej reguły decyzyjnej do podziału wektorów cech w węźle drzewa, hiperpłaszczyzna powinna przecinać możliwie dużo dipoli mieszanych. W idealnym przypadku, jeżeli wszystkie dipole mieszane są przecięte, mamy do czynienia z liniowym odseparowaniem podzbiorów uczących (przypadek 2-klasowy). Z drugiej strony powinniśmy się starać unikać przecinania dipoli czystych. Przecięcie ich oznacza bowiem, że wektory cech z tej samej klasy skierowane zostaną do różnych poddrzew.

Bazując na powyższym rozważaniu można zaproponować [6] prostą miarę jakości podziału (funkcję kryterialną) postaci:

$$\Psi_1(\mathbf{v}) = \alpha_c \cdot \frac{d_c}{D_c} + \alpha_m \cdot \frac{d_m}{D_m} \tag{4}$$

gdzie:

d_c – liczba dipoli czystych, które zostały przecięte przez $H(\mathbf{v})$,

d_m – liczba dipoli mieszanych, które nie zostały przecięte przez $H(\mathbf{v})$,

α_c, α_m – relatywny wpływ (cena) poszczególnych rodzajów dipoli na wartość funkcji.

Należy zwrócić uwagę na fakt, że cena α_m związana z dipolami mieszanymi powinna być znacząco większa niż cena α_c odnosząca się do dipoli czystych. Podstawowym celem jest przecinanie dipoli mieszanych. W drugiej kolejności staramy się unikać przecinania dipoli czystych. W przeciwnym razie możemy doprowadzić do sytuacji, w której żaden dipol nie będzie przecięty, a takie rozwiązanie jest zwykle bezużyteczne. Poszukiwanie optymalnych hiperpłaszczyzn $H(\mathbf{v})$ opiera się na minimalizacji funkcji kryterialnej $\Psi_1(\mathbf{v})$ (4).

Jeżeli bierzemy pod uwagę wszystkie dipole i jeżeli przez $p(k)$ oznaczymy obiekty z klasy ω_k położone po pozytywnej (dodatniej) stronie hiperpłaszczyzny $H(\mathbf{v})$, a przez $n(k)$ odpowiednio po stronie negatywnej, to d_m i d_c możemy obliczyć w następujący sposób:

$$d_m = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \left(p(i) \cdot p(j) + n(i) \cdot n(j) \right), \quad d_c = \sum_{i=1}^{K} p(i) \cdot n(i). \tag{5}$$

Ponieważ funkcja Ψ_1 ma wiele minimów lokalnych, jej minimalizacja nie jest prostym zadaniem optymalizacyjnym. W przypadku takich właśnie funkcji dobrze sprawdzają się metody takie jak *algorytmy ewolucyjne* (*genetyczne*) [17]. Niestety, nie należą one do najszybszych metod i, zwłaszcza w przypadku dużych zbiorów danych, tworzenie drzewa decyzyjnego może wymagać sporo czasu.

Alternatywne podejście do poszukiwania hiperpłaszczyzny zainspirowane zostało algorytmami uczenia *Perceptronu* [12], które w przypadku liniowej separowalności zbioru (2 klasy) dają rozwiązania optymalne. W klasycznej funkcji perceptronowej z każdym błędnie sklasyfikowanym przykładem wiążemy karę w postaci iloczynu skalarnego, która może być traktowana jako odległość obiektu od hiperpłaszczyzny (w przypadku unormowania wektora wag). Minimalizowana funkcja kryterialna jest sumą tych kar. Dodatkowo wprowadzenie marginesów (ang. *margin*) pozwala znajdować rozwiązania bardziej „stabilne", a funkcja kryterialna pozostaje nadal wypukła i odcinkowo-liniowa (ang. *piecewise-linear*). Do optymalizacji funkcji tego typu zostały zaproponowane efektywne algorytmy, takie jak algorytmy wymiany rozwiązań bazowych ([2], [8]). Aby móc wykorzystać zalety tych technik, dipolowa funkcja kryterialna również musi mieć analogiczne właściwości.

W naszym przypadku z wektorem cech $\mathbf{y}^j$ możemy związać dwa typy składowych funkcji kary: „pozytywną"

$$\varphi_j^+(\mathbf{v}) = \begin{cases} \delta^j - \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle < \delta^j \\ 0 & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle \geq \delta^j \end{cases}, \tag{6}$$

lub „negatywną"

$$\varphi_j^-(\mathbf{v}) = \begin{cases} \delta^j + \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle > -\delta^j \\ 0 & if \left\langle \mathbf{v}, \mathbf{y}^j \right\rangle \leq -\delta^j \end{cases}, \tag{7}$$

gdzie δ^j ($\delta^j \geq 0$) jest marginesem, którego wartość może być różna dla różnych przypadków (we wszystkich prezentowanych w pracy eksperymentach miała wartość 1.0). Nazwy funkcji kary odpowiadają stronom w stosunku do poszukiwanej hiperpłaszczyzny $H(\mathbf{v})$, po których powinien się znaleźć wektor cech $\mathbf{y}^j$ w wyniku minimalizacji funkcji kary. Inaczej mówiąc, w przestrzeni wag wykorzystanie pozytywnej funkcji kary powoduje spychanie wektora wag na dodatnią stronę hiperpłaszczyzny związanej z wektorem cech. Opierając się na składowych funkcjach kary można łatwo osiągnąć wymagany efekt w odniesieniu do dipoli. I tak, aby wymusić przecinanie dipola mieszanego $\left(\mathbf{y}^i, \mathbf{y}^j\right)$ przez $H(\mathbf{v})$, należy związać z obiektami tworzącymi dipol przeciwne funkcje kary, np.: ujemną $\varphi_j^-(\mathbf{v})$ oraz dodatnią $\varphi_i^+(\mathbf{v})$ (lub odwrotnie). Możemy zdefiniować funkcję kary związaną z dipolem mieszanym jako sumę odpowiednich funkcji:

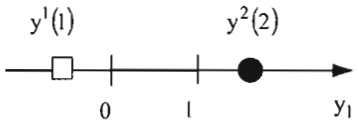
$$\varphi_{ij}^m(\mathbf{v}) = \varphi_i^+(\mathbf{v}) + \varphi_j^-(\mathbf{v}) \qquad (\text{lub } \varphi_{ij}^m(\mathbf{v}) = \varphi_i^-(\mathbf{v}) + \varphi_j^+(\mathbf{v})) \tag{8}$$

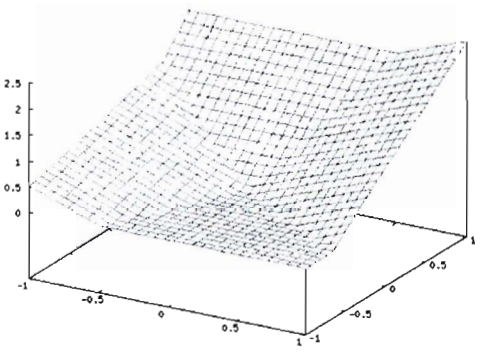
W przypadku dipola czystego, aby uniknąć jego przecięcia, wiążemy z obiektami tworzącymi ten dipol funkcje kary tego samego znaku, dzięki czemu obiekty te powinny się znaleźć po tej samej stronie hiperpłaszczyzny. Funkcja kary związana z czystym dipolem przybiera postać następującą:

$$\varphi_{ij}^c(\mathbf{v}) = \varphi_i^+(\mathbf{v}) + \varphi_j^+(\mathbf{v}) \qquad (\text{lub } \varphi_{ij}^c(\mathbf{v}) = \varphi_i^-(\mathbf{v}) + \varphi_j^-(\mathbf{v})) \tag{9}$$

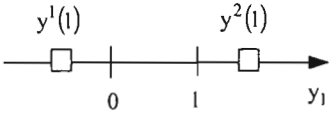
Aby móc lepiej wyobrazić wygląd funkcji kary, rozważmy pojedyncze dipole (mieszany i czysty) w jednowymiarowej przestrzeni cech. Na rysunku 5 przedstawione zostały odpowiadające im składowe funkcje kary w przestrzeni wag.

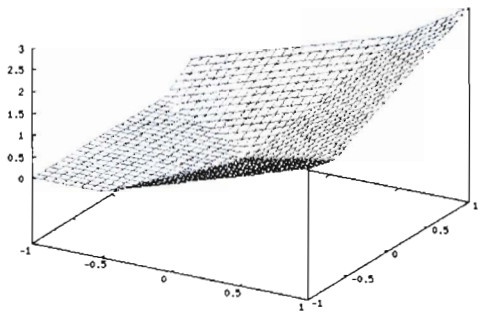
a)





b)





Rysunek 5. Pojedyncze dipole (mieszany – a oraz czysty – b) w 1-wymiarowej przestrzeni cech oraz wykresy składowych funkcji kary $\varphi_{12}^{m}(\mathrm{v})$ oraz $\varphi_{12}^{c}(\mathrm{v})$, związanych z danym typami dipoli, przedstawione w 2-wymiarowej przestrzeni wag

Wybór odpowiednich postaci funkcji kary, związanych z danym dipolem, nazywany jest *orientowaniem dipola* [4]. Aby zbytnio nie komplikować rozważań, można przyjąć, że orientacja dipoli jest dowolna, ale ustalona.

Dipolowa funkcja kryterialna [7] jest ważoną sumą funkcji kar związanych z poszczególnymi dipolami:

$$\Psi_2(\mathbf{v}) = \frac{1}{D_m} \cdot \sum_{\left(y^i, y^j\right) \in I^m} \alpha_{ij} \cdot \varphi_{ij}^{m}(\mathbf{v}) + \frac{1}{D_c} \cdot \sum_{\left(y^i, y^j\right) \in I^c} \alpha_{ij} \cdot \varphi_{ij}^{c}(\mathbf{v}) \qquad (10)$$

gdzie:

α_{ij} – relatywny wpływ (cena) dipola $\left(\mathbf{y}^i, \mathbf{y}^j\right)$ na wartość funkcji kryterialnej,

I^m (I^c) – zbiór dipoli mieszanych (czystych).

Warto zauważyć, że o ile w funkcji $\Psi_1(\mathbf{v})$ wszystkie dipole jednego typu były traktowane w ten sam sposób (miały przypisany ten sam wpływ α_m lub α_c), o tyle w przypadku funkcji $\Psi_2(\mathbf{v})$ każdy dipol może być traktowany niezależnie (cena jest bowiem związana z pojedynczym dipolem). Rodzi to możliwość proponowania różnorodnych strategii wyceniania dipoli i przez to wpływania na kształt funkcji kryterialnej a tym samym na optymalną hiperpłaszczyznę. Przykładowo można różnicować ceny dipoli, wykorzystując ich długości. W przedstawionych w tej pracy eksperymentach zastosowana została najprostsza z możliwych strategii ze stałymi cenami.

Dipolowa funkcja kryterialna (z ustaloną orientacją dipoli) jest, oczywiście, wypukła i odcinkowo-liniowa (jako ważona suma takich właśnie funkcji). Dzięki temu do jej minimalizacji, tak jak wspomniano wcześniej, możemy wykorzystać algorytmy wymiany rozwiązań bazowych. W rzeczywistości wykorzystana metoda jest kombinacją powyższych algorytmów i prostego algorytmu ustalania orientacji dipoli.

W pewnych sytuacjach (zwłaszcza gdy liczba klas jest większa niż 2) bardzo dobre rezultaty mogą być uzyskane poprzez minimalizację *kryterium rangowego* [5], które może być traktowane jako szczególny przypadek kryterium dipolowego. Celem kryterium rangowego jest odseparowanie obiektów z wyróżnionej klasy od wszystkich pozostałych obiektów. Można to osiągnąć poprzez przypisanie obiektom z klasy wyróżnionej składowych funkcji kary pozytywnych, natomiast pozostałym obiektom – negatywnych. W ten sposób dążymy do sytuacji, w której możliwie wiele obiektów z klasy wyróżnionej znajdzie się po dodatniej stronie hiperpłaszczyzny; pozostałe obiekty usytuowane są po stronie ujemnej. Warto zauważyć, że z punktu widzenia zastosowania metody dipolowej kryterium rangowe oznacza wykorzystanie tylko mieszanych dipoli, stworzonych z obiektów klasy wyróżnionej i pozostałych obiektów, przy czym orientacja jest taka sama dla wszystkich dipoli.

4.2. Metody optymalizacji funkcji kryterialnej

W przypadku optymalizacji funkcji, o której wiemy, że jest odcinkami-stała (ang. *piecewise constant*) i zapewne posiada minima lokalne, niezbędne jest zastosowanie metod, które przede wszystkim są w stanie unikać rozwiązań suboptymalnych. Szybkość działania jest w tej sytuacji czynnikiem mniej istotnym. Takimi technikami z pewnością są szeroko rozumiane algorytmy genetyczne. Do minimalizacji funkcji kryterialnej Ψ_1 zaprojektowany został wyspecjalizowany algorytm ewolucyjny, dzięki czemu udało się poprawić efektywność takiego podejścia. W przypadku funkcji kryterialnej Ψ_2 zastosowane zostało odmienne rozwiązanie.

Już sama funkcja zaprojektowana została z myślą o konkretnej technice optymalizacyjnej (algorytmach wymiany rozwiązań bazowych) i w związku z tym posiada określone własności: jest wypukła i odcinkowo-liniowa (ang. *piecewise linear*).

4.2.1. Algorytmy wymiany rozwiązań bazowych

Algorytmy wymiany rozwiązań bazowych ([2], [8]) są efektywnymi iteracyjnymi metodami poszukiwania minimum wypukłych, odcinkowo-liniowych funkcji kryterialnych, utworzonych na podstawie zbioru danych. Są to metody bliskie programowaniu liniowemu, a ich siła bierze się między innymi z wykorzystania specyfiki optymalizowanej funkcji. Podstawowa obserwacja niezbędna do zrozumienia zasady działania algorytmów wskazuje, że hiperpłaszczyzny związane z wektorami cech (przesuniętymi odpowiednio o $+\delta$ lub $-\delta$) rozbijają przestrzeń wag na wypukłe obszary, w których funkcja kryterialna jest liniowa. Obszary te nazywane są obszarami *korekcyjnymi*, a ich wierzchołki wyznaczone są przez punkty przecięcia granicznych hiperpłaszczyzn (N+1 hiperpłaszczyzn w wierzchołku niezdegenerowanym). Z podstawowych twierdzeń programowania liniowego wiemy, że optimum funkcji rozpatrywanego typu zostanie osiągnięte w jednym z wierzchołków obszaru korekcyjnego. Jeśli oprzemy się na powyższym, nasuwa się idea algorytmu polegająca na przemieszczaniu się wzdłuż krawędzi (wyznaczonych przez hiperpłaszczyzny) od jednego wierzchołka do drugiego. Kolejne odwiedzane wierzchołki powinny mieć nie większą wartość funkcji kryterialnej i, o ile nie będziemy odwiedzać ponownie tych samych wierzchołków, mamy zagwarantowane odnalezienie minimum. Poszczególne algorytmy różnią się między sobą sposobem wyboru drogi, po której podążamy, aby osiągnąć optimum. W dalszym opisie skoncentrujemy się na strategii *krawędzi największego spadku*.

Współrzędne każdego wierzchołka $\mathbf{v}(n)$ obszaru korekcyjnego w n-tym kroku algorytmu można wyznaczyć rozwiązując układ N+1 równań liniowych, który w postaci macierzowej wygląda następująco:

$$\mathbf{B}(n)\cdot\mathbf{v}(n)=\boldsymbol{\delta}(n) \tag{11}$$

gdzie $\mathbf{B}(n)$ jest macierzą, zwaną n-tą *bazą*, której wierszami jest N+1 niezależnych wektorów cech $\mathbf{y}^j$, związanych z hiperpłaszczyznami przecinającymi się w wierzchołku $\mathbf{v}(n)$. Składowe wektora marginesów $\boldsymbol{\delta}(n)$ mogą przyjmować odpowiednio wartości $+\delta^j$ lub $-\delta^j$ zgodnie z położeniem hiperpłaszczyzny związanej z danym wektorem cech $\mathbf{y}^j$ w bazie $\mathbf{B}(n)$. Tak więc baza $\mathbf{B}(n)$ oraz wektor $\boldsymbol{\delta}(n)$ jednoznacznie wyznaczają wierzchołek $\mathbf{v}(n)$:

$$\mathbf{v}(n)=\mathbf{B}^{-1}(n)\cdot\boldsymbol{\delta}(n) \tag{12}$$

W każdym kroku algorytmu następuje przejście z $\mathbf{v}(n)$ do następnego wierzchołka $\mathbf{v}(n+1)$, co jest równoznaczne ze zmianą bazy $\mathbf{B}(n)$ na $\mathbf{B}(n+1)$ oraz $\boldsymbol{\delta}(n)$ na $\boldsymbol{\delta}(n+1)$. Jeśli dwie kolejne bazy powstają przez wymianę pojedynczego wektora cech, to do wyliczenia $\mathbf{B}^{-1}(n+1)$ na podstawie $\mathbf{B}^{-1}(n)$ wystarczy zastosować *transformację Gaussa-Jordana.*

Aby wybrać kierunek przechodzenia pomiędzy wierzchołkami obszarów korekcyjnych, wykorzystujemy średni wektor korekcji $\mathbf{k}(\mathbf{v}) = -\nabla\Psi_2(\mathbf{v})$. Gradient funkcji kryterialnej $\nabla\Psi_2(\mathbf{v})$ jest stały w pojedynczym obszarze korekcyjnym i można go łatwo policzyć jako ważoną sumę gradientów składowych funkcji kary, które wynoszą odpowiednio:

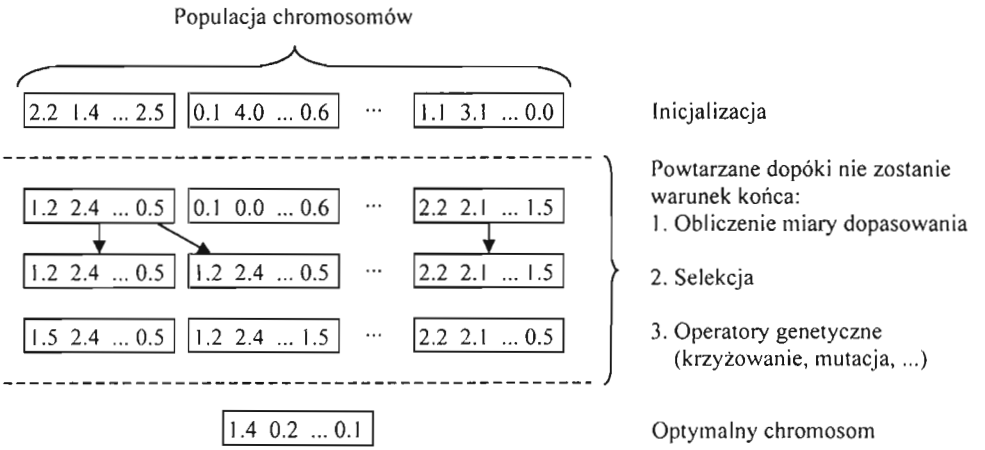
$$\nabla\varphi_j^+(\mathbf{v}) = \begin{cases} \mathbf{y}^j & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle < \delta^j \\ 0 & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle \geq \delta^j \end{cases} \text{ oraz } \nabla\varphi_j^-(\mathbf{v}) = \begin{cases} -\mathbf{y}^j & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle \geq -\delta^j \\ 0 & \text{if } \langle \mathbf{v}, \mathbf{y}^j \rangle < -\delta^j \end{cases} \tag{13}$$

Po obliczeniu rzutów średniego wektora korekcji $\mathbf{k}(\mathbf{v})$ na hiperpłaszczyzny przecinające się w wierzchołku $\mathbf{v}(n)$ i związane z wektorami cech tworzącymi bazę wybierany jest kierunek, w którym ów rzut jest największy. Tym samym wyznaczamy wektor cech opuszczający bazę aktualną (*kryterium wyjścia*). Następnie poruszamy się w tym kierunku aż do osiągnięcia minimum w kolejnym wierzchołku $\mathbf{v}(n+1)$, co jest równoznaczne ze wskazaniem wektora wchodzącego do nowej bazy $\mathbf{B}(n+1)$ (*kryterium wejścia*). Kolejne kroki algorytmu są następnie powtarzane aż do osiągnięcia minimum globalnego (*kryterium stopu*).

4.2.2. Algorytm ewolucyjny

Algorytmy ewolucyjne (AE) są stochastycznymi technikami optymalizacji, które zostały zainspirowane przez proces ewolucji [17]. Ich największą zaletą w stosunku do metod gradientowych jest zdolność unikania optimów lokalnych a także możliwość wykorzystania specyfiki problemu podczas jego optymalizacji (np. poprzez zdefiniowanie specjalnych operatorów genetycznych czy odpowiednią reprezentację). Dzięki temu zostały one z powodzeniem zastosowane w wielu praktycznych zagadnieniach. W pewnym uproszczeniu zasada działania AE polega na utrzymywaniu populacji osobników (zwanych chromosomami), z których każdy koduje jedno rozwiązanie danego problemu. W kolejnych iteracjach algorytmu chromosomy ulegają zmianom wskutek działania operatorów genetycznych, przy czym osobniki lepiej przystosowane mają większą szansę na przeżycie i wydanie potomstwa. Jakość rozwiązań zakodowanych w poszczególnych chromosomach jest oceniana na podstawie funkcji dopasowania (ang. *fitness function*). W sposób

schematyczny AE przedstawiono na rysunku 6. W dalszej części paragrafu skoncentrowano się na specyficznych cechach algorytmu wykorzystanego do minimalizacji funkcji kryterialnej Ψ_1.



Rys. 6. Schemat działania algorytmu ewolucyjnego

Reprezentacja i inicjalizacja. Ponieważ pojedynczy chromosom ma kodować współczynniki hiperpłaszczyzny $H(\mathbf{v})$ najbardziej naturalna jest reprezentacja w postaci $N+1$ liczb rzeczywistych, która bezpośrednio odpowiada rozszerzonemu wektorowi wag $\mathbf{v}$. Do inicjalizacji populacji początkowej wykorzystano następujący prosty algorytm: dla każdego osobnika losujemy dwa wektory cech $\mathbf{x}^i$, $\mathbf{x}^j$ z różnych klas (dipol mieszany) i wybieramy wagi tak, aby $H(\mathbf{v})$ separowała te wektory:

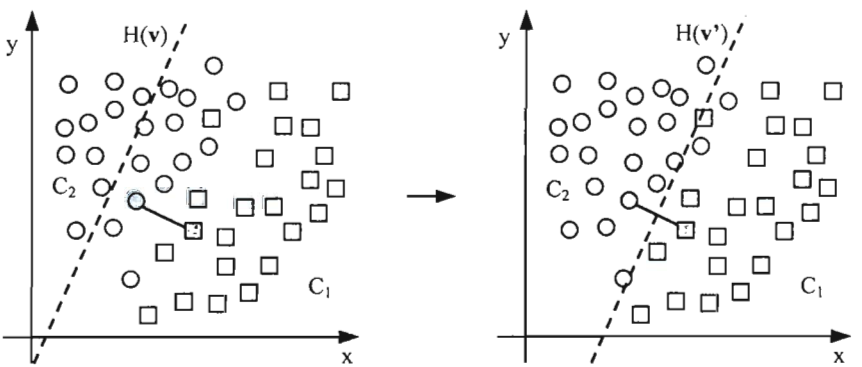
$$\mathbf{w} = \mathbf{x}^i - \mathbf{x}^j = \left[x_1^i - x_1^j, \ldots, x_N^i - x_N^j\right] \text{ oraz } \theta = \frac{1}{2}\left\langle \mathbf{w}, \mathbf{x}^i \right\rangle + \frac{1}{2}\left\langle \mathbf{w}, \mathbf{x}^j \right\rangle. \qquad (14)$$

Tak zdefiniowana hiperpłaszczyzna $H(\mathbf{v})$ leży w połowie odległości pomiędzy wektorami cech tworzącymi dipol $\left(\mathbf{x}^i, \mathbf{x}^j\right)$ i jest prostopadła do prostej przez nie przechodzącej.

Funkcja dopasowania. Algorytm ten służy do minimalizowania funkcji kryterialnej Ψ_1, w związku z czym jest ona wykorzystana do oceny przystosowania osobników.

Selekcja. Jako operator selekcji wykorzystana została proporcjonalna selekcja z liniowym skalowaniem, przy czym chromosom z najlepszą funkcją dopasowania jest zawsze kopiowany do kolejnej populacji (model elitarny).

Operatory genetyczne. Oprócz standardowego operatora krzyżowania arytmetycznego i mutacji wykorzystaliśmy dodatkowy specjalizowany operator genetyczny, który nazwany został operatorem *dipolowym*. Jego działanie jest następujące: losowany jest dipol i sprawdzane jest czy przecina go hiperpłaszczyzna $H(\mathbf{v})$. W zależności od typu dipola i od tego czy przemieszczenie jest w jego przypadku potrzebne dokonywana jest zmiana położenia $H(\mathbf{v})$ poprzez modyfikację pojedynczej wylosowanej cechy. Na rysunku 7 zilustrowano przykładowe działanie operatora w przypadku wylosowania dipola mieszanego, który nie jest w tej sytuacji przecinany. W prezentowanych eksperymentach prawdopodobieństwa wykorzystania operatorów wynosiły odpowiednio: krzyżowanie – 0,8, mutacja – 0,1, dipolowy – 0,5.



Rysunek 7. Przykład działania operatora dipolowego, który dla wylosowanego dipola mieszanego (który nie jest jeszcze przecięty) przemieszcza hiperpłaszczyznę tak, aby go przecięła

Warunek stopu. Algorytm zatrzymuje się, gdy przez pewną określoną liczbę iteracji funkcja dopasowania nie ulega poprawie bądź osiągnięta zostaje z góry założona maksymalna liczba iteracji (w prezentowanych eksperymentach wartości te wynosiły odpowiednio: 200 i 1000 iteracji).

5. Wielowymiarowe drzewa decyzyjne

Strategia generowania wielowymiarowego drzewa decyzyjnego wykorzystana w prezentowanym systemie nie odbiega od standardowo stosowanej rekurencyjnej metody „top-down" (opisana została w paragrafie 2), przy czym liniowe reguły decyzyjne w poszczególnych węzłach drzewa uzyskiwane są w wyniku zastosowania procedur opisanych w poprzednim paragrafie. Pozostałe kwestie związane z indukcją drzewa, takie jak selekcja cech oraz unikanie przetrenowania, zostaną przedstawione poniżej.

5.1. Selekcja cech

Użyteczność klasyfikatora zależy często nie tylko od jego skuteczności (mierzonej jakością klasyfikacji), ale również od możliwości zrozumienia jego działania i interpretacji sposobu proponowania decyzji. W takim przypadku dążenie do uproszczenia drzewa decyzyjnego jest jak najbardziej uzasadnione. Co więcej, w wielu sytuacjach, poprzez usunięcie z testów cech zaszumionych bądź nadmiarowych, uzyskujemy rozwiązania nie tylko prostsze, ale również bardziej poprawne. Dlatego też prawie wszystkie metody poszukiwania wielowymiarowych drzew decyzyjnych zostały wyposażone w mechanizm selekcji cech. Zwykle w każdym węźle dobór najbardziej użytecznych cech jest przeprowadzany niezależnie od pozostałych części drzewa[4]. W najbardziej optymistycznym przypadku może to prowadzić do testu wykorzystującego tylko pojedynczą cechę. Z drugiej strony musimy zdawać sobie sprawę, że problem wyboru optymalnego podzbioru atrybutów jest bardzo złożony (liczba możliwych podzbiorów rośnie wykładniczo wraz ze wzrostem liczby cech). W związku z tym stosowane w praktycznych zastosowaniach algorytmy selekcji cech są metodami heurystycznymi. W pracy [10] został przedstawiony przegląd metod selekcji cech w kontekście wielowymiarowych drzew decyzyjnych. Zdaniem autorów najlepsze wyniki można uzyskać stosując zaproponowaną przez nich metodę *Heuristic Sequential Search*. Jest to kombinacja dwu powszechnie znanych sekwencyjnych metod: wyboru (*Forward Selection*) i eliminacji (*Backward Elimination*). Działa ona w ten sposób, że początkowo znajduje najlepsze testy składające się ze wszystkich cech oraz z pojedynczej cechy. Następnie porównuje je i, w zależności od wyniku, uruchamia standardową procedurę sekwencyjnego wyboru lub eliminacji.

Stosowanie w bezpośredni sposób powyższej metody okazało się nie być zbyt efektywne (w sensie złożoności czasowej), zwłaszcza w przypadku danych z dużą

[4] W niektórych bardziej specyficznych zastosowaniach (np. diagnostyce medycznej), gdzie poszczególne cechy mogą mieć przypisany (często bardzo różny) koszt wykorzystywane są podejścia oparte na powtórnym użyciu cech występujących w wyższych partiach drzewa.

ilością cech [7]. Dlatego też opracowane zostały metody, w których selekcja cech została wbudowana w proces minimalizacji funkcji kryterialnej.

W przypadku algorytmu wymiany rozwiązań bazowych zaproponowane podejście polega na modyfikacji kryterium wyjścia (służącego do wyboru wektora cech, opuszczającego bazę) i krokowym uruchamianiu algorytmu. Należy przy tym wyjaśnić, że w momencie startu algorytmu wszystkie wagi są wyzerowane, w bazie znajdują się „sztuczne" wektory jednostkowe, a wprowadzenie wektora cech do bazy w wierszu k-tym równoznaczne jest z wykorzystaniem k-tej cechy. W pierwszym kroku wprowadzane są wektory cech tylko do dwu wierszy bazy (przy czym jeden odpowiada progowi $-\theta$) i jest to równoznaczne ze znalezieniem testu z pojedynczą cechą. Wybór wektorów cech opuszczających bazę odbywa się standardowo na podstawie wartości rzutu wektora korekcji. W kolejnych krokach dopuszczone jest wykorzystanie dodatkowo jednej cechy (jednego wiersza w bazie) i następuje sekwencja minimalizacji, w której wektory cech mogą opuszczać bazę tylko z tych miejsc. Każdorazowo po zakończeniu sekwencji minimalizacji otrzymujemy informację o wartości funkcji kryterialnej (bądź ilości przeciętych dipoli) odpowiadających rozpatrywanej liczbie cech. Dzięki takiemu mechanizmowi na podstawie pojedynczego przebiegu algorytmu wymiany rozwiązań bazowych możemy dokonać wyboru optymalnej liczby atrybutów tworzących hiperpłaszczyznę.

Jeśli chodzi o algorytm ewolucyjny, to minimalny zestaw zmian obejmuje przede wszystkim modyfikację funkcji dopasowania, która w przypadku wykorzystywania selekcji cech wygląda następująco:

$$Fitness(\mathbf{v}) = \Psi_1(\mathbf{v}) \cdot \left[(1-\alpha) + \alpha \frac{n}{N} \right] \tag{15}$$

gdzie:

n – liczba cech wykorzystana w teście,

α – parametr regulujący wpływ złożoności testu na jego ogólną ocenę ($\alpha \in 0..1$).

Tak zdefiniowana funkcja umożliwia premiowanie rozwiązań prostszych, a poprzez wartości α użytkownik ma możliwość wpływania na dążenie algorytmu do ograniczania liczby cech wykorzystanych w testach (we wszystkich eksperymentach $\alpha = 0{,}2$). Oprócz zmiany funkcji dopasowania niezbędne jest również ułatwienie usuwania cech z testu, zwłaszcza w przypadku zastosowanej reprezentacji rzeczywistej. Osiągnięte to zostało poprzez zmodyfikowanie mutacji, tak aby znacząco zwiększyć prawdopodobieństwo przypisania współczynnikom wartości 0, odpowiadającej wykluczeniu cechy z kombinacji liniowej.

Innym problem, który związany jest bezpośrednio z selekcją cech, jest zapewnienie odpowiedniej liczby wektorów cech w stosunku do liczby atrybutów [12]. Problem ten jest szczególnie istotny w dolnych partiach drzewa decyzyjnego („bli-

sko liści"), gdzie wskutek wcześniejszych podziałów liczba dostępnych danych uczących w węźle ulega znacznej redukcji. Jeżeli liczba obiektów jest zbyt mała w stosunku do rozpatrywanej liczby cech, to nie możemy jednoznacznie wskazać optymalnej hiperpłaszczyzny, gdyż istnieje wiele równoważnych orientacji. Sytuacja ta może być określona mianem braku możliwości dopasowania do danych uczących (ang. *underfitting*). Aby jej uniknąć, należy ograniczyć rozpatrywaną liczbę cech w teście, tak aby liczba obiektów stanowiła wielokrotność liczby atrybutów (w prezentowanych eksperymentach przyjęto, że liczba wektorów cech musi być co najmniej 5-krotną wielokrotnością liczby cech).

5.2. Unikanie przetrenowania

W przypadku generowania klasyfikatorów na podstawie zbiorów danych, niezależnie od ich formy (dotyczy to w równym stopniu drzew decyzyjnych, zbiorów reguł decyzyjnych czy też sieci neuronowych), mamy często do czynienia z problemem unikania zbytniego dopasowania do danych (ang. *over-fitting*). Zwykle klasyfikator bardzo dobrze rozpoznaje przypadki wykorzystywane do uczenia, co niestety nie znaczy, że równie dobrze będzie sobie radził z danymi, których wcześniej nie poznał. Problem ten ma szczególne znaczenie podczas przetwarzania rzeczywistych, zwykle zaszumionych, zbiorów danych.

W odniesieniu do drzew decyzyjnych podstawowym parametrem kontrolującym ich zdolności generalizacyjne jest rozmiar (ilość węzłów i liści). Aby zapewnić odpowiedni rozmiar drzewa, stosowane są powszechnie dwie techniki [9]:

- przerwanie rekurencyjnego podziału obiektów w węzłach przed osiągnięciem perfekcyjnej klasyfikacji na zbiorze uczącym. Zdefiniowanie optymalnego kryterium stopu okazuje się jednak zagadnieniem bardzo trudnym.
- dodatkowe przycinanie (ang. *post-pruning*) drzewa uruchamiane po zakończeniu jego indukcji. Metoda ta jest oczywiście bardziej czasochłonna, tym niemniej eliminuje niepewność związaną ze zbyt wczesnym zatrzymaniem rekurencyjnego podziału i umożliwia dokładną analizę uzyskanego drzewa. Główna idea podcinania polega na zastępowaniu wybranego poddrzewa poprzez liść (lub ewentualnie jedną z gałęzi), o ile klasyfikuje on lepiej niż całe poddrzewo. Algorytmy przycinania rozpoczynają swoje działanie od najniższych partii drzewa, a następnie przesuwają w stronę korzenia, próbując zredukować coraz większe poddrzewa. Poszczególne algorytmy różnią się od siebe głównie sposobem wyznaczania jakości klasyfikacji rozpatrywanych części drzewa, przy czym warto zaznaczyć, że musi być ona określona inaczej niż poprzez reklasyfikację. W najprostszym przypadku wydziela się z danych uczących część wektorów cech, które nie są wykorzystywane podczas tworzenia drzewa, natomiast służą do oceny poddrzew w czasie przycinania. Metoda ta-

ka ma jedną podstawową wadę: ogranicza ilość danych służących do indukcji drzewa.

Przeprowadzono wiele badań eksperymentalnych, które jednoznacznie pokazują, że druga wymieniona powyżej technika daje lepsze rezultaty. Tymczasem najczęściej obie techniki są stosowane równocześnie. W prezentowanym systemie wykorzystana jest bardzo prosta reguła stopu – jeżeli liczba obiektów w węźle jest mniejsza niż M_{st}, to węzeł automatycznie staje się liściem bez względu na przynależność obiektów do klas (we wszystkich eksperymentach prezentowanych w pracy $M_{st} = 5$). W systemie zaimplementowano kilka spośród wielu algorytmów przycinania [13], tym niemniej we wszystkich eksperymentach wykorzystano zmodyfikowaną metodę „pesymistyczną", zaproponowaną przez Quinlana w systemie C4.5 [21]. Jest to metoda heurystyczna, która wyróżnia się tym, że nie wymaga podziału zbioru uczącego na części wykorzystywane do indukcji i walidacji.

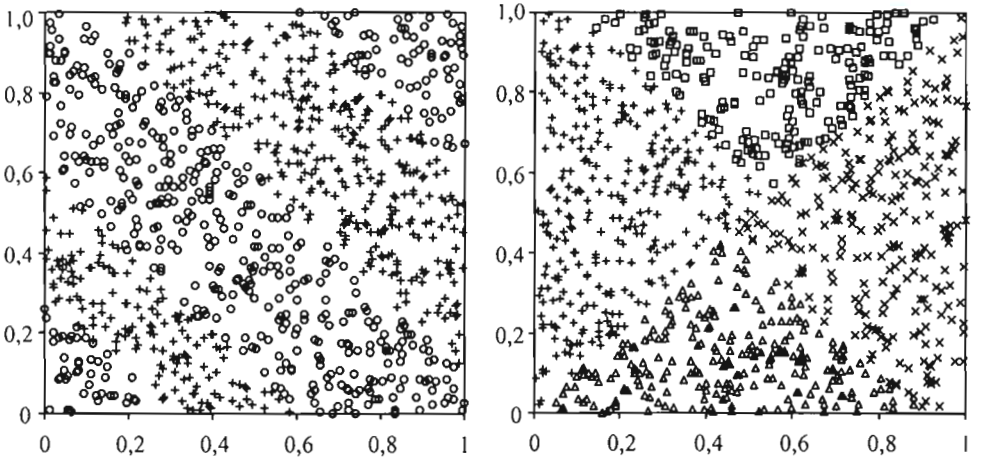
6. Wyniki eksperymentalne

Aby sprawdzić, na ile zaproponowane algorytmy spełniają swoje zadanie, przeprowadzono dwie serie eksperymentów. W pierwszej operowano na sztucznie wygenerowanych zbiorach danych, dla których jednoznacznie zdefiniowano podział na klasy i dzięki temu znane jest rozwiązanie optymalne. Druga seria eksperymentów została przeprowadzona na rzeczywistych zbiorach danych, obejmujących szeroki zakres problemów praktycznych pochodzących z repozytorium Uniwersytetu Kalifornijskiego w Irwine [1]. Zbiory tam zgromadzone są często wykorzystywane przy porównywaniu algorytmów tworzących wszelkiego typu klasyfikatory. Z jednego z takich porównań [16] zaczerpnięte zostały wyniki uzyskane przez inne systemy (C4.5 i OC1).

Wszystkie prezentowane w tej pracy rezultaty uzyskano przy wykorzystaniu bądź powtórzonej 5-krotnie kroswalidacji 10-przedziałowej, bądź zbioru testowego (o ile był dostępny). Złożoność drzew mierzona jest ilością liści, natomiast podawane czasy dotyczą pojedynczego uruchomienia algorytmów indukcji drzew przy wykorzystaniu całego zbioru uczącego. Obliczenia wykonano na komputerze klasy PC (PII 350MHz). W prezentowanych poniżej tabelach skrót MDT-EA odnosi się do prostszej metody wykorzystującej algorytm ewolucyjny do minimalizacji funkcji kryterialnej, natomiast MDT-BEA odpowiada metodzie opartej na algorytmie wymiany rozwiązań bazowych.

6.1. Zbiory syntetyczne

Wszystkie zbiory analizowane w tym podpunkcie zbudowane są z 2000 wektorów cech należących do 2 klas. Cechy przyjmują wartości rzeczywiste z przedziału [0,1]. Dwa pierwsze zbiory prezentują analogiczny liniowo separowalny przypadek (stąd ich nazwa – LS), a różnią się liczbą wykorzystanych zmiennych (odpowiednio 2 i 10 atrybutów). Hiperpłaszczyzny decyzyjne rozdzielające klasy są zdefiniowane następująco: $x_1 < x_2$ oraz $x_1 + x_2 + x_3 + x_4 + x_5 < x_6 + x_7 + x_8 + x_9 + x_{10}$. Dwa kolejne zbiory są 2-wymiarowe i zostały przedstawione na rysunku 8. Analogiczne testy na syntetycznych zbiorach danych przeprowadzone zostały w pracy [18]. Oryginalne zbiory nie są, niestety publicznie dostępne, stąd bezpośrednie porównanie uzyskanych wyników nie jest możliwe.



Rys. 8. Wykresy zbiorów syntetycznych: „Zebra" i „Szachownica"

Tabela 2

Porównanie na syntetycznych zbiorach danych

Zbiór danych	MDT-EA			MDT-BEA		
	Jakość kl.[%]	Złożoność	Czas [s]	Jakość kl. [%]	Złożoność	Czas [s]
LS2	99,85	2	10,5	99,85	2	8,5
LS10	98,30	8	19,8	99,55	2	180
„Zebra"	99,35	6	23,2	99,35	9	25,2
„Szachownica"	99,55	6	16,9	99,25	13	36,0

W przypadku pierwszego zbioru (LS2) oba algorytmy poradziły sobie bez trudu, znajdując optymalne rozwiązanie. Przeniesienie problemu na większą liczbę wymiarów pokazało, że zagadnienie to przestało być trywialne. Prostsza metoda

oparta na algorytmie ewolucyjnym, odnalazła podstawowy podział, tym niemniej pojawiły się problemy z dokładnym dopasowaniem i stąd spory wzrost złożoności. Podobnie zachowuje się OC1 (na podstawie [18]), który uzyskał 97,2% przy 14 liściach. Jeśli chodzi o dwa kolejne zbiory, to pierwszy algorytm odnalazł rozwiązania prawie optymalne, natomiast drugi, zgodnie z oczekiwaniami, miał nieco większe problemy. Wpływ dipoli mieszanych utworzonych z dalekich dipoli spowodował przesunięcia hiperpłaszczyzn i w związku z tym nie udało się osiągnąć optymalnej złożoności (odpowiednio 5 i 4 liście), tym niemniej jakość klasyfikacji jest bardzo wysoka.

6.2. Zbiory rzeczywiste

Tabela 4 zawiera charakterystykę rzeczywistych zbiorów danych wykorzystanych w drugiej serii eksperymentów, natomiast tabele 5 i 6 przedstawiają wyniki uzyskane przez poszczególne algorytmy (odpowiednio jakość klasyfikacji i złożoność drzew w 5 i czas obliczeń w 6). Czasy obliczeń podane są wyłącznie dla algorytmów „dipolowych", gdyż nie dysponujemy analogicznymi danymi dla pozostałych algorytmów.

Tabela 4

Charakterystyka rzeczywistych zbiorów danych

Zbiór danych	Liczba obiektów (część testowa)	Liczba cech	Liczba klas
Breast (Wisconsin)	683	9	2
CMC	1473	9	3
Heart (Cleveland)	270	13	2
Pima	532	7	2
Segmentation	2310	19	7
Smoking	1855(1000)	8	3
Vehicle	846	18	4
Waveform	600(3000)	21	3

Tabela 5

Jakość klasyfikacji i złożoności drzew uzyskane przez poszczególne algorytmy

Zbiór danych	Jakość klasyfikacji [%]				Złożoność drzew [liczba liści]			
	MDT-EA	MDT-BEA	C4.5	OC1	MDT-EA	MDT-BEA	C4.5	OC1
Breast	95,2	96,6	95,8	95,9	11	4	11	5
CMC	49,0	53,0	51,7	52,2	243	129	143	8
Heart	78,2	80,6	80,4	77,8	20	10	23	3
Pima	72,4	74,7	75,8	75,3	49	14	18	5
Segmentation	95,1	96,4	96,8	94,3	51	28	42	26
Smoking	60,5	71,1	69,5	69,5	122	23	1	2
Vehicle	66,9	76,2	72,3	68,4	88	36	65	41
Waveform	77,5	80,0	73,9	78,2	39	25	54	15

Uzyskane wyniki pokazują, że metoda generowania drzewa oparta na algorytmie wymiany rozwiązań bazowych (MDT-BEA) poradziła sobie ze zbiorami rzeczywistymi najlepiej spośród analizowanych systemów, uzyskując najwyższą jakość klasyfikacji w 6 na 8 zbiorów. Wiązało się to jednak ze wzrostem złożoności tworzonych klasyfikatorów. Interesujące jest również, że w porównaniu do poprzednio stosowanej selekcji cech [7] udało się ograniczyć czas obliczeń od 3 do 5 razy w zależności od zbioru danych, przy zachowaniu bardzo zbliżonych wyników w sensie jakości klasyfikacji. Zgodnie z oczekiwaniami, prostsza z metod dipolowych, oparta na algorytmie ewolucyjnym (MDT-EA) poradziła sobie tym razem gorzej. Zarówno jakość klasyfikacji jak i złożoność uzyskiwanych w ten sposób drzew decyzyjnych ustępują pozostałym systemom.

Tabela 6

Czas potrzebny do wygenerowania wielowymiarowego drzewa decyzyjnego na podstawie całego zbioru uczącego (w sekundach)

Zbiór danych	MDT-EA	MDT-BEA
Breast	28	10
CMC	192	82
Heart	29	8
Pima	38	16
Segmentation	453	697
Smoking	175	70
Vehicle	197	115
Waveform	98	66

7. Podsumowanie i planowane badania

W pracy przedstawiono algorytmy generowania wielowymiarowych drzew decyzyjnych przy wykorzystaniu dipolowych funkcji kryterialnych. Opierając się na dotychczasowych wynikach badań można stwierdzić, że zaproponowana metoda (wykorzystująca algorytm wymiany rozwiazań bazowych) charakteryzuje się jakością działania co najmniej porównywalną z najbardziej znanymi systemami tworzącymi klasyfikatory tego typu. Należy jednak zauważyć, że istnieje jeszcze cały szereg możliwych usprawnień, które zdaniem autorów powinny wpłynąć zarówno na poprawę jakości klasyfikacji jak i na zmniejszenie złożoności klasyfikatora. Przykładowo możliwe jest zaproponowanie nowych strategii wyceny dipoli a także, co się z tym bezpośrednio wiąże, wykorzystanie kryterium rangowego alternatywnie, obok kryterium dipolowego, w tym samym węźle. Rozważane jest wprowadzenie dodatkowego kroku polegającego na lokalnym dopasowaniu uzy-

skanej hiperpłaszczyzny, mające na celu wyeliminowanie możliwego negatywnego wpływu dalekich wektorów cech. Planowane jest również opracowanie wersji rozproszonej algorytmu, która powinna umożliwić przetwarzanie znacznie większych zbiorów danych w sensownym czasie.

Przedstawione w pracy metody mogą być stosowane w różnorodnych zagadnieniach praktycznych. Obecnie trwają prace nad ich zastosowaniem w rozpoznawaniu zmian patologicznych wątroby na obrazach tomografii komputerowej, gdzie jako cechy wykorzystywane są charakterystyki teksturalne, obliczone dla analizowanych obszarów.

Referencje:

[1] Blake, C., Merz, C., (1998) *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html] Irvine, CA: University of California, Department of Information and Computer Science.

[2] Bobrowski, L., (1991) *Design of piecewise linear classifiers from formal neurons by some basis exchange technique*, Pattern Recognition, 24(9): 862-870.

[3] Bobrowski, L., (1996) *Piecewise-linear classifiers, formal neurons and separability of the learning sets*, In Proc. of Int. Conf. on Pattern Recognition ICPR'96 (Vienna), 224-228.

[4] Bobrowski, L., (1999) *Data mining procedures related to the dipolar criterion function*, Applied Stochastic Models and Data Analysis – Quantitative Methods in Business and Industry Society, INE (Lisboa), 43-50.

[5] Bobrowski, L., (2000) *Strategie projektowania sieci neuropodobnych*, Biocybernetyka i Inżynieria Biomedyczna 2000, Tom 6, Akademicka Oficyna Wydawnicza EXIT, 295-321.

[6] Bobrowski, L., Krętowska, M., Krętowski, M., (1998) *Generowanie sieci neuropodobnych oraz drzew decyzyjnych w oparciu o kryteria dipolowe*, Symulacja w badaniach i rozwoju, 17-28.

[7] Bobrowski, L., Krętowski M., (2000) *Induction of Multivariate Decision Trees by using Dipolar Criteria*, Principles of Data Mining and Knowledge Discovery. 4th European Conference, PKDD'2000. Springer LNCS 1910, 331-336.

[8] Bobrowski, L., Niemiro, W., (1984) *A method of synthesis of linear discriminant function in the case of nonseparability*, Pattern Recognition, 17: 205-210.

[9] Breiman, L., Friedman, J., Olshen, R., Stone C., (1984) *Classification and Regression Trees*, Wadsworth International Group.

[10] Brodley, C., Utgoff, P., (1995) *Multivariate decision trees*, Machine Learning, 19: 45-77.

[11] Chai, B., Huang, T., Zhuang, X., Zhao, Y., Sklansky, J., (1996) *Piecewise-linear classifiers using binary tree structure and genetic algorithm*, Pattern Recognition, 29(11): 1905-1917.
[12] Duda, O.R., Heart, P.E., Stork D.G., (2001) *Pattern Classification*, Wydanie drugie, zmienione, John Wiley & Sons.
[13] Esposito, F., Malerba, D., Semeraro, G., (1997) *A comparative analysis of methods for pruning decision trees*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(5): 476-491.
[14] Gama, J., Brazdil, P., (1999) *Linear tree*, Inteligent Data Analysis, 3(1): 1-22.
[15] Kass, G. V., (1980) *An exploratory technique for investigating large quantities of categorical data*, Applied Statistics, 29(2): 119-127.
[16] Lim, T., Loh, W., Shih, Y., (2000) *A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms*, Machine Learning 40(3): 203-228.
[17] Michalewicz, Z., (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer.
[18] Murthy, S., Kasif, S., Salzberg, S., (1994) *A system for induction of oblique decision trees*, Journal of Artificial Intelligence Research, 2: 1-33.
[19] Murthy, S., (1998) *Automatic construction of decision trees from data: A multi-disciplinary survey*, Data Mining and Knowledge Discovery, 2:345-389.
[20] Shah, S., Sastry, P.S., (1999) *New algorithms for learning and pruning oblique decision trees*, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 29(4): 494-505.
[21] Quinlan, J., (1986) *Induction of decision trees.* Machine Learning 1(1): 81-106.
[22] Quinlan, J., (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers.
[23] Utgoff, P., Brodley, C., (1991) *Linear machine decision trees*, (COINS Technical report 91-10), Amherst, MA: University of Massachusetts, Department of Computer and Information Science.

INDUCTION OF MULTIVARIATE DECISION TREES FROM DATASETS

Abstract: In the paper, new methods for induction of classifiers in the form of multivariate (oblique) decision trees are presented. A linear decision function is used at each non-terminal node of the binary tree for splitting the data. The strategies for finding the decision hyper-plane are based on the concept of dipoles and as optimization procedures they use evolutionary algorithms or basis exchange algorithms. To eliminate redundant and noisy features we embedded into the process of searching for the optimal linear combination in each node the feature selection mechanism. To avoid over-fitting and to increase generalization the tree is pruned back after the growing phase. The presented methods

were experimentally verified on publicly available datasets and compared with other decision tree systems.

Key words: multivariate (oblique) decision trees, linear decision function, dipolar criteria

Artykuł zrealizowano w ramach pracy badawczej W/II/1/00.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Walenty Oniszczuk[1], Cezary Zwolski[2]

MODELE SYMULACYJNE I ANALIZA PRZECIĄŻEŃ PRZEŁĄCZNIKÓW SIECIOWYCH W SIECIACH ATM

Streszczenie: w pracy przedstawiono modele symulacyjne kilku typów przełączników sieciowych (multiplekserów), zbudowane z wykorzystaniem języka obiektowo zorientowanego MODSIM. Następnie opisano całą serię eksperymentów symulacyjnych, wyniki których posłużyły do analizy przeciążeń występujących w sieci, jak również do obliczania parametrów charakteryzujących jakość obsługi pakietów (ang. Quality of Service (QoS) requirements) w nowoczesnych sieciach komputerowych zbudowanych w technologii ATM (ang. Asynchronous Transfer Mode) i charakteryzujących się różnorodnością i wysoką szybkością przesyłanych informacji. W sieciach tego typu z reguły występują niejednorodne, mające zmienne natężenie, strumienie pakietów, co poważnie utrudnia (lub uniemożliwia) ich modelowanie analitycznymi metodami teorii kolejek, więc modelowanie symulacyjne jest często jedynym sposobem analizy pracy wybranych fragmentów takich sieci.

Słowa kluczowe: symulacja, MODSIM, modele przełączników sieciowych, ATM

1. Wprowadzenie

W dobie szybkiego rozwoju technologii i wzrostu intensywności przepływu informacji, jakość usług świadczonych za pomocą sieci komputerowych nabiera szczególnego znaczenia [4], [5], [6]. Najszersze perspektywy rozwoju stoją przed sieciami typu ATM (ang. Asynchronous Transfer Mode) i Gigabit Ethernetem. Sieci tego typu potrafią przeprowadzić transmisję danych przy najlepszych parametrach jakościowych, a w szczególności służą one do transmisji audiowizualnych w czasie rzeczywistym [3], [7].

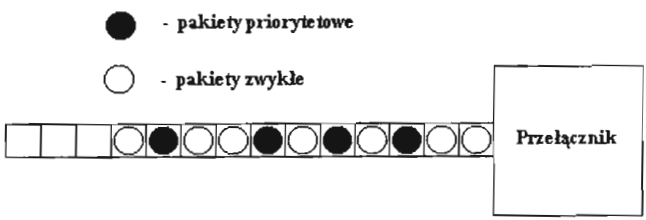
[1] Katedra Systemów Czasu Rzeczywistego, Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45a, 15-351 Białystok
[2] Dyplomant Instytutu Informatyki Politechniki Białostockiej

Istotnym czynnikiem ograniczającym niepohamowany wzrost przepustowości takich sieci jest konieczność buforowania pakietów w jej węzłach i związane z tym problemy [3], [5]. W niniejszej pracy przedstawiono i porównano niektóre szeroko stosowane przełączniki z różnymi algorytmami szeregowania, włączając w to priorytetowanie pakietów. W tym celu zbudowano szereg modeli symulacyjnych w języku obiektowo zorientowanym MODSIM [1], [2], które posłużyły do serii eksperymentów, będących podstawą analizy pracy poszczególnych przełączników sieciowych, przy zmiennej intensywności strumieni napływających pakietów.

2. Modele przełączników sieciowych [3]

2.1 Model przełącznika z naturalnym regulaminem szeregowania

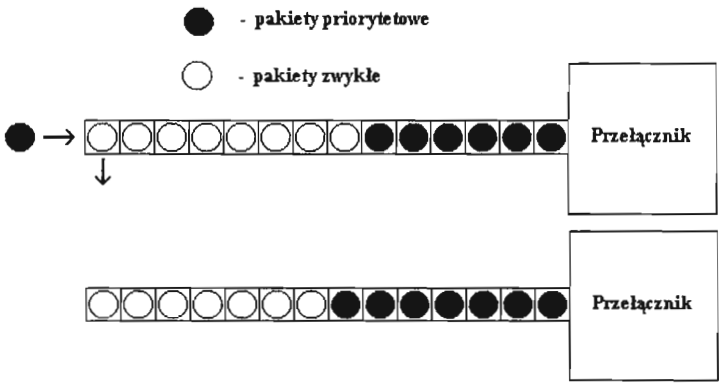
Jest to najprostszy algorytm działania przełącznika (naturalny regulamin szeregowania) [3]. Działa on na zasadzie kolejki FCFS (ang. First Come First Served), czyli pakiety obsługiwane są zgodnie z kolejnością z jaką przychodzą do bufora (rys. 1). Priorytet nie ma tu znaczenia, bo zgłoszenia traktowane są równorzędnie. Jest to najstarszy algorytm, nieposiadający mechanizmów wpływania na przeciążenia sieci.



Rys. 1. Model przełącznika z regulaminem naturalnym

Modyfikacją przełączników z naturalnym regulaminem szeregowania są przełączniki priorytetowe, w których priorytet nie służy jedynie do podjęcia decyzji o umieszczeniu pakietu w kolejce, lecz wpływa także na kolejność szeregowania pakietów. W pierwszej kolejności obsługiwane są pakiety z wyższym priorytetem.

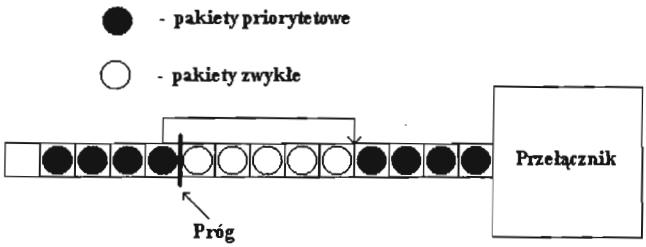
2.2 Model przełącznika z wypychająco-priorytetowym regulaminem szeregowania



Rys. 2. Model przełącznika z regulaminem wypychająco-priorytetowym

Jest to szeroko stosowany regulamin szeregowania pakietów w buforach przełączników sieciowych, nazywany często przestrzennym algorytmem z wypychaniem (od angielskiego push-out space algorithm) (rys. 2). Ogólna zasada działania tego algorytmu oparta jest na następującej regule: nadchodzący pakiet priorytetowy, gdy nie zastaje wolnego miejsca w kolejce oczekujących zgłoszeń, a są w tej kolejce pakiety niższego priorytetu, wyszukuje jeden z nich o najniższym priorytecie i zajmuje jego miejsce. Jeśli przyjąć, że wielkości k i l oznaczają odpowiednio liczbę pakietów zwykłych i priorytetowych, a n – pojemność kolejki, to, dopóki suma pakietów obu klas nie przekracza pojemności bufora $(k+l < n)$, pakiety są obsługiwane zgodnie z regulaminem priorytetowym. Kiedy wszystkie miejsca w kolejce są już zajęte $(k+l = n)$, nadchodzące zgłoszenia o niższym priorytecie są tracone, a zgłoszenie o wyższym priorytecie wyszukują pakiety zwykłe (począwszy od ostatniego) i usuwają je, robiąc w ten sposób miejsce dla siebie. Wchodzący pakiet ustawiany jest przed zgłoszeniami normalnymi. Pakiety priorytetowe są tracone tylko wtedy, gdy wszystkie miejsca w buforze są zajęte przez zgłoszenia ich klasy.

2.3. Model przełącznika z progowo-priorytetowym regulaminem szeregowania



Rys. 3. Model przełącznika z regulaminem progowo-priorytetowym

Innym, często stosowanym w przełącznikach algorytmem szeregowania jest regulamin progowy (rys. 3), w którym w buforze występuje bariera (próg), nieprzekraczalna dla pakietów zwykłych. Jeśli przyjmujemy, że wielkości k i l są odpowiednio liczbą pakietów zwykłych i priorytetowych w kolejce, a n – jest pojemnością bufora i oznaczymy przez N próg, przy czym N musi być nie większe niż n, to, dopóki suma pakietów zwykłych i priorytetowych nie osiągnie granicy jaką stanowi próg $(k+l < N)$, zgłoszenia są obsługiwane zgodnie z regulaminem priorytetowym. Po osiągnięciu progu $(k+l = N)$ nowo nadchodzące pakiety o niższym priorytecie są tracone, a do kolejki przyjmuje się wyłącznie zgłoszenia priorytetowe. Strzałka na modelu wskazuje, że nowo przechodzące pakiety ustawiają się przed zgłoszeniami zwykłymi. Proces gubienia pakietów o wyższym priorytecie następuje, gdy bufor jest już pełny i nie ma miejsc na nadchodzące pakiety $(k+l = n)$.

3. Eksperymenty symulacyjne z modelami przełączników sieciowych

Aby prześledzić mechanizmy działania przełączników sieciowych oraz określić parametry jakości obsługi (QoS), zbudowano ich modele w języku MODSIM [1], [2] i przeprowadzono całą gamę eksperymentów symulacyjnych dla niestandardowych i zmiennych w czasie intensywności strumieni pakietów.

Programowa realizacja generatora pakietów została pomyślana tak aby między dłuższymi okresami niższej niż przeciętna intensywności następowały okresy skokowego jej wzrostu, co dość dokładnie odzwierciedla natężenie ruchu w rzeczywistych sieciach (dane wejściowe generatora: parametry związane z intensywnością pakietów zwykłych i priorytetowych). Aby porównać różnorakie mechani-

zmy szeregowania kolejek do przełącznika, w eksperymentach nie zmieniano danych wejściowych, które dobrano w następujący sposób:

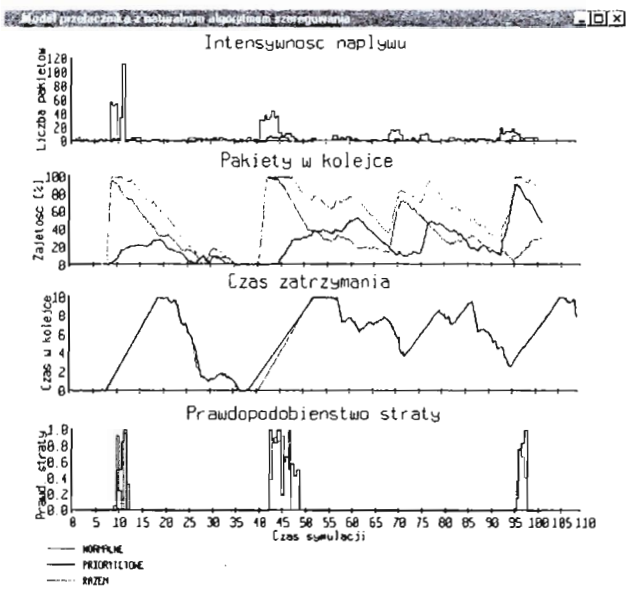
```
Czas symulacji:                          100.0 jednostek czasu
Średni czas obsługi:                     0.10 jednostki czasu
Rozmiar kolejki:                         100
Par. regulujący
intensywność pakietów normalnych:        0.60 jednostki czasu
Par. regulujący
intensywność pakietów priorytetowych:    0.90 jednostki czasu
```

Powyższe dane wejściowe zostały wybrane w trakcie serii badań oraz analiz rezultatów uzyskanych za pomocą modeli symulacyjnych i dość trafnie odzwierciedlają one procesy zachodzące w przełącznikach.

W trakcie każdego przebiegu eksperymentu symulacyjnego zbierano statystyki dotyczące podstawowych parametrów pracy systemu, takich jak:

- *intensywność napływu* – liczba wygenerowanych pakietów w danym przedziale czasowym,
- *pakiety w kolejce* – liczba pakietów w kolejce w danej chwili,
- *czas zatrzymania* – czas pobytu w kolejce danego pakietu,
- *prawdopodobieństwo straty* – stosunek liczby pakietów straconych do wygenerowanych, (w zadanym przedziale czasowym).

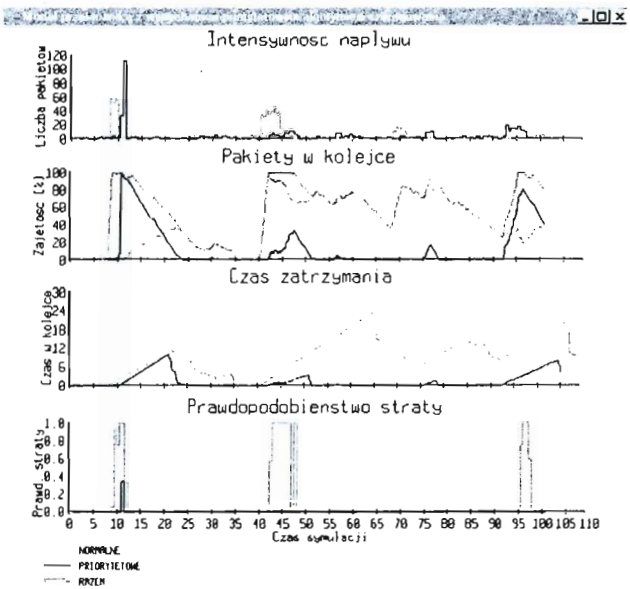
Wyniki eksperymentów pokazane zostały na rysunkach 4–6.



Rys. 4. Wyniki symulacji dla regulaminu naturalnego

Pierwszy diagram na tych rysunkach przedstawia zmienną w czasie intensywność strumienia pakietów. Na początku symulacji występowała mała intensywność napływu zarówno pakietów zwykłych jak i priorytetowych. Później nastąpił gwałtowny, krótkoterminowy napływ pakietów normalnych. Wartość szczytowa w tym okresie była bliska sześćdziesięciu. Zaraz potem wystąpił krótkotrwały, ale lawinowy napływ pakietów priorytetowych, który osiągnął apogeum przy wartości ponad 110 pakietów, co oznaczało ponad dwudziestokrotne przekroczenie norm gwarantujących płynność obsługi. Po okresie wysokiej aktywności generatora nastąpił czas „ciszy", itd.

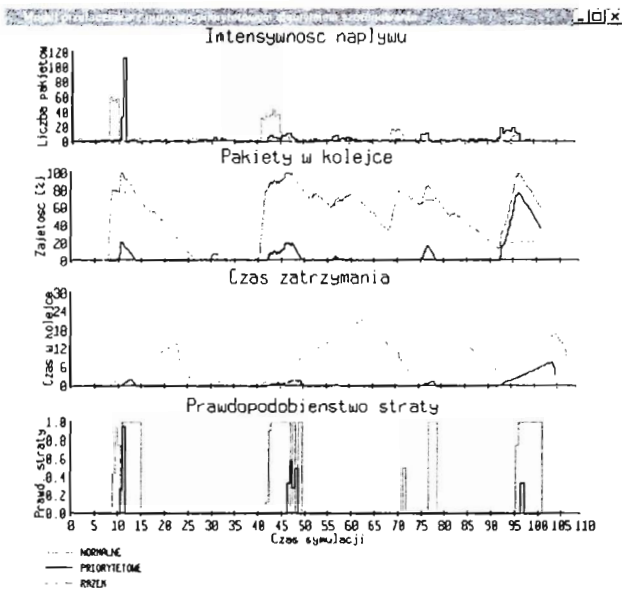
Opisane wyżej fluktuacje intensywności napływu pakietów miały decydujący wpływ na pozostałe parametry, charakteryzujące pracę sieci. Wykresy nazwane „pakiety w kolejce" charakteryzują stopień zapełnienia buforów przez pakiety oczekujące na obsługę.



Rys. 5. Wyniki symulacji dla algorytmu wypychająco-priorytetowego

Ogólna liczba pakietów w kolejce dla algorytmu naturalnego oraz wypychająco-priorytetowego jest taka sama ze względu na to, że w obu przypadkach wszystkie pakiety są przyjmowane do bufora, jeśli nie jest on pełny. Co prawda mechanizm wypychania powoduje, że w razie zapełnienia kolejki zgłoszenia priorytetowe mają szansę na wejście do systemu, ale zawsze jeden pakiet uprzywilejowany zastępuje jeden zwykły; suma zatem pozostaje bez zmian. Inaczej jest z regulaminem progowo-priorytetowym. Wartość progu stanowi dodatkową barierę, której

nie mogą przekroczyć pakiety normalne i dzięki temu zgłoszenia priorytetowe mają dla siebie więcej miejsc w buforze. Sprawia to, że wykres ten różni się od dwóch pozostałych. Zupełnie inny przebieg mają wykresy dla poszczególnych klas pakietów. Przy algorytmie naturalnym widać całkowitą losowość przebiegu charakterystyk, które na przemian się przeplatają, w zależności od intensywności strumienia wejściowego. W dwóch pozostałych algorytmach wykresy liczby pakietów priorytetowych układają się na dużo niższym poziomie. Dzieje się tak dzięki priorytetowemu algorytmowi obsługi, a wyjątkiem od tej reguły jest moment, kiedy następuje wypychanie informacji zwykłych.



Rys. 6. Wyniki symulacji dla przełącznika progowo-priorytetowego

Trzeci z kolei wykres, to parametr nazwany „czas zatrzymania", czyli charakterystyka opóźnienia w przesyłaniu pakietów.

Wykresy czasu zatrzymania pakietów normalnych i priorytetowych dla algorytmu naturalnego prawie się pokrywały (brak priorytetowego szeregowania w buforze). Maksymalny czas pobytu w kolejce wystąpił wtedy, gdy bufor był pełny. W dwóch pozostałych modelach przełącznika zatrzymania pakietów uprzywilejowanych zazwyczaj były bliskie zeru (obsługa „na bieżąco"), a pakiety zwykłe musiały oczekiwać w kolejce znacznie dłużej.

Ostatni z serii wykresów, to tzw. „prawdopodobieństwo straty" pakietów, będące jednym z najważniejszych parametrów charakteryzujących jakość oferowanych w sieci usług (QoS).

W modelu z regulaminem naturalnym, przy skokowym wzroście intensywności strumienia wejściowego rozpoczął się proces tracenia pakietów i to zarówno zwykłych jak i priorytetowych (ograniczona pojemność buforów). Początkowo została zgubiona niewielka ilość pakietów normalnych (ok. 6%), lecz w chwilę po tym miejsca w kolejce nie znalazło ponad 90% pakietów, a potem stracone zostały wszystkie wygenerowane pakiety zwykłe. Podobna sytuacja miała miejsce z priorytetowymi pakietami, przy czym okres ten był już nieco krótszy i nie osiągnął wartości maksymalnej (100%). Inny charakter wykresów można zobaczyć dla regulaminu wypychająco-priorytetowego, w którym gubione były przede wszystkim pakiety zwykłe, a prawdopodobieństwo straty sięgało 100%. Wszystko po to, aby zrobić miejsce pakietom uprzywilejowanym, które w stu procentach zostały przyjęte i obsłużone przez system. Wykres prawdopodobieństw straty dla modelu z regulaminem progowo-priorytetowym wygląda trochę inaczej; straty wystąpiły w pięciu przedziałach czasowych z krótkim okresem tracenia pakietów priorytetowych i kasowaniem pakietów zwykłych.

4. Porównanie wyników symulacji i wnioski końcowe

Eksperymenty symulacyjne z modelami przełączników sieciowych dostarczają oprócz charakterystyk oddających chwilowe zachowania systemu dostarczają również wartości uśrednionych (tab. 1). W tabeli tej dokonano zestawienia dla prezentowanych modeli przełączników sieciowych, uwypuklając parametry efektywności ich pracy.

Jak wiadomo, parametrami QoS (jakości obsługi) dla każdego typu przełącznika powinny być przede wszystkim:

a) poprawność transmisji, czyli jak najmniejszy współczynnik strat pakietów priorytetowych,
b) prędkość transmisji pakietów priorytetowych (minimalizacja zatrzymań w buforach),
c) możliwie najmniejszy współczynnik strat również dla pakietów zwykłych, itp.

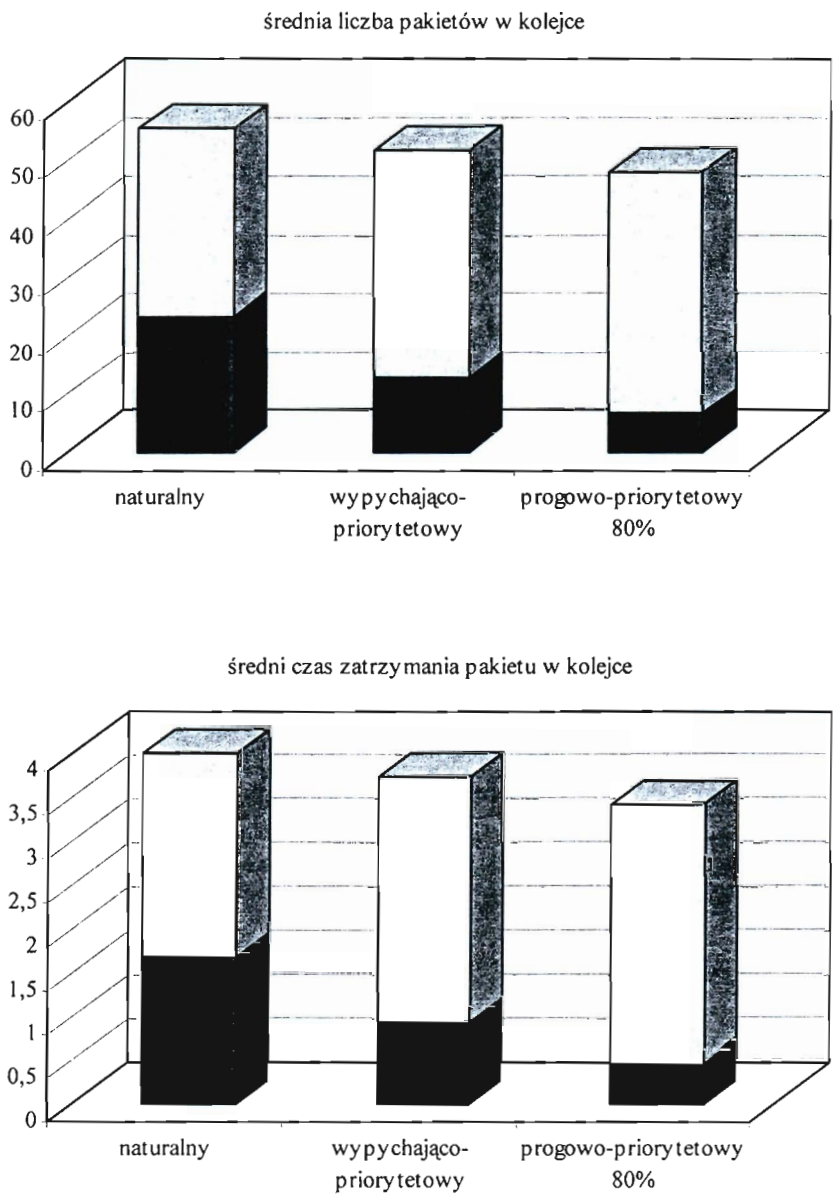
Pamiętać należy o tym, że najistotniejsze są parametry charakteryzujące przesył pakietów priorytetowych, bo pakiety zwykłe pełnią tutaj rolę raczej drugorzędną. Dlatego też, dla uwypuklenia różnic pomiędzy zaimplementowanymi algorytmami eksperymenty symulacyjne przeprowadzono przy danych powodujących niedopuszczalnie wysokie prawdopodobieństwa straty i to zarówno informacji normalnych jak i uprzywilejowanych. Parametry wejściowe zostały dobrane tak, aby, po pierwsze – odzwierciedlić zmienność przepływu danych w rzeczywistych systemach, a po drugie – umożliwić obserwację mechanizmów działania prezentowanych modeli.

Tabela 1

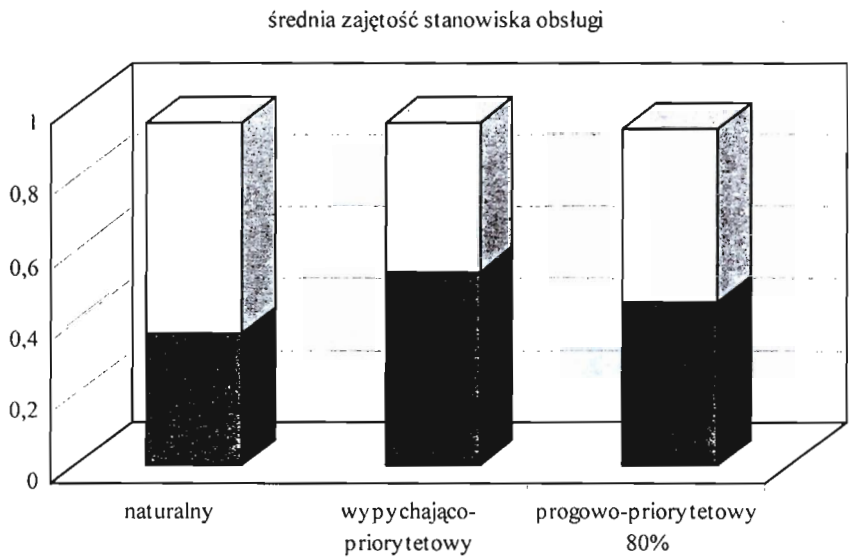
Zestawienie wyników dla parametrów jakości obsługi (QoS)

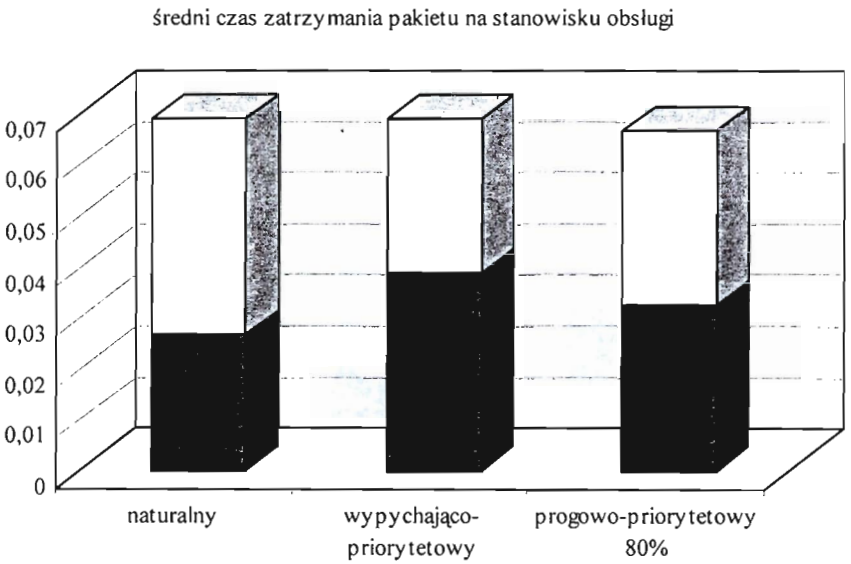
	naturalny		wypychająco-priorytetowy		progowo-priorytetowy	
	zwykłe	priorytet.	zwykłe	priorytet.	zwykłe	priorytet.
średnia liczba w kolejce	55,3628		51,8036		48,0543	
	31,9459	23,4169	38,9089	12,8947	41,5044	6,5499
średnia liczba na stanowisku obsługi	0,9552		0,9552		0,9405	
	0,5851	0,3702	0,4142	0,5410	0,4810	0,4595
średnia liczba w systemie	56,3180		52,7588		48,9948	
	32,5310	23,7870	39,3231	13,4357	41,9854	7,0094
średni czas w kolejce	3,9944		3,7376		3,4098	
	2,3049	1,6895	2,8073	0,9304	2,9450	0,4648
średni czas na stanowisku obsługi	0,0689		0,0689		0,0667	
	0,0422	0,0267	0,0299	0,0390	0,0341	0,0326
średni czas w systemie	4,0634		3,8066		3,4765	
	2,3471	1,7162	2,8372	0,9694	2,9792	0,4974
liczba zgłoszeń obsłużonych	1040		1040		1007	
	637	403	451	589	515	492
prawdopodobieństwo obsługi	0,6892		0,6892		0,6673	
	0,7222	0,6427	0,5113	0,9394	0,5839	0,7847

Oprócz tabel 1 i 2, rysunki 7–10 dokładnie odzwierciedlają zalety każdego z algorytmów szeregowania.

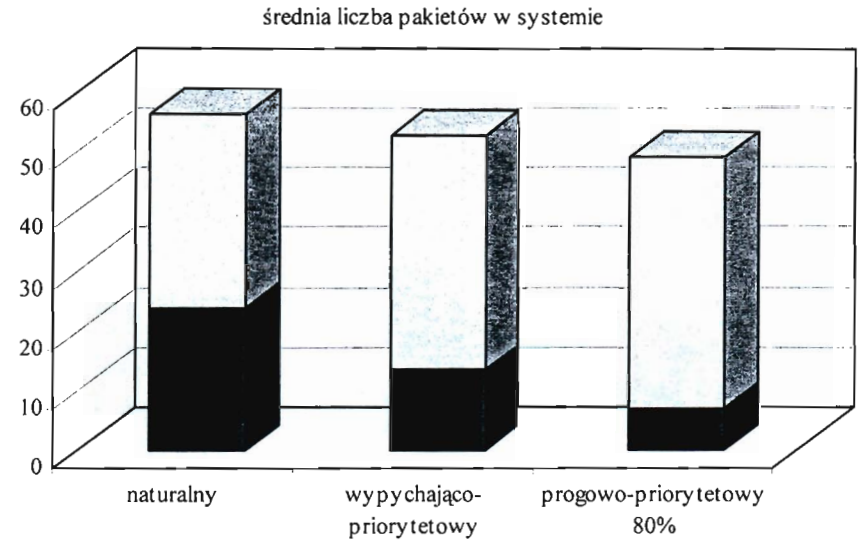


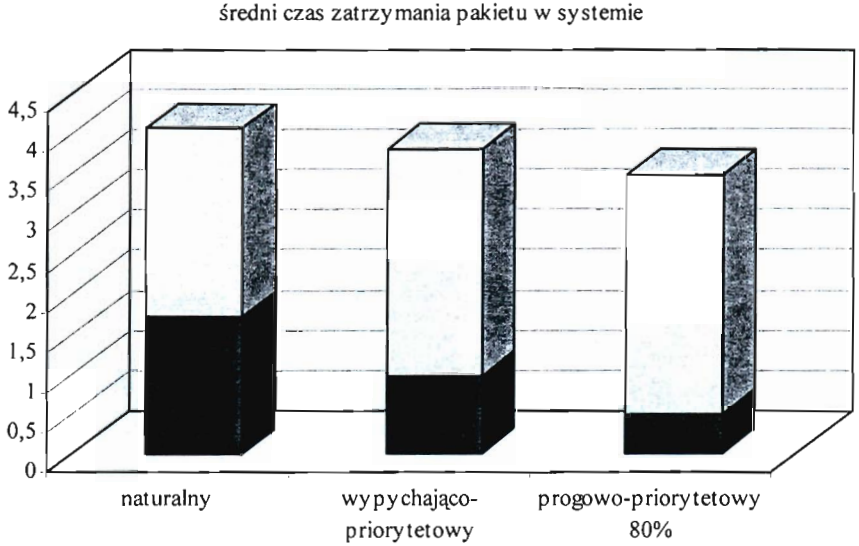
Rys. 7. Średnia liczba i czas zatrzymania pakietów w kolejce: ■ – priorytetowe, □ – normalne



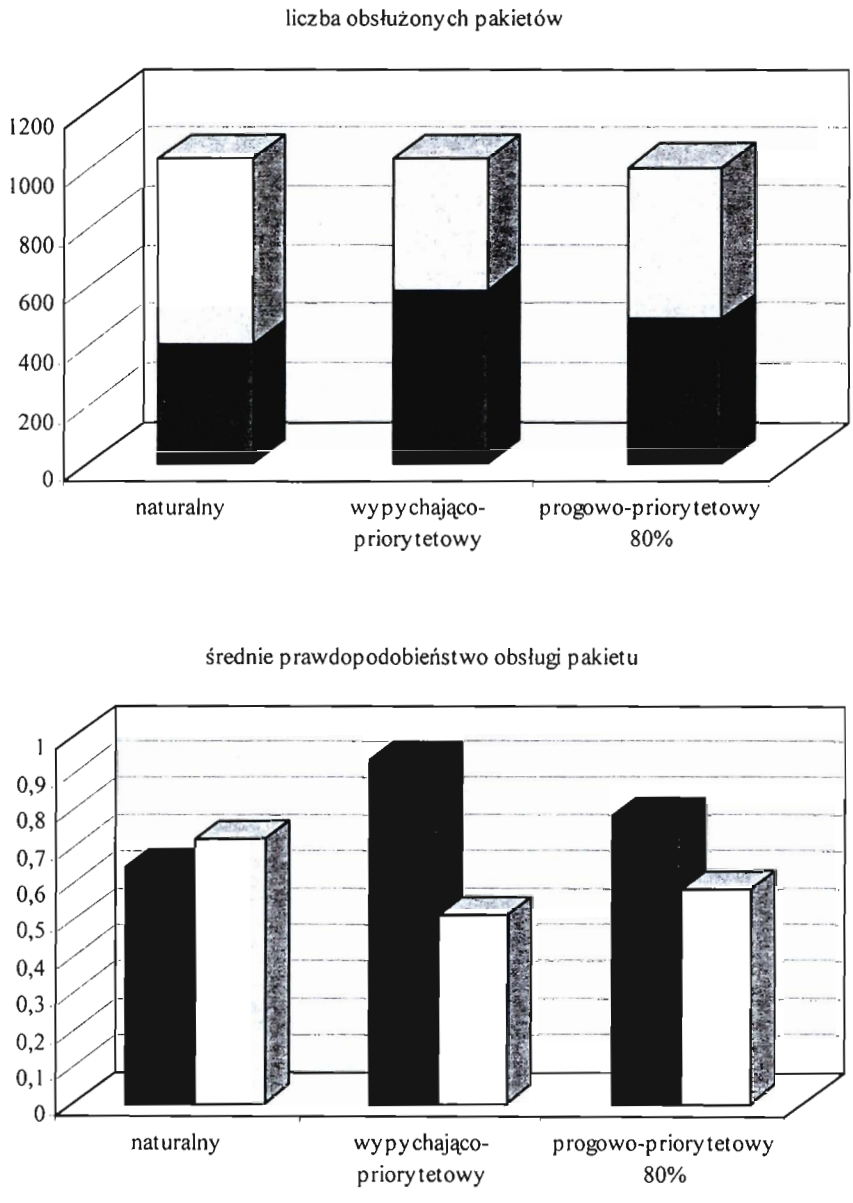


Rys. 8. Średnie obciążenie i czas obsługi pakietów w przełącznikach: ■ – priorytetowe, □ – normalne





Rys. 9. Średnia liczba i czas zatrzymania pakietów w przełącznikach: ■ – priorytetowe, □ – normalne



Rys. 10. Średnia liczba obsłużonych pakietów i prawdopodobieństwo obsługi: ■ – priorytetowe, □ – normalne

Na rysunku 7 pokazane jest porównanie średniej liczby i czasu zatrzymania pakietów w kolejce dla każdego typu przełącznika. Średnia liczba pakietów w kolejce pozwala dobrać właściwą pojemność bufora, a czas zatrzymania pokazuje, czy obsługa przebiega sprawnie. Oba parametry są oczywiście ściśle ze sobą powiązane, co zresztą widać na zamieszczonych rysunkach, a najlepsze wskaźniki QoS dla pakietów uprzywilejowanych otrzymano dla algorytmu progowo-priorytetowego, który zapewnił najszybsze przesłanie wygenerowanych danych. Okazał on się ponad 3,5 razy szybszy niż naturalny i dwukrotnie szybszy niż wypychająco-priorytetowy.

Innym ważnym parametrem QoS jest obciążenie i czas zatrzymania pakietów na stanowisku obsługi. Wykresy pokazane na rys. 8 świadczą o stopniu wykorzystania zasobu jakim jest stanowisko obsługi, które prawie maksymalnie jest obciążone, co wskazuje na jego odpowiednie wykorzystanie. Średni czas na stanowisku pozostawał w granicach 0,06-0,07, czyli na zadawalającym poziomie. Najwięcej zgłoszeń priorytetowych obsługiwano przy algorytmie wypychająco-priorytetowym.

Na rysunku 9 pokazane jest porównanie średniej liczby i czasu zatrzymania pakietów w systemie, a rysunek 10 pokazuje porównanie liczby obsłużonych pakietów i prawdopodobieństwo obsługi dla każdego przełącznika. Są to parametry decydujące o jakości transmisji (szczególnie multimedialnych).

Tabela 2

Zestawienie liczby obsłużonych pakietów i prawdopodobieństwa obsługi

algorytm	ilość obsłużonych pakietów priorytetowych	ilość obsłużonych pakietów zwykłych	prawdopodobieństwo obsługi pakietów priorytetowych	prawdopodobieństwo obsługi pakietów zwykłych
naturalny	403	637	0,64	0,72
wypychająco-priorytetowy	589 wzrost o 46%	451 spadek o 29%	0,94 wzrost o 30%	0,51 spadek o 21%
progowo-priorytetowy	492 wzrost o 22%	515 spadek o 19%	0,78 wzrost o 14%	0,58 spadek o 14%

Tabela 2 zawiera końcowe porównania prezentowanych przełączników. Przy algorytmie wypychająco-priorytetowym zanotowano 46% wzrost liczby obsłużonych zgłoszeń priorytetowych w porównaniu z regulaminem naturalnym, przy 29% spadku liczby pakietów zwykłych. Dla porównania model progowo-priorytetowy: wzrost o 22% (spadek o 19% liczby pakietów zwykłych).

Porównajmy dane dotyczące prawdopodobieństwa obsługi. Parametr ten wzrósł (względem regulaminu FIFO) aż o 30% przy algorytmie wypychająco-priorytetowym (przy 21% spadku prawdopodobieństwa zgłoszeń normalnych). Dla

porównania model progowo priorytetowy: wzrost o 14% (spadek o 14% prawdopodobieństwa pakietów zwykłych i o 3% wszystkich).

5. Podsumowanie

Rezultaty eksperymentów z przełącznikami sieciowymi pokazały, że podstawowe miary jakości obsługi (QoS), takie jak ilość obsłużonych pakietów oraz prawdopodobieństwo obsługi są lepsze w przypadku algorytmu wypychania niż progowego i oczywiście dla algorytmu naturalnego. Powoduje to znacznie bardziej efektywne wykorzystanie zasobu systemowego jakim jest stanowisko obsługi. Uwzględniając wnioski wynikające z przebiegu charakterystyk czasowych można stwierdzić, że najlepsze wyniki osiągnął model przełącznika z regulaminem wypychająco-priorytetowym. Nie jest on jednak niezastąpiony. W systemach, które większy nacisk kładą na prędkość transmisji niż na jej poprawność, algorytmy progowe z racji lepszych parametrów czasowych mogą okazać się bardziej użyteczne.

Literatura

[1] *MODSIM II The Language for Object-Oriented Simulation*, Reference Manual, CACI Products Co., 1991.

[2] Koleśnik K.: *MODSIM II – Procesowy język symulacyjny zorientowany obiektowo*, Pro Dialog No 3/1995, Nakom, Poznań, 1995.

[3] Czachórski T.: *Modele kolejkowe w ocenie efektywności sieci i systemów komputerowych*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice, 1999.

[4] Oniszczuk W.: *Metody modelowania*, Wydawnictwa Politechniki Białostockiej, Białystok, 1995.

[5] Zgrzywa A.: *Ocena wydajności systemów informacyjnych metodami kolejkowymi*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 1998.

[6] Wilcox P. A., et al. *Advanced distributed simulatiom: a review of developments and their implication for data collection and analysis*. Simulatiom Practice and Theory, vol.8, 2000, pp. 201-231.

[7] Li Z. G., at al. *A switched priority scheduling mechanism for ATM switches with multi-class output buffers*. Computer Networks, vol. 35, 2001, pp. 203-221.

SIMULATION MODELS AND ANALYSIS OF THE ATM SWITCHES OVERLOADING

Abstract: In this paper models of simulation for several types of network switches (multiplexers) are presented. They are constructed on the bases of the object-oriented language MODSIM. A set of simulation experiments are described, which allowed to analyze the ATM network overloading and to calculate the Quality of Service (QoS) parameters. In this type of networks usually un-stationary, heterogeneous, variable-time cell streams take place, which do not allow analyzing such networks by analytical modeling using queueing network theory and hence the simulation is the only way to do it.

Key words: Simulation, MODSIM, Network Switches Models, ATM

Artykuł zrealizowano w ramach pracy badawczej własnej W/II/5/00

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Aleksander Ostanin[1]

AN INTEGRATED LABORATORY FOR MODELING, SIMULATION, DIGITAL SIGNAL PROCESSING AND CONTROL

Abstract: In this study, we discuss how engineering problems are defined in the classroom versus how such problems are formulated in the real-world. We present some result on integrated environment that allows students to bridges gap between software modeling, simulation and testing of actual systems through a common visual programming interface.

Key words: integrated laboratory, simulated, control algorithms, inverted pendulum

1. Introduction

There is a fundamental difference between how engineering problems are defined in the classroom versus how such problems are formulated in the real-world. In a classic engineering learning environment, faculty prepare lectures and demonstrations while students attend lectures and complete assignments. Too often, unfortunately, this exposure provides only a factual understanding of a subject. Real-world problems are too often ignored or reduced to trivialized case studies. Research is generally considered a holy activity and not to be shared with novices found below the Ph.D. level. The question, of course, is how to correct this condition.

The Integrated Laboratory bridges gap between software modeling, simulation and testing of actual systems through a common visual programming interface. Students can explore the iterative process of developing a model and then refining the model until computer simulation result agree with experimental measurements. The key components of the Integrated Laboratory are networked a simulation software environment MATLAB/SIMULINK and dSPACE digital signal processing hardware units. An integrated environment that allows students to cycle through the modeling, simulating and real-time stages of the experiment would help im-

[1] Department of Computer Science, Bialystok Technical University

prove student understanding of system, signal theory, algorithms of optimization and control applications.

The visual programming environment of MATLAB/SIMULINK provides a common interface for both computer simulation and real-time execution of algorithms on dSPACE digital signal processing hardware. The MATLAB/SIMULINK/dSPACE combination allows students to develop a model for a system, perform computer simulations of the system based on the model, and compare the simulation results with experimental measurements of an actual system. All of this can be done during the laboratory or class period using an integrated set of tools. The integrated environment facilitates iteration of modeling and design procedures in order to match simulation results with measured results and meet design specifications. The graphical nature of the MATLAB/SIMULINK interface removes the difficulty that is often associated with programming digital signal processors and controllers.

The Integrated Laboratory can be use in several courses at Bialystok Technical University, including: *Signals and Communication Systems, Modeling and Simulation of Dynamic Systems, Probability, Random Process, and Estimation, Computer-Controlled Systems, Neural Networks, Advanced Control et al.* The Integrated Laboratory may be used in several other courses and student's projects, including numerical methods, digital signal processing, methods of optimization, automatic control systems, Advanced Master's courses, and academic courses for professionals. An interesting characteristic of the Laboratory is that all of the components are connected to the Internet, making it possible to remotely share the Laboratory and experiments with other departments, institutes and universities.

2. Laboratory Hardware and Software

The main hardware components are 10 Sun dSPACE DS1102 Miniboxes and 10 Sun SPARCstation 5 workstations. As shown in Fig.1, the DSPACE Miniboxes and Sun workstations communicate with each other through the Internet. Each Minibox and workstation has its own Internet Protocol address and is therefore remotely accessible from any Internet connection. The Internet access to the laboratory opens up the possibility of developing experiments that can be shared among several universities, as described in [1,2]. For example, a Word Wide Web interface can be established that allows a remote user to control a physical system while viewing a video sequence of the actual system.

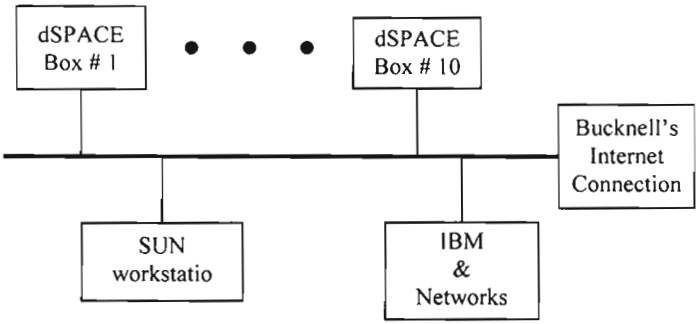


Fig. 1. Main laboratory components

Each DS1102 Minibox contains a 40-MHz Texas Instruments TMS320C31 digital signal processor along with memory and input/output circuits. Other hardware components of the DS1102 Minibox include the following:

- 128k x 32-b zero wait state memory,
- two 16-b and two 12-b analog-to-digital converters (A/D),
- four 12-b digital-to-analog converters (D/A),
- 16-b digital input/output, and
- Ethernet/Internet connection.

The dSPACE Minibox can be programmed in C or assembly language using a compiler, assembler, and linker. A script is provided by dSPACE for downloading the resulting object code to a specified Minibox. This "low-level" programming interface is most appropriate for more advanced courses which emphasize implementation details.

The simplest way to program the Minibox is through SIMULINK, which is a graphical interface to MATLAB. Systems are described in SIMULINK using block diagrams. SIMULINK contains an extensive library of predefined blocks for signal and system analysis. For example, blocks are available for generating signals, displaying signals, describing linear and nonlinear systems, designing various type of analog and digital filters, and much more. Students can develop their own blocks for special purpose functions. Furthermore, since SIMULINK runs on top of MATLAB, all of the programming facilities and built-in functions in MATLAB are accessible.

Once an algorithm or system is described by a block diagram in SIMULINK, it can either be *simulated* on the Sun workstation or compiled, downloaded, and executed in *real-time* on a dSPACE Minibox. The simulation runs on the Sun workstation only. The simulation is valuable for testing how an algorithm or system responds to various input signals or changes in design parameters. Test signals

can be generated easily within SIMULINK, and system parameters can be changed by "pointing and clicking" with a mouse.

The real-time algorithm is executed on the dSPACE Minibox and is used to process external signals and connect to physical system. The real-time version can be developed rapidly from the simulation version, since the block diagrams are nearly identical for both cases. The example presented in the following subsection illustrates the progression from simulation to real-time implementation. An interesting feature of the interface between the SIMULINK software and the dSPACE hardware is "on-the-fly" updating of parameters during real-time execution. Parameters that are changed in the SIMULINK block-diagram are updated immediately on the dSPACE hardware without recompiling the program. An additional software tool (graphical interface) serves as a virtual oscilloscope for real-time data collection and display. It can be used to transfer measured data directly to MATLAB for analysis. The combination of MATLAB, SIMULINK, dSPACE, and graphical interface provides an integrated set of tools for simulation, identification, real-time implementation, and testing of signal processing, control algorithms and programs.

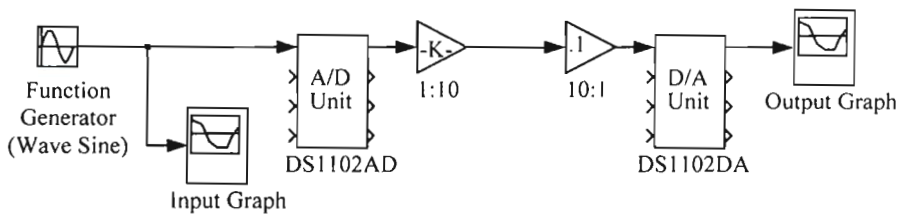


Fig. 2. SIMULINK model of simple example for simulation

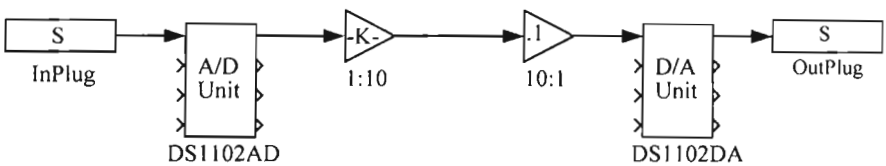


Fig. 3. SIMULINK model of simple example for real-time implementation

Illustrative Example

An example is presented that samples a signal with the A/D and reconstructs that signal with the D/A at the output. The example illustrates the similarity and difference between the SIMULINK block diagrams for the simulation and real/time versions.

The block diagram for simulation is shown in Fig. 2, and the real/time block diagram is shown in Fig. 3. The system measures the voltage on A/D channel 1 and writes it back out on D/A channel 1. The gains of 10 and 0.1 are included to compensate for the A/D and D/A units, which convert external voltage levels in the range -10 to 10 V into quantized values in the range -1 to 1 V. Note that for the simulation block diagram in Fig. 2, the sine wave input signal is included, as are graph blocks to display the input and output signals. The real-time block diagram in Fig. 3 replaces these components by "plugs" to indicate that a real function generator will be plugged into channel 1 of the Minibox A/D, and a real oscilloscope will be plugged into D/A channel 1 to display the output signal. In general, a simulation is converted into a real-time block diagram simply by replacing input/output blocks by plugs.

This example is quite instructive for students. If the input is an analog sine wave, then the output is a zero-order-hold reconstructed version of the sampled and quantized sine wave. Thus, student can explore the sampling theorem and aliasing by varying the sine wave frequency and the sampling rate. Attaching the D/A output to both an oscilloscope and an audio speaker provided an excellent demonstration of aliasing that students can both see and hear. Replacing the sine wave with speech or music is an interesting extension. These demonstrations are accessible to students at all levels, including multidisciplinary courses.

3. Inverted Pendulum Control System

We describe the application of the integrated MATLAB/SIMULINK/dSPACE environment to a Inverted Pendulum Control System. From a pedagogic point of view the Inverted Pendulum System is an ideal experimental demonstration. We describe the complete process of developing a mathematical model for the Inverted Pendulum System (IPS), simulating various control algorithms using the *model* for the IPS, controlling the *actual* IPS with the dSPACE Minibox, comparing simulation results with actual system behavior, and improving the system model and controller design as necessary.

Interesting pedagogical features of this approach are:

- Students develop a mathematical model for a real system and estimate parameters of the model, in contrast to the common textbook approach of "assuming" a model for a system.
- Students use the model to obtain computer simulation of the nonlinear IPS with SIMULINK.
- Student measure data on the actual system and then compare the data with the simulation results.

- The entire process is performed using the MATLAB/SIMULINK/dSPACE environment. Mathematical modeling, computer simulations, and real-time implementation are performed at the same time using the same tools. Students explore and understand differences between simulation and actual results, rather than simply observing the disagreement in a lab report.
- Students can document their results in a report using standard word processors on the workstations. Plots from MATLAB/SIMULINK/dSPACE are easily imported into the the reports.

Inverted Pendulum System

The object of this classical problem is to maintain a rod in the upright position as the cart to which it is hinged moves horizontally (see Fig. 4). The cart is driven by an electric motor. Cart position and rod angle are measured with potentiometers. The Inverted Pendulum is unstable in that it may fall over any time in any direction unless a suitable control force is applied. Here we consider only a two-dimensional problem in which the Pendulum moves only in the plane of the page. Assume that the Pendulum mass is concentrated at the end of the rod, as shown in Fig. 4. (The rod is massless).

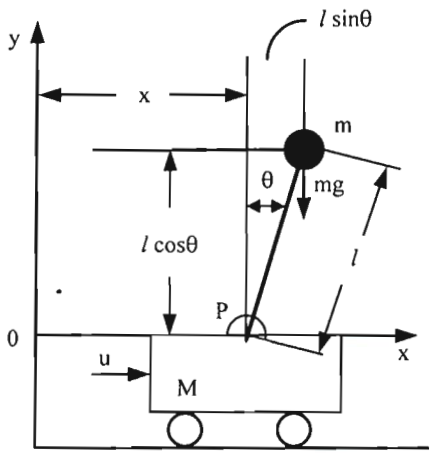


Fig. 4. Schematic of the Inverted Pendulum on a Cart

The control force u is applied to the cart. This system and low-cost experiments can be used at Bialystok Technical University and can be developed at other universities. A block diagram of the IPCS is shown in Fig. 5. Based on the error between the desired position of the cart and the measured position of the cart, the controller applies a voltage to the DC motor that is used to drive the cart.

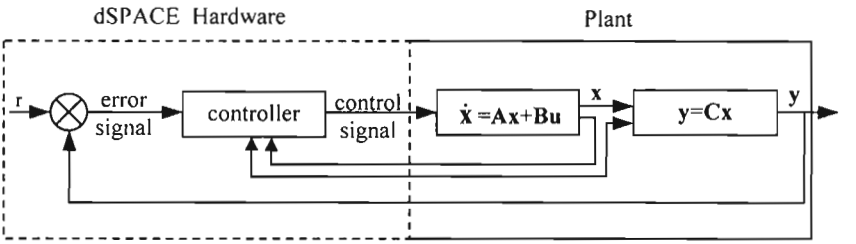


Fig. 5. Inverted Pendulum Control System

In order to *simulate* the IPCS in Fig. 5 with SIMULINK, models are need for the plant. Fig. 6 contains the SIMULINK block diagram that corresponds to Fig. 5 with a controller. The details of the model for the " IPS Plant " block in Fig. 6 are developed later.

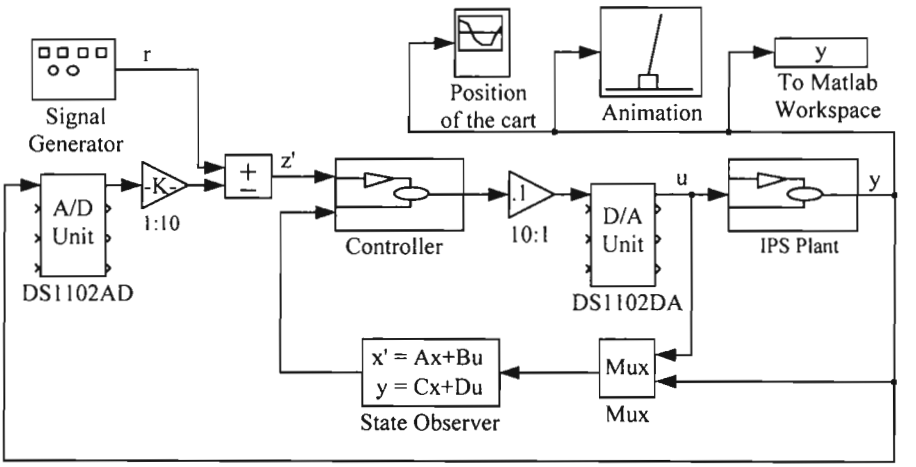


Fig. 6. SIMULINK block diagram for IPCS

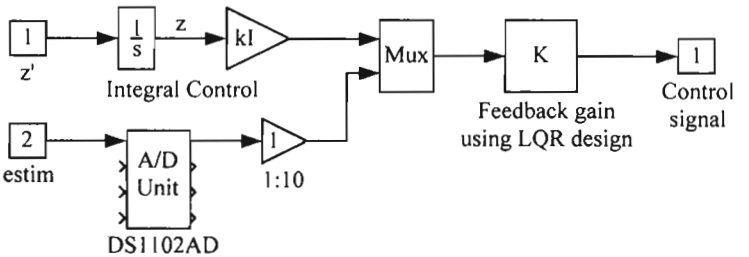


Fig. 7. SIMULINK block diagram for Controller

For *real-time* implementation, the blocks inside the dotted line in Fig. 5 are executed by the dSPACE Minibox, and the actual Inverted Pendulum System are used instead of the models. The only changes need in the SIMULINK block diagram of Fig. 6 for real-time implementation are to replace the input to the A/D and the output from the D/A by plugs (see Fig. 3). The controller is identical in the simulation and real-time block diagrams (see Fig. 7). Thus, once an algorithm is developed in a SIMULINK, that same algorithm can be executed in real-time on the dSPACE Minibox. Such "rapid prototyping" is an important advantage of the MATLAB/SIMULINK/dSPACE environment in industrial and real projects.

Modeling the IPS

Consider the IPS as shown in Fig. 4. Define the angle of the rod from the vertical line as θ. Then applying Newton's second law to the x direction of motion of the mass m and to the rotational motion it around point P we obtain the following nonlinear differential equation [3]:

$$(M+m)\ddot{x} - ml(\sin\theta)\dot{\theta}^2 + ml(\cos\theta)\ddot{\theta} = u \tag{1}$$

$$m\ddot{x}\cos\theta + ml\ddot{\theta} = mg\sin\theta \tag{2}$$

Since we must keep the Inverted Pendulum vertical, we can assume that θ (t) and $\dot{\theta}$ (t) are small quantities such that $\sin\theta \cong \theta$, $\cos\theta \cong 1$, and $\theta\dot{\theta}^2 \cong 0$. Under these conditions, the Eqs. (1) and (2) can be linearized as follows:

$$(M+m)\ddot{x} + ml\ddot{\theta} = u \tag{3}$$

$$m\ddot{x} + ml\ddot{\theta} = mg\theta \tag{4}$$

These linearized equations are valid as θ and $\dot{\theta}$ are small. Equations (3) and (4) define a mathematical model of the Inverted Pendulum System.

The linearized system equations, Eqs. (3) and (4), can be modified to

$$Ml\ddot{\theta} = (M+m)g\theta - u \tag{5}$$

$$M\ddot{x} = u - mg\theta \tag{6}$$

Equation (5) was obtained by eliminating m and $\ddot{x}$ from Eqs. (3) and (4). Equation (6) was obtained by eliminating $\ddot{\theta}$ from Eqs. (3) and (4). Define state variables $x1$, $x2$, $x3$, and $x4$ by

$$x1 = \theta,\ x2 = \dot{\theta},\ x3 = x,\ x4 = \dot{x}.$$

Note that angle θ indicates the rotation of the pendulum rod about point P, and x is the location of the cart. We consider x as the output of the system, or

$$y = x = x3$$

Then, in terms of vector-matrix equation, we have

$$\begin{bmatrix} \dot{x}1 \\ \dot{x}2 \\ \dot{x}3 \\ \dot{x}4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ g(M+m)M & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -mg/M & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} + \begin{bmatrix} 0 \\ -1/Ml \\ 0 \\ 1/M \end{bmatrix} u \tag{7}$$

$$y = [0\ 0\ 1\ 0] \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} \tag{8}$$

Equations (7) and (8) give a state-space representation of the Inverted Pendulum System. (Note that the state-space representation of the system is not unique. There are infinitely many such representations).
The mass of cart, M, the mass of the rod, m, the length of the rod, l, are all measured directly. The parameters determined for this example are:

$M = 2$ kg, $m = 0.1$ kg, $l = 0.5$ m.

By substituting these values in Eqs. (7) and Eqs. (8), we obtain:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 20.601 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -0.4905 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}, \quad \mathbf{C} = [0\ 0\ 1\ 0], \quad \mathbf{D} = 0$$

The plant eigenvalues for the Inverted Pendulum are {0, 0, 4.54, –4.54}. The eigenvalues indicate that the plant is unstable, which is expected with an Inverted Pendulum .

The measurement and data analysis used to arrive at the model in (7) and (8) are performed using the MATLAB/SIMULINK/dSPACE environment. The process of estimating parameters in the model through simple measurements and calculations is a valuable learning exercise for students. Deciding how to revise the model in light of measured data is another important lesson in the exercise. The model in (7) and (8) is used in the following subsection to simulate the IPCS.

Controller Design

It is desired to keep the IPS upright as much as possible and yet control the position of the cart, for instance, by moving the cart in a step fashion. To control the position of the cart, we need to build a type 1 servo system. The basic idea in designing the type 1 servo system here is to design a stable regulator system that will bring a zero steady-state error to a step-function input. The IPS mounted on a cart does not have an integrator (type 0 plant). Therefore, we feed the position signal y (which indicates the position of the cart) back to the input and insert an integrator in the feedforward path between the error comparator and the plant, as shown in Fig. 6 and Fig 7.

From the block diagrams of Fig. 6 and Fig. 7 the equation for the Inverted Pendulum Control System are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \tag{9}$$

$$y = \mathbf{C}\mathbf{x} \tag{10}$$

$$u = -\mathbf{K}\mathbf{x} + kIz \tag{11}$$

$$\dot{z} = r - y = r - \mathbf{C}\mathbf{x} \tag{12}$$

where $\mathbf{x}$ = state vector of the plant (n-vector); u = control signal (scalar); y = output signal (scalar); z = output of the integrator (state variable of the system, scalar); r = reference input signal (step function, scalar); $\mathbf{A} = n\text{x}n$ constant matrix; $\mathbf{B} = n\text{x}1$ constant matrix; $\mathbf{C} = 1\text{x}n$ constant matrix; $\mathbf{K}$ = state-feedback matrix; kI = integral-feedback gain (scalar)

Define $\mathbf{x}e(t) = \mathbf{x}(t) - \mathbf{x}(\infty)$; $ze(t) = z(t) - z(\infty)$; and $ue(t) = u(t) - u(\infty)$

For the type 1 servo system, we have the state error equation as given by

$$\dot{\mathbf{e}} = \hat{\mathbf{A}}\mathbf{e} + \hat{\mathbf{B}}ue$$

where

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix}, \quad \hat{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} \mathbf{x}e \\ ze \end{bmatrix}$$

and the control signal is given by

$$ue = -\hat{\mathbf{K}}\hat{\mathbf{e}}$$

where

$$\hat{\mathbf{K}} = [\mathbf{K} \vdots -kI] = [k1 \;\; k2 \;\; k3 \;\; k4 \vdots -kI],$$

and $\hat{\mathbf{e}} = (x1e, \tilde{x}2e, x3e, \tilde{x}4e, x5e)$ is the estimate of the state. (Note that $x5e = ze$).

We shall determine the necessary state feedback gain matrix such that the following quadratic performance index is minimized:

$$J = \int_0^\infty (\hat{\mathbf{e}}'\mathbf{Q}\hat{\mathbf{e}} + ue'R\,ue)\,dt,$$

where $\mathbf{Q}$ and R should be chosen properly so that the response of the system is acceptable. (The purpose of using the quadratic performance index is to assure the stability of the system).

Let us choose $\mathbf{Q}$ and R as follows:

$\mathbf{Q}$ = diag (10, 1, 100, 1, 1), $R = 1$

Our emphasis is on state variable $x1e$ and $x3e$ because θ and x are directly measured. Their derivatives ($\dot{\theta}$ and $\dot{x}$) must be generated numerically. This led us to use a full-state observer.

The controller for this problem was designed using a modification of the LQG method [4]. The modification was that a reduced order observer was used in place of a full order Kalman filter. The LQG methodology was used because of the multivariable nature of the problem (single-input two-output system). The separation principle is used to independently determine the controller and observer gains.

The controller gain was found with a state weighting of $\mathbf{Q}$ and the control weighting of R. The resulting control gain is

$\mathbf{K}$ = [–64.93 –14.48 –10.85 –9.29] and kI = –0.52

The closed loop eigenvalues for this gain are

{–5 –5 –4.99 and $-1 \pm 1.73j$}

The control law stabilizes the plant and the dominant closed loop pole indicates that the system has a settling time of approximately 4.5 sec.

The next step is the design of the observer. Because we directly measuring the rod angle and the cart position, we only need to estimate the remaining two states: the rod angular velocity and the cart velocity. In this case, we design a reduced order observer. In order to find the observer gain **L**, we again use the LQG approach. The matrices **A**22 and **A**12 are read off the 2x2 partitioning of the plant **A** matrix [4]. These matrices play the role of the **A** and **C** matrices in the full order case. The weighting matrices $\mathbf{Q}_0$ and $\mathbf{R}_0$ are used to tune the observer. The selected weights, after several trials, are

$$\mathbf{Q}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 300 \end{bmatrix}, \quad \mathbf{R}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

These weights were used to obtain desirable observer poles, and simultaneously come up with reasonable numbers for the observer parameters. It is desirable, from an implementation point of view, that the observer parameters not vary over a large range. Define $\mathbf{w}$ as $\mathbf{w} = \hat{\mathbf{e}} - \mathbf{L}y$.

The observer gain is

$$\mathbf{L} = \begin{bmatrix} 0.15 & 0.55 \\ 0.55 & 25.12 \end{bmatrix}$$

This resulted in

$$\dot{\mathbf{w}} = \begin{bmatrix} -1.33 & -0.55 \\ 75.42 & -24.83 \end{bmatrix} \mathbf{w} + \begin{bmatrix} -0.74 & -17.59 \\ 0.89 & 101.41 \end{bmatrix} y + \begin{bmatrix} 0.37 \\ -24.60 \end{bmatrix} ue,$$

where the observer poles of –24 and –3.2 indicate that the observer are great than the system poles. This results in the observer having a settling time that is faster than the system.

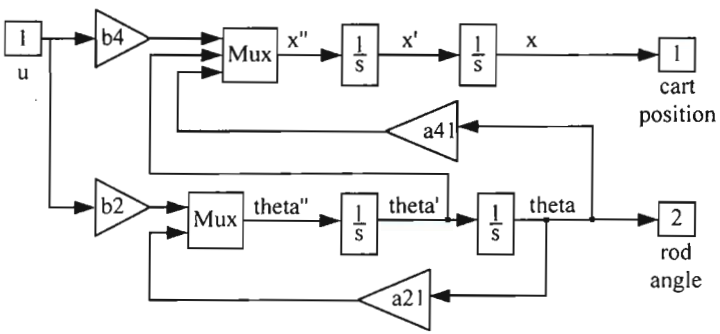


Fig. 8. IPS SIMULINK model

Simulation and Real-Time Implementation Results

A SIMULINK model for the IPS is easily developed from (7) and (8) and is shown in Fig. 8. The block diagram in Fig. 8 is encapsulated into the " IPS Plant " block in Fig. 6. Note that we have used SIMULINK to develop a flexible computer simulation of a reasonably complex system. An important payoff from our effort in constructing the *simulation* block diagram is that the *real-time* implementation with dSPACE requires only a few more keystrokes!

Figs. 9 through Fig. 11 displays the results of SIMULINK simulations and control of the IPS. The optimal gain *k1* obtained using LQG design is plotted in Fig. 9. Also shown is the optimal gain *k1* for values of *R* of 0.01; 0.1; 10 and 100,

with **Q** unchanged. As the value of R is reduced, the contributing of the control effort to the cost function J is reduced. Hence the control effort will be increased in order to reduce the magnitudes of the states. This effect is seen in Fig. 9. Similar results were obtained for optimal gains *k2*, *k3*, and *k4*.

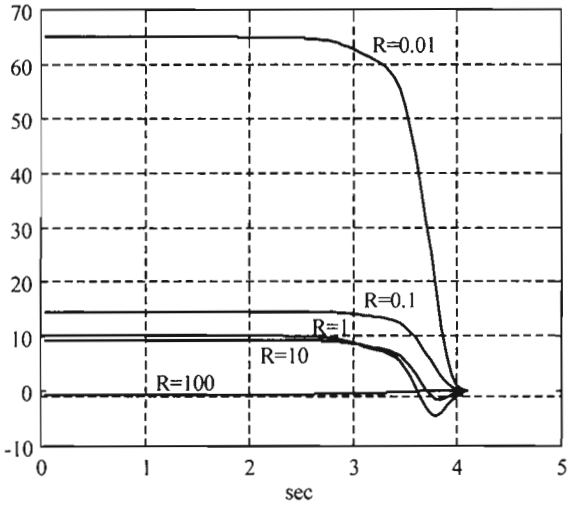


Fig. 9. Feedback gain *k1* evolution with R

As seen in Fig. 9, the feedback gain $k1(t)$ is almost constant except for the last several time. This fact is important, because the deviation from the optimal performance due to the approximation will appear only near the end of the control process. In practical systems, instead of using a time-varying gain matrix **K**(t), we approximate such a gain matrix by the constant gain matrix **K**. In the actual design, it is necessary to consider different matrices **K** (which correspond to several different sets of the weighting matrices **Q** and **R**) and carry out computer simulation to find the one that yields the best overall system performance. Then choose the best one as the matrix **K**.

Fig. 10 shows the Cart position $y(t)$ and Pendulum angle $\theta(t)$ when the system is subjecte to a unit-step input. An interesting point in the position curve is that the cart moves backward for the first 0.6 sec or so to make the Pendulum fall forward. Then the cart accelerates to move in the positive direction. Such response is a typical response behavior of the nonminimum-phase system.

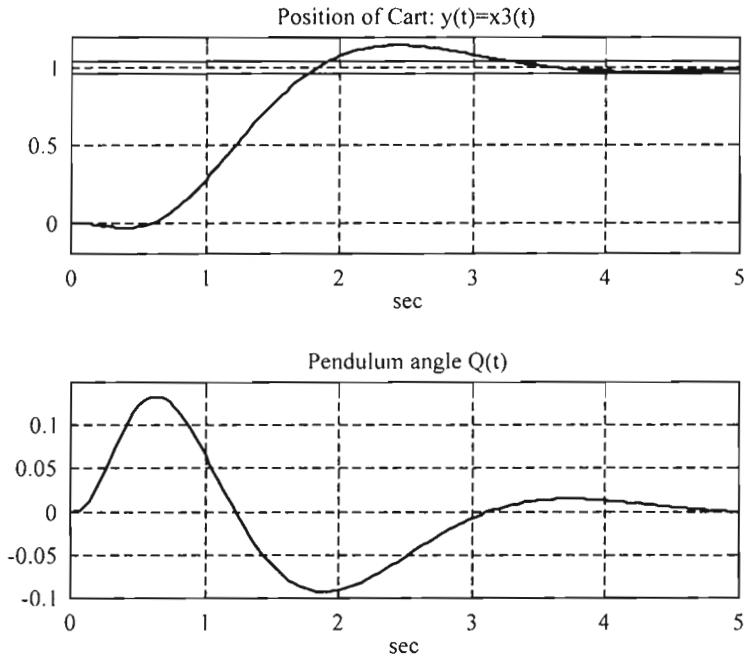


Fig. 10. Unit-Step response of the Pendulum on a Cart

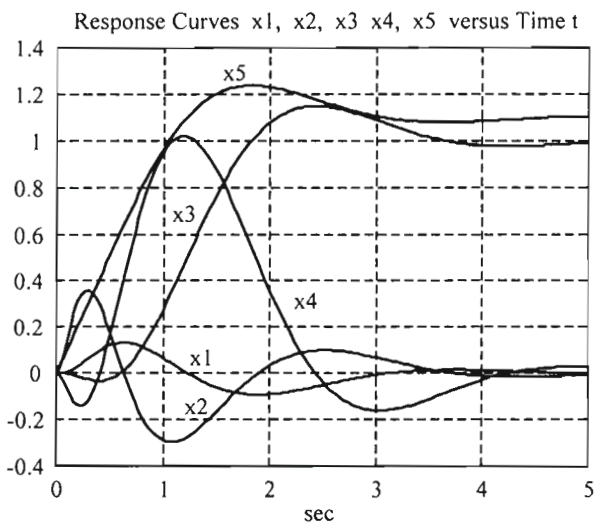


Fig.11. Response curves x1, x2, x3, x4, and x5

Fig. 11 shows $x1(t)$, $x2(t)$, $x3(t)$, $x4(t)$ and output of the integrator [$z(t)$] $x5(t)$ versus t on one diagram. From Fig. 10 and Fig. 11, it can be seen that the state variables almost reach steady state for 5 sec.

The simulation results agree closely with the measured response of the system [5]. The simulations correctly predicted the steady-state error and the oscillation frequency for the case of Integral Control. The refined model in (7) and (8) was found to be important in order for simulation results to agree with actual system results.

4. Concluding Remarks

An Integrated Laboratory environment has been described that uses MATLAB/SIMULINK software and dSPACE hardware for modeling, simulation, digital signal processing, and control. MATLAB/SIMULINK provides a visual programming interface for modeling, simulation, and DSP programming, so these potentially difficult processes become more accessible to students. The entire Laboratory is connected to the Internet, making possible some interesting proposals for remotely sharing laboratory facilities between different departments and universities.

Inverted Pendulum System is a textbook example of an inherently unstable control system. It is simple to describe and easy to understand, and although most of us can balance a broomstick with a little practice, it looks like a difficult problem.

Students can used the Integrated Laboratory for modeling, simulation, optimization and control of systems other than the Inverted Pendulum System. Examples include:

- Quadratic Optimal Control of a Servo System,
- Design of a Controller for a Disk Read/Write Head,
- Temperature Control of a Nonlinear Heating/Cooling System,
- Simulation of a Automatic Ship Steering System ,
- Cellular Automobile Security System,
- Telephone Initiated Home Monitoring System,
- Automated Drink Machine that Talks,
- Digital Tracking System Design,
- Design of a Controller and Full-Order Estimator for a Space Satellite's,
- Magnetic Levitation of a Steel Ball,
- Modeling, Simulation and Control a Robot Arm,
- Control of Coupled Carts,
- Active Suspension Control,
- Robust Design for Hybrid Systems,

- Control Model of Driver Steering Behavior,
- Control of Flexible Spacecrafts,
- Environment Chamber Control System,
- The Pilot-Ejecting Benchmark Problem,
- Design of an Automatic Aircraft Landing System,
- Design of a Minimum-Energy Control System,
- Design a Robot for Riding a Motorcycle,
- The PHYSBE Blood-Circulation Model,
- Electrocardiogram and Treadmill Project,
- Intelligent Control Systems *et al.* [7-12].

In the Inverted Pendulum experiment the system configuration is given and the entire work is limited to the design of the controller and computer simulation. To the realization a complete engineering development process and the final design in examples, mentioned above, at Faculty of Computer Science PB the prerequisite course are require in:

- system dynamics and modeling (Euler, Neuton and Lagrange's equation),
- modern state space control theory (state feedback, observer),
- programming language in MATLAB/SIMULINK,
- machine shop skill.

The Laboratory may also be used for a variety of digital signal processing applications. In the DSP projects, SIMULINK may be used to develop an algorithm and test it with simulated signals. Then the SIMULINK diagram is modified slightly to produce a real-time algorithm that is downloaded to the dSPACE hardware and used to process real signals.

One of the challenges posed to educators in [6] is " How can we assure that students will learn the basics so that they will be able to tackle any subject and develop an expertise suitable for their career? ". I believe that a well planned Laboratory works and design project are a step forward in the direction of meeting this challenge. I also think that the next generation of informatics engineer will have to integrate concepts of computer science and control theory much more than today. We are seeing an increasing number of applications of linguistic, logic-based, heuristic, neural and hybrid methods in control. Some of the most exciting growth areas, as I mentioned above, may be in biological and biomedical control systems, micro-mechanical systems, mechatronics systems, intelligent control systems, virtual reality, and smart materials. The average person has come to expect computer control in automobiles, consumer electronics, appliances, commercial and residential buildings, and a host of other places. This means that the future of control is extremely bright.

Engineers in the 21st century will work in an integrated environment that requires additional skills and capabilities not typically included in a traditional engineering program. The path of a concept through design, development, testing, and, finally, to the marketplace requires successful interactions of multidisciplinary teams of people who must communicate through verbal interaction, formal presentation, and readable documentation and reports. These communications will rely on computers, the Internet, video-conferencing, and multimedia interactions.

References

[1] Aburdene M.F. *et al.*: *Computer-controlled laboratory experiments*, Proc. of 1995 ASEE Annual Conf., June 1995, pp. 2088-2095.

[2] Kozick R.J., Aburdene M.F.: *A dynamic parameter estimation experiment that is remotely accessible via Internet*, Proc. of 1996 ASEE Annual Conf., June 1996.

[3] Ogata K.: *Designing Linear Control Systems with MATLAB*. Prentice Hall, 1994.

[4] Phillips C.L., Nagle H.T.: *Digital Control System Analysis and Design,* 3ed., Prentice Hall, 1995.

[5] Shahian B., Hassul M.: *Control System Design using MATLAB*. Prentice Hall, 1993.

[6] Masten M.K.: *An industrial challenge to control system educators*, Proc. of 1991 IFAC Conf. Advances in Control Education, Boston, MA, pp.1-6, June 1991.

[7] Etter D.M., Bordogna J.: *Engineering education for the 21st century*, Proc. of IEEE Int. Conf. Acustics, Speech and Signal Processing, Apr. 1994.

[8] Ostanin A.: *Problems in building a MATLAB Vibroprotection Toolbox.*/ GMUS NEWSLETTER, Rundbrief No.11- Winter 1996.

[9] Ostanin A., Leoshin A.: *A Hierarchical Data Base for Simulation of Vibration Isolation Systems*. MODELLING, MEASUREMENT & CONTROL, B, Vol.59, No.2, 1995, pp. 57 - 63.

[10] Ostanin A.: *Adaptive Control System of Vibrations of Multi-Support Vehicles*. Proc. of 14th Int. Conf. on Syst. Sci., ICSE, September 11th-14th, Wroclaw, Poland, 2001, Vol.3, pp.145-149.

[11] *Cyfrowe przetwarzanie sygnałów. Ćwiczenia laboratoryjne.* (Praca zbiorowa pod red. Andrzeja Wojtkiewicza), Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2000.

[12] *Komputerowe wspomaganie w obliczeniach naukowo-technicznych. Przykłady zastosowań pakietów MATLAB i Maple V.* (Praca zbiorowa pod red. Macieja Szymkata), CCATIE, Kraków, 1998.
[13] Osowski S., Toboła A.: *Analiza i projektowanie komputerowe obwodów z zastosowaniem językówMATLAB i PCNAP.* Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 1997.
[14] Mathews J. H., Fink K. D.: *Numerical Methods Using MATLAB*, 3ed., Prentice Hall, 1999.
[15] Chen K., Gibbin P., Irving A.: *Mathematical Exploration with MATLAB.* Cambridge University Press, 1999.
[16] Kaczorek T.: *Teoria sterowania i systemów.* WNT, Warszawa,1996.

This research was supported by the Białystok University of Technology Rector's Grant W/II/0/2001

ZINTEGROWANE STANOWISKA LABORATORYJNE DO MODELOWANIA, SYMULACJI, CYFROWEGO PRZETWARZANIA SYGNAŁÓW I STEROWANIA

Streszczenie: W pracy dyskutujemy o tym jak problemy inżynierskie są definiowane w uczelni, a jak to się przedstawia w prawdziwym świecie. Przedstawione zostały niektóre wyniki badań, otrzymanych na zintegrowanym otoczeniu MATLAB/SIMULINK/dSPACE, który pozwala studentom do nawiązania mostów między oprogramowaniem do modelowania, symulacji i testowania faktycznych systemów przez wspólne wzrokowe połączenie interfejsowe.

Słowa kluczowe: zintegrowane stanowiska laboratoryjne, symulacja, algorytmy sterowania, wahadło odwrócone

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Khalid Saeed[1], Marcin Kozłowski[2], Andrzej Kaczanowski[3]

METODA DO ROZPOZNAWANIA OBRAZÓW AKUSTYCZNYCH IZOLOWANYCH LITER MOWY

Streszczenie: W artykule jest opisany system do rozpoznawania obrazów akustycznych izolowanych liter mowy. Czytelnik zapozna się z tym jak przebiegał proces przetwarzania wstępnego, parametryzacji sygnału mowy, wyboru cech charakterystycznych i klasyfikacji. Omówione zostaną zastosowane w pracy algorytmy i procedury.

Słowa kluczowe: przetwarzanie, rozpoznawanie, identyfikacja mowy, metody Burga i Toeplitza

1. Wprowadzenie

Mowa jest zjawiskiem bardzo złożonym. Oceniamy ją jednak jako coś bardzo prostego, naturalnego. Jednakże, jeśli się bardziej zagłębimy w mechanizmy powstawania i odbioru dźwięków oraz próbujemy wykorzystać je w technice, możemy uświadomić sobie jak skomplikowana jest mowa. Powstające dźwięki zawierają mnóstwo informacji. Najważniejsze z nich to: semantyczne – związane z treścią wypowiedzi, osobnicze – pozwalające rozpoznać osobę mówiącą oraz emocjonalne, czyli te, dzięki którym możemy stwierdzić, w jakim stanie emocjonalnym jest osoba mówiąca.

Systemy służące do rozpoznawania mowy są jeszcze rzadkością. Istnieją co prawda komercyjne programy, służące do tego celu, ale żaden z nich nie rozpoznaje języka polskiego.

[1] Politechnika Białostocka, Instytut Informatyki, Wiejska 45A, 15-351 Białystok, Polska, aidabt@ii.pb.bialystok.pl
[2] mkoz@interia.pl
[3] kaczanowski@idea.net.pl

Niniejsza praca dotycząca rozpoznawania mowy ma za zadanie zbudowanie algorytmów, na bazie których możliwe będzie zbudowanie systemu do rozpoznawania mowy w czasie rzeczywistym.

Do tej pory większość algorytmów klasyfikacji w systemach rozpoznawania mowy jest oparta na zasadach działania sieci neuronowych. Zastosowanie tego typu algorytmów zarówno do rozpoznawania mowy jak i do rozpoznawania pisma jest omówione m.in. w artykule D.J. Burra [1]. Innym często stosowanym podejściem jest wykorzystanie ukrytych modeli Markova, o czym możemy się przekonać m.in. w artykule L. Grada [2].

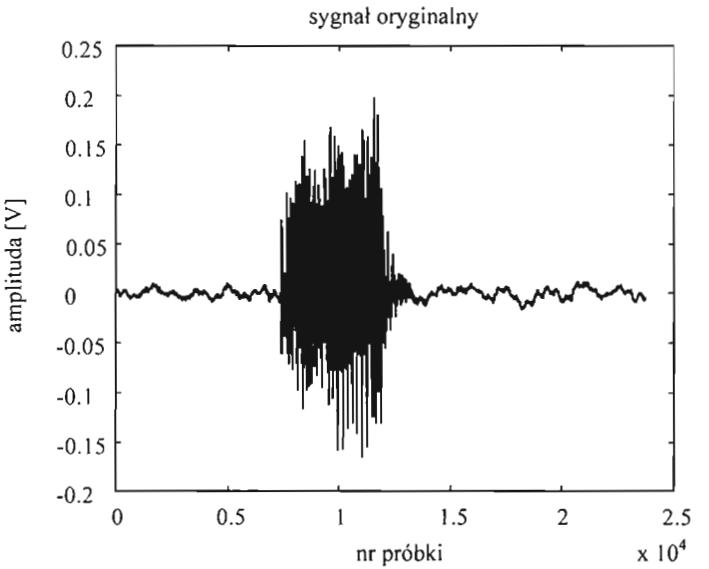
W tym artykule sygnał mowy potraktowano jako obraz, tzn. przedstawiono sygnał mowy w sposób graficzny w celu zastosowania wcześniej opracowanych metod klasyfikacji i rozpoznawania pisma traktowanego również jako obraz [3]. Wspomnianych metod użyto do identyfikacji „obrazu" mowy. Ze względu na skomplikowany charakter mowy i niejednoznaczności w jej obrazowym opisie nie jest to powszechnie stosowane podejście. Znane są inne metody parametryzacji, dające w wielu przypadkach zadowalające rezultaty. Mowa tu o współczynnikach LPC [4] (liniowego kodowania predykcyjnego), momentach widmowych [5], czy badaniu przejść przez zero [6].

Każda z wyżej wymienionych metod ma swoje zalety ale, też i wady.

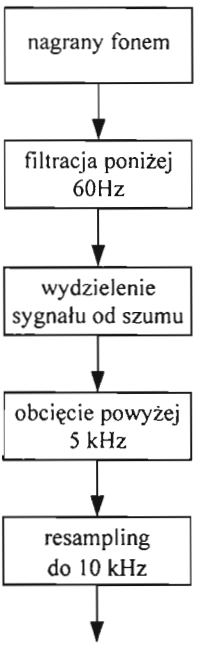
Szukając prostego a zarazem jak najbardziej stacjonarnego sposobu obrazowej reprezentacji sygnału mowy, autorzy niniejszego artykułu zastosowali jedną z metod estymacji widma częstotliwościowego opierając się na liniowym modelu predykcyjnym. Taką metodą okazała się być najlepsza metoda Burga [7]. To właśnie bazując na obrazach widma uzyskanych metodą Burga, autorzy dokonali szeregu analiz i porównań, których wyniki przedstawili w poniższym artykule.

2. Wstępne przetwarzanie sygnału

Niniejszy system rozpoznaje tylko pojedyncze fonemy. Więc na jego wejściu występuje sygnał jak na rysunku 1.



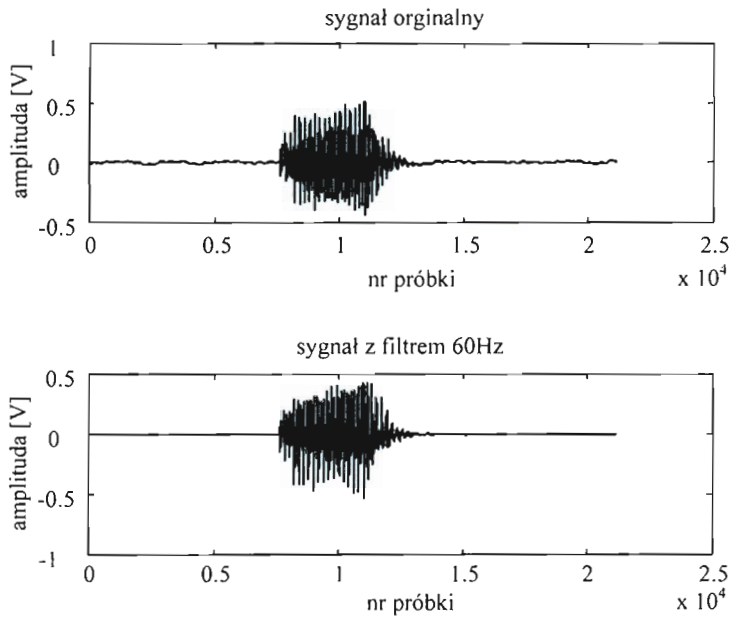
Rys. 1. Nagrana głoska do analizy



Rys. 2. Schemat blokowy systemu przygotowania sygnału mowy do rozpoznania

Jak widać, nagranie to ma „ciszę" przed dźwiękiem i po dźwięku, który należy analizować. Dźwięk ten zawiera także bardzo dużo próbek, którą też należy zmniejszyć, tak aby nie stracić najważniejszych cech sygnału. Cały proces przygotowania dźwięku do rozpoznawania przedstawia rysunek 2.

Nagrany sygnał mowy może zawierać tzw. przydźwięk sieci. Jest to sygnał o częstotliwości około 50 Hz. Może on w jakimś stopniu wpływać na wyniki. Aby więc mieć pewność dokładnej filtracji tej częstotliwości, zastosowano filtr dolnozaporowy, ustawiony na częstotliwość 60 Hz. Jak pokazały nam eksperymenty, niskie częstotliwości, nawet do 100 Hz, nie wpływają na jakość rozpoznawania mowy. Na rysunku 3 przedstawione zostało porównanie sygnału mowy bez filtracji (górny rysunek) i z filtracją (rysunek dolny).



Rys. 3. Sygnał mowy przed i po zastosowaniu filtru dolnozaporowego 60 Hz

Jak pokazuje powyższy przykład, prawie w każdym sygnale występuje składowa o częstotliwości do 60 Hz. Objawia się ona lekkim pulsowaniem sygnału w miejscu tzw. „ciszy" (w czasie, kiedy nie ma sygnału właściwego). Po wyeliminowaniu powyższej składowej sygnał jest pozbawiony niepotrzebnych częstotliwości, a przez to dokładniejsza jest jego analiza. Potrzebna jest więc na wstępie filtracja sygnału. Zastosowany jest tu filtr cyfrowy ustawiony na częstotliwość 60 Hz, aby mieć pewność wyeliminowania z sygnału składowej 50 Hz, a jednocześnie, aby nie

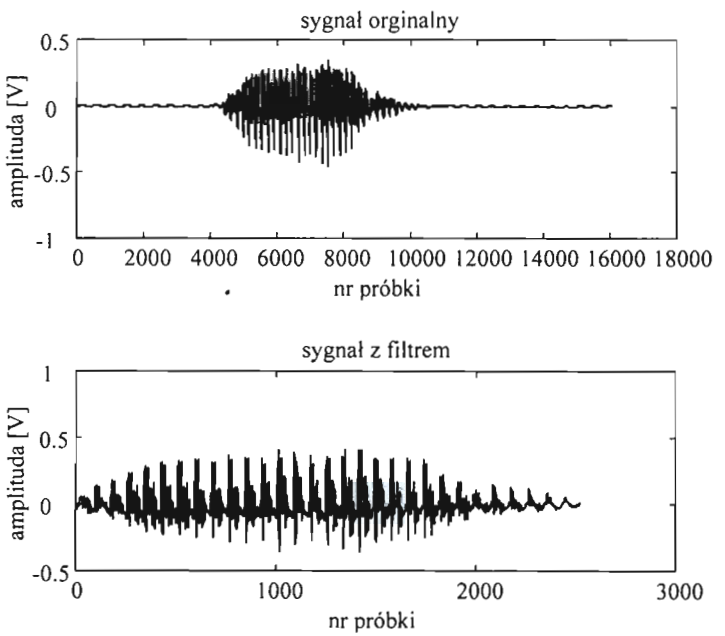
tracić zbędnej mocy obliczeniowej przez ustawienie parametrów filtru zbliżonych do ideału.

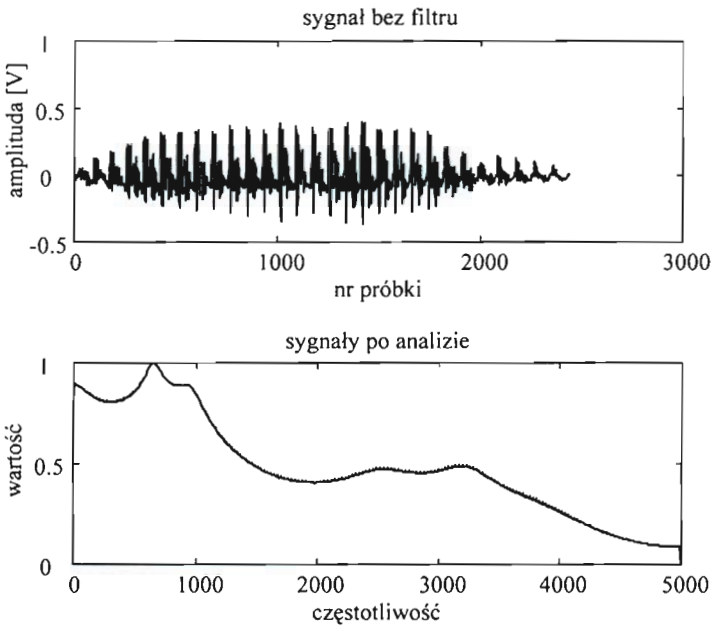
Kolejnym krokiem w przygotowaniu sygnału jest odcięcie sygnału z szumu. Jest to najważniejsza część przygotowania sygnału do rozpoznawania. Każdy algorytm, który służy do rozpoznawania sygnału mowy na wejściu otrzymuje pozbawiony zakłóceń dźwięk mowy.

Opisany poniżej algorytm składa się z dwóch etapów. Na wstępie sygnał w górnym paśmie jest wzmacniany, a następnie jest wycinany „czysty" sygnał.

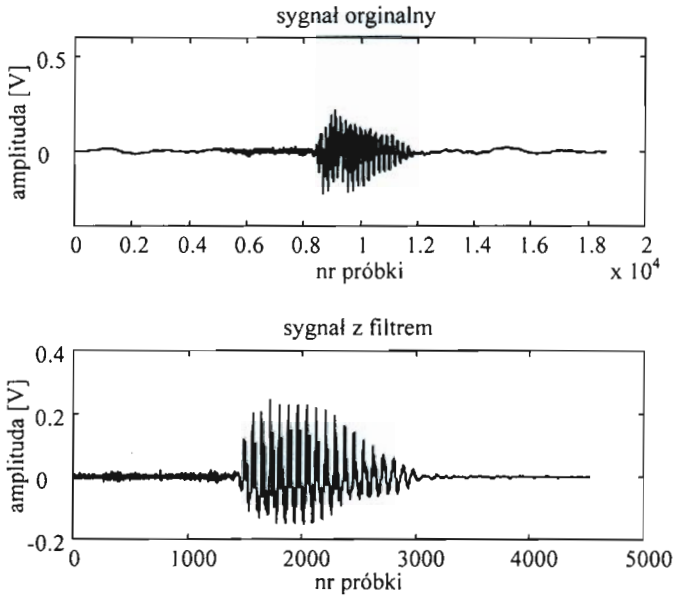
Na rysunkach 4 i 5 przedstawiono spółgłoskę i samogłoskę przygotowaną ze wzmocnieniem powyżej 3 kHz i bez niego oraz porównano widma dla poszczególnych głosek.

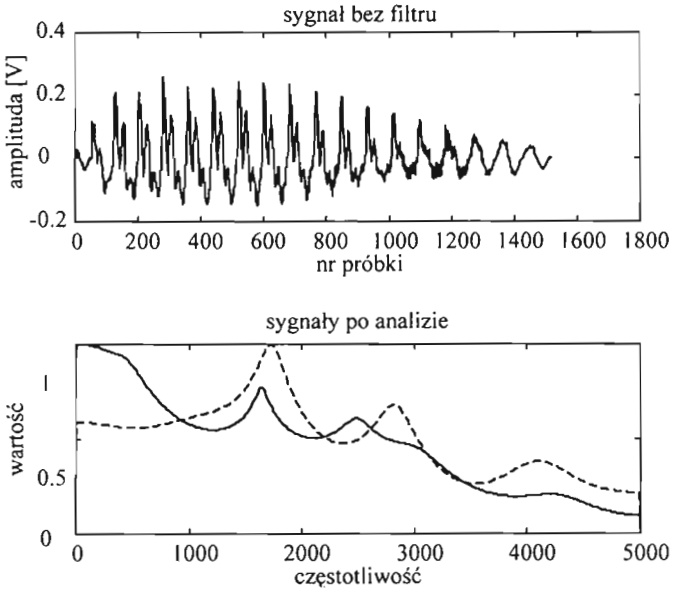
W pierwszym etapie wzmacniany jest sygnał o częstotliwościach powyżej 3 kHz [8]. Wykonuje się to, ponieważ spółgłoska składa się przeważnie z dwóch części. Pierwsza z nich o małej amplitudzie zawiera cechy charakterystyczne dla danej głoski, a druga część brzmi bez pierwszej jak samogłoska. Dlatego też wzmocnienie wysokich częstotliwości sygnału powoduje zwiększenie amplitudy pierwszej części spółgłoski, a dzięki temu dokładniejsze jest jej wydzielenie. Zabieg ten jest więc niezbędny przy spółgłoskach, co pokazują przykłady na rysunkach 4 i 5.





Rys. 4. Przygotowany do analizy sygnał „O" bez filtracji i z filtracją, oraz porównanie jego analizy: linia ciągła – sygnał z filtrem; przerywana – bez filtru





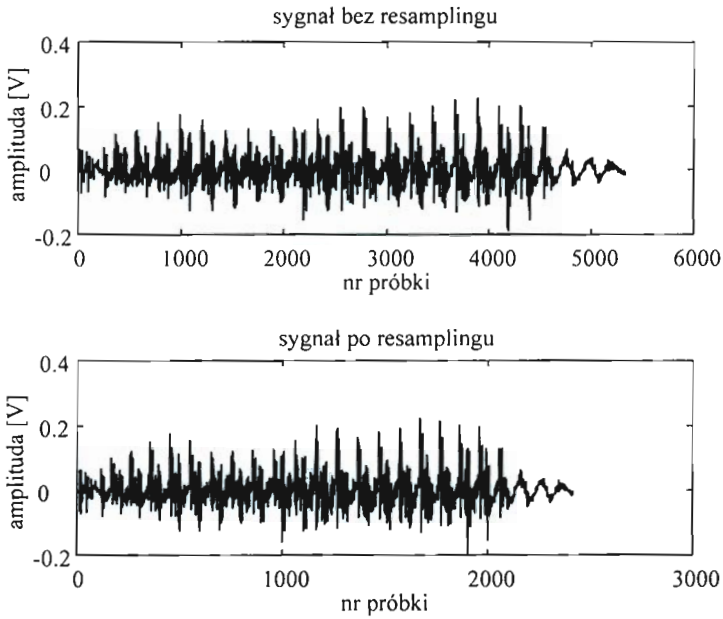
Rys. 5. Przygotowany do analizy sygnał „F" z filtrem i bez oraz porównanie jego analizy: linia ciągła – z filtrem; przerywana – bez filtru

Początkowo sygnał jest próbkowany z częstotliwością 22 kHz. Zawiera on więc składowe o częstotliwości od 0 do 11 kHz [8], gdyż częstotliwość próbkowania zgodnie z twierdzeniem Nyquista jest co najmniej dwa razy większa od najwyższej częstotliwości składowej sygnału próbkowanego. Jak stwierdziliśmy eksperymentalnie, charakterystyczne cechy sygnału mowy, które nas interesują pod kątem jego rozpoznawania, zawierają się w przedziale do 5 kHz. Ponieważ cały czas staramy się, aby system działał możliwie jak najszybciej, możemy więc maksymalnie zredukować liczbę próbek w sygnale, a zatem także i liczbę obliczeń. Wykonujemy to w dwóch etapach. Najpierw stosujemy filtr dolnoprzepustowy, ustawiony na częstotliwość 5 kHz. Otrzymujemy sygnał posiadający częstotliwości w zakresie od 0 do 5 kHz, czyli przedział, w którym zawarte są wszystkie jego charakterystyczne cechy niezbędne do rozpoznawania.

Mając taki sygnał, możemy przejść do kolejnego etapu jego przygotowania do rozpoznawania, a mianowicie do *Resamplingu,* czyli zmiany liczby próbek w sygnale. Jest to ostatnia faza w przygotowywaniu obrazów akustycznych do ich klasyfikacji. Dzięki temu *Resamplingowi* zmniejszana jest liczba próbek w analizowanym sygnale. Nie traci się natomiast nic z charakterystyki sygnału.

W naszym przypadku, po zmniejszeniu częstotliwości maksymalnej sygnału do 5 kHz, częstotliwość próbkowania powinna wynosić co najmniej 10 kHz. Wynika to z faktu, że częstotliwość próbkowania równa jest co najmniej podwojonej

częstotliwości Nyquista, która w naszym przypadku wynosi właśnie 5 kHz. Dzięki takiemu prostemu zabiegowi ponad dwukrotnie zmniejsza się liczba próbek w sygnale (z 22000 do 10000 na sekundę), zatem kolejne etapy rozpoznawania mowy operować będą na mniejszej liczbie danych. Poprawi się zatem szybkość działania systemu. Na rysunku 6 przedstawiono porównanie sygnału przed i po zastosowaniu resamplingu.



Rys. 6. Przygotowana litera do wydobycia wektora cech przed i po resamplingu

Tak przygotowany sygnał jest sygnałem wejściowym dla kolejnego etapu w systemie rozpoznawania mowy, a mianowicie: parametryzacji sygnału mowy.

3. Parametryzacja sygnału mowy

Spośród wielu sposobów parametryzacji sygnału mowy autorzy pracy wybrali metodę opartą na analizie widma częstotliwościowego. Dlaczego tak się stało? Otóż celem pracy jest spojrzenie na sygnał mowy jako na obraz i znalezienie cech charakterystycznych obrazów mowy, które umożliwią najlepszą parametryzację. Wbrew pozorom nie jest to proste zadanie, tak ze względu na mechanizmy powstawania mowy, jak też ze względu na proces jej postrzegania.

Najbardziej obiecującym sposobem parametryzacji według autorów wydaje się być analiza częstotliwościowa widma sygnału mowy. Zaletą tego sposobu parametryzacji jest niewątpliwie możliwość analizy poszczególnych składowych częstotliwościowych zawartych w sygnale mowy, zależnych od sposobu artykulacji, a więc w pewien sposób identyfikujących poszczególne składowe mowy (np. fonemy). Brak możliwości analizy sygnału w dziedzinie czasu nie jest dobrym rozwiązaniem w przypadku rozpoznawania całych słów, czy mowy ciągłej, natomiast fakt ten nie wpływa znacząco na skuteczność rozpoznawania izolowanych części mowy. Idąc dalej tym tropem, autorzy niniejszego artykułu poszukiwali sposobu na wygładzenie bardzo nieregularnego kształtu obwiedni widma, jaki otrzymano w wyniku zastosowania Szybkiej Transformaty Fouriera – FFT. Do tego celu użyto jednej z metod estymacji widma częstotliwościowego, opartej na zasadzie liniowego kodowania predykcyjnego.

Podstawową zasadą leżącą u podstaw liniowego kodowania predykcyjnego [9] jest to, że próbka sygnału $u(n)$ może być w przybliżeniu przedstawiona jako liniowa kombinacja poprzednich P próbek dla $n > 0$ (rys. 7).

$$\tilde{u}(n) = -\sum_{p=1}^{P} a_p u(n-p) \tag{1}$$

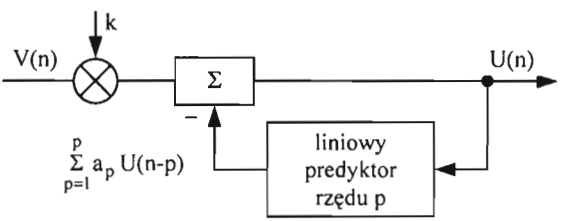
gdzie:

a_p – współczynniki predykcji,

$p = 1, 2, ..., P$,

P – rząd predykcji,

$\tilde{u}(n)$ – estymator n-tej próbki.



Rys. 7. Idea predykcji liniowej

Różnica pomiędzy $u(n)$ a $\tilde{u}(n)$ nazywana jest błędem predykcji $e(n)$.

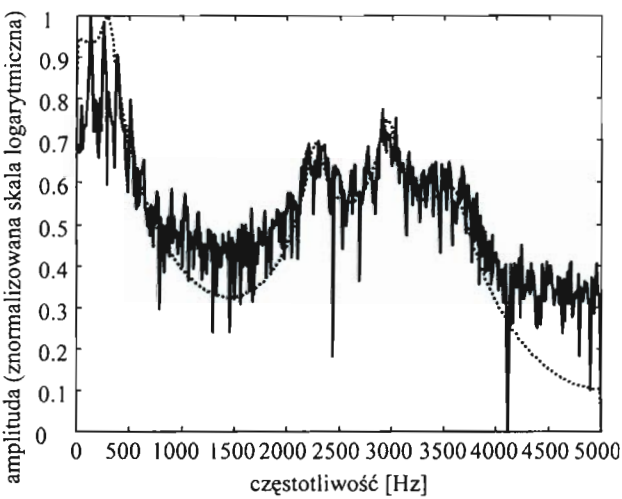
$$e(n) = u(n) - \tilde{u}(n) = u(n) + \sum_{p=1}^{P} a_p u(n-p) \tag{2}$$

Metodą wykorzystującą tę zasadę, jest metoda Burga, której implementacja, zawarta w pakiecie „Signal Processing Toolbox" programu Matlab, okazała się być punktem wyjściowym do dalszych analiz.

Metoda Burga jest metodą bazującą na minimalizacji błędów predykcji (2) z zastosowaniem iteracyjnego algorytmu Levinsona – Durbina [10,11].

W odróżnieniu do innych autoregresywnych metod obliczania współczynników predykcji, w tej metodzie pominięto etap wyznaczania funkcji autokorelacji, od razu estymując współczynniki odbicia. Metoda Burga daje dobre rezultaty przy wyznaczaniu krótkookresowego widma sygnału, dla danych będących złożeniem impulsów sinusoidalnych o podobnych częstotliwościach. Dodatkowo dokładność estymowanych wyników jest tym większa, im mniejszy jest poziom szumu (dlatego badane próbki mowy należy wcześniej wstępnie przygotować – patrz punkt 2). Zaletą tej metody jest również stabilność przy przesunięciu sygnału, tzn. widmo częstotliwościowe uzyskane dla dwóch identycznych sygnałów przesuniętych względem siebie wygląda tak samo w obu przypadkach. W związku z tym nie jest tak ważny moment, od którego zaczniemy analizować sygnał.

Dla większości głosek widmo uzyskane tą metodą ma łagodną obwiednię, gdzie bardzo wyraźnie widoczne są ekstrema lokalne (rys. 8).



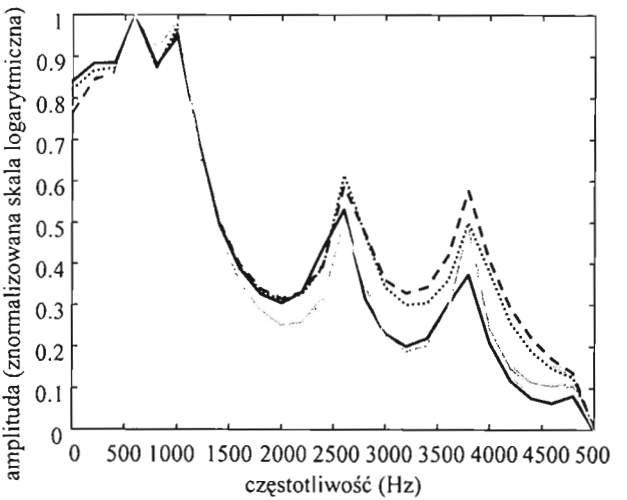
Rys. 8. Porównanie widma uzyskanego metodą klasycznego FFT i metodą Burga: —— – FFT, ····· – Burg

Uzyskane obrazy akustyczne w postaci widma częstotliwościowego zostały przeanalizowane kilkoma sposobami, a w szczególności metodą wykorzystania algorytmu minimalnych wartości własnych macierzy Toeplitza [12]. Algorytm ten służy do testowania realizowalności funkcji przenoszenia dla systemów analogowych i cyfrowych. Udowodnił on swoją przydatność przy wydobyciu cech obrazu w procesie identyfikacji pisma [13] i rozpoznawania różnych obrazów [3]. Postanowiono więc sprawdzić możliwości wspomnianego algorytmu w zastosowaniu do rozpoznawania mowy.

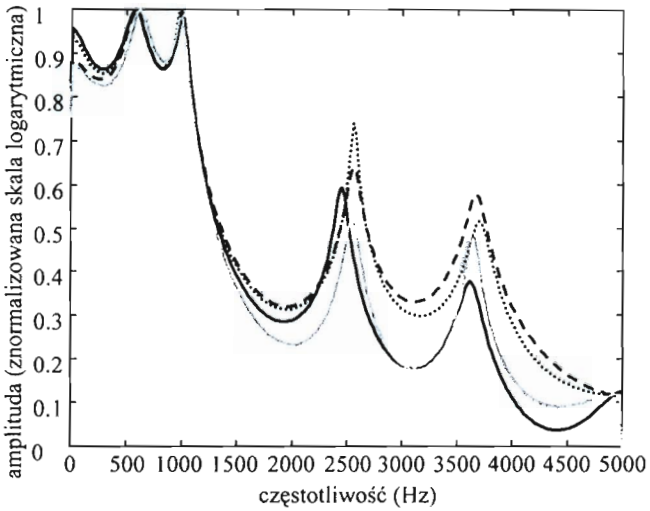
Estymacja widma metodą Burga wymagała podania parametrów dotyczących rzędu modelu predykcji, a także wielkości FFT.

Rozmiar FFT powinien być tak dobrany, żeby zapewniać gładką obwiednię widma (im więcej próbek tym lepiej); z drugiej strony nie może to być zbyt duża liczba próbek ze względu na efektywność działania algorytmu.

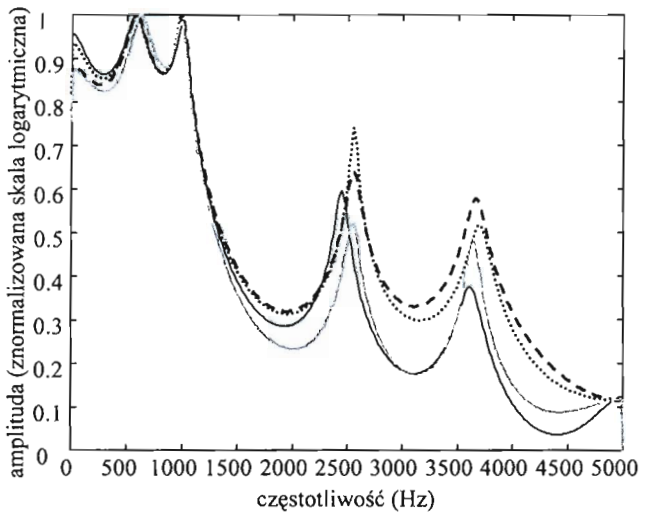
Na rysunku 9, 10 oraz 11 zostaną przedstawione różne warianty, jeśli chodzi o rozmiar FFT (dalej nazywany *nfft*) w metodzie Burga.



Rys. 9. Widmo estymowane metodą Burga dla FFT o rozmiarze 50 punktów (cztery powtórzenia litery „o"): —— – głoska o1, ····· – głoska o2, – – – głoska o3, ~~~ – głoska o4



Rys. 10. Widmo estymowane metodą Burga dla FFT o rozmiarze 500 punktów (cztery powtórzenia litery „o"): —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4



Rys. 11. Widmo estymowane metodą Burga dla FFT o rozmiarze 1000 punktów (cztery powtórzenia litery „o"): —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4

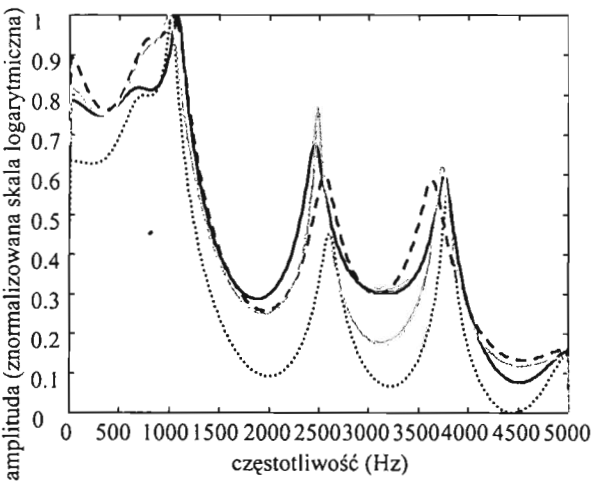
Na powyższych wykresach możemy zauważyć, że kształt obwiedni widma dla 50-punktowej transformaty FFT (*nfft* = 50) jest jeszcze nieregularny, co oznacza,

że jest to za mały rozmiar do dobrego opisania charakteru widma. Gdy *nfft* = 500 i dla *nfft* = 1000 nie widać wyraźnych różnic, co oznacza, że wybór tak dużej jak 1000 punktowa transformata FFT – jest nieuzasadniony. Przy zastosowaniu 500 punktowej transformaty FFT, ze względu na symetryczność przekształcenia Fouriera, tak naprawdę otrzymujemy widmo od 0 do 5000 Hz z rozdzielczością $f_{\max} / \left(\frac{nfft}{2} \right) = f_{\max} / 250 = 20\text{Hz}$ w rozpatrywanym przypadku.

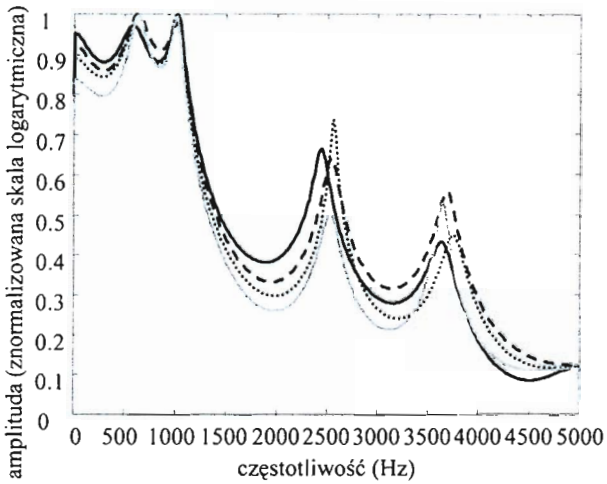
Bardzo istotnym parametrem analizy sygnału jest rozmiar próbki. Przy częstotliwości próbkowania 10000 Hz, 1ms sygnału jest reprezentowana przez 10 próbek, wobec czego duży rozmiar FFT oznacza analizę długiego okresu mowy. Nie jest to pożądane rozwiązanie, gdyż czas wypowiedzi pojedynczych fragmentów mowy (głosek) zawiera się w granicach kilkudziesięciu milisekund.

Na rysunku 12, 13 oraz 14 przedstawiony zostanie wpływ długości badanego fragmentu próbki na kształt widma przy zastosowaniu wcześniej wybranej 500 punktowej transformaty FFT.

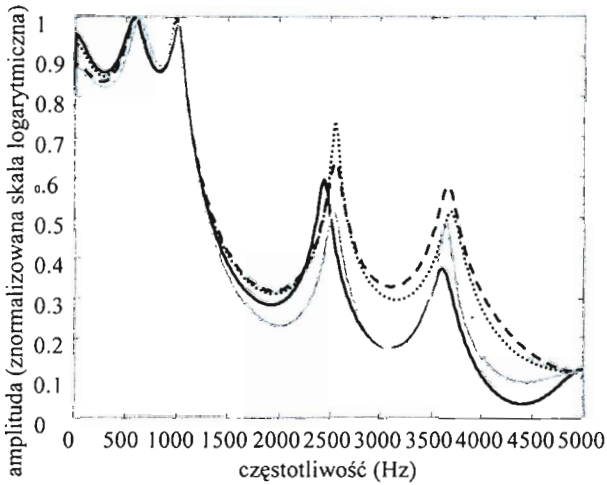
Na rysunku 12 widzimy kształt widma przy analizie 10ms fragmentu próbki. Kształt ten dla poszczególnych wypowiedzi litery „o" nie jest tak podobny jak na rysunku 13 czy 14. Z kolei rysunki 13 oraz 14 nie wykazują wyraźnych różnic, co pozwala stwierdzić, że do analizy wybór 50ms fragmentu próbki (500 próbek dla f_s = 10000 Hz) jest wystarczający.



Rys. 12. Widmo estymowane metodą Burga dla długości próbki równej 100, czyli 10ms (cztery powtórzenia litery „o"): —— – głoska o1, ····· – głoska o2, - - – głoska o3, —— – głoska o4

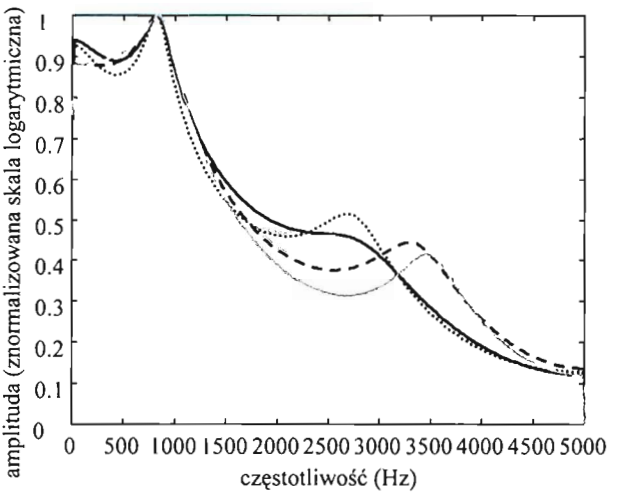


Rys. 13. Widmo estymowane metodą Burga dla długości próbki równej 500, czyli 50ms (cztery powtórzenia litery „o"): —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4

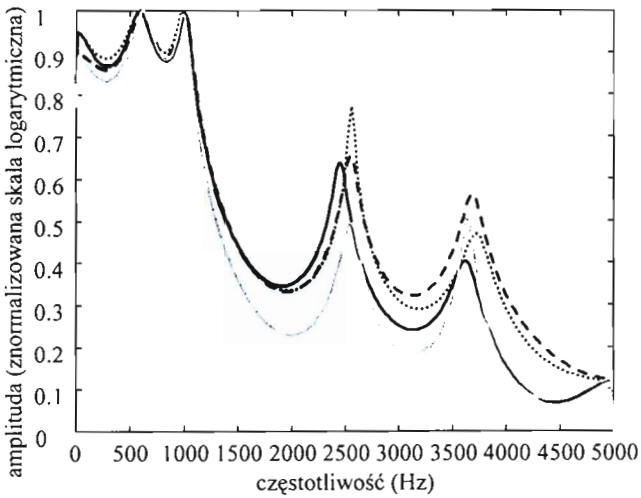


Rys. 14. Widmo estymowane metodą Burga dla długości próbki równej 1000, czyli 100ms (cztery powtórzenia litery „o"): —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4

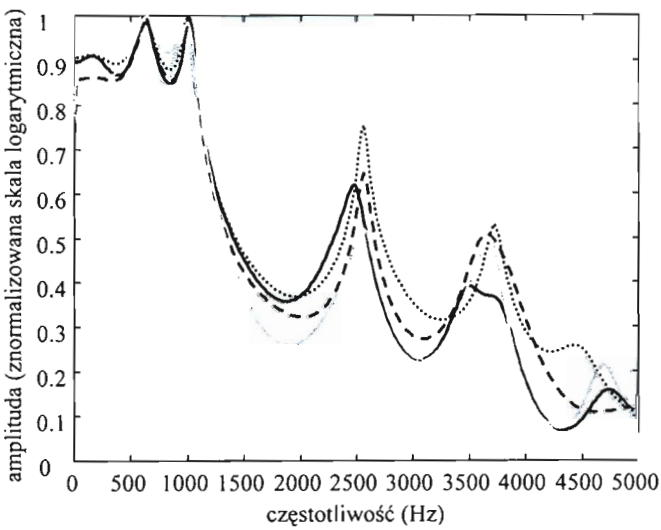
Na kolejnych rysunkach (rys. 15, 16 oraz 17) jest pokazane jaki wpływ ma rząd modelu predykcji dla potrzeb estymacji widma metodą Burga. Ogólnie mówiąc, im więcej formantów (lokalne maksimum) obwiedni widma spodziewamy się dla analizowanego sygnału, tym wyższego rzędu predykcję powinniśmy zastosować. Rząd ten powinien być około dwukrotnie wyższy od spodziewanej liczby formantów. Na podstawie próbnych analiz, autorzy arbitralnie ustalili rząd zastosowanego modelu predykcji, a wyniki tych próbnych analiz są przedstawione na rysunkach 15, 16, oraz 17. Z rysunków tych wynika, że zadowalające rezultaty osiąga się przy wyborze rzędu 12 (rys. 16). Rząd 6 (rys. 15) jest zbyt niski – na wykresie widma nie widać wyraźnych formantów, a dla rzędu 18 (rys. 17) kształt obwiedni widma wykazuje zbyt wyraźne różnice dla poszczególnych wypowiedzi badanej litery (w tym przypadku litery „o").



Rys. 15. Widmo estymowane metodą Burga – rząd modelu 6: —— – głoska o1, ····· – głoska o2, – – – głoska o3, —·— – głoska o4



Rys. 16. Widmo estymowane metodą Burga – rząd modelu 12: —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4



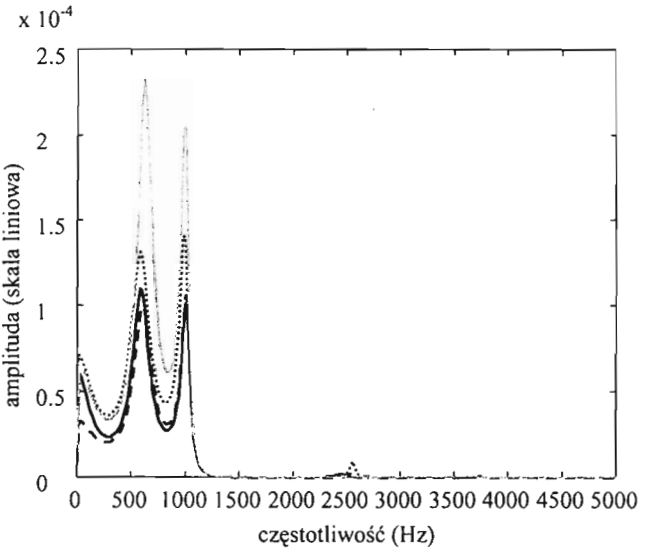
Rys. 17. Widmo estymowane metodą Burga – rząd modelu 18: —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4

Istotnym elementem mającym wpływ na kształt obwiedni widma jest zastosowanie skali logarytmicznej dla wyrażenia amplitudy poszczególnych składowych widma oraz normalizacja uzyskanych wartości, tak aby mieściły się w prze-

dziale <0;1>, tzn. 0 – minimalna wartość amplitudy dla dowolnej składowej widma, 1 – dla maksymalnej wartości amplitudy dowolnej składowej widma.

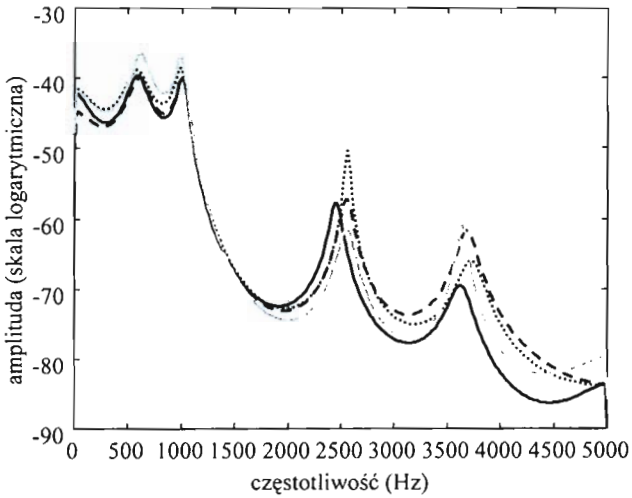
Na kolejnych rysunkach 18, 19 oraz 20 przedstawione zostaną kolejne etapy normalizacji widma.

Na pierwszym rysunku (rys. 18) widzimy widmo w skali liniowej na osi amplitudy. Na wykresie tym niewidoczne są składowe widma powyżej częstotliwości 1500 Hz z uwagi na małe amplitudy w tym zakresie.

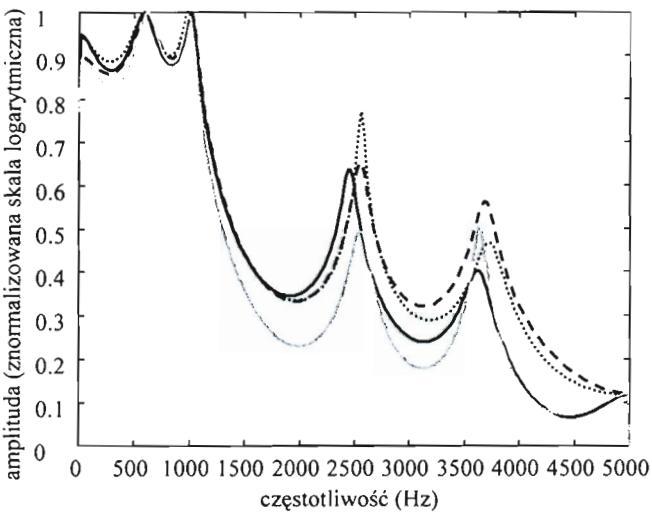


Rys. 18. Widmo estymowane metodą Burga – skala liniowa: —— – głoska o1, ····· – głoska o2, – – – głoska o3, — — – głoska o4

Na rys. 19 widmo jest bardziej wyraźne i widzimy jaki jest poziom sygnału w zakresie wyższych częstotliwości. Natomiast na Rys. 20 pokazane jest widmo po zastosowaniu wspomnianej wcześniej normalizacji do przedziału <0;1>.



Rys. 19. Widmo estymowane metodą Burga – skala logarytmiczna: —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4

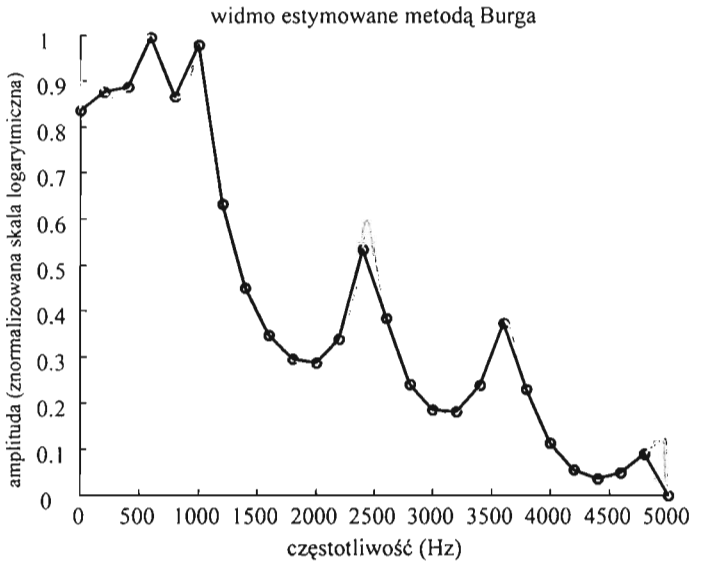


Rys. 20. Widmo estymowane metodą Burga – znormalizowana skala logarytmiczna: —— – głoska o1, ····· – głoska o2, – – – głoska o3, —— – głoska o4

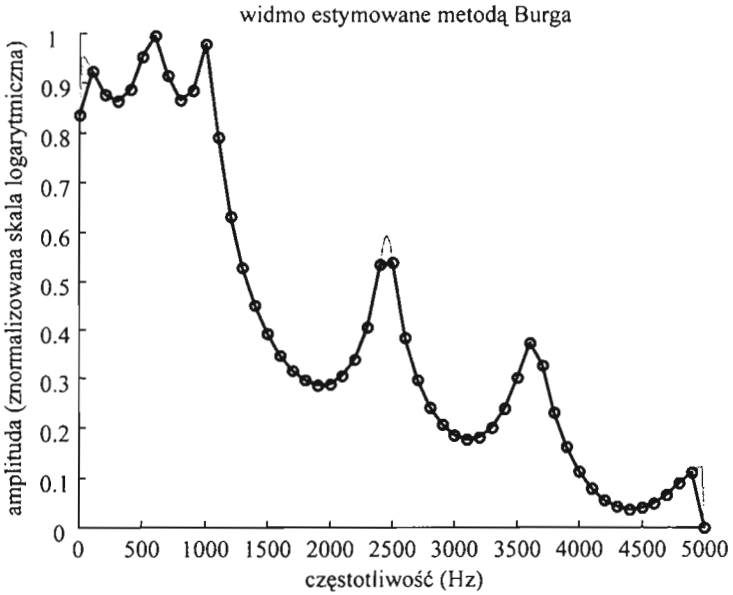
Dodatkowo przy zastosowaniu algorytmu minimalnych wartości własnych istotny jest sposób wyboru punktów charakterystycznych opisujących obwiednię widma. Proces ten jest pokazany na rysunku 21 oraz 22 i omówiony następująco:

W tym przypadku widmo składa się z 250 punktów, a wybrany zostanie co 10 punkt i dla porównania co 5 punkt wykresu.

Z wykresów widać jednoznacznie, że wybór co 10 punktu widma prowadzi do zniekształcenia obwiedni, natomiast wybór tylko co 5 punktu wykresu nie wprowadza istotnych zmian co do kształtu widma, co oznacza, że rozdzielczość co 5 punkt jest wystarczająca do przeniesienia cech charakterystycznych widma składającego się z 250 punktów, powodując jednocześnie zmniejszenie rozmiaru wektora cech 5-krotnie.



Rys. 21. Wybór punktów charakterystycznych (co 10): —— – wszystkie punkty, o – co 10 punkt



Rys. 22. Wybór punktów charakterystycznych (co 5): —— – wszystkie punkty, o – co 5 punkt

4. Klasyfikacja

W celu przeprowadzenia klasyfikacji dla metody bezpośredniego porównania obwiedni widma autorzy użyli klasyfikatora minimalnoodległościowego z funkcją podobieństwa w postaci odległości Euklidesowej:

$$D^{Mi}(x,y) = \left[\sum_{p=1}^{P} (x_p - y_p)^2 \right]^{1/2} \tag{3}$$

W metodach opartych na algorytmie minimalnych wartości własnych zmodyfikowanej macierzy Toeplitza [3], została zastosowana modyfikacja klasycznego algorytmu minimalnoodległościowego. Przyjęto następujące ustalenia:

Oznaczmy ciąg wzorcowych minimalnych wartości własnych jako:

$$\psi = \frac{\psi_2}{\psi_1}, \frac{\psi_3}{\psi_2}, \ldots, \frac{\psi_n}{\psi_{n+1}}, \tag{4}$$

gdzie $\psi_1, \psi_2, \ldots, \psi_n$ – kolejne minimalne wartości własne macierzy wzorcowej, a ciąg minimalnych wartości własnych klasyfikowanych jako:

$$\varphi = \frac{\varphi_2}{\varphi_1}, \frac{\varphi_3}{\varphi_2}, \ldots, \frac{\varphi_n}{\varphi_{n+1}}, \tag{5}$$

gdzie $\varphi_1, \varphi_2, \ldots, \varphi_n$ – kolejne minimalne wartości własne macierzy klasyfikowanej. Klasyfikacja polega na znalezieniu minimum funkcji wyrażonej jako:

$$f(\psi, \varphi) = \sqrt{\sum_{i=1}^{n} (\psi_i - \varphi_i)^2}. \tag{6}$$

5. Wyniki rozpoznawania

Znaczenie parametrów we wszystkich tabelach zawierających wyniki jest następujące:

- **Parametry metody** – zastosowane parametry estymacji widma metodą Burga i parametry wyboru cech charakterystycznych w przypadku metod opartych na algorytmie minimalnych wartości własnych;
- **Rząd metody Burga** – rząd predykcji użyty do realizacji estymacji widma metodą Burga;
- **Rozmiar FFT** – rozmiar przekształcenia Fouriera w metodzie Burga – wpływa na rozdzielczość widma – ponieważ przekształcenie jest symetryczne, to np. po zadaniu rozmiaru FFT jako 500, otrzymujemy połowę wartości widma, czyli 250;
- **Długość próbki** – długość fragmentu sygnału poddanego analizie – ponieważ zastosowana częstotliwość próbkowania f_s wynosi 10000 Hz, to np. 100 ms, oznacza 1000 próbek;
- **Okno** – zastosowano mnożenie fragmentu próbki sygnału przez funkcję okna – w większości przypadków okno Blackmana;

W przypadku metod opartych na minimalnych wartościach własnych, dodatkowo podane są parametry wyboru punktów charakterystycznych. Ich znaczenie jest następujące:

- **Co n-ty punkt** – oznacza, że ze wszystkich punktów należących do widma poddany analizie zostanie co n-ty punkt.
- **Wybieramy punkty od n_1 do n_2** – w niektórych wariantach analizy wybór punktów charakterystycznych obwiedni widma następował tylko w wybranym zakresie, tzn. np. jeśli widmo składało się ze 100 punktów, to podanie parametru: „*wybieramy punkty od 10 do 90*" oznacza wybranie tylko 80 środkowych punktów widma;

- **Wszystkie punkty** – oznacza, że jako punkty charakterystyczne potraktowane zostaną wszystkie punkty tworzące obwiednię widma;
- **Normalizacja** – oznacza czy została dokonana normalizacja punktów charakterystycznych lub wartości własnych – brak tego parametru oznacza, że normalizacja została dokonana (w większości przypadków tak właśnie jest przyjęte);

Znaczenie pozostałych parametrów jest następujące:

- **Wzorzec** – wymienione są poszczególne głoski, które w danym wariancie metody były klasyfikowane;
- **Liczba błędnie sklasyfikowanych próbek** – dla danej klasy oznacza liczba błędnie przydzielonych do innych klas próbek;
- **Liczba badanych próbek** – dla każdej klasy liczba ta wynosi 5;
- **Razem** – podsumowanie ilości wszystkich próbek poddanych analizie i wszystkich próbek źle sklasyfikowanych;
- **Procent poprawności** – liczba poprawnie sklasyfikowanych głosek w stosunku do wszystkich, które zostały poddane analizie w danej metodzie.

5.1. Metoda bezpośredniego porównania obwiedni widma [14]:

Tabela 1

Wyniki rozpoznawania z wykorzystaniem metody bezpośredniego porównania obwiedni widma

parametry metody: rząd metody Burga: 12, długość FFT: 500, długość próbki: 100ms																								razem:	procent poprawności:
wzorzec:	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	r	s	t	u	w	y	z			92,73%
ilość błędnie sklasyfikowanych próbek:	0	2	1	0	0	0	0	1	0	0	0	0	0	0	0	2	0	0	0	1	0	1		8	
liczba badanych próbek:	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5		110	

5.2. Metoda z wykorzystaniem algorytmu minimalnych wartości własnych [12] na podstawie różnic odległości punktów charakterystycznych [14]:

Tabela 2

Wyniki rozpoznawania z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie różnic odległości punktów charakterystycznych

Parametry metody: rząd metody Burga: 12, długość FFT: 50, długość próbki: 100ms, wybór wszystkich punktów w zakresie od 5 do 20,							razem:	procent poprawności:
wzorzec:	a	e	i	o	u	y		90,00%
ilość błędnie sklasyfikowanych próbek:	0	0	0	1	1	1	3	
liczba badanych próbek:	5	5	5	5	5	5	30	

Tabela 3

Wyniki rozpoznawania z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie różnic odległości punktów charakterystycznych

parametry metody: rząd metody Burga: 12, długość FFT: 50, długość próbki: 100ms, wybór wszystkich punktów w zakresie od 5 do 20,																						razem:	procent poprawności:
wzorzec:	a	b	c	d	e	f	g	h	i	k	l	m	N	o	p	r	s	t	u	w	y	z		
ilość błędnie sklasyfikowanych próbek:	0	5	5	5	0	5	5	3	1	4	5	5	5	1	3	4	3	1	1	4	1	1	67	39,09%
liczba badanych próbek:	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	110	

5.3. Metoda z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie funkcji przenoszenia [12,14]:

Tabela 4

Wyniki rozpoznawania z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie funkcji przenoszenia

parametry metody: rząd metody Burga: 12, długość FFT: 50, długość próbki: 100ms, wybór wszystkich punktów w zakresie od 1 do 25							razem:	procent poprawności:
wzorzec:	a	e	i	o	u	y		
ilość błędnie sklasyfikowanych próbek:	0	0	0	0	0	0	0	100,00%
liczba badanych próbek:	5	5	5	5	5	5	30	

Tabela 5

Wyniki rozpoznawania z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie funkcji przenoszenia

parametry metody: rząd metody Burga: 12, długość FFT: 50, długość próbki: 100ms, wybór wszystkich punktów w zakresie od 1 do 25																								razem:	procent poprawności:
wzorzec:	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	r	s	t	u	w	y	z			
ilość błędnie sklasyfikowanych próbek:	1	3	1	0	0	4	5	5	0	3	3	0	0	0	0	2	0	2	0	5	3	2	39	64,55%
liczba badanych próbek:	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	110		

5.4. Metoda z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie odwrotnej funkcji przenoszenia [14]:

Tabela 6

Wyniki rozpoznawania z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie odwrotnej funkcji przenoszenia

parametry metody: rząd metody Burga: 12, długość FFT: 100, długość próbki: 100ms, wybór co 2 punktu w zakresie od 5 do 45							razem:	procent poprawności:
wzorzec:	a	e	i	O	u	y		
ilość błędnie sklasyfikowanych próbek:	0	0	0	0	2	0	2	93,33%
liczba badanych próbek:	5	5	5	5	5	5	30	

Tabela 7

Wyniki rozpoznawania z wykorzystaniem algorytmu minimalnych wartości własnych na podstawie odwrotnej funkcji przenoszenia

parametry metody: rząd metody Burga: 12, długość FFT: 100, długość próbki: 100ms, wybór co 2 punktu w zakresie od 5 do 45																											razem:	procent poprawności:
wzorzec:	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	r	s	t	u	w	y	z						
ilość błędnie sklasyfikowanych próbek:	0	1	2	1	0	4	2	5	1	4	5	0	0	0	1	2	1	2	3	2	1	3					40	63,64%
liczba badanych próbek:	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5					110	

6. Podsumowanie

Spośród wszystkich zastosowanych metod zdecydowanie najlepsze i bardzo obiecujące rezultaty osiągnięto przy bezpośrednim porównaniu poszczególnych (wszystkich) wartości widma z odpowiadającymi im wartościami widm wzorcowych. W przypadku zastosowania 22 klas wzorców w tym przypadku osiągnięta została skuteczność rozpoznania na poziomie powyżej 90%. Duży wpływ na jakość rozpoznawania miały parametry estymacji widma. Najlepsze rezultaty w metodzie opartej na bezpośrednim porównaniu obwiedni widma osiągnięto dla następujących parametrów: rząd modelu predykcji: 12, rozmiar FFT – 500 punktów, długość próbki – 100ms (1000 punktów dla częstotliwości próbkowania f_s = 10000 Hz). Ponadto lepsze wyniki pozwoliło uzyskać zastosowanie funkcji okna Blackmana [15].

W pozostałych metodach zastosowane zostały różne modyfikacje algorytmu minimalnych wartości własnych macierzy Toeplitza [12]. Zastosowanie tego algorytmu przy klasyfikacji w procesie rozpoznawania alfabetu łukowego dało bardzo

dobre rezultaty [13]. W przypadku sygnału mowy (obrazu widma) musimy zdawać sobie sprawę z większej złożoności obrazu. W przypadku rozpoznawania pisma klasyfikowane były pojedyncze znaki, tymczasem obrazy sygnału mowy w jakiejkolwiek formie przedstawiają sobą złożenie wielu elementów, tworzących razem skomplikowane kształty, znacznie różniące się między sobą, pomimo wypowiadania dźwięków przez tę samą osobę i pomimo jednakowego znaczenia lingwistycznego.

Badania nad zastosowaniem metod opartych na algorytmie minimalnych wartości własnych w odniesieniu do rozpoznawania mowy ciągle trwają. Uzyskana skuteczność jest spowodowana efektem minimalizacji liczby punktów charakterystycznych obrazu i przez to zacieraniem ich cechy dyskryminacyjnej. Kolejną wyraźną przyczyną z pewnością jest fakt, iż widmo częstotliwościowe wielu głosek jest zbyt do siebie podobne.

Można osiągnąć bardzo dobre wyniki w przypadku bezpośredniego porównania składowych widma ze składowymi widma wzorcowego, ale to jeszcze nie pozwala w pełni uniezależnić się od wpływu estymacji widma metodą Burga.

Poprawy wyników rozpoznawania należy oczekiwać w przypadku poszerzenia zestawu cech charakterystycznych obrazu mowy. Należałoby wypróbować inne techniki parametryzacji sygnału mowy. Zestawienie ich, łącznie z wynikami uzyskanymi przez autorów pracy, mogłoby umożliwić uzyskiwanie większej poprawności rozpoznawania.

Takimi technikami są np.: zastosowanie innych metod oceny podobieństwa kształtu badanej obwiedni widma do kształtu obwiedni wzorcowej, czy też zastosowanie przekształcenia widma do postaci, którą nazywamy cepstrum. Opis wspomnianych innowacji jest zawarty m.in. w literaturze: „Digital Speech Processing – Synthesis and Recognition" [16].

Autorzy niniejszego artykułu mają nadzieję, że wyniki ich pracy zostaną w przyszłości wykorzystane do stworzenia jeszcze lepszych i wydajniejszych metod analizy sygnału mowy i że ich praca będzie stanowiła bazę do kolejnych analiz i poszukiwań niezawodnych metod rozpoznawania mowy, w szczególności opartych na algorytmie minimalnych wartości własnych macierzy, utworzonych z charakterystyk obrazów mowy.

Bibliografia

[1] D. J. Burr: *Experiments on Nuural Net Recognition of Spoken and Written Text,* IEEE Transactions on Acoustic, Speech, and Signal Processing, Vol. 36, July 1988.

[2] L. Grad: *Rozpoznawanie słów izolowanych przy wykorzystaniu ukrytych modeli Markowa,* Biuletyn IAiR WAT, nr 12, 2000.

[3] K. Saeed: *Computer Graphics Analysis: A Method for Arbitrary Image Shape Description,* V. 10, Number 2, 2001, pp. 185-194, MGV – International Journal on Machine Graphics and Vision, Institute of Computer Science, Polish Academy of Sciences, Warsaw.

[4] R. W. Schafer, L. R. Rabiner: *System for Automatic Formant Analysis of Voiced Speech,* J. Acoust. Soc. Amer. V. 47, February 1970.

[5] L. Grad: *Obrazowa reprezentacja sygnału mowy,* Biuletyn IAiR WAT, nr 11, 2000.

[6] Cz. Basztura: *Modele analizy i procedury w komputerowym rozpoznawaniu głosów,* Prace naukowe ITiA Politechniki Wrocławskiej, nr 30, 1989.

[7] L. S. Marple: *Digital Spectral Analysis,* Englewood Cliffs, NJ: Prentice Hall, 1987.

[8] A. Kaczanowski: *Przygotowanie sygnału mowy do jego rozpoznawania*, Praca magisterska nr 449, Instytut Informatyki PB, Białystok, 2001.

[9] R. Tadeusiewicz: *Sygnał mowy,* WKiŁ, Warszawa, 1988.

[10] N. Levinson: *The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction,* Journal Math. Phys. V. 25, 1947.

[11] J. Durbin: *Efficient Estimation of Parameters in Moving Average Models,* Biometrika, V. 46, part 1, 2, 1969.

[12] Khalid Saeed: *Experimental Algorithm for Testing The Realization of Transfer Functions*, Proceedings of the Fourteenth IASTED International Conference, Austria, 1995.

[13] R. Niedzielski: *Kryterium do rozpoznawania znaków maszynowych alfabetu łukowego,* Praca magisterska nr 293, Instytut Informatyki PB, Białystok, 1999.

[14] M. Kozłowski: *Klasyfikacja wstępnie przygotowanych obrazów akustycznych izolowanych liter mowy i przygotowanie ich do rozpoznawania.,* Praca magisterska nr 446, Instytut Informatyki PB, Białystok, 2001.

[15] R. G. Lyons: *Wprowadzenie do cyfrowego przetwarzania sygnałów,* WKiŁ, Warszawa, 1999.

[16] Sadaoki Furui: *Digital Speech Processing, Synthesis, and Recognition,* Marcel Dekker, Inc. 2001.

A METHOD FOR SPOKEN-LETTER-IMAGE RECOGNITION

Summary: A new trial on speech recognition from graphical point of view is introduced. Isolated spoken letters are considered. After recording, the speech signal is processed as an image by Power Spectrum Estimation using Burg's method. For feature extraction, classification and hence recognition, the algorithm of minimal eigenvalues of Toeplitz matrices together with other methods of speech processing and recognition are used. A number of examples on applications and comparison are presented in the work. Although the efficiency of the method is not very high, the results are encouraging for algorithm extension to cover word recognition.

Key words: Speech, Image Processing and Recognition, Burg's and Toeplitz Methods

Artykuł zrealizowano w ramach pracy własnej nr W/WI/3/2002

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Khalid Saeed[1], Mariusz Rybnik,
Marek Tabędzki, Marcin Adamski

ALGORYTM DO ŚCIENIANIA OBRAZÓW: IMPLEMENTACJA I ZASTOSOWANIA

Streszczenie: Praca ta prezentuje niektóre zastosowania algorytmu z wcześniejszego artykułu [2]. Zastosowany algorytm do pisma ręcznego jest aplikowany tutaj na obrazach, takich jak fotografie, obrazy o charakterze medycznym i podpisy. Ścienianie jest ważne w wielu zastosowaniach z dziedziny rozpoznawania wzorców, np. kompresja, transmisja lub przechowywanie danych. Otrzymano interesujące wyniki, które są prezentowane w tym artykule. Porównując uzyskane rezultaty z rezultatami innych algorytmów uznano, że są one porównywalne z najnowszymi podejściami w dziedzinie ścieniania liter, słów i zdań pisma ręcznego. Jednak, w przypadku rysunków, podpisów i innych bardziej skomplikowanych obrazów, nasz algorytm daje lepsze rezultaty.

Słowa kluczowe: algorytm ścieniania; ścienianie słów, tekstów i obrazów

1. Wstęp

Ścienianie jest bardzo ważną fazą w rozpoznawaniu wzorców w niemal wszystkich metodach i podejściach do klasyfikacji. Pomimo tego, w większości przypadków, ścienianie nie jest tak precyzyjne jak powinno być, choć autorzy algorytmów deklarują swoje metody jako niemal idealne [2,3,4,5]. Rzeczywiście, są pewne algorytmy, które prowadzą do niemal jednopikselowego ścienionego obrazu [2,4,5]. Te metody wykazują swoją skuteczność w działaniu na piśmie ręcznym, wyrazach lub w pewnych specjalnych zastosowaniach [3,6] ale nie na fotografiach lub pewnych skomplikowanych obrazach zbliżonych do medycznych [2,7]. Ponieważ nie wszyscy autorzy ujawniają szczegóły algorytmów i zazwyczaj przemilczają usterki, które są dyskutowane w zestawieniu [8], porównanie bywa problematyczne. W wielu przypadkach autorzy tej pracy implementowali algorytmy innych,

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok
aidabt@ii.pb.bialystok.pl

opierają się na podstawowych ideach wymienionych w materiałach, i następnie porównywali wyniki ze swoimi w takich samych warunkach. Nasze eksperymenty, niemal we wszystkich rozważanych przypadkach, wykazały dużą efektywność w utrzymywaniu kontynuacji (braku przerw) szkieletu obrazu bez straty podstawowych cech obrazu. To jest bardzo ważne, szczególnie w kompresji, transmisji lub zapisywaniu dużych danych po ścienianiu.

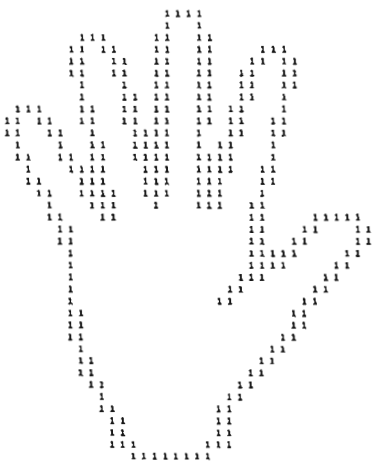
2. Rozważania teoretyczne

W rozdziale tym opisany jest algorytm i jego implementacja. Generalne wytyczne do ścieniania pisma, słów i zdań w różnych językach są dane w [2]. Tutaj przedstawiamy główne fazy algorytmu, które prowadzą do konturu gotowego do rozpoznawania.

2.1. Algorytm ścieniania

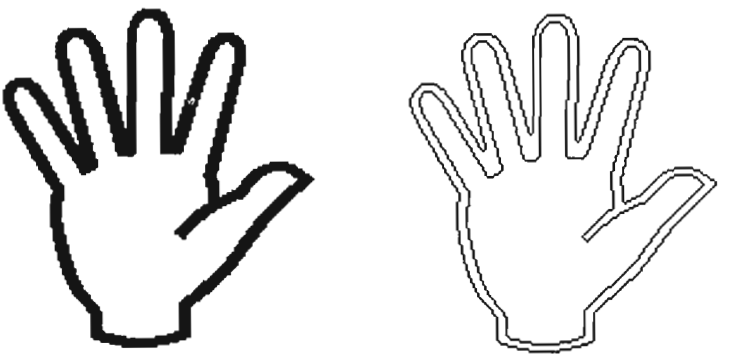
Algorytm jednopikselowego ścieniania prezentowany w [3] jest modyfikowany [2], by otrzymać nieprzerwany obraz. Nieprzerwany obraz definiujemy jako obraz z ciągłym szkieletem, który jest jednopikselowy. To jest podstawa do większości zastosowań algorytmu, którego najważniejsze fazy przedstawiamy poniżej. Do ilustracji jego działania wykorzystamy rysunek ludzkiej dłoni.

1. Obraz jest przedstawiany jako mapa bitowa, czarne piksele są zaznaczane jedynkami:



2. Jedynki konturu (które dotykają zer tła) są zmieniane na dwójki; a te w kątach w trójki. Wtedy otrzymujemy następujący obraz:

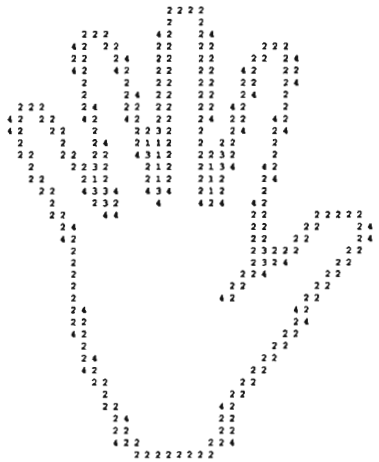
Uwaga: Zauważmy, że ta faza może zakończyć działanie algorytmu, pozostawiając finalny kontur zależnie od okoliczności. W przypadku niektórych zastosowań przetwarzania obrazów to jest podstawowy cel ścieniania. Rysunek 1 pokazuje dłoń razem z konturem po fazie 2:



Rys. 1. Obraz dłoni ścieniony do konturu

Jednak aby pokazać całość metody ścieniania do konturu jednopikselowego, będącego najczęściej pożądanym i stosowanym przypadkiem, rozważmy następne fazy [2].

3. Wyszukajmy piksele, które mają dwóch, trzech lub czterech przylegających do siebie sąsiadów i zmieńmy je na czwórki:



Zauważmy, że jest osiem takich układów w każdym przypadku, zależnie od sposobu w jaki piksele otaczające piksel testowany x są ułożone. Waga każdego z tych pikseli jest brana z następującej tabeli:

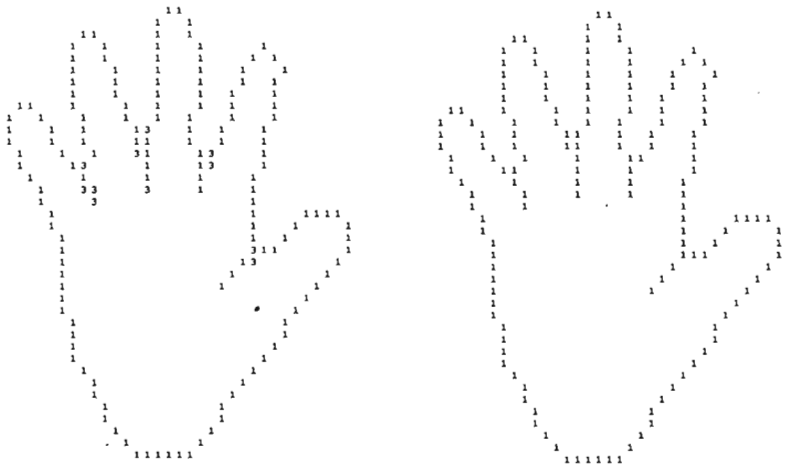
128	1	2
64	x	4
32	16	8

Mamy więc 24 różne możliwości z dwoma, trzema lub czterema sąsiadami.
Suma s sąsiadujących pikseli, z zakresu 0 do 255, decyduje, kiedy usunąć piksel. Następująca tablica usunięć prezentuje sumy, dla których piksele zostają usunięte:

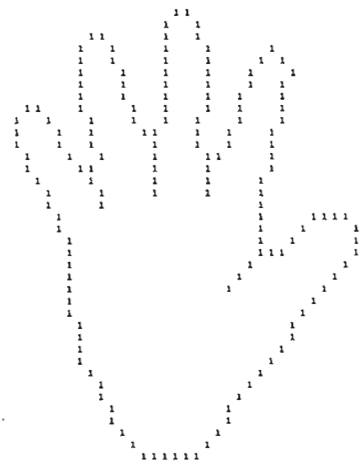
3	5	7	12	13	14	15	20
21	22	23	28	29	30	31	48
52	53	54	55	56	60	61	62
63	65	67	69	71	77	79	80
81	83	84	85	86	87	88	89
91	92	93	94	95	97	99	101
103	109	111	112	113	115	116	117
118	119	120	121	123	124	125	126
127	131	133	135	141	143	149	151
157	159	181	183	189	191	192	193
195	197	199	205	207	208	209	211
212	213	214	215	216	217	219	220
221	222	223	224	225	227	229	231
237	239	240	241	243	244	245	246
247	248	249	251	252	253	254	255

4. Usuwamy czwórki, aby otrzymać następujący kształt:

5. Sprawdzamy czy dwójki i trójki w obrazie powyżej są potrzebne do utrzymania ciągłości obrazu, jeżeli tak, to oznaczamy je jedynkami, by otrzymać:



Całość algorytmu jest zwykle powtarzana aż do otrzymania finalnego obrazu:



2.2. Schemat blokowy algorytmu

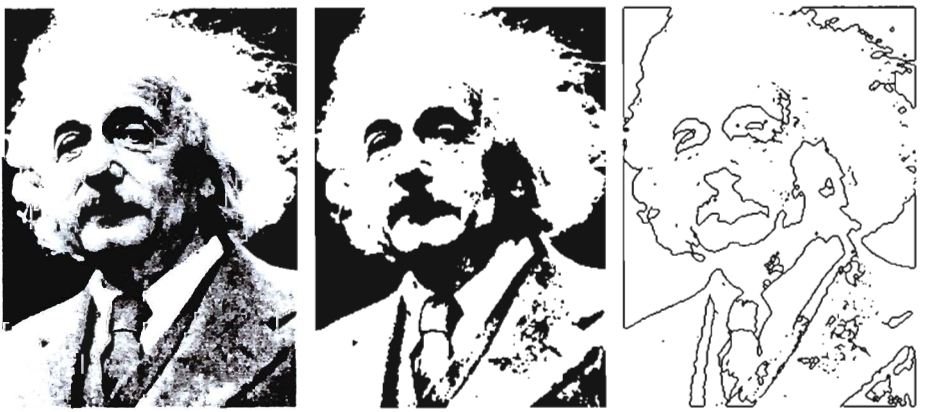
Algorytm został zaimplementowany przy użyciu języka *MFC C++* i nazwany *KMM*. Schemat blokowy programu komputerowego jest przedstawiony w Dodatku. Algorytm testowano przy użyciu wielu przykładów, a otrzymane wyniki były porównywane z innymi podejściami pod względem szybkości, złożoności i kosztu.

3. Przykłady

Przykłady rozważane tutaj różnią się od wymienionych w [2], jako że jesteśmy zainteresowani raczej obrazami niż tekstami. Prezentujemy fotografię Einsteina, podpis i przykład obrazu medycznego.

3.1 Obrazy

Algorytm prezentowany w tej pracy był testowany na wielu obrazach. Rozważano obrazy oraz fotografie – kolorowe i czarno-białe, między innymi przetworzono fotografię Einsteina. Rysunek 2 pokazuje oryginalną fotografię Einsteina, czarno-białą wersję oraz ścieniony kontur. Zauważmy, że ważne cechy obrazu pozostały niezmienione.



Rys. 2. Fotografia Einsteina ścieniona w celu kompresji lub zapisywania danych

3.2 Podpisy

Innym przykładem użytym do prezentacji ekstrakcji wektorów cech jest typowe rozpoznawanie podpisu [9]. Chociaż algorytm użyty w rozpoznawaniu podpisu nie potrzebuje ścieniania przed klasyfikacją, ścienianie upraszcza proces opisywania i klasyfikacji. Rysunek 3 pokazuje próbkę podpisu z jego ścienionym kształtem, wszystkie punkty charakterystyczne pozostały niezmienione.

Rys. 3. Ścienianie podpisu

3.3 Obrazy medyczne

Rozważmy obraz medyczny pokazany na rysunku 4 w trzech różnych formach: kolorowej, czarno-białej i w formie ścienionego szkieletu wewnątrz konturu. To jest jeden z kilku testowanych przykładów ścieniania obrazów medycznych w celu przygotowania do kompresji przed transmisją lub zapisaniem jako ścieniony obraz.

Rys. 4. Przetwarzanie obrazu medycznego

4. Wnioski

Algorytm opisany w [2] jest wystarczający do otrzymania dobrych rezultatów ścieniania do rozpoznawania wzorców [10,11]. Przerwa w ciągłości linii formujących szkielet obrazu pozwala jednak czasem na wyodrębnienie wielu więcej praktycznych cech do klasyfikacji i opisania obrazu, jak to opisano w [3]. Ta metoda jednak nie daje jednopikselowego szkieletu. Algorytm prezentowany w tej pracy spełnia założenia szkieletu jednopikselowego. Warunek ten jest wymagany przez wiele zastosowań w dziedzinie przetwarzania obrazów. Algorytm prezentowany tutaj wykazuje znaczący postęp i w niektórych przypadkach większą efektywność.

Bibliografia

[1] K. Saeed, M. Rybnik, M. Tabedzki: *More Results and Applications about the Algorithm of Thinning Images to One-Pixel-width,* 9th CAIP International Conference on Computer Analysis of Images and Patterns, pp. 601-609, Sept. 5-7, Warsaw 2001, Springer-Verlag Heidelberg: Berlin 2001.

[2] K. Saeed: *Text and Image Processing: Non-Interrupted Skeletonization WSES/IEEE-CSCC'01*, World Scientific and Engineering Society Multi-Conference on Circuits, Systems, Communications and Computers, July 8-15, (Advances in Signal Processing and Computer Technology, pp. 350-354, WSES Press), Crete, 2001.

[3] K. Saeed, R. Niedzielski: *Experiments on Thinning of Cursive-Style Alphabets*, ITESB'99, Inter. Conf. on Information Technologies, June 24-25, Mińsk, 1999.

[4] Yung-Sheng Chen: *The Use of Hidden Deletable Pixel Detection to Obtain Bias-Reduced Skeletons in Parallel Thinning*, Proceedings of 9th ICPR'96 – IEEE, Vol.1, pp. 91-95, Aug. 25-29, Vienna, 1996.

[5] Y.Y. Zhang, P.S.P. Wang: *A Parallel Thinning Algorithm with Two-Subiteration that Generates One-Pixel-Wide Skeletons*, Proceedings of 9th ICPR'96 - IEEE, Vol.2, pp. 457-461, Aug. 25-29, Vienna, 1996.

[6] G. Ososkov, A. Stadnik: *Face Recognition by a new type of Neural Networks*, Advances in Neural Networks and Applications, pp. 304-308, WSES Press, MA 2001.

[7] K. Saeed: *New Approaches for Cursive Languages Recognition: Machine and Hand Written Scripts and Tests*, Invited Paper in Proceedings of World Scientific and Engineering Society WSES-NNA'2001 Conference February 12-14, (Advances in Signal Processing and Computer Technology, pp. 92-97, WSES Press), Tenerife, 2001.

[8] M. Ghuwar and W. Skarbek: *Recognition of Arabic Characters – A Survey*, Polish Academy of Science, Manuscript No.740, Warsaw, 1994.

[9] A. Hodun: *Signature Recognition*, B.Sc. Thesis, Bialystok University of Technology, Bialystok, 2001, Poland.

[10] K. Saeed: *Three-Agent System for Cursive-Scripts Recognition*, Proc. CVPRIP'2000 Computer Vision, Pattern Recognition and Image Processing, 5th Joint Conference on Information Sciences JCIS'2000, Vol.2, pp. 244-247, February 27 – March 3, New Jersey, 2000.

[11] K. Saeed: *A Projection Approach for Arabic Handwritten Characters Recognition*, New Trends and Applications in Computational Intelligence, pp. 106-111, Springer-Verlag Heidelberg: Berlin, 2000.

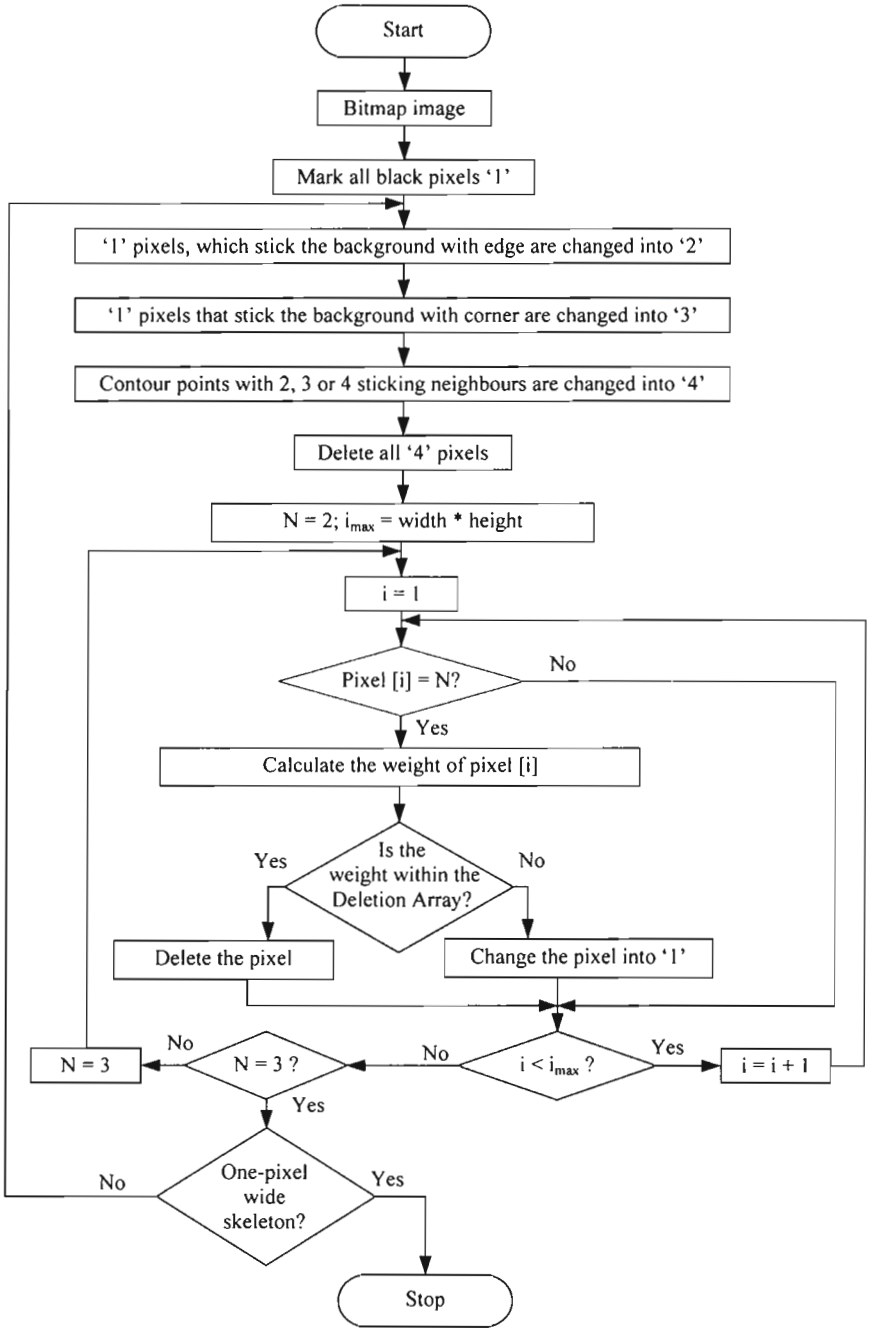
AN ALGORITHM FOR IMAGE THINNING: IMPLEMENTATION AND APPLICATIONS

Summary: The article presents a new algorithm with its computer implementation and applications in texts, pictures and medical organs description. This is very important for a number of applications in pattern recognition, like, for example, data compression, transmission or saving. Some interesting results have been obtained and presented in this paper. Comparing with results of other methods, we can conclude that if it comes to thinning of scripts, words or sentences our method is as good as some of the latest known-to-us approaches. However, when it comes to pictures, signatures or other more complicated images, the given in this work examples may prove better and more precise results than a number of other known methods.

Key words: image processing; image digitalization, skeletonization, thinning

Artykuł zrealizowano w ramach pracy własnej nr W/WI/3/2002

Dodatek: Schemat blokowy algorytmu KMM



ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Walery Sołowjew, Adam Klimowicz[1]

SYNTEZA UKŁADÓW KOMBINACYJNYCH NA JEDNYM UNIWERSALNYM UKŁADZIE PAL Z WYKORZYSTANIEM MONTAŻOWEGO ŁĄCZENIA WYJŚĆ

Streszczenie: W artykule został opisany algorytm syntezy układów kombinacyjnych z łączeniem montażowym wyjść, dopuszczający użycie tylko jednego uniwersalnego układu PAL, a także jego modyfikacje pozwalające zastosować ten algorytm do syntezy na jednym „klasycznym" układzie PAL oraz do syntezy na jednym bloku funkcjonalnym złożonego układu programowalnego. Algorytm wykorzystuje właściwości architektury współczesnych uniwersalnych układów PAL, takie jak różna liczba linii iloczynów podłączona do jednej makrokomórki i możliwość wyboru polaryzacji sygnału wyjściowego. Określono też warunki realizacji systemu funkcji boolowskich przy pomocy tego algorytmu. Wyniki działania algorytmu porównano z innymi znanymi metodami oraz z wynikami uzyskanymi za pomocą systemu MAX+Plus II.

Słowa kluczowe: synteza logiczna, układy kombinacyjne, PLD, projektowanie układów cyfrowych.

1. Wprowadzenie

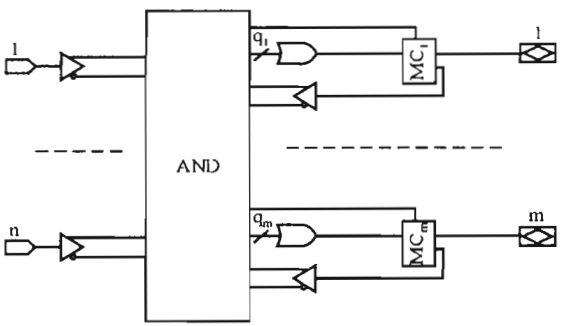
Projektowanie współczesnych systemów cyfrowych opiera się obecnie w dużym stopniu na wykorzystaniu programowalnych układów logicznych PLD (*Programmable Logic Devices*). Dlatego ważnym zadaniem jest znalezienie jak najlepszych metod syntezy układów cyfrowych korzystając z PLD. Każdy system cyfrowy zawiera w sobie część kombinacyjną, tak więc znalezienie optymalnych algorytmów syntezy układów kombinacyjnych na bazie PLD jest zagadnieniem bardzo istotnym i wartym zainteresowania.

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok, e-mail: walsol@ii.pb.bialystok.pl, aklim@ii.pb.bialystok.pl

Metody syntezy logicznej można podzielić na dwupoziomowe i wielopoziomowe. Na potrzeby syntezy logicznej zorientowanej na PLD najbardziej przydatne są metody dwupoziomowe, ze względu na to, że w strukturze PLD sygnał przechodzi przez dwa poziomy logiczne: programowalną matrycę AND i matrycę OR. Wśród metod syntezy dwupoziomowej możemy wyróżnić metody dokładne (klasyczne) opracowane kilkadziesiąt lat temu m.in. przez W. Quine'a [12] oraz E. McCluskey'a [11] oraz metody przybliżone np. Espresso [8]. Metody wielopoziomowe, oparte na faktoryzacji i dekompozycji [1, 2, 3, 4, 5], są najbardziej efektywne do realizacji układów kombinacyjnych na FPGA (*Field Programmable Gate Arrays*). Nie można jednak konkretnie podać, które ze sposobów syntezy są najlepsze do syntezy układów kombinacyjnych na bazie PLD. Najczęściej mają one dość abstrakcyjny charakter i nie są dość efektywne w konkretnych zastosowaniach. Ważne jest to, aby już na wczesnym poziomie syntezy logicznej wykorzystywać specyficzne cechy danej architektury [6, 7, 13], a w szczególności zwracać uwagę na warunki konieczne umożliwiające stosowanie metody, liczbę wykorzystanych elementów układu programowalnego oraz liczbę poziomów układu i, co się z tym wiąże, na parametry czasowe układu.

W artykule zostanie pokazana metoda syntezy układów kombinacyjnych z wykorzystaniem montażowego łączenia wyjść PLD. Metoda ta charakteryzuje się efektywnym wykorzystaniem właściwości uniwersalnych układów PAL (rys. 1), w szczególności:

- możliwości programowania polaryzacji sygnału wyjściowego; do realizacji układu można wykorzystać funkcję lub jej negację, w zależności od złożoności danej postaci funkcji,
- różnej liczby linii iloczynów (termów) podłączonych do poszczególnych makrokomórek, co pozwala realizować proste funkcje na makrokomórkach o małej liczbie związanych z nimi termów, a złożone – na makrokomórkach o dużej liczbie podłączonych do nich termów,
- możliwości wykorzystania wyjściowych makrokomórek jako wejść układu (poprzez sprzężenia zwrotne).



Rys. 1. Struktura układu PAL

2. Założenia wstępne

Niech układ kombinacyjny, zadany zbiorem funkcji boolowskich Y, będzie przedstawiony w postaci sumy iloczynów (SOP – *Sum of Products*) w formie dwóch macierzy: A – zawierającej wartości ze zbioru {0, 1, -} oraz B – zawierającej wartości ze zbioru {0, 1}.

Macierze A i B mają jednakową liczbę wierszy, odpowiadających iloczynom logicznym. Kolumny macierzy A odpowiadają argumentom, a macierzy B – funkcjom zbioru Y. Na przecięciu wiersza i oraz kolumny j macierzy A stawia się jedynkę – jeśli wejściowa zmienna x_j, $x_j \in X$ wchodzi w *i-ty* iloczyn w postaci prostej, zero – jeżeli wchodzi w postaci zanegowanej i kreskę ("-") jeśli x_j nie wchodzi do *i-tego* iloczynu. Na przecięciu wiersza i oraz kolumny j macierzy B stawia się jedynkę, jeśli *i-ty* iloczyn wchodzi w skład funkcji y_j, $y_j \in Y$. Liczbę funkcji boolowskich należących do zbioru Y oznaczmy jako N, natomiast liczbę argumentów funkcji – jako L.

Układ programowalny PAL jest opisany za pomocą następujących parametrów: m – liczby makrokomórek, n – liczby wejść układu PAL oraz zbioru $Q = \{q_1,...,q_k\}$, gdzie q_j – liczba termów podłączonych do makrokomórki MC_j, $q_j \in Q$, $j=1...m$ (rys.1). Przy realizacji zbioru funkcji boolowskich na jednym uniwersalnym PAL, konieczne jest, spełnienie następujących warunków (1):

$$N \leq m; \qquad (1)$$
$$L \leq n + m - N.$$

Przy naruszeniu też warunku (2):

$$Q(\tilde{y}_i) \leq q_j, \qquad (2)$$

zadanie może być rozwiązane poprzez zbudowanie układu jednopoziomowego, w którym złożone funkcje otrzymujemy drogą złączenia (przez OR) potrzebnych wyjść PAL.

3. Metoda syntezy

Istota przedstawionej metody syntezy jest następująca. Jeżeli do realizacji którejkolwiek z funkcji nie wystarcza termów podłączonych do jednej makrokomórki PAL, wykorzystuje się drugą makrokomórkę, jeśli termów jest dalej za mało bierze się następną makrokomórkę, itd. Zbiór funkcji boolowskich może być realizowany na PAL, jeżeli dla każdej funkcji y_i, $y_i \in Y$ można znaleźć wydzieloną grupę makrokomórek (może być to też tylko jedna makrokomórka), dla której cał-

kowita liczba termów jest nie mniejsza niż wielkości $Q(\tilde{y}_i)$, ($\tilde{y}_i$ – jedna z funkcji y_i lub $\bar{y}_i$). Następnie wyjścia grupy makrokomórek przeznaczone do realizacji funkcji y_i łączy się przy pomocy montażowego OR. Połączenie wyjść PAL posiada następujące cechy:

- jeśli do tego stosuje się połączenie montażowe, to realizowana funkcja powinna być przedstawiona w postaci zanegowanej, ponieważ montażowe połączenie OR pracuje tylko w logice inwersyjnej;
- w przypadku logiki prostej do połączenia wyjść PAL należy stosować elementy logiczne;
- jeśli postać realizowanej funkcji (y_i lub $\bar{y}_i$) wybiera się w zależności od najmniejszej wartości $Q(y_i)$ lub $Q(\bar{y}_i)$, to elementy logiczne, stosowane dla łączenia wyjść PAL przez OR, powinny mieć możliwość programowania logicznego poziomu wyjściowego sygnału dla otrzymania koniecznej postaci funkcji.

Niech wszystkie realizowane funkcje przedstawione będą w postaci zanegowanej; możliwe jest także wykorzystanie montażowego łączenia wyjść PAL. Oczywiste jest, że realizacja zbioru funkcji boolowskich na jednym PAL możliwa jest w przypadku spełnienia warunku:

$$\sum_{i=1}^{N} Q(\tilde{y}_i) \le \sum_{q_j \in Q} q_j. \qquad (3)$$

Algorytm syntezy układu kombinacyjnego na jednym uniwersalnym PAL ma następującą postać:

Algorytm 1.

1. Przyporządkowuje się zmienne wejściowe zewnętrznym wyprowadzeniom PAL. Wykorzystuje się przy tym w pierwszym rzędzie wejścia PAL, potem wyjścia, zaczynając od tych, które mają najmniejszą liczbę związanych z nimi termów (pozostawia się największe możliwości do realizacji funkcji wyjściowych).
2. Dla każdej funkcji y_i, $y_i \in Y$ znajduje się postać sumy iloczynów (SOP) jej negacji $\bar{y}_i$.
3. W celu utworzenia zbioru realizowanych funkcji Y^* kolejno rozpatruje się funkcje y_i, $i=1...N$. Jeśli dla którejś funkcji jest spełniona zależność:

   $$Q(y_i) < Q(\bar{y}_i) \qquad (4)$$

   oraz spełniony jest warunek (2), to do zbioru Y^* dołącza się funkcję y_i, inaczej do zbioru Y^* dołącza się funkcję $\bar{y}_i$.

4. Wykorzystuje się makrokomórki PAL do realizacji funkcji należących do zbioru Y^*. Do tego celu dla każdej funkcji $\tilde{y}_i$, $\tilde{y}_i \in Y^*$ znajduje się najmniejszą grupę wolnych makrokomórek, których suma ilości termów jest nie mniejsza niż $Q(\tilde{y}_i)$. Aby zminimalizować liczbę elementów grupy, na początku znajdujemy grupę złożoną z jednej makrokomórki, potem z dwóch, trzech, itd. Spośród grup z jednakową liczbą elementów wybiera się grupę, dla której różnica między całkowitą liczbą termów makrokomórek i wartością $Q(\tilde{y}_i)$ jest minimalna. Dla funkcji, które weszły do zbioru Y^* w prostej postaci, zawsze wybiera się pojedynczą makrokomórkę. Jeśli dla przynajmniej jednej funkcji nie udało się znaleźć pasującej grupy makrokomórek, to zbiór funkcji boolowskich nie może być realizowany na PAL z zadanymi parametrami, wykonuje się zatem przejście do punktu 6.
5. Realizowane funkcje przyporządkowuje się odpowiednim wyjściom PAL.
6. Koniec.

Zauważmy, że w rezultacie wykonania punktów 3 i 4 funkcje w postaci prostej będą zawsze realizowane na jednej makrokomórce PAL, a funkcje w postaci zanegowanej mogą być realizowane zarówno na jednej jak i na kilku makrokomórkach. W ostateczności jeśli jakaś makrokomórka PAL realizuje funkcję w postaci prostej, to zanegowaną postać funkcji tworzy się drogą programowania logicznego poziomu wyjściowego sygnału. Jeśli funkcja realizowana jest na kilku makrokomórkach, to odpowiadające wyjścia PAL łączy się za pomocą montażowego OR.

Przedstawiony powyżej algorytm można także z powodzeniem zastosować przy *syntezie układów kombinacyjnych zorientowanych na „klasyczne" układy PAL* [14], „Klasyczne" układy PAL, w odróżnieniu od uniwersalnych, nie mają makrokomórek z dwoma sprzężeniami zwrotnymi i z każdą makrokomórką związane jest taka sama liczba q termów. Oprócz tego „klasyczne" PAL nie pozwalają programować poziomów wyjściowych sygnałów logicznych. Dlatego, jeśli założyć $q_j := q$ dla wszystkich $q_j \in Q$, a także w punkcie 3 algorytmu założyć $Y^* := \{\bar{y}_1, \ldots, \bar{y}_N\}$, to pokazaną tu metodę syntezy można także przyjąć jako sposób realizacji układów kombinacyjnych na podstawie „klasycznego" PAL.

Kolejna modyfikacja tego algorytmu pozwala na zastosowanie go do *syntezy układów kombinacyjnych na jednym bloku funkcjonalnym złożonego układu PLD* (CPLD – *Complex Programmable Logic Devices*)[14]. Układ CPLD opisują następujące parametry: liczba bloków funkcjonalnych E, liczba wejść dedykowanych d_I, liczba makrokomórek w jednym bloku funkcjonalnym m, maksymalna liczba termów jaka może być podłączona do jednej makrokomórki q_{max} oraz całkowita liczba termów w danym bloku funkcjonalnym q_Σ. Realizacja układu kombinacyjnego na jednym bloku funkcjonalnym CPLD polega na wykorzystaniu do realizacji

wszystkich funkcji boolowskich makrokomórek jednego bloku funkcjonalnego. Warunki możliwości trywialnej realizacji układu funkcji boolowskich na jednym bloku funkcjonalnym CPLD mają postać:

$$\begin{aligned} &L \le d_l + (E\text{-}1)\cdot m + m - N; \\ &L \le n; \\ &N \le m - max(0, L - d_l - (E\text{-}1)\cdot m); \\ &Q(\tilde{y}_i) \le q_{max}, \text{ dla wszystkich } y_i, y_i \in Y; \\ &\sum_{i=1}^{N} Q(\tilde{y}_i) \le q_{\Sigma}, \end{aligned} \tag{5}$$

gdzie:
$Q(\tilde{y}_i) = min(Q(y_i), Q(\bar{y}_i)), y_i \in Y$ – minimalna liczba termów, wystarczająca do realizacji funkcji y_i.

Pierwsza nierówność w warunku (5) zapewnia dostateczną liczbę zewnętrznych wyprowadzeń CPLD dla przyjęcia wartości argumentów funkcji; druga – dostateczną liczbę wejść bloku funkcjonalnego dla przyjęcia wartości argumentów funkcji; trzecia – liczbę wyprowadzeń bloku funkcjonalnego dla realizacji wartości funkcji; czwarta – liczbę termów jednej makrokomórki dla realizacji dowolnej funkcji; piąta – liczbę termów bloku CPLD do realizacji wszystkich funkcji zbioru Y.

Zauważmy, że jeżeli termy CPLD rozdzielane są klastrami (*ang. cluster*), to słuszne są następujące zależności (6):

$$\begin{aligned} &Q(\tilde{y}_i) :=]Q(\tilde{y}_i) / q_{CL}[; \\ &q_{max} :=]q_{max} / q_{CL}[; \\ &q_{\Sigma} :=]q_{\Sigma} / q_{CL}[, \end{aligned} \tag{6}$$

gdzie:
q_{CL} – liczba termów wchodząca w skład jednego klastra,
$]A[$ – najmniejsza liczba całkowita, większa lub równa A.

Dla każdej funkcji y_i należy wprowadzić pojęcie rangi funkcji Q_i, $Q_i = min(Q(y_i), Q(\bar{y}_i)), y_i \in Y$. W przypadku, gdy dla którejkolwiek z funkcji y_i, $y_i \in Y$, naruszono warunek:

$$Q_i \le q_{max}, \qquad i=1...N, \tag{7}$$

wówczas układ kombinacyjny może być realizowany na jednym bloku funkcjonalnym CPLD drogą połączenia wyjść przy pomocy montażowego OR. Do tego można zastosować poniższy algorytm (modyfikację algorytmu 1).

Algorytm 2.

1. Przyporządkowuje się zmienne wejściowe zewnętrznym wyprowadzeniom CPLD, przy czym w pierwszej kolejności wykorzystuje się dedykowane wejścia CPLD, potem dowolne dwukierunkowe wyprowadzenia innych bloków funkcjonalnych, a na końcu dwukierunkowe wyprowadzenia bloku funkcjonalnego.
2. Dla każdej funkcji y_i, $y_i \in Y$ znajduje się postać sumy iloczynów jej negacji $\bar{y}_i$.
3. Tworzy się zbiór realizowanych funkcji Y^*. Do tego kolejno rozpatruje się funkcje y_i, $i=1...N$. Jeśli dla którejś funkcji ma miejsce zależność:

$$Q(y_i) < Q(\bar{y}_i) \tag{8}$$

i spełnione są warunki:

$$\begin{aligned} &|X_t| \le n; \\ &|Y_t| \le m; \\ &Q_t \le q_\Sigma, \end{aligned} \tag{9}$$

gdzie:

$X_t = \bigcup_{y_i \in Y_t} X(y_i)$; $Q_t = \sum_{y_i \in Y_t} Q(\tilde{y}_i)$; $X(y_i)$ – zbiór argumentów funkcji y_i, $y_i \in Y$.

to do zbioru Y^* włącza się funkcję y_i; w innym przypadku do zbioru Y^* włącza się funkcję $\bar{y}_i$.

4. Wykonuje się przyporządkowanie makrokomórek do realizacji funkcji zbioru Y^*. Do tego celu dla każdej funkcji $\tilde{y}_i$, $\tilde{y}_i \in Y^*$ przyporządkowuje się minimalną grupę z k_i wolnych makrokomórek, których liczba termów jest wystarczająca do realizacji funkcji $\tilde{y}_i$, gdzie:

$$k_i =]\, Q_i / q_{max} [. \tag{10}$$

Aby zminimalizować liczbę elementów grupy, na początku znajduje się grupę złożoną z jednej makrokomórki, potem z dwóch, z trzech itd. Pośród grup z jednakową liczbą elementów wybiera się grupę, dla której różnica między sumaryczną liczbą termów makrokomórek i wielkością Q_i jest minimalna. Dla funkcji, wchodzących do zbioru Y^* w postaci prostej, zawsze wybierana jest pojedyncza makrokomórka. Jeśli dla którejkolwiek funkcji nie udało się znaleźć pasującej grupy makrokomórek, to zbiór funkcji boolowskich nie może być realizowany na CPLD z zadanymi parametrami. Zatem wykonuje się przejście do punktu 6.
5. Realizowane funkcje przyporządkowuje się odpowiednim wyjściom CPLD.
6. Koniec.

4. Badania eksperymentalne

Do badań eksperymentalnych wykorzystano oprogramowanie, będące implementacją przedstawionej w artykule metody syntezy. Oprogramowanie to, jako dane wejściowe przyjmuje pliki opisujące układ kombinacyjny w standardzie systemu ESPRESSO [9], natomiast na wyjściu otrzymywane są pliki w formatach systemów automatycznego projektowania układów cyfrowych na bazie PLD, a w szczególności: MAX+Plus II, CUPL, Mach XL, Xilinx Xact. Program umożliwia wybór układu PLD do syntezy z gotowej biblioteki oraz samodzielne tworzenie teoretycznych układów PLD.

Badania przeprowadzono w dwóch etapach. W pierwszym etapie przeanalizowano zależności między danymi wejściowymi a wynikami pośrednimi i końcowymi oraz porównano wyniki działania algorytmu z jego modyfikacją dla „klasycznych" PAL. oraz z *algorytmem syntezy układów jednopoziomowych* [13]. Drugi etap polegał na zastosowaniu plików wyjściowych programu w pakiecie MAX+Plus II. Porównano wyniki otrzymane przy zastosowaniu algorytmu z łączeniem wyjść PAL przez montażowe OR z wynikami uzyskanymi z MAX+Plus II.

Jako dane wejściowe do badania algorytmu posłużyły standardowe przykłady testowe opisane w pracy [9]. Pod uwagę brano następujące parametry:

- ogólną liczbę elementarnych koniunkcji opisujących funkcje w postaci prostej (*p(Y)*), zanegowanej (*p(!Y)*) oraz w przypadku połączonych obu typów funkcji (*p(~Y)*),
- koszt (C_q) wyrażony liczbą wykorzystanych makrokomórek (tab. 1),
- liczbę funkcji realizowanych w postaci prostej (%y) i zanegowanej (%!y), wyrażoną w procentach (tab. 2).

Badania przeprowadzono dla teoretycznych układów PAL, mających nieograniczoną liczbę makrokomórek oraz wejść, tak aby zapewnić realizację wszystkich badanych przykładów niezależnie od ich złożoności. Pod uwagę brano różne liczby termów podłączonych do jednej makrokomórki (q = 4, 8, 16, 32).

Tabela 1

Wyniki syntezy uzyskane przy pomocy algorytmu wykorzystującego łączenie wyjść przez OR

Nazwa	Argumenty	Funkcje	Liczba iloczynów			Koszt [MC]			
pliku	L	N	p(Y)	p(!Y)	p(~Y)	C4	C8	C16	C32
5xp1	7	10	74	77	151	23	15	11	10
9sym	9	1	86	72	158	18	9	5	3
Alu4	14	8	631	529	1160	134	66	36	21
Apex1	45	45	902	1528	2430	369	196	111	64
Apex2	39	3	1065	445	1510	112	57	29	15
Apex3	54	50	628	908	1536	205	113	75	61
Apex4	9	19	1004	1068	2069	273	141	71	43
B12	15	9	53	34	87	12	9	9	9
Bw	5	28	110	130	240	36	28	28	28
Clip	9	5	148	148	296	39	21	12	7
Con1	7	2	9	9	18	2	2	2	2
Cordic	23	2	914	923	1837	232	116	59	30
Cps	24	109	596	1823	2419	312	161	117	102
Duke2	22	29	200	426	626	110	54	31	29
E64	65	65	65	2145	2210	65	65	65	65
Ex1010	10	10	452	888	1337	226	117	60	30
Ex5	8	63	304	342	646	74	64	63	63
inc	7	9	44	56	100	16	11	9	9
Misex1	8	7	32	45	77	12	7	7	7
Misex2	25	18	29	142	171	19	18	18	18
Misex3	14	14	1232	644	1876	165	86	42	22
Misex3c	14	14	222	390	612	99	53	30	21
Pdc	16	40	138	330	468	76	48	41	40
Rd53	5	3	31	36	67	10	5	3	3
Rd73	7	3	141	141	282	36	19	10	6
Rd84	8	4	283	274	555	68	35	18	10
Sao2	10	4	73	74	147	20	11	5	4
Seq	41	35	1399	1787	3186	445	233	124	71
Spla	16	46	458	787	1245	177	97	58	49
Squar5	5	8	29	36	65	11	8	8	8
T481	16	1	481	360	841	90	45	23	12
Table3	14	14	530	967	1495	246	128	64	34
Table5	17	15	550	1120	1670	282	140	72	38
Vg2	25	8	110	257	367	66	31	15	10
Xor5	5	1	16	16	32	4	2	1	1
Z5xp1	7	10	74	77	151	23	15	11	10
Z9sym	9	1	86	72	157	18	9	5	3
Średnia	**17,14**	**19,27**	**356,73**	**516,38**	**872,81**	**111,49**	**60,41**	**36,43**	**25,89**

Tabela 2

Średnia liczba funkcji realizowanych w postaci prostej i w postaci inwersyjnej

q=4		q=8		q=16		q=32	
%y	%!y	%y	%!y	%y	%!y	%y	%!y
23%	**76%**	**33%**	**67%**	**40%**	**60%**	**44%**	**56%**

Analizując otrzymane wyniki można zauważyć, co następuje:

- wzrost wartości q (liczby termów podłączonych do jednej makrokomórki) powoduje wzrost liczby funkcji realizowanych w postaci prostej, a co za tym idzie wykorzystanie do realizacji poszczególnych funkcji pojedynczych makrokomórek.
- koszt maleje wraz ze wzrostem q, przy czym nie jest to spadek liniowy (różnica, wraz ze wzrostem q - maleje).

Jako algorytmów porównawczych użyto metody realizacji poprzez złączenie wyjść układu przez montażowe OR dla „klasycznych" układów PAL, w której założono, że każda funkcja będzie realizowana w postaci zanegowanej oraz algorytmu syntezy układów jednopoziomowych. Wyniki pokazano w tabeli 3, gdzie K_q oznacza koszt układu realizowanego algorytmem dla klasycznych PAL (q = 4, 8, 16, 32), natomiast $\Delta K_q\%$ oznacza różnicę procentową między algorytmami.

$$\Delta K_q\% = \frac{K_q - C_q}{\max(K_q, C_q)} \tag{11}$$

Tabela 3

Porównanie algorytmu z wykorzystaniem łączenia wyjść przez OR z wersją dla „klasycznych" układów PAL

Nazwa	K4	ΔK4%	K8	ΔK8%	K16	ΔK16%	K32	ΔK32%
5xp1	24	4%	15	0%	11	0%	10	0%
9sym	18	0%	9	0%	5	0%	3	0%
Alu4	134	0%	69	4%	37	3%	21	0%
Apex1	397	7%	211	7%	117	5%	76	16%
Apex2	112	0%	57	0%	29	0%	15	0%
Apex3	238	14%	126	10%	78	4%	61	0%
Apex4	275	1%	143	1%	76	7%	44	2%
B12	13	8%	9	0%	9	0%	9	0%
Bw	45	20%	29	3%	28	0%	28	0%
Clip	39	0%	21	0%	12	0%	7	0%
Con1	3	33%	2	0%	2	0%	2	0%
Cordic	232	0%	116	0%	59	0%	30	0%
Cps	506	**38%**	290	**44%**	192	**39%**	114	**11%**
Duke2	119	8%	68	21%	43	28%	29	0%

c.d. tabeli 3

E64	65	0%	65	0%	65	0%	65	0%
Ex1010	227	0%	117	0%	60	0%	30	0%
Ex5	102	27%	64	0%	63	0%	63	0%
inc	17	6%	11	0%	9	0%	9	0%
Misex1	13	8%	7	0%	7	0%	7	0%
Misex2	41	**54%**	23	**22%**	18	**0%**	18	**0%**
Misex3	165	0%	86	0%	44	5%	23	4%
Misex3c	102	3%	54	2%	32	6%	22	5%
Pdc	100	24%	60	20%	44	7%	40	0%
Rd53	10	0%	6	17%	3	0%	3	0%
Rd73	36	0%	19	0%	10	0%	6	0%
Rd84	70	3%	36	3%	19	5%	10	0%
Sao2	20	0%	11	0%	6	17%	4	0%
Seq	460	3%	243	4%	129	4%	74	4%
Spla	213	17%	116	16%	72	19%	49	0%
Squar5	12	8%	9	11%	8	0%	8	0%
T481	90	0%	45	0%	23	0%	12	0%
Table3	246	0%	128	0%	68	6%	37	8%
Table5	285	1%	145	3%	76	5%	43	12%
Vg2	66	0%	37	16%	20	25%	12	17%
Xor5	4	0%	2	0%	1	0%	1	0%
Z5xp1	24	4%	15	0%	11	0%	10	0%
Z9sym	18	0%	9	0%	5	0%	3	0%
Średnia	**122,73**	**8%**	**66,84**	**6%**	**40,30**	**5%**	**26,97**	**2%**

Z przeprowadzonych badań wynika, że algorytm z łączeniem wyjść przez OR jest o 5% lepszy pod względem kosztu od modyfikacji dla „klasycznych" PAL (największe wartości sięgają 54% dla układu *Misex2*). Można tutaj zauważyć zależność pomiędzy różnicą kosztów obu algorytmów a liczbą wykorzystanych funkcji w postaci prostej. Różnica ta jest większa od średniej (tzn. od 5%) wówczas, gdy co najmniej połowa funkcji jest realizowana w postaci prostej. Różnica kosztów maleje wraz ze wzrostem parametru q, stąd wniosek, że przedstawiony algorytm syntezy ma dobre właściwości dla PAL o małych wartościach parametru q. W algorytmie syntezy układów jednopoziomowych możliwość realizacji układu zależy od parametrów (n, m) konkretnej struktury PAL, więc przy założonych parametrach PAL ($n=65$, $m=10$) duża część testów nie mogła być zrealizowana. W przypadku zrealizowanych testów koszt był taki sam jak dla badanego algorytmu syntezy z wykorzystaniem łączenia przez OR.

Praktyczne testy wykonano przy pomocy pakietu MAX+Plus II. Do testowania wykorzystano układ *EP1810* z rodziny CLASSIC firmy Altera o następujących

parametrach: $n=16$, $m=48$, $q_j=8$ [15]. Do późniejszej analizy brano pod uwagę następujące rezultaty (tab. 4):

- koszt, wyrażony liczbą wykorzystanych makrokomórek układu, oznaczony odpowiednio: dla MAX+Plus II – C_M, dla algorytmu z łączeniem przez OR – C_{OR} oraz różnicę między tymi parametrami $\Delta C\%$ wyrażoną w procentach, obliczaną analogicznie do wzoru (11),
- szybkość działania układu, czyli opóźnienie (w nanosekundach), oznaczone odpowiednio: dla MAX+Plus II – D_M, dla algorytmu z łączeniem przez OR – D_{OR} oraz różnicę między tymi parametrami $\Delta D\%$, wyrażoną w procentach, obliczaną analogicznie do wzoru (11),
- liczbę użytych układów programowalnych, oznaczoną odpowiednio: dla MAX+Plus II – F_M, dla algorytmu z łączeniem przez OR – F_{OR} oraz różnicę między tymi parametrami $\Delta F\%$ wyrażona w procentach, obliczaną analogicznie do wzoru (11).

W trakcie przeprowadzania testów dopuszczono zastosowanie więcej niż jednego układu programowalnego, w celu sprawdzenia, czy można zastosować przedstawiony tu algorytm do syntezy na kilku układach programowalnych.

Stosując przetworzone przez oprogramowanie pliki wejściowe zrealizowano o 20% więcej układów testowych niż korzystając tylko z samego pakietu, a sam proces obliczeń trwał znacznie krócej. Podczas przeprowadzonych badań stwierdzono, że algorytm z wykorzystaniem łączenia wyjść przez OR jest lepszy od MAX+Plus II o 17% pod względem kosztu oraz liczby użytych układów i o 36% pod względem szybkości działania układu. W najlepszym wypadku opisany w tym artykule algorytm był pod względem kosztu lepszy od MAX+Plus II aż o 85% (czyli ponad 6 razy), a pod względem szybkości lepszy o 93% (15 razy lepszy) dla układu *Ex1010*. Ogólnie dla 58% realizowanych układów algorytm przyniósł poprawę zarówno pod względem kosztów jak i szybkości działania, a zaledwie dla 16% układów (4 przypadki) zaobserwowano słabsze wyniki. Najgorszy wynik otrzymano dla układu *Vg2*, gdzie opisywany algorytm był gorszy od MAX+Plus II o 42%. Jest to spowodowane zapewne faktem, że pakiet MAX+Plus II zastosował w tym wypadku algorytm syntezy wielopoziomowej, korzystniejszy w wypadku danego układu funkcji boolowskich. Zastosowanie przedstawionego tu algorytmu syntezy znacznie usprawniło pracę z pakietem MAX+Plus II.

Tabela 4

Porównanie wyników uzyskanych dla algorytmu z łączeniem wyjść przez OR i uzyskanych dla pakietu MAX+Plus II

Nazwa	Koszt (MC)			Liczba układów			Opóźnienie		
pliku	CM	COR	ΔC%	FM	FOR	ΔF%	DM	DOR	ΔD%
5xp1	21	15	29%	1	1	0%	32	20	38%
B12	9	9	0%	1	1	0%	20	20	0%
Bw	28	28	0%	1	1	0%	20	20	0%
Clip	30	21	30%	1	1	0%	32	20	38%
Con1	2	2	0%	1	1	0%	20	20	0%
Cps	235	161	31%	7	4	43%	44	22	50%
Duke2	51	54	-6%	2	2	0%	34	22	35%
Ex1010	770	117	**85%**	17	3	**82%**	304	20	**93%**
Ex5	65	64	2%	2	2	0%	20	20	0%
inc	14	11	21%	1	1	0%	32	20	38%
Misex1	7	7	0%	1	1	0%	20	20	0%
Misex2	18	18	0%	1	1	0%	20	22	-9%
Misex3c	99	53	46%	3	2	33%	84	20	76%
Rd53	9	5	44%	1	1	0%	32	20	38%
Rd73	22	19	14%	1	1	0%	32	20	38%
Rd84	113	35	69%	4	1	75%	104	20	81%
Sao2	14	11	21%	1	1	0%	32	20	38%
Seq	407	233	43%	19	8	58%	122	22	82%
Squar5	8	8	0%	1	1	0%	20	20	0%
Table3	114	128	-11%	5	3	40%	54	20	63%
Table5	101	140	-28%	12	4	67%	42	22	48%
Vg2	18	31	**-42%**	1	1	**0%**	34	22	**35%**
Xor5	3	2	33%	1	1	0%	32	20	38%
Z5xp1	21	15	29%	1	1	0%	32	20	38%
Średnia	**90,79**	**49,46**	**17%**	**3,58**	**1,83**	**17%**	**50,75**	**20,50**	**36%**

5. Podsumowanie

Z przeprowadzonej analizy można wyciągnąć wniosek, że algorytm z możliwością łączenia wyjść za pomocą montażowego połączenia OR nadaje się do realizacji układów szybkich ze względu na wprowadzenie tylko jednego poziomu logiki. Opóźnienie dla tej metody jest stałe (spowodowane wykorzystaniem tylko jednego poziomu logiki plus opóźnienie związane z połączeniem montażowym OR). Za pomocą tego algorytmu, jak wynika z badań, realizuje się szerszy obszar układów kombinacyjnych niż za pomocą systemu MAX+Plus II. Wadą tego algorytmu jest to, że potrzebuje on dość dużo wyprowadzeń zewnętrznych układu PLD,

a także wymusza konieczność stosowania wyjść z otwartym kolektorem (dla logiki CMOS – odpowiednio – z otwartym drenem) oraz dodatkowych rezystorów umożliwiających montażowe łączenie wyjść.

Dalszy rozwój badań powinien pójść w kierunku wykorzystania do łączenia części funkcji wyjściowych drugiego poziomu PLD zamiast łączenia montażowego, a także zastosowania na potrzeby PLD algorytmów syntezy wielopoziomowej, które mogłyby dostatecznie dobrze wykorzystać w swym działaniu wszelkie cechy szczególne architektury układów programowalnych.

Literatura

[1] Brayton R.K., McMullen C.: *Decomposition and factorization of Boolean expressions*; In The Proc. Of the Int. Symposium on Circuits and Systems (ISCAS-82), April 1982, pp. 49-55.

[2] Brayton R.K.: *Factoring Logic Functions*; IBM Journal of Research and Development. Vol. 31, No. 2, March 1987, pp. 187-198.

[3] Brzozowski J.A., Łuba T.: *Decomposition of Boolean Functions Specified by Cubes. – Part 1: Theory of Serial Decomposition Using Blankets*; Research Report CS-97-01, University of Waterloo, Canada, 1997, REVISED, October 1998.

[4] Łuba T. *Multi-Level Logic Synthesis Based on Decomposition*; Microprocesors and Microsystems – 1994, Vol. 18, a 8, p. 429-437.

[5] Rawski M., Nowicka M., Łuba T.: *Algorytm dekompozycji nierozłącznej w odwzorowaniu technologicznym dla układów FPGA*; Materiały I Krajowej Konferencji Naukowej – Reprogramowalne Układy Cyfrowe, Szczecin, 12-13 marca 1998, s. 83-89.

[6] Kania D.: *Wybór sposobu realizacji wielowyjściowych funkcji logicznych w strukturach CPLD typu PAL*; Materiały III Krajowej Konferencji Naukowej – Reprogramowalne Układy Cyfrowe, Szczecin, 10-11 kwietnia 2000, s. 89-96.

[7] Kania D.: *Realizacja układów kombinacyjnych w strukturach MACH*; Kwartalnik Elektroniki i Telekomunikacji, 2001, 47, z. 1, s. 65-74.

[8] Brayton R.K., Hatchel G.D., McMullen C.T., Sangiovanni-Vincentelli A.L.: *Logic minimization algorithms for VLSI synthesis*; Kluwer Academic Publisher, Boston 1984

[9] Yang S.: *Logic Synthesis and Optimization Benchmarks User Guide*; Microelectronics Center of North Carolina, Research Triangle Park, NC, 1991

[10] Micheli G.: *Synteza i optymalizacja układów cyfrowych*; WNT, 1998.

[11] McCluskey E.: *Minimization of Boolean Functions*; The Bell System Technical Journal, November 1956, Vol 35, pp. 1417-1444.

[12] Quine W.: *The Problem of Simplifying Truth Functions*; American Mathematical Monthly, 1952, Vol. 59, pp. 521-531.
[13] Sołowiew W., Bułatowa I.: *Synteza jednopoziomowych układów kombinacyjnych na PLD*; Materiały III Krajowej Konferencji Naukowej – Reprogramowalne Układy Cyfrowe, Szczecin, 10-11 kwietnia 2000, s. 39-45.
[14] Sołowiew W.: Projektowanie systemów cyfrowych na bazie programowalnych układów logicznych; Wyd. „Gorąca Linia – Telekom", Moskwa, 2001.
[15] Atera Corp.: *Data Book*. San Jose, CA, 1996

SYNTHESIS OF COMBINATORIAL LOGIC ON SINGLE PAL DEVICE USING WIRED-OR METHOD OF PAL OUTPUTS JOINING

Summary: This article contains a description of an algorithm of synthesis of combinatorial logic schemes, which uses wired-OR method of joining outputs, limited to use only one universal PAL device and some modifications, which allow to use this algorithm to synthesis on single "classic" PAL device and one functional block of complex programmable device. This algorithm uses features of modern universal PAL devices, such as different number of terms connected to single macrocell and possibility of selection of output signal polarity. Conditions allowing to realize boolean function system using this algorithm are described. Work results are compared to results given from other known methods and given from MAX+Plus II system.

Key words: logic synthesis, combinatorial logic, PLD, digital systems design

Artykuł zrealizowano w ramach pracy badawczej własnej W/WI/7/02.

ZESZYTY NAUKOWE POLITECHNIKI BIAŁOSTOCKIEJ 2002
Informatyka – Zeszyt 1

Jarosław Stepaniuk[1], Leszek Góralczuk

ALGORTYM GENEROWANIA REGUŁ PIERWSZEGO RZĘDU WYKORZYSTUJĄCY METODY ZBIORÓW PRZYBLIŻONYCH

Streszczenie: W pracy przedstawiono algorytm generowania reguł pierwszego rzędu, tzn. zależności, które w poprzedniku mają koniunkcję formuł atomowych bądź ich negacji a w następniku formułę atomową. Technikę zbiorów przybliżonych wykorzystano w procesie doboru literałów mogących wchodzić w skład przesłanki generowanej reguły. Kryterium doboru opiera się na tym, aby reguła po dołączeniu do jej przesłanki kandydującego literału jak najlepiej rozróżniała przykłady pozytywne i negatywne, które do tej pory nie były rozróżnialne.

Słowa kluczowe: zbiory przybliżone, indukcyjne programowanie logiczne

1. Wstęp

Zastosowanie Zbiorów Przybliżonych (RS) [9], [14], [5], [6] w Indukcyjnym Programowaniu Logicznym (ILP) ma na celu usprawnienie procesu uczenia się z danych niepełnych. W artykule przedstawiono algorytm, który na podstawie indukcyjnego programowania logicznego i zbiorów przybliżonych umożliwia odkrywanie reguł pierwszego rzędu. Prezentowany algorytm bazuje na systemie otoczeń, przy czym na wejściu otrzymuje system decyzyjny pierwszego rzędu, a na wyjściu produkuje zbiór reguł pierwszego rzędu [8]. Idea wykorzystania zbiorów przybliżonych do znajdowania reguł bazuje na schemacie sekwencyjnego pokrywania. Główną operacją schematu sekwencyjnego pokrywania jest utworzenie pewnej koniunkcji literałów, która posłuży jako część warunkowa reguły – przesłanka [13]. Skutkiem dołączania kolejnych literałów do przesłanki jest zwiększenie liczby poprawnie pokrytych przykładów pozytywnych i zmniejszanie liczby pokrytych przez regułę przykładów negatywnych. Tym, co odróżnia przedstawiany

[1] Wydział Informatyki, Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok

algorytm od innych, jest kryterium wyboru dołączanych literałów do przesłanki [3]. Kryterium opiera się na tym, aby reguła po dołączeniu literału jak najlepiej rozróżniała przykłady negatywne i pozytywne. Dopiero spośród takich literałów-kandydatów wybieramy tego, który po dołączeniu do części warunkowej reguły sprawia, że reguła jak najlepiej pokrywa zbiór przykładów. Aby ocenić rozróżnialność przykładów, stosuje się technikę zbiorów przybliżonych.

W ostatniej części pracy przedstawiono wyniki eksperymentów przeprowadzonych przy wykorzystaniu systemów ILP i opisywanego algorytmu. Eksperymenty przeprowadzono na zbiorze danych opisujących położenie bloków tekstu w dokumentach.

2. Indukcyjne programowanie w logice i zbiory przybliżone

2.1. Wybrane pojęcia indukcyjnego programowania logicznego

Indukcyjne programowanie w logice (ILP) jest obszarem badań znajdującym się na pograniczu dwóch dziedzin: uczenia się maszyn *Machine Learning* (ML) i programowania logicznego *Logic Programming* (LP) [13], [3]. Indukcyjne logiczne programowanie wykorzystuje do reprezentacji wiedzy język rachunku predykatów. Umożliwia to przedstawianie nie tylko klasyfikacji pojedynczych obiektów dziedziny do pewnych kategorii, lecz także związków zachodzących między różnymi obiektami. System ILP może wykorzystywać wiedzę dziedzinową bardziej naturalnie i efektywnie, ponieważ w ILP zarówno przykłady, wiedza dziedzinowa jak i wygenerowane reguły są wyrażane w tym samym języku.

Głównym celem systemu ILP jest znajdowanie reguł dla relacji celu. Relacje te reprezentowane są przez predykaty docelowe. Wnioskowanie odbywa się na podstawie wiedzy trenującej, przedstawionej jako przykłady oraz na podstawie znanych już zależności, które zachodzą pomiędzy relacjami – czyli wiedzy dziedzinowej.

W systemach ILP istnieje problem przy przetwarzaniu niedokładnych danych wejściowych. Uzyskany wynik może okazać się niewłaściwy, jeśli dane wejściowe są niedokładne, a w konsekwencji tej niedokładności indukowane hipotezy wyjściowe H mogą być błędne. Standardowe algorytmy ILP nie radzą sobie dobrze z takiego typu problemami. Nie sprawdzają się one dla danych niedokładnych, niekompletnych, zbyt ogólnikowych i nieścisłych, występujących zarówno w informacji trenującej, wiedzy dziedzinowej, jak też i wygenerowanych regułach.

Na niedokładne dane wejściowe składają się:

1. Zakłócone dane (ang. *noisy data*) np. błędne wartości argumentów w przykładach, błędna klasyfikacja przykładów.

2. Za rzadkie dane np. podane obserwacje zawarte w przykładach uczących są niewystarczające do otrzymania właściwej hipotezy H.
3. Brakujące dane, np. niektóre argumenty w niektórych przykładach uczących mają nieznane wartości.
4. Dane sprzeczne, np. niektóre spośród przykładów należą zarówno do E^+ jak i E^-.

2.2. Wybrane pojęcia teorii zbiorów przybliżonych

Zbiory przybliżone to jedna z technik służących do identyfikacji i rozpoznawania wzorców w danych. Jest ona szczególnie użyteczna w przypadku danych niekompletnych, opiera się na aproksymacji zbiorów. Obiekt zostaje przydzielony do konkretnej klasy na podstawie relacji nierozróżnialności R, zdefiniowanej na zbiorze U wszystkich obiektów.

Oznaczmy przestrzeń aproksymacji jako AS

$$AS = (U,R) \tag{2.1}$$

gdzie U jest zbiorem obiektów, natomiast R jest relacją nierozróżnialności na zbiorze U. Czyli U jest podzielone przez R na klasy nierozróżnialności [4].

Jeżeli $X \subseteq U$, to obie aproksymacje, dolną i górną, możemy zdefiniować następująco:

$$\underline{Apr_{AS}}(X) = \bigcup_{[x]_R \subseteq X} [x]_R = \{x \in U | [x]_R \subseteq X\} \tag{2.2}$$

$$\overline{Apr_{AS}}(X) = \bigcup_{[x]_R \cap X \neq \emptyset} [x]_R = \{x \in U | [x]_R \cap X \neq \emptyset\} \tag{2.3}$$

Gdzie $[x]_R$ oznacza klasę równoważności, elementu x.

3. Algorytm konstrukcji reguł

3.1. Pojęcia wstępne

W dalszych rozważaniach zakładamy, że generowane reguły nie są regułami rekurencyjnymi, tzn. żaden predykat nie może występować jednocześnie w części warunkowej i decyzyjnej reguły.

W celu sprecyzowania, które predykaty mają się znajdować w części warunkowej reguły, a które w części decyzyjnej, będziemy używać, podobnie jak w PROGOLU [12], poleceń *modeh* i *modeb*:

modeh – definiuje predykaty wchodzące w skład konkluzji reguł.
modeb – definiuje predykaty mogące wchodzić w skład przesłanek reguł.

Definicja relacji mogącej wchodzić w skład przesłanki reguły ma postać:

$$modeb(r, p(t_1,...,t_n)),$$

gdzie
$r \geq 1$ jest maksymalną liczbą wystąpień predykatu w przesłance,
p jest nazwą n-elementowej relacji,
t_i jest i-tym argumentem relacji.

Dla każdego argumentu t_i można określić czy zmienna odpowiadająca temu argumentowi jest wejściowa, czy też wyjściowa. Wejściowe argumenty poprzedzane są znakiem '+', wyjściowe znakiem '–'. Analogicznie wygląda definicja *modeh*.

Przykład. $modeb(1, ojciec(+osoba, -osoba))$ definiuje predykat o nazwie ojciec, który może maksymalnie raz wchodzić w skład przesłanki reguły. Predykat ojciec jest dwuargumentowy, oba argumenty są argumentami typu osoba, z czego pierwszy jest argumentem wejściowym, natomiast drugi jest argumentem wyjściowym.

Jeżeli dany jest literał a, to przez $In(a)$ oznaczamy zbiór zmiennych wejściowych literału a, natomiast przez $Out(a)$ zbiór zmiennych wyjściowych literału a.

Zbiór zmiennych wejściowych koniunkcji literałów jest definiowany jako suma zbiorów zmiennych wejściowych poszczególnych literałów.

3.2 System decyzyjny pierwszego rzędu

System decyzyjny pierwszego rzędu definiowany jest jako czwórka $\langle B, E, M_h, M_b \rangle$, gdzie B jest zbiorem reguł wyróżnionych jako wiedza dziedzinowa, E jest zbiorem przykładów $E = E_h \cup E_b$, gdzie E_h oznacza przykłady odpowiadające predykatom zdefiniowanym w M_h, E_b oznacza przykłady odpowiadające predykatom zdefiniowanym w M_b, M_h jest zbiorem deklaracji wskazujących predykaty mogące wchodzić w skład konkluzji reguły (*modeh*), M_b jest zbiorem deklaracji określających predykaty mogące wchodzić w skład przesłanki reguły (*modeb*) [3].

Zarówno przykłady odpowiadające predykatom zdefiniowanym w M_h, jak i przykłady odpowiadające predykatom zdefiniowanym w M_b, można podzielić na pozytywne i negatywne: $E_h = E_h^+ \cup E_h^-$ i $E_b = E_b^+ \cup E_b^-$.

Przykładowa definicja predykatu wchodzącego w skład M_h:

$modeh(1, dziadek(+osoba, +osoba))$

Przykładowa definicja predykatu wchodzącego w skład M_b:

$modeb(1, ojciec(+osoba, -osoba))$

W M_h może znajdować się wiele deklaracji predykatów docelowych, w $E_h = E_h^+ \cup E_h^-$ mogą znajdować się przykłady dla wszystkich predykatów zadeklarowanych w M_h. Oznaczmy więc symbolem $E_h^+(p)$ zbiór przykładów należących do E_h^+, w których występuje predykat p i analogicznie przez $E_h^-(p)$ zbiór przykładów należących do E_h^-, w których występuje predykat p.

Głębokość Niech h będzie konkluzją, a b przesłanką reguły. Głębokość zmiennej X w regule IF b THEN h, którą będziemy również zapisywać jako $h \leftarrow b$, definiujemy następująco:

1. Jeżeli $X \in In(h)$, to Głębokość$(h \leftarrow b, X) = 0$
2. Jeżeli $X \in Out(b)$, to Głębokość$(h \leftarrow b, X) = \min$ (Głębokość$(h \leftarrow c, U)) + 1$ dla $c \subset b$ i $U \in In(b)$
3. ∞ w przeciwnym przypadku.

Głębokość reguły jest równa maksymalnej głębokości zmiennych w niej występujących.

Przykład. Zbadajmy głębokość reguły $z(A) \leftarrow q(A,B) \wedge q(B,C)$ przy następujących deklaracjach predykatów $modeh(1, z(+typ))$, $modeb(2, q(+typ, -typ))$. Głębokość zmiennej A wynosi 0, ponieważ $A \in In(h)$. Ponieważ zmienna B jest zmienną wyjściową wprowadzaną przez literał $q(A,B)$, głębokość B jest równa głębokości A plus jeden, czyli 1. Głębokość zmiennej C wynosi głębokość B plus jeden, czyli 2.

Generowanie kandydatów

Zdefiniujmy funkcję $Generuj\_Kandydatów(h,b,k)$, gdzie h jest konkluzją reguły, b jest przesłanką reguły oraz k jest maksymalną głębokością zmiennych. Niech q będzie literałem kandydującym do dodania do przesłanki b takim, że [3]:

$q = p(s_1,...,s_n)$ jeżeli $modeb(r, p(t_1,...,t_n))$ jest w M_b i dla $1 \le i \le n$

a) $s_i = X$ jeżeli $t_i = +t$, $X \in Out(b) \cup In(h)$ i $X \notin Out(neg(b))$

b) $s_i = X$ jeżeli $t_i = -t$, i X jest nową zmienną, która nie występuje ani w h ani w b oraz spełniony jest warunek Głębokość$(h \leftarrow b \wedge q, X) \le k$.

p pojawia się co najwyżej $r-1$ razy w b, $neg(b)$ oznacza zbiór literałów negatywnych wchodzących w skład przesłanki.

Do przesłanki można dodać kandydata, którego deklaracja jest zamieszczona w M_b, oraz spełnione są warunki:

a) Argument wejściowy kandydata musi być zmienną, która występuje jako argument wyjściowy w literałach już zawartych w przesłance, lub zmienną, która występuje jako argument wejściowy literału docelowego. Zmienne reprezentujące argumenty wyjściowe przesłanki opisanej przez negatywny literał nie mogą być używane jako zmienne wejściowe w argumentach innych literałów wchodzących do przesłanki reguły – dlatego $X \notin Out(neg(b))$.

b) Argument wyjściowy musi być nową zmienną, która nie występuje jako argument wejściowy predykatu docelowego, ani nie występuje jako argument wyjściowy w przesłance. Należy sprawdzić czy po dodaniu do przesłanki kandydata reguła będzie miała odpowiednią głębokość. $Głębokość(h \leftarrow b \wedge q, X) \leq k$.

Literał p, na podstawie którego został stworzony kandydat q może pojawiać się co najwyżej $r-1$ razy w przesłance b.

Przykład. Chcemy znaleźć opis relacji *dziadek* – będzie to nasz predykat docelowy. Opis tworzymy na podstawie relacji *ojciec* i *rodzic*. Deklaracje predykatów:

M_h: $modeh(1, dziadek(+osoba, +osoba))$

M_b: $modeb(1, ojciec(+osoba, -osoba))$, $modeb(1, ojciec(+osoba, +osoba))$
$modeb(1, matka(+osoba, -osoba))$, $modeb(1, matka(+osoba, +osoba))$
$modeb(1, rodzic(+osoba, +osoba))$

Załóżmy, że mamy do dyspozycji poniższą wiedzę dziedzinową i dane treningowe:

$$B = \begin{cases} rodzic(X,Y) \leftarrow ojciec(X,Y) \\ rodzic(X,Y) \leftarrow matka(X,Y) \end{cases} \quad E_b^+ = \begin{cases} ojciec(Adam, Tadeusz) \\ ojciec(Adam, Eliza) \\ ojciec(Tadeusz, Katarzyna) \\ matka(Anna, Tadeusz) \\ matka(Anna, Eliza) \\ matka(Katarzyna, Donald) \\ matka(Katarzyna, Zuzanna) \end{cases}$$

$$E_h^+ = \begin{cases} dziadek(Adam, Katarzyna) & (e_1) \\ dziadek(Tadeusz, Donald) & (e_2) \end{cases}$$

$$E_h^- = \begin{cases} \neg dziadek(Anna, Katarzyna) & (e_3) \\ \neg dziadek(Anna, Eliza) & (e_4) \\ \neg dziadek(Adam, Tadeusz) & (e_5) \end{cases}$$

Zakładamy, że w literale docelowym $dziadek(A,B)$ występują zmienne A i B.
Niech przesłanka $b = \{\neg matka(A,C)\}$.
Generujemy zbiór kandydatów do dołączenia do przesłanki:
$ojciec(A,D)$, $ojciec(B,D)$, $ojciec(A,A)$, $ojciec(A,B)$, $ojciec(B,A)$, $ojciec(B,B)$,
$matka(A,A)$, $matka(A,B)$, $matka(B,A)$, $matka(B,B)$,
$rodzic(A,A)$, $rodzic(A,B)$, $rodzic(B,A)$, $rodzic(B,B)$.

Powyżej nie ma kandydatów dla predykatu $matka(+osoba,-osoba)$, ponieważ predykat matka znajduje się już w przesłance, a w deklaracji $modeb(1, matka(+osoba,-osoba))$ maksymalna liczba wystąpień tego predykatu w przesłance została ograniczona do jednego wystąpienia.
Wśród kandydatów zabrakło również:
$ojciec(C,D)$, $ojciec(A,C)$, $ojciec(C,C)$, $ojciec(C,A)$, $ojciec(C,B)$, $ojciec(C,C)$,
$matka(A,C)$, $matka(C,C)$, $matka(C,A)$, $matka(C,B)$, $matka(C,C)$,
$rodzic(A,C)$, $rodzic(C,C)$, $rodzic(C,A)$, $rodzic(C,B)$, $rodzic(C,C)$,
ponieważ zmienna C została wprowadzona poprzez literał negatywny $\neg matka(A,C)$, wobec tego nie może być użyta jako argument wejściowy w żadnym z kandydatów.

3.3. System otoczeń

Pokrycie Niech b będzie przesłanką i h niech będzie literałem docelowym. Pokrycie reguły $h \leftarrow b$ oznaczamy jako $Pokrycie(h \leftarrow b)$.

Reguła $h \leftarrow b$ pokrywa przykład $h(A_1, A_2, ..., A_n)$, jeżeli istnieje podstawienie nadające wartości wszystkim zmiennym występującym w regule, dla którego spełnione są wszystkie literały zawarte w przesłance b. Oznaczamy przez $Pokrycie^+(h \leftarrow b)$ zbiór przykładów pozytywnych pokrytych przez regułę $h \leftarrow b$ i przez $Pokrycie^-(h \leftarrow b)$ zbiór przykładów negatywnych pokrytych przez regułę $h \leftarrow b$. Definiujemy miarę trafności reguły $h \leftarrow b$ jako

$$Oceń\_Pokrycie(h \leftarrow b) = \frac{|Pokrycie^+(h \leftarrow b)|}{|Pokrycie^-(h \leftarrow b)| + 1}. \qquad (3.1)$$

Załóżmy, że $\{H1, H2, ..., Hn\}$ jest zbiorem reguł i $e \in E_h$ jest przykładem.

System otoczeń $NS_{\{H1,H2,...,Hn\}}(e)$ jest sumą pokryć wszystkich reguł, które pokrywają element e tzn.

$$NS_{\{H1,H2,...,Hn\}}(e) = \bigcup_{i=1}^{n}\{Pokrycie(Hi) \mid e \in Pokrycie(Hi)\} \tag{3.2}$$

Dolna i górna aproksymacja zbioru przykładów X dla zbioru reguł $\{H1,H2,.....,Hn\}$:

$$\underline{Apr_{\{H1,H2,...,Hn\}}}(X) = \{e \in E_h \mid NS_{\{H1,H2,...,Hn\}}(e) \subseteq X\} \tag{3.3}$$

$$\overline{Apr_{\{H1,H2,...,Hn\}}}(X) = \{e \in E_h \mid NS_{\{H1,H2,...,Hn\}}(e) \cap X \neq \emptyset\} \tag{3.4}$$

Przykład. Załóżmy, że dla literału docelowego $h = dziadek(A,B)$ mamy przesłankę $b = \{ojciec(A,C)\}$. Rozpatrujemy dodanie następujących literałów $q = ojciec(C,B)$ i $\neg q = \neg ojciec(C,B)$. Wyznaczmy pokrycia dla przesłanki $b \wedge q = \{ojciec(A,C) \wedge ojciec(C,B)\}$
oraz przesłanki $b \wedge \neg q = \{ojciec(A,C) \wedge \neg ojciec(C,B)\}$

$$NS_{\{h\leftarrow b\wedge q,\, h\leftarrow b\wedge\neg q\}}(e_1) = Pokrycie(h \leftarrow b \wedge q) = \{e_1\},$$
$$NS_{\{h\leftarrow b\wedge q,\, h\leftarrow b\wedge\neg q\}}(e_2) = Pokrycie(h \leftarrow b \wedge \neg q) = \{e_2, e_5\},$$
$$NS_{\{h\leftarrow b\wedge q,\, h\leftarrow b\wedge\neg q\}}(e_3) = \emptyset,\ NS_{\{h\leftarrow b\wedge q,\, h\leftarrow b\wedge\neg q\}}(e_4) = \emptyset,$$
$$NS_{\{h\leftarrow b\wedge q,\, h\leftarrow b\wedge\neg q\}}(e_5) = Pokrycie(h \leftarrow b \wedge \neg q) = \{e_2, e_5\},$$
$$\underline{Apr_{\{h\leftarrow b\wedge q,\, h\leftarrow b\wedge\neg q\}}}(E_h^{+}) = \{e_1\} \text{ i } \overline{Apr_{\{h\leftarrow b\wedge q,\, h\leftarrow b\wedge\neg q\}}}(E_h^{+}) = \{e_1, e_2, e_5\}.$$

3.4. Algorytm znajdujący zbiór reguł

Algorytmy sekwencyjnego pokrywania [13] polegają na takim dodawaniu kolejnych literałów do przesłanki reguły, aby uzyskać jak najlepsze pokrycie przykładów. W kolejnych iteracjach do przesłanki dołączamy taki literał, który rozróżnia jak najwięcej spośród dotychczas nierozróżnialnych przykładów.

Niech $Dis\{H1,H2,...,Hn\}$ oznacza zbiór par przykładów rozróżnialnych przez zbiór reguł $\{H1,H2,.....,Hn\}$. Określmy kiedy przykład e_i jest odróżnialny od przykładu e_j. Po pierwsze – przykłady muszą należeć do różnych klas decyzyjnych. Ponieważ mamy dwie klasy decyzyjne – przykłady pozytywne i przykłady negatywne – przyjmujemy, że $e_i \in E_h^{+}$, natomiast $e_j \in E_h^{-}$.

Przykład e_i jest odróżnialny od przykładu e_j za pomocą zbioru reguł $\{H1,H2,...,Hn\}$ wtedy i tylko wtedy, gdy $NS_{\{H_1,H_2,...,H_n\}}(e_i) \neq \emptyset$ lub $NS_{\{H_1,H_2,...,H_n\}}(e_j) \neq \emptyset$ i $NS_{\{H_1,H_2,...,H_n\}}(e_i) \cap NS_{\{H_1,H_2,...,H_n\}}(e_j) = \emptyset$.

Algorytm generujący pojedynczą regułę [3] opiera się na sekwencyjnym dołączaniu literałów do przesłanki b. W kolejnych iteracjach dołączamy do b tego kandydata, który rozróżnia jak najwięcej spośród dotychczas nierozróżnialnych przykładów. Ponieważ reguła może być nieskończona, określamy jej maksymalną głębokość jako $k \in <0,\infty)$.

Zakładamy, że na początku wszystkie przykłady są nierozróżnialne i algorytm sukcesywnie powiększa zbiór przykładów rozróżnionych przez rozszerzanie przesłanki reguły. Algorytm kończy iteracje, gdy wszystkie przykłady zostaną rozróżnione $(R = \emptyset)$ lub R nie może być dalej zredukowany. W drugim przypadku przykłady nie mogą już zostać bardziej rozróżnione; przyjmujemy wtedy, że przykłady negatywne, które są pokryte przez wygenerowaną regułę są wyjątkami. Należy zaznaczyć, że nie dla wszystkich zbiorów redukcja ta jest dobrą metodą, ponieważ mogą istnieć inne przesłanki, które lepiej rozróżniają przykłady.

Główna idea procedury generującej zbiór reguł.

Wejście: h – literał docelowy.
$< B, E, M_h, M_b >$ – system decyzyjny pierwszego rzędu.
k – maksymalna głębokość generowanej reguły.
apr – aproksymacja $apr \in \{dolna, górna\}$
Wyjście: SH – zbiór reguł.

function *Generuj_Zbiór_Reguł*$(h,< B,E,M_h,M_b >,k,apr)$
begin
1. $E_{curr} = E_h^+(h)$, $SH = \emptyset$
2. **while** $E_{curr} \subseteq E$ **do**
3. H = *Generuj_Regułę*$(h,< B,E,M_h,M_b >,k,apr)$
4. $E = E - Pokrycie^+(H)$
5. $SH = SH \cup \{H\}$

done
end.

Komentarze do opisu algorytmu *Generuj_Zbiór_Reguł*:
Na wejściu podajemy $< B,E,M_h,M_b >$ – system decyzyjny pierwszego rzędu, literał docelowy h, którego definicja musi znajdować się w M_h, k – maksymalną

głębokość generowanej reguły oraz *apr* – rodzaj aproksymacji. Na wyjściu algorytm zwraca regułę H.

1. E_{curr} to wszystkie przykłady należące do E_h^+, które są obserwacjami dla predykatu h.
2. Powtarzamy pętlę dopóki SH nie pokrywa wszystkich przykładów pozytywnych.
3. Generujemy regułę H.
4. Z przykładów E usuwamy przykłady pozytywne pokryte przez znalezioną regułę H.
5. Dodajemy znalezioną regułę do zbioru reguł.

Teraz przedstawimy główną ideę procedury *Generuj_Regułę*.

function *Generuj_Regułę*$(h,< B,E,M_h,M_b >,k,apr)$
begin
1. $R=\{< e_i,e_j > \mid \forall e_i \in E_h^+(h), \forall e_j \in E_h^-(h)\}$ $b=\varnothing$
2. **while** $R \neq \varnothing$ **do**
   3. $\forall q \in Generuj\_Kandydatów(h,b,k)$
      oblicz $R(q)=R \cap Dis\{h \leftarrow b \wedge q, h \leftarrow b \wedge \neg q\}$
   4. $p=q$ dla którego $|R(q)|$ jest maksymalne
   5. **if** $R(p)=\varnothing$ **then**
      $\forall e \in Pokrycie^-(h \leftarrow b)$
      $b=b \wedge \neg e$
      exit while
   6. **if** $Oceń\_Pokrycie(h \leftarrow b \wedge p) > Oceń\_Pokrycie(h \leftarrow b \wedge \neg p)$
      then $bp=p$
      else $bp=\neg p$
   7. **if** apr = *górna* **and** $Pokrycie^+(h \leftarrow b) < Pokrycie^+(h \leftarrow b \wedge bp)$ **then**
      exit while;
   8. $b=b \wedge bp$ $R=R-R(p)$
done
9. **return** $h \leftarrow b$
end.

Komentarze do opisu procedury *Generuj_Regułę*:

1. Określamy R jako zbiór par przykładów, które są rozróżnialne. R jest zbiorem par przykładów relacji h, które nie należą do tej samej klasy decyzyjnej.

Ponieważ występują dwie klasy decyzyjne – przykłady pozytywne $E_h^+(h)$ i przykłady negatywne $E_h^-(h)$ – więc jest to zbiór par, w których pierwszy element jest przykładem pozytywnym, drugi jest przykładem negatywnym.

2. Powtarzamy pętlę dopóki zbiór par przykładów R jest niepusty.
3. Dla wszystkich kandydatów q otrzymanych za pomocą metody *Generuj_Kandydatów*(h,b,k) obliczamy $R(q)$ – zbiór par należących do R, które rozróżnia przesłanka $b \wedge q$ lub $b \wedge \neg q$.
4. Spośród zbioru kandydatów wybieramy takiego (oznaczmy go przez p), dla którego liczność zbioru $R(p)$ jest największa.
5. Jeżeli $R(p) = \varnothing$, to kończymy iterację. Zbiór par przykładów nie może zostać bardziej zredukowany, przyjmujemy wtedy, że przykłady negatywne, które są pokryte przez wygenerowaną regułę, są wyjątkami i dodajemy je do przesłanki reguły.
6. Określamy czy ze znakiem negacji czy bez tego znaku ma być dodany kandydat p do przesłanki b. Jeżeli reguła $h \leftarrow b$ po dodaniu do przesłanki b kandydata pozytywnego p ma lepsze pokrycie niż po dodaniu do przesłanki kandydata negatywnego $\neg p$, to $bp = p$ w przeciwnym przypadku $bp = \neg p$.
7. Jeżeli obliczamy górną aproksymację i po dodaniu do przesłanki b kandydata bp zmniejszyła się liczba pokrywanych przykładów pozytywnych, kończymy iterację.
8. Do przesłanki dodajemy kandydata $b = b \wedge bp$. Ze zbioru par R usuwamy te pary, które rozróżnia przesłanka po dodaniu do niej wybranego kandydata.
9. Zwracamy regułę H.

Przykład. W celu zilustrowania działania metody *Generuj_Regułę* posłużmy się przykładem przytoczonym w paragrafie 3.2. $k = 1$, $apr = dolna$, $h = dziadek(A,B)$.

Kolejne kroki działania algorytmu:

1. $R = \{< e_1,e_3 >,< e_1,e_4 >,< e_1,e_5 >,< e_2,e_3 >,< e_2,e_4 >,< e_2,e_5 >\}$.
2. $R \neq \varnothing$.
3. kandydaci $q = \{ ojciec(A,C),\ ojciec(B,C),$
   $ojciec(A,A),\ ojciec(A,B),\ ojciec(B,A),\ ojciec(B,B),$
   $matka(A,C),\ matka(B,C),$
   $matka(A,A),\ matka(A,B),\ matka(B,A),\ matka(B,B),$
   $rodzic(A,A),\ rodzic(A,B),\ rodzic(B,A),\ rodzic(B,B)\ \}$.
   Dla każdego kandydata obliczmy $R(q) = R \cap Dis(h \leftarrow \{b \wedge q\}, h \leftarrow \{b \wedge \neg q\})$.

4. Na podstawie wyników poprzedniego punktu wybieramy kandydata p o największej liczności $R(p)$. Najlepszy rezultat otrzymaliśmy dla 7-ego kandydata $p = matka(A,C)$.
   $R(p) = R \cap Dis(dziadek(A,B) \leftarrow matka(A,C), dziadek(A,B) \leftarrow \neg matka(A,C)) =$
   $\{< e_1, e_3 >, < e_1, e_4 >, < e_2, e_3 >, < e_2, e_4 >\}$
   $|R(p)| = 4$.
5. $R(p) \neq \emptyset$.
6. $Oceń\_Pokrycie(dziadek(A,B) \leftarrow matka(A,C)) <$
   $Oceń\_Pokrycie(dziadek(A,B) \leftarrow \neg matka(A,C))$
   wobec tego $bp = \neg matka(A,C)$.
7. $apr \neq górna$.
8. $b = \neg matka(A,C)$. $R = R - R(p) = \{< e_1, e_5 >, < e_2, e_5 >\}$.
2. $R \neq \emptyset$. Powtarzamy obliczenia – drugi przebieg pętli.
3. kandydaci $q = \{ ojciec(A,D), ojciec(B,D),$
   $ojciec(A,A), ojciec(A,B), ojciec(B,A), ojciec(B,B),$
   $matka(A,A), matka(A,B), matka(B,A), matka(B,B),$
   $rodzic(A,A), rodzic(A,B), rodzic(B,A), rodzic(B,B) \}$.
   Dla każdego kandydata obliczmy $R(q) = R \cap Dis(h \leftarrow \{b \wedge q\}, h \leftarrow \{b \wedge \neg q\})$.
4. Wybieramy kandydata p o największej liczności $R(p)$ – $p = ojciec(A,B)$
   $R(ojciec(A,B)) = \{< e_1, e_5 >, < e_2, e_5 >\}$
   $|R(p)| = 2$.
5. $R(p) \neq \emptyset$.
6. $Oceń\_Pokrycie(dziadek(A,B) \leftarrow \{\neg matka(A,C) \wedge ojciec(A,B)\}) <$
   $Oceń\_Pokrycie(dziadek(A,B) \leftarrow \{\neg matka(A,C) \wedge \neg ojciec(A,B)\})$
   wobec tego $bp = \neg ojciec(A,B)$.
7. $apr \neq górna$.
8. $b = \neg matka(A,C) \wedge \neg ojciec(A,B)$. $R = R - R(p) = \emptyset$.
3. $R = \emptyset$. Kończymy działanie procedury *Generuj_Regułę*.

4. Badania eksperymentalne

Badania eksperymentalne przeprowadzono na bazie danych opisujących jednostronicowe dokumenty. Każdy dokument [2] określony jest przez dwa odmienne typy struktur reprezentujące zarówno jego zawartość jak i jego organizację – czyli

strukturę geometryczną i logiczną. Strukturę geometryczną charakteryzują własności takie jak: tekst, linie (pionowe czy poziome), elementy graficzne itd. [1]. Drugą strukturę charakteryzuje zawartość dokumentu taka jak: tytuł, paragraf, sekcja, rozdział, tabela itd. Każdy z logicznych obiektów może, oczywiście, być opisany przez zbiór atrybutów. Atrybutami są własności i relacje zachodzące pomiędzy komponentami. Nas będą interesowały zwłaszcza relacje pomiędzy komponentami, bo za ich pośrednictwem będziemy rozpoznawać bloki dokumentu.

Mamy pięć literałów docelowych. Określają one logiczne komponenty, które mogą być wyspecyfikowane w próbkach listów. W eksperymentach rozpatrujemy 30 pojedynczych stron dokumentów. Wykonano 6 eksperymentów: 20 dokumentów stanowiło podstawę do danych trenujących, pozostałych 10 generowało dane testowe.

Eksperymenty przeprowadzono przy użyciu czterech systemów: FOIL [10], PROGOL [12], LINUS [11] oraz za pomocą prezentowanego algorytmu. W poniższej tabeli zamieszczono wyniki otrzymane dla tych czterech systemów. Kolumna A przedstawia procent przykładów pozytywnych poprawnie zaklasyfikowanych, kolumna B przedstawia procent przykładów negatywnych błędnie zaklasyfikowanych jako pozytywne.

Tabela 1

Wyniki eksperymentów

systemy	FOIL		PROGOL		LINUS		algorytm dolna aprok.		algorytm górna aprok.	
wyniki	A	B	A	B	A	B	A	B	A	B
Eksperyment 1										
logic_type_rif	100	9,09	100	15,91	100	14,77	83,33	9,38	100	11,93
logic_type_sender	100	7,06	100	20,06	100	19,49	80,00	7,06	80	7,06
logic_type_date	92,31	13,39	7,69	1,99	100	25,36	84,62	10,83	84,62	10,83
logic_type_receiver	100	5,65	100	24,86	100	14,12	50,00	5,65	80,00	5,65
logic_type_logo	100	5,65	100	6,21	100	5,65	90,00	5,65	100	5,65
Eksperyment 2										
logic_type_rif	89,47	7,54	100	26,09	73,68	10,43	15,79	7,54	100	10,14
logic_type_sender	91,67	6,82	66,67	7,67	100	19,03	66,67	6,53	91,67	6,82
logic_type_date	100	11,9	100	13,03	100	15,58	90,91	7,37	90,91	12,18
logic_type_receiver	90	5,37	100	12,43	100	12,43	60,00	5,65	100	5,65
logic_type_logo	100	5,65	100	5,65	100	5,65	100	5,65	100	5,65

Eksperyment 3										
logic_type_rif	94,12	8,07	100	13,54	100	26,51	0,00	8,07	100	10,66
logic_type_sender	91,67	6,82	100	19,03	100	27,56	66,67	6,53	91,67	6,82
logic_type_date	100	12,15	100	11,58	100	15,82	100	7,63	100	12,71
logic_type_receiver	80	5,37	100	12,43	100	24,86	0,00	5,65	100	5,65
logic_type_logo	100	5,65	100	5,65	100	5,65	100	5,65	100	5,65
Eksperyment 4										
logic_type_rif	100	10,26	92,31	12,82	100	26,78	0,00	9,12	92,31	9,12
logic_type_sender	100	6,8	90,91	15,58	90,91	24,65	63,64	6,8	90,91	6,8
logic_type_date	92,31	3,42	100	15,95	100	15,10	84,62	6,84	84,62	6,84
logic_type_receiver	90	6,21	100	18,64	90	24,29	100	5,65	100	5,65
logic_type_logo	100	5,65	100	6,21	100	5,65	80	5,65	100	5,65
Eksperyment 5										
logic_type_rif	100	7,98	84,62	13,39	100	27,35	84,62	9,12	100	11,68
logic_type_sender	40	5,73	40	5,73	40	5,73	40	5,73	40	5,73
logic_type_date	84,62	8,55	92,31	10,54	100	15,10	92,31	6,84	100	11,97
logic_type_receiver	80	6,5	100	24,86	90	17,23	50	5,65	100	5,65
logic_type_logo	100	5,65	100	6,21	100	6,78	10	5,65	100	5,65
Eksperyment 6										
logic_type_rif	68,75	8,33	93,75	13,51	100	24,71	6,25	8,33	100	9,77
logic_type_sender	92,31	6,55	84,62	6,84	92,31	24,22	38,46	6,27	92,31	6,55
logic_type_date	84,62	1,71	92,31	10,54	100	15,10	84,62	6,84	100	15,1
logic_type_receiver	90	5,08	100	12,43	100	24,86	70	5,65	100	5,65
logic_type_logo	100	5,65	100	6,21	100	5,65	100	5,65	100	5,65

Należy zaznaczyć, że w systemach FOIL, PROGOL i LINUS nie wykorzystuje się zjawiska dolnej i górnej aproksymacji. Wyniki otrzymane za pomocą prezentowanego algorytmu są zbliżone do wyników otrzymanych za pomocą systemu FOIL, są natomiast lepsze od wyników otrzymanych za pomocą systemów PROGOL i LINUS. Najgorsze wyniki otrzymano dla predykatu logic_type_sender w 5-tym eksperymencie. Najlepsze wyniki dla wszystkich systemów uzyskano dla predykatu logic_type_logo. Najgorsze rezultaty otrzymano dla systemu LINUS.

W systemie FOIL maksymalna głębokość zmiennych w otrzymanych regułach wynosi 2. W systemie FOIL dopuszczalne są reguły rekurencyjne i takie w rozwiązaniu występują. W przesłankach nie używamy literałów negatywnych. W przypadku systemu PROGOL wynik jest o wiele bardziej spójny niż otrzymany przy udziale systemu FOIL. PROGOL może produkować reguły, w skład których

wchodzą jedynie literały pozytywne - w przesłankach nie występują literały negatywne. Dla omawianego algorytmu otrzymano zbiory zawierające od 2 do 11 reguł, maksymalna głębokość zmiennych wynosi 4, w regułach występują literały negatywne. Pewną wadą wyników otrzymanych przy zastosowaniu dolnej aproksymacji jest za duża szczegółowość rozwiązania, uzyskujemy wiele reguł i przesłanki składają się z wielu literałów. Przy zastosowaniu górnej aproksymacji rozwiązaniem jest jedna reguła dla każdego z przeprowadzonych eksperymentów.

Podsumowanie

W przedstawionym algorytmie podstawowym kryterium wyboru literałów, które mają być dodane do przesłanki reguły, jest to, żeby po ich dodaniu przesłanka jak najlepiej rozróżniała przykłady pozytywne od negatywnych. Jest to podejście odmiennie niż reprezentowane w systemach FOIL, PROGOL czy LINUS, gdzie preferuje się literały, po których dodaniu część warunkowa reguły pokrywa jak najwięcej przykładów pozytywnych. Można zauważyć, że wyniki otrzymane przy udziale FOIL'a są nieco lepsze od tych otrzymanych za pomocą PROGOL'a, choć reguły otrzymane za pomocą PROGOL'a posiadają w przesłankach mniejszą liczbę literałów. Dokładność zbioru reguł otrzymanych dla omawianego algorytmu nie odbiega od dokładności zbioru reguł otrzymanych przy udziale FOIL'a i jest lepsza od dokładności reguł otrzymanych przy użyciu PROGOL'a i LINUS'a.

Ciekawym pomysłem, o który można rozszerzyć algorytm może okazać się generowanie reguł na podstawie przykładów negatywnych.

Literatura

[1] F. Esposito, D. Malerba, G.Semerano, M.Pazzani: *A Machine Learning Approach To Document Understanding,* Proceedings of the Second International Workshop on Multistrategy Learning, R. S. Michalski, G. Tecuci, (Eds.), 1993, 276-292.

[2] ftp://ftp.mlnet.org/ml-archive/general/data/doc-understanding/ – baza danych dokumentów.

[3] H. Midelfart, J. Komorowski: *A Rough Set Approach to Inductive Logic Programming,* W. Ziarko, Y. Yao, (Eds.), Proceedings of the 2nd International Conference on Rough Sets and Current Trends in Computing (RSCTC-2000), 2000, 158-166.

[4] J. Stepaniuk: *Knowledge Discovery by Application of Rough Set Models,* L. Polkowski, S. Tsumoto, T.Y. Lin, (Eds.), Rough Sets: New Developments, Physica-Verlag, Heidelberg, 2000, 137-233.
[5] L. Polkowski, A. Skowron (Eds.): *Rough Sets in Knowledge Discovery 1: Methodology and Applications.* Physica-Verlag, Heidelberg, 1998.
[6] L. Polkowski, A. Skowron (Eds.): *Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems.* Physica-Verlag, Heidelberg, 1998.
[7] N. Lavrac, S. Dzeroski: *Inductive Logic Programming, Techniques and Applications,* Ellis Horwood Limited 1994.
[8] P. Cichosz: *Systemy uczące się.* WNT, 2000.
[9] S.K. Pal, A. Skowron (Eds.): *Rough Fuzzy Hybridization: A New Trend in Decision-Making.* Springer-Verlag, Singapore, 1999.
[10] System FOIL: http://www.cse.unsw.edu.au/~quinlan/.
[11] System LINUS: ftp://ftp.mlnet.org/ml-archive/ILP/public/software/linus/.
[12] System PROGOL: http://www-users.cs.york.ac.uk/~stephen/progol.html.
[13] T. Mitchell: *Machine Learning.* McGraw Hill, 1997.
[14] Z. Pawlak: *Rough Sets. Theoretical Aspects of Reasoning about Data,* Kluwer Academic Publishers, Dordrecht, 1991.

AN ALGORITHM GENERATING FIRST ORDER RULES BASED ON ROUGH SET METHODS

Summary: The aim of this paper is to introduce and investigate an algorithm for finding first order rules. Rough set theory is used in the process of selecting literals, which may be part of the rule. The criterion of selecting literals reads as follows: only those literals are selected, which adding to the rule makes that the rule discerns the most examples from those, which were yet undiscerned.

Key words: rough sets, inductive logic programming