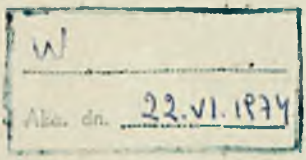
ALGORYTMY

Vol. XI • No. 19 • 1974

INSTYTUT MASZYN MATEMATYCZNYCH

A L G O R Y T M Y
vol. XI N° 19 1974



K o m i t e t R e d a k c y j n y

Antoni MAZURKIEWICZ /red. nacz./, Krzysztof MOSZYŃSKI, Zdzisław PAWLAK,
Jan WIERZBOWSKI, Andrzej WIŚNIEWSKI, Ryszard ZIELIŃSKI
Romana NITKOWSKA /sekr. red./

Adres Redakcji: Warszawa, ul. Krzywickiego 34, tel. 28-37-29

Druk IMM z. 54/74 n. 550. GP.II/1416/68.

TREŚĆ
СОДЕРЖАНИЕ
CONTENTS

ALGORYTMY, XI, N° 19, 1974, 5-22 518.5

PROVING PROPERTIES OF PROCESSES

Antoni MAZURKIEWICZ
Computation Centre of PAS, P.O.B. 22, PKiN
00-901 Warszawa, Poland

Received 31th Oct., 1973

In the paper some problems concerning computing processes, as input - output dependence, resulting relation properties, cycling, termination, etc., are considered. Some algebraic methods of proving such properties are given.

1. INTRODUCTION

In this paper we shall deal with processes. Generally speaking, a process (deterministic or not) is defined by (a) a set S, of states, representing instantaneous properties of objects involved in the process, and (b) a binary relation T, $T \subseteq S \times S$, the transition relation, or the next-state relation, describing elementary actions of the process, i.e. the way of changing states into the next ones. The total action of a process is a result of the repeated execution of elementary actions; it can be then described by the iteration T^* of the transition relation T. By proving properties of processes we shall mean here proving some properties of total actions of processes by means of properties of their elementary actions; that is, we want to express properties of T^* ("global" properties) by properties of T ("local" properties).

We are mostly interested in computing processes, where states are state-vectors of programs and the transition relation is defined by program statements; other examples of processes are:

*) This paper was prepared for and presented at the Second Symposium on MFCS, Strbske Pleso, September 1973.

phrase structure grammars defining derivations some strings from other, or Turing machines, where states are instantaneous descriptions and the transition relation describes single moves of the machine.

The first (known to the author) formal notion of a process, due to Pawlak, was used in a modified version for proving algorithms (Mazurkiewicz 1970), and next was generalized to the notion of iterative systems (Blikle 1972, Blikle and Mazurkiewicz 1973). This generalization is a basic for further development of the process theory, e.g. theory of abstract algorithms (Mazurkiewicz 1972a, 1972b, Blikle and Mazurkiewicz 1973) or the theory of interacting processes (Winkowski 1972).

The main purpose of this paper is:

- to investigate possibly most general models of computing processes, in order to get a common basis for dealing with more specified models;
- to find possibly most general methods for proving properties of computing processes (as correctness, termination, etc.);
- to clarify mutual relationships between different existing methods of proving such properties.

Pawlak's processes are probably the simplest and the most general models of computer programs; investigating such processes we may expect to obtain the deepest and the most fundamental results concerning computing processes. For that reason we shall use Pawlak's processes as our basic notion.

2. NOTATION

Let X, Y be sets. Every subset Q of $X \times Y$ is called a binary relation (or simply relation) in $X \times Y$. By xQy we mean $(x, y) \in Q$. If $A \subseteq X$, then I_A denotes the diagonal of A (or the identity in A), i.e. such a relation in $X \times X$ that

$$x \, I_A \, y \iff x \in A \text{ and } x = y.$$

Let X, Y, Z be sets, $Q \subseteq X \times Y$, $R \subseteq Y \times Z$. By QR we denote the composition of Q and R, i.e. a binary relation in $X \times Z$ such that

$$x\ QR\ z \iff (\exists y)\ xQyRz.$$

If $Q \subseteq X \times Y$, then Q^{-1} is a relation in $Y \times X$ such that $xQ^{-1}\ y \iff yQx$. Let $S \subseteq X \times X$. For each integer $n \geq 0$ we define recursively the n-th power of S, putting $S^0 = I_X$, $S^{n+1} = S^n S$. The iteration and positive iteration of S are defined respectively as

$$S^* = \bigcup \{S^n : n \geq 0\},\ S^+ = SS^*.$$

Let $A \subseteq X$, $B \subseteq Y$, $Q \subseteq X \times Y$. By AQ we denote the image of A under Q, by QB the coimage of B under Q, i.e.

$$AQ = \{y : (\exists x \in A)\ xQy\},\quad QB = \{x : (\exists y \in B)\ xQy\}.$$

It follows from this definition that QY is the domain of Q and XQ is the range of Q. The letters i, j, k, m, n (with indices, if necessary) will denote integers. The set of all integers will be denoted by J.

3. PAWLAK'S PROCESSES

By a Pawlak's process (or simply, process) we shall mean an ordered quadruple

$$P = (S, B, F, T),$$

where S is a set (of states of P), B is a subset of S (the set of initial states of P), P is a subset of S (the set of final states of P), and $T \subseteq S \times S$ is a binary relation (the transition relation of P). This definition differs slightly from the original one introduced by Pawlak; however, the above version will be more suitable for our purposes. By a computation in P we shall mean a sequence of states

$$(s_0, s_1, \ldots, s_n, \ldots),$$

finite or infinite, such that $s_{i-1}Ts_i$ for all $i \geqslant 1$. Infinite computations are said to be divergent. It follows from this definition that $sT^{\infty}t$ iff there exists a finite computation in P, beginning with s and ending with t. We say that a computation $(s_0, s_1, \ldots, s_n)$, $n \geqslant 0$, starts from s, if $s_0 = s$; and that it halts, if $s_n \in F$. The relation

$$Res_P = T^{\infty} \cap (B \times F)$$

is called the resulting relation of P (we shall omit the subscript P, if P is understood by a context). From this definition it follows that $Res = I_B T^{\infty} I_F$. The resulting relation of P describes the effect of the whole action of P and will be the main subject of our subsequent considerations. If s Res t, then we say that P computes t from s, or that t is a result of P for s. If there exists such t that s Res t, then we say that P halts (or terminates) for s. We say that P diverges for s, if there is a divergent computation in P which starts from s. By $Loop_P$ (or Loop, if P is understood) we denote the set of all initial states for which P diverges.

A process defined as above is in general a nondeterministic device; if we assume $T^{-1}T \subseteq I_S$ and $TS \subseteq S - F$ (i.e.T is a partial function undefined on F), then we get a deterministic process. It can be proved (Pawlak, 1971) that the resulting relation of a deterministic process is a partial function. If we assume $S - F \subseteq TS$ (i.e. the domain of T contains all nonfinal states), then the process is said to be total. One can observe that in general a process can halt and diverge for the same state; it is not possible for deterministic processes. It can be also observed that a process can neither halt nor diverge for the same state; it is not possible for total processes.

Example 1. Consider a program

a: x := 1; y :=0; b: if y = z then stop; c: y := y+1; x := x×y;
goto b;

with integer variables x, y, z. This program can be represented by a process P = (S, B, F, T), with

$S = \{a, b, c, d\} \times J^3$, $B = \{a\} \times J^3$, $F = \{d\} \times J^3$, $T = T_1 \cup T_2 \cup T_3 \cup T_4$, where

$T_1 = \{((a, x, y, z), (b, 1, 0, z)) : x, y, z \in J\}$,

$T_2 = \{((b, x, y, z), (d, x, y, z)) : y = z, x, y \in J\}$,

$T_3 = \{((b, x, y, z), (c, x, y, z)) : y \neq z, x, y, z \in J\}$,

$T_4 = \{((c, x, y, z), (b, x*(y+1), y+1, z)) : x, y, z \in J\}$.

We can observe that the defined process is deterministic and total. The states of this process are quadruples (e, x, y, z), where e is a label, x, y, z are actual values of program variables. The transition relation of P is the union of the relations T_1, T_2, T_3, T_4, which correspond to instructions of the program. Observe that the if-statement is represented by two relations with disjoint domains. This example will be used later on.//

In the sequel we shall deal with methods of proving some assertions about processes. Let $P = (S, B, F, T)$ be a process, $A \subseteq B$, $Z \subseteq F$, $Q \subseteq B \times F$, and $L \subseteq B$. The following assertions will be the subject of our investigations:

(1) $A\ Res \subseteq Z$ (results of P for states with property A have property Z),

(2) $Res\ Z \subseteq A$ (if a state with property Z is a result of P for s, then s has property A),

(3) $Res \subseteq Q$ (if P computes t from s, then sQt holds),

(4) $L \subseteq Loop$ (P diverges for each state with property L),

(5) $A \subseteq Res\ Z$ (from each state with property A P computes a state with property Z),

(6) $Z \subseteq A\ Res$ (each state with property Z is a result of P for a state with property A),

(7) $Q \subseteq Res$ (if sQt holds, then P computes t from s),

(8) $Loop \subseteq L$ (if P diverges for s, then s has property L).

Assertion (1) is a classical problem in programming theory; if we want to prove properties of program results assuming some properties of input data, then we have to prove an assertion like (1). Assertion (3) says that Q is an upper approximation of the resulting relation; in other words, Q express properties of resulting relation. Assertions (1) - (4) say nothing about termination of P; each of them has form of an implication "if P halts, then...". The next four assertions are in a sense dual to the previous four: they have form of implications "if ..., then P halts and..."; assertion (5) says that if s has property A, then P halts for s and gives a result with property Z. Assertion (7) gives a lower approximation of resulting relation; combined with (3) can be used for determining the resulting relation. Methods for proving assertions (1) - (4) base on the so called invariant properties of processes, described in sections 4 and 5 of the paper; methods for proving (5) - (8) use the so called inductive properties, described in section 6.

4. INVARIANT PROPERTIES

Let $P = (S, B, F, T)$ be a process, fixed for the rest of this section, and let G be a subset of S, or a relation in $S \times S$. We say that G is <u>right</u> (<u>left</u>) <u>invariant</u> for P, or simply, right (left) invariant, if

$$GT \subseteq G \quad (TG \subseteq G).$$

To make this definition clear, suppose P represents a program and G is a set of its state-vectors. G is right invariant, if no instruction of the program, applied to state-vectors in G, leads outside of G, i.e. if each instruction of P preserves the property G. The trivial examples of right, and in the same time, left invariant sets (relations) are the empty set (empty relation) $\emptyset$, and the whole set S (the full relation $S \times S$).

Invariant sets play an important part in proving assertions (1) and (2); in fact, the method of assigning meaning to programs (Floyd, 1967) is based on properties of right invariant sets. Left invariant relations play a similar part in proving

algorithms by tail functions (Mazurkiewicz, 1970). Both methods result from the following theorem.

Theorem 1. $GT \subseteq G \Leftrightarrow GT^{\circledast} \subseteq G$; $TG \subseteq G \Leftrightarrow T^{\circledast}G \subseteq G$.
Proof. Assume $GT \subseteq G$; by induction we get $GT^n \subseteq G$ for every $n \geqslant 0$, hence $GT^{\circledast} \subseteq G$. Assume $GT^{\circledast} \subseteq G$; hence, in particular, $GT \subseteq G$. The second part of the proof is similar.//

Thus, roughly speaking, we can say that if a property is right (left) invariant for P, then it is preserved by any computation ("backward" computation) in P.

Corollary 1. Let $A, Z, D \subseteq S$. If the following conditions are satisfied:

$$\text{(i)}\ DT \subseteq D, \quad \text{(ii)}\ A \subseteq D \frown B, \quad \text{(iii)}\ D \frown F \subseteq Z,$$

then $A\ \mathrm{Res} \subseteq Z$.
Proof. Suppose (i), (ii), (iii) are satisfied. By (i) and Theorem 1 $DT^{\circledast} \subseteq D$; since $\mathrm{Res} \subseteq T^{\circledast}$, $D\ \mathrm{Res} \subseteq D$, hence $D\ \mathrm{Res} \subseteq (D \frown F)$. By (ii) and (iii) it turns out $A\ \mathrm{Res} \subseteq Z$.//

This corollary gives a method for proving assertion (1). Similarly we can prove the following corollary, giving a method for proving assertion (2):

Corollary 2. Let $A, Z, D \subseteq S$. If the following conditions are satisfied:

$$\text{(i)}\ TD \subseteq D, \quad \text{(ii)}\ Z \subseteq D \frown F, \quad \text{(iii)}\ D \frown B \subseteq A,$$

then $\mathrm{Res}\ Z \subseteq A$.//

Example 2. Consider the process in Example 1. Let $n \geqslant 0$ be fixed and let $D = D_1 \cup D_2 \cup D_3 \cup D_4$ be a subset of S such that

$$D_1 = \{(a, x, y, n) : x, y \in J\}, \quad D_2 = \{(b, y!, y, n) : \leqslant n\},$$
$$D_3 = \{(c, y!, y, n) : y < n\}, \quad D_4 = \{(d, n!, n, n)\}.$$

We shall prove that D is right invariant for P. Indeed, $DT = (D_1 \cup D_2 \cup D_3 \cup D_4)\ (T_1 \cup T_2 \cup T_3 \cup T_4) = D_1T_1 \cup D_2T_2 \cup D_2T_3 \cup D_3T_4$ (all remaining terms vanish); as it can be directly checked, $D_1T_1 \subseteq D_2$, $D_2T_2 = D_4$, $D_2T_3 = D_3$, $D_3T_4 = D_2$, hence $DT = D_2 \cup D_3 \cup D_4 \subseteq D$. Since $D \frown B = D_1$, $D \frown F = D_4$, by Corollary 1 we conclude that

$D_1\mathrm{Res} \subseteq D_4$, i.e. $\{(a, x, y, n) : x, y \in J\}\ \mathrm{Res} \subseteq \{(d, n!, n, n)\}$,

provided $n \geq 0$. Thus, if the program in Example 1, starting with arbitrary values of x, y and with $n \geq 0$ as a value of z, ultimately halts, then the resulting values of x, y, z will be respectively n!, n, n. //

Corollary 3. Let $R \subseteq S \times S$. If $I_B \cup RT \subseteq R$ ($I_F \cup TR \subseteq R$), then $Res \subseteq RI_F$ ($Res \subseteq I_B R$).

Proof. Assume $I_B \cup RT \subseteq R$; since $RT \subseteq R$, R is right invariant and $RT^* \subseteq R$; $I_B \subseteq R$, hence $I_B T^* \subseteq R$, what gives $Res = I_B T^* I_F \subseteq RI_F$. The second part of the proof is similar.//

This corollary can be used for proving assertion (3). If we apply right invariant relation, the method is called the head relation method; if the left invariant relation is applied, then we have the tail relation method which is a generalization of the tail function method.

Example 3. Consider the process in Example 1. Let $f: J \times J \to J$ be a total function, defined as follows:

$$f(x, y) = \begin{cases} (x+1)(x+2) \ldots (y-1)y, & \text{for } x < y, \\ 1 & \text{for } x \geq y. \end{cases}$$

We have then $f(0, y) = y!$, $f(x, x) = 1$, $(x+1)f(x+1, y) = f(x, y)$ for $x < y$. Let $R = R_1 \cup R_2 \cup R_3 \cup R_4$ be a relation in $S \times S$, where

$$R_1 = \{((a, x, y, z), (d, z!, z, z)) : x, y \in J, z \geq 0\},$$

$$R_2 = \{((b, x, y, z), (d, xf(y, z)\ z, z)) : x, y \in J, z \geq y\},$$

$$R_3 = \{((c, x, y, z), (d, xf(y, z), z, z)) : x, y \in J, z > y\},$$

$$R_4 = \{((d, x, y, z), (d, x, y, z)) : x, y, z \in J\}.$$

We shall prove that R is a left invariant relation for P. Indeed; $TR = (T_1 \cup T_2 \cup T_3 \cup T_4)(R_1 \cup R_2 \cup R_3 \cup R_4) = T_1R_2 \cup T_2R_4 \cup T_3R_3 \cup T_4R_2$ (all remaining terms are empty). Since it can be easily checked that $T_1R_2 = R_1$, $T_2R_4 \subseteq R_2$, $T_3R_3 \subseteq R_2$, and $T_4R_4 = R_3$, we obtain $TR \subseteq R_1 \cup R_2 \cup R_3 \subseteq R$, i.e. R is left invariant. Since $I_F = R_4 \subseteq R$, $I_B R = R_1$, by Corollary 3 we infer that $Res \subseteq R_1$, that is,

$$Res \subseteq \{((a, x, y, z), (d, z!, z, z)) : x, y \in J, z \geq 0\}. //$$

At the end of this section we formulate some theoretical properties of invariant sets and relations.

Theorem 2. Let $A \subseteq S$. The family of all right (left) invariant sets containing A and ordered by the set-theoretical inclusion is a complete lattice with the least element $AT^{\mathcal{H}}$ ($T^{\mathcal{H}}A$), and the greatest element S. Similarly, let $Q \subseteq S \times S$; the family of all right (left) invariant relations containing Q and ordered by the set-theoretical inclusion is a complete lattice with the least element $QT^{\mathcal{H}}$ ($T^{\mathcal{H}}Q$), and the greatest element $S \times S$.

Proof. It is easy to prove that the union and intersection of any number of right (left) invariant sets containing a set $A \subseteq S$ is a right (left) invariant set containing A. $AT^{\mathcal{H}}$ ($T^{\mathcal{H}}A$) is clearly right (left) invariant; suppose D is right (left) invariant and $A \subseteq D$; then $AT^{\mathcal{H}} \subseteq DT^{\mathcal{H}}$ ($T^{\mathcal{H}}A \subseteq T^{\mathcal{H}}D$), and since D is right (left) invariant, $AT^{\mathcal{H}} \subseteq D$ ($T^{\mathcal{H}}A \subseteq D$). Thus, $AT^{\mathcal{H}}$ ($T^{\mathcal{H}}A$) is the least element of the lattice. S is clearly the greatest element of this lattice. The second part of the proof is similar.//

5. DIVERGENT COMPUTATIONS

In this section we give some methods for proving assertion(4) concerning divergent computations. Let $P = (S, B, F, T)$ be a process, fixed for the rest of this section, and let Lp denote the set of all states $s \in S$ for which P diverges. Clearly, $\text{Loop} = \text{Lp} \cap B$. The set Lp will be called the divergence set for P. The following theorem characterizes Lp in terms of the transition relation of P.

Theorem 3. Lp is the greatest subset of S satisfying the inclusion $\text{Lp} \subseteq T\,\text{Lp}$.

Proof. If $s \in \text{Lp}$, then there exists an infinite computation $s, s_1, \ldots$ in P. Of course, $s_1 \in \text{Lp}$ and sTs_1, hence $s \in T\text{Lp}$. That is, $\text{Lp} \subseteq T\text{Lp}$. Suppose now that U is a subset of S such that $U \subseteq TU$. We shall prove that $U \subseteq \text{Lp}$. Let $s_0 \in U$; suppose that for some $i \geqslant 0$ $s_i \in U$. Since $U \subseteq TU$, there is $s_{i+1} \in U$ such that $s_i T s_{i+1}$; thus, there is infinite sequence $(s_0, s_1, \ldots, s_i, \ldots)$ such that $s_i T s_{i+1}$ for all $i \geqslant 0$. It means that $s_0 \in \text{Lp}$, hence $U \subseteq \text{Lp}$.//

Corollary 4. For each $L \subseteq S$, $L \subseteq TL \Rightarrow L \cap B \subseteq Loop$.
Proof. By Theorem 3 $L \subseteq TL \Rightarrow L \subseteq Lp$; hence $L \subseteq TL$ implies $L \cap B \subseteq Lp \cap B = Loop$.//

This corollary can be applied for proving assertion (4), as it is shown in the following example.

Example 4. Consider the process in Example 1. Let L be a subset of S, $L = L_1 \cup L_2 \cup L_3$, where,

$$L_1 = \{(a, x, y, z): x, y, \in J, z < 0\},$$
$$L_2 = \{(b, x, y, z): x, y \in J, z < y\},$$
$$L_3 = \{(c, x, y, z): x, y \in J, z \leqslant y\}.$$

We shall prove that $L \subseteq TL$. Indeed, $L_1 \subseteq T_1 L_2$, $L_2 \subseteq T_3 L_3$, $L_3 \subseteq T_4 L_2$, hence $L = L_1 \cup L_2 \cup L_3 \subseteq T_1 L_2 \cup T_3 L_3 \cup T_4 L_2 \subseteq TL$. It means that $L \cap B = L_1 \subseteq Loop$, i.e.

$$\{(a, x, y, z): x, y \in J, z < 0\} \subseteq Loop.$$

That is, if the considered program stars with arbitrary values of x, y and with a negative value of z, then it diverges.//

Now we give a general theorem which can be useful for proving some facts connected with divergent computations.

Theorem 4. Let $T \subseteq S \times S$, and let X_1, X_2, X_3, X_4 be the greatest sets, satisfying respectively the conditions: $X_1 \subseteq TX_1$, $X_2 = TX_2$, $X_3 \subseteq T^+X_3$, $X_4 = T^+X_4$. Then $X_1 = X_2 = X_3 = X_4$.
Proof. Since $X_1 \subseteq TX_1$, $TX_1 \subseteq TTX_1$, hence $TX_1 \subseteq X_1$, because X_1 is the greatest set satisfying $X \subseteq TX$. That is, $X_1 = TX_1$, hence $X_1 \subseteq X_2$. Clearly, $X_2 \subseteq X_1$, thus $X_1 = X_2$. Putting T^+ instead T we obtain in the same way $X_3 = X_4$. Now, since $X_2 = TX_2$, we get by induction $X_2 = T^+X_2$, hence $X_2 \subseteq X_3$; since $X_3 = T^+X_3 = T^+T^+X_3 = TT^+X_3 = TX_3$, we obtain $X_3 \subseteq X_2$. That is, $X_2 = X_3$, and finally $X_1 = X_2 = X_3 = X_4$.//

The following corollaries to Theorem 4 can be applied in many practical situations.

Corollary 5. For each $L \subseteq S$, $L \subseteq T^+L \Rightarrow L \subseteq Lp$.//

Corollary 6. $T^*Lp \subseteq Lp$.

Proof. By Theorem 4, $Lp = TLp$, hence Lp is left invariant; by Theorem 1 $T^*Lp \subseteq Lp$.//

We use these corollaries in the following way. Let P represents a program, let $T_1, T_2, \ldots, T_n$, $n \geqslant 1$, $T_i \subseteq T$, $1 \leqslant i \leqslant n$, represent some instructions of this program. If U is a set of state-vectors such that $U \subseteq T_k T_{k+1} \ldots T_n U$, $1 \leqslant k < n$, and a state-vector s belongs to the set $T_1 T_2 \ldots T_{k-1} U$, then this program diverges for s.

The divergence set Lp is characterized implicitly as the greatest set satisfying the inclusion $Lp \subseteq TLp$. If the process P is deterministic, then we can obtain an explicit formula for Lp, given by the following theorem.

Theorem 5. If P is deterministic, then $Lp = \bigcap \{T^n S : n \geqslant 0\}$.

Proof. Denote $\bigcap \{T^n S : n \geqslant 0\}$ by M. Assume $s \in Lp$; hence for each $n \geqslant 0$ there is s_n such that $sT^n s_n$, i.e. $s \in M$. Assume $s \in M$; it means that for each $n \geqslant 0$ there is s_n such that $sT^n s_n$. Let

$$(s_0, s_1, \ldots, s_n, \ldots) \qquad (*)$$

be a sequence with the property $sT^n s_n$ for each $n \geqslant 0$. We shall show that $(*)$ is a computation in P. Indeed, since $sT^{n+1} s_{n+1}$, $sT^n T s_{n+1}$, hence there is $t \in S$ such that $sT^n t$ and tTs_{n+1}. But T is deterministic, hence, by definition, T and consequently T^n is a partial function; that is, $t = s_n$ what gives $s_n T s_{n+1}$. That is, $(*)$ is a computation in P, hence $s = s_0 \in Lp$. Therefore $Lp = M$.//

6. INDUCTIVE RELATIONS

In this section we give some methods for proving assertions (5)-(8). All these methods have as a basis the notion of inductive relations, defined as below. Let $P = (S, B, F, T)$ be a process, fixed for the sequel. Let U, V be subsets of S, or relations in $S \times S$. We shall write

$$U \ Rd \ V \qquad (U \ Ld \ V),$$

if there exists an ordinal number α and a transfinite sequence $(G_0, G_1, \ldots, G_\alpha)$ of subsets of S, or relations in $S \times S$, with the following properties:

(i) $U = G_0$, (ii) $V = G_\alpha$,

(iii) for each $\gamma \leqslant \alpha$ $G_\gamma \subseteq \bigcup\{G_\beta T^* : \beta < \gamma\}$ $G_\gamma \subseteq \bigcup\{T^* G_\beta : \beta < \gamma\}$, resp.).

In particular, U Rd V (U Ld V), if $U = G_0$, $V = G_\alpha$, and for each $\gamma < \alpha$

$G_{\gamma+1} \subseteq G_\gamma T^{n\gamma}$ $(G_{\gamma+1} \subseteq T^{n\gamma} G_\gamma)$ for some $n\gamma \geqslant 0$,

$G_\gamma \subseteq \bigcup\{G_\beta : \beta < \gamma\}$, if γ is a limit ordinal.

This last condition is useful in many practical cases, as we shall see in examples. It follows from the above definition that Rd and Ld are binary relations in $(2^S \times 2^S) \cup (2^{S\times S} \times 2^{S\times S})$, reflexive and transitive. We shall call them, respectively, right and left inductive relations for P. It can be observed that these notions are in a sense analoguous to the notion of computation relation and its inverse; the following theorem express this analogy.

Theorem 6. Let U, V be subsets of S or relations in $S \times S$. Then

$$U \ Rd \ V \Longleftrightarrow V \subseteq UT^* \quad \text{and} \quad U \ Ld \ V \Longleftrightarrow V \subseteq T^* U.$$

Proof. If $V \subseteq UT^*$, then (U, V) is the sequence mentioned in the definition of Rd, hence U Rd V. Assume, conversely, that U Rd V; hence, there exists a sequence $(G_0, G_1, \ldots, G_\alpha)$ with $G_0 = U$, $G_\alpha = V$, such that $G_\gamma \subseteq \bigcup\{G_\beta T^* : \beta < \gamma\}$, for each $\gamma \leqslant \alpha$. Clearly, $G_0 \subseteq UT^*$; suppose $G_\beta \subseteq UT^*$ for each $\beta < \gamma$; then $G_\gamma \subseteq \bigcup\{G_\beta T^* : \beta < \gamma\} \subseteq \bigcup\{UT^* : \beta < \gamma\} = UT^*$. Therefore, by transfinite induction, $G_\alpha \subseteq UT^*$, i.e. $V \subseteq UT^*$. The second part of the proof is similar.//

Corollary 7. Let $A \subseteq B$, $Z \subseteq F$; if A Rd Z, then $Z \subseteq A$ Res; if Z Ld A, then $A \subseteq$ Res Z.

Proof. Since A Rd Z, $Z \subseteq AT^{\omega}$; since $A \subseteq B$, $Z \subseteq AI_B T^{\omega}$ and since $Z \subseteq F$, $Z \subseteq AI_B T^{\omega} I_F = A$ Res. The second part of the proof is similar.//

This corollary can be used for proving assertions (5) and (6).

Corollary 8. If $A \subseteq S$ and F Ld A, then P halts for each s in A. Proof follows directly from Corollary 7.//

Example 5. Consider the process in Example 1. Define the sequence:

$$D_0 = \{(d, x, y, z) : x, y, z \in J\},$$
$$\dots$$
$$D_{m+1} = \{(b, x, y, z) : x, y \in J, z = y+m\}, \quad 0 \leqslant m < \omega$$
$$\dots$$
$$D_{\omega} = \{(b, x, y, z) : x, y \in J, z \geqslant y\},$$
$$D_{\omega+1} = \{(a, x, y, z) : x, y \in J, \quad z \geqslant 0\}.$$

Since $D_1 \subseteq T_2 D_0$, $D_{m+1} \subseteq T_3 T_4 D_m'$ for $0 \leqslant m$, $D_{\omega} = \cup\{D_m : m \geqslant 1\}$, $D_{\omega+1} = T_1 D_{\omega}$, we have $D_0 \mathrm{Ld} D_{\omega+1}$ and by Corollary 8 the process terminates for each state (a,x,y,z) with $x, y \in J$, and $z \geqslant 0$, because $D_0 = I_F$.//

Corollary 9. Let $Q \subseteq S \times S$. If I_B Rd Q, then $QI_F \subseteq$ Res; if I_F Ld Q, then $I_B Q \subseteq$ Res.
Proof is similar to the previous one.//

This corollary can be applied for proving assertion (7).

Example 6. Consider the process in Example 1. Define the sequence:

$$Q_0 = \{((d, x, y, z), (d, x, y, z)) : x, y, z \in J\},$$
$$\dots$$
$$Q_{m+1} = \{((b, 1, 0, m), (d, m!\, m, m))\}, \quad 0 \leqslant m < \omega$$
$$\dots$$
$$Q_{\omega} = \{((b, 1, 0, z), (d, z!, z, z)) : z \geqslant 0\},$$
$$Q_{\omega+1} = \{((a, x, y, z), (d, z!, z, z)) : x, y \in J, \quad z \geqslant 0\}.$$

Since $Q_1 \subseteq T_2 Q_0$, $Q_{m+1} \subseteq T_3 T_4 Q_m$ for $m \geq 0$, $Q_\omega = \cup \{Q_m : m > 1\}$, $Q_{\omega+1} = T_1 Q_\omega$, we have Q_0 Ld $Q_{\omega+1}$; by Corollary 9 we obtain $Q_{\omega+1} \subseteq$ Res, i.e.

$$\{((a, x, y, z), (d, z!, z, z)) : x, y \in J, \quad z \geq 0\} \subseteq \text{Res};$$

since in Example 3 we proved the inverse inclusion for the considered process, we obtain finally

$$\text{Res} = \{((a, x, y, z), (d, z!, z, z)) : x, y \in J, \quad z \geq 0\}.//$$

In order to prove assertion (8), i,e. that Loop $\subseteq$ L, observe that this inclusion is equivalent to the following one:

$$(S - L) \subseteq T^{\omega} (S - TS),$$

and thus it can be proved with the aid of Theorem 6.

The length of sequences defining inductive relations for a process depends upon the process complexity; in all previous examples we have dealt with a simple "one-loop" process. Now we give a more sophisticated example.

Example 7. Let $f: J \times J \to J$, $g: J \times J \times J \to J$ be total functions. In this example we shall consider termination properties of the following program scheme:

p: z:= 0;
a: y:= 0;
b: if f(x, y) = 0 then goto c; y := y+1; goto b;
c: z := g(x, y, z);
if x = 0 then stop; x := x-1; goto a;

with integer variables x, y, z. We assume the computation of f, g causes no side-effects. This program can be represented by the process P = (S, B, F, T) with
$S = \{p, a, b, c, d\} \times J^3$, $B = \{p\} \times J^3$, $F = \{d\} \times J^3$,
$T = T_1 \cup T_2 \cup T_3 \cup T_4 \cup T_5 \cup T_6$, where

$T_1 = \{((p, x, y, z), (a, x, y, 0)) : x, y, z \in J\}$,
$T_2 = \{((a, x, y, z), (b, x, 0, z)) : x, y, z \in J\}$,

$$T_3 = \{((b, x, y, z), (c, x, y, g(x, y, z))) : x, y, z \in J, f(x, y) = 0\},$$
$$T_4 = \{((b, x, y, z), (b, x, y+1, z)) : x, y, z \in J, f(x, y) \neq 0\},$$
$$T_5 = \{((c, x, y, z), (d, x, y, z)) : x, y, z \in J, x = 0\},$$
$$T_6 = \{((c, x, y, z), (a, x-1, y, z)) : x, y, z \in J, x \neq 0\}.$$

Denote by A the following subset of B:

$$\{(p, x, y, z) : x, y, z \in J, x \geqslant 0, (\forall k: 0 \leqslant k \leqslant x)(\exists n \geqslant 0) f(k, n) = 0\}.$$

We prove that A = Res F, i.e. that starting from the beginning the process P terminates if and only if $x \geqslant 0$ and for each non-negative integer k not greater than x there is at least one root of the equation $f(k, x) = 0$. At first, we prove that $A \subseteq$ Res F. To this effect define a sequence of subsets of S:

$$D_0 = \{(d, x, y, z) : x, y, z \in J\},$$
$$D_{n+1} = \{(b, 0, y, z) : x, y, z \in J, f(0, y+n) = 0\}, \quad 0 \leqslant n < \omega,$$
$$D_{\omega \cdot (m+1)} = \{(b, m, y, z) : x, y, z \in J, (\exists i \geqslant 0) f(m, y+1) = 0, (\forall k: 0 \leqslant k < m)(\exists n \geqslant 0) f(k, n) = 0\}, \quad 0 \leqslant m < \omega$$
$$D_{\omega \cdot (m+1)+n+1} = \{(b, m+1, y, z) : x, y, z \in J, (\forall k: 0 \leqslant k \leqslant m)(\exists i \geqslant 0) f(k, 1) = 0, f(m+1, y+n) = 0\}, \quad 0 \leqslant n < \omega, \quad 0 \leqslant m < \omega$$
$$D_{\omega \cdot \omega} = \{(b, x, y, z) : x, y, z \in J, x \geqslant 0, (\forall k: 0 \leqslant k < x)(\exists n \geqslant 0) f(k, n) = 0, (\exists i \geqslant 0) f(x, y+1) = 0\}.$$
$$D_{\omega \cdot \omega + 1} = \{(p, x, y, z) : x, y, z \in J, x \geqslant 0, (\forall k: 0 \leqslant k < x)(\exists n \geqslant 0) f(k, n) = 0\}$$

Since

$$D_1 = T_3 T_5 D_0,$$
$$D_{\gamma+1} \subseteq T_4 D_\gamma, \quad \gamma \text{ is an isolated ordinal},$$
$$D_\gamma \subseteq \bigcup \{D_\beta : \beta < \gamma\}, \quad \gamma \text{ is a limit ordinal}, \gamma \leqslant \omega \cdot \omega$$
$$D_{\gamma+1} \subseteq T_3 T_6 T_2 D_\gamma, \quad \gamma \text{ is a limit ordinal}, \gamma < \omega \cdot \omega$$
$$D_{\omega \cdot \omega + 1} \subseteq T_1 T_2 D_{\omega \cdot \omega}$$

we infer that D_0 Id A, and since $D_0 = F$, $A \subseteq B$, $A \subseteq$ Res F. Now, we prove that Res $F \subseteq A$. Let $U \subseteq S$, $U = U_1 \cup U_2 \cup U_3 \cup U_4 \cup U_5$, where

$U_1 = \{(p, x, y, z): x, y, z \in J,\ x \geqslant 0, (\forall k: 0 \leqslant k < x)(\exists n \geqslant 0) f(k,n) = 0\}$,

$U_2 = \{(b, x, y, z): x, y, z \in J,\ x \geqslant 0,\ (\forall k: 0 \leqslant k < x)(\exists n \geqslant 0) f(k, n) = 0,$
$(\exists i \geqslant 0) f(x, y+1) = 0\}$,

$U_3 = \{(a, x, y, z): x, y, z \in J,\ x \geqslant 0,\ (\forall k: 0 \leqslant k < x)(\exists n \geqslant 0) f(k, n) = 0\}$,

$U_4 = \{(c, x, y, z): x, y, z \in J,\ x \geqslant 0,\ (\forall k: 0 \leqslant k < x)(\exists n \geqslant 0) f(k, n) = 0\}$,

$U_5 = \{(d, x, y, z): x, y, z \in J\}$.

U is left invariant; indeed, $TU = (T_1 \cup T_2 \cup T_3 \cup T_4 \cup T_5 \cup T_6)$ $(U_1 \cup U_2 \cup U_3 \cup U_4 \cup U_5) = T_1U_3 \cup T_2U_2 \cup T_3U_4 \cup T_4U_2 \cup T_5U_5 \cup T_6U_3 \subseteq U_1 \cup U_3 \cup U_2 \cup U_2 \cup U_4 \cup U_4 \subseteq U$. Since $U \cap B = U_1$, $U \cap F = U_5$, $U_1 = A$, $U_5 = F$, we have Res $F \subseteq A$. That is

$$A = \text{Res } F.$$

In such a way assertion (2) combined with assertion (5) can be applied to solve termination problems for processes.//

7. FINAL REMARKS

The methods of proving assertions about processes, presented here, are applicable not only for procedure-free programs (as it could be suggested by examples), but also to programs with recursive procedures. Such programs can be described by processes with a special form of states: each state should contain an instantaneous description of a stack mechanism. For example, the recursive procedure

proc f: if x = 0 then y := 1 else begin x := x-1; f;
x := x+1; s := s×x end;

can be represented by the process $P = (S, B, F, T)$ with $S = \{f, a\}^* \times J^2 = V^* \times J^2$, $B = \{f\} \times J^2$, $F = \{\epsilon\} \times J^2$, $T = T_1 \cup T_2 \cup T_3$, where

$T_1 = \{((fw, 0, y), (w, 0, 1)) : y \in J, w \in V^*\}$,

$T_2 = \{((fw, x, y), (faw, x-1, y)) : x, y \in J, x \neq 0, w \in V^*\}$,

$T_3 = \{((aw, x, y), (w, x+1, y(x+1))) : x, y \in J, w \in V^*\}$.

Each word w over the alphabet V = {f, a} describes an actual state of procedure-calls stack. Clearly, all presented methods can be applied for such processes.

References

[1] BLIKLE A.: Complex Iterative Systems, Bull. Acad. Polon. Sci., Ser. Sci. Math. Phys. Astronom., 20 (1972), 57-62

[2] BLIKLE A., MAZURKIEWICZ A.: An Algebraic Approach to the Theory of Programs, Algorithms, Languages and Recursiveness, Proc. Symp. on Math. Found. of CS, Jabłonna (1972).

[3] FLOYD R.W.: Assigning Meanings to Programs, Proc. of Symposia in Applied Mathematics, vol. XIX (1967) 19-32.

[4] MAZURKIEWICZ A.: Proving Algorithms by Tail Functions, Information and Control, 18 (1970) 220-226.

[5] MAZURKIEWICZ A.: Iteratively Computable Relations, Bull. Acad. Polon. Sci., Ser. Sci. Math. Phys. Astronom., 20 (1972) 793-798, (a)

[6] MAZURKIEWICZ A.: Recursive Algorithms and Context-free Languages, ibid., 20 (1972) 799-803, (b)

[7] MAZURKIEWICZ A.: Podstawy teorii programowania maszyn cyfrowych, Problemy Przetwarzania Informacji, 2, WNT, Warszawa (in Polish) (in press).

[8] PAWLAK Z.: Maszyny Programowane, Algorytmy, 10 (1969) 5-19 (in Polish).

[9] WINKOWSKI J.: Interacting Abstract Machines, Bull. Acad. Polon. Sci., Ser. Sci. Math. Phys. Astronom., 20 (1972) 181-184 .

DOWODZENIE WŁASNOŚCI PROCESÓW

Streszczenie

W pracy rozważa się pewne problemy dotyczące procesów obliczeniowych, takie jak zależność wyników od danych, własności relacji wynikowych, pętlenie się, zatrzymywanie itp. Podaje się pewne metody algebraiczne dowodzenia takich własności.

ДОКАЗАТЕЛЬСТВА СВОЙСТВ ПРОЦЕССОВ

Резюме

В работе рассматриваются некоторые проблемы касающиеся к вычислительным процессам, такие как взаимосвязь результатов и данных, свойства выходных отношений, зацикливание, проблема останова и т.д. Приводится некоторые алгебраические методы доказательства таких свойств.

Warszawa, ul. Krzywickiego 34
A.M.

ALGORYTMY, XI, N° 19, 1974, 23-28

517.27:519.21

A COMMENT ON EFFECTIVENESS OF A RANDOM PROCEDURE FOR SEEKING MAXIMA

Ryszard ZIELIŃSKI

Instytut Matematyczny
Polskiej Akademii Nauk
Warszawa, ul. Śniadeckich 8

Received 14.10.1972

The note deals with the following method of construction of the sequence (x_n) in the problem of seeking maximum of a function f: $x_{n+1} = x_n + \xi$ if $f(x_n + \xi) > f(x_n)$ and $x_{n+1} = x_n$ otherwise, ξ being a random vector. Estimation from above of the probability of $\{f(x_n + \xi) > f(x_n)\}$ for convex functions is given.

Let $f: X \longrightarrow R$, $X \subset R^m$, be a given function. The problem consists in calculating the point $\bar{x} \in X$ such that $f(\bar{x}) \geq f(x)$ for any $x \in X$. Consider the following stochastic method of successive approximations. Choose $x_0 \in X$ as an arbitrary point. If $x_0, x_1, x_2, \ldots, x_{n-1}$ are given calculate x_n according to the following algorithm (see [1], [2], [3]): sample a point ξ uniformly distributed on the sphere $S_m(0,R)$ with radius R, centered at the origin. Put

$$x_n = \begin{cases} x_{n-1} + \xi, & \text{if } f(x_{n-1} + \xi) > f(x_{n-1}) \\ x_{n-1}, & \text{otherwise} \end{cases}$$

The random event $A_n = \{f(x_{n-1} + \xi) > f(x_{n-1})\}$ will be referred to as a success. In what follows an approximation from

above of probability of the success if f is a convex function will be given (it will be easy to see that the approximation may be extended for any measurable function).

Let ϱ denote the Euclidean metric in R^m. Let L_x be the family of semi-axes which begin at the point x and let $\varrho(y,l) = \min_{z \in l} \varrho(y,z)$ be the distance between the point y and $l \in L_x$. Denote

$$S_m(x_{n-1}, R) = \left\{ x : \varrho(x, x_{n-1}) = R \right\}$$

$$\Omega(x_{n-1}, R) = \left\{ x : \varrho(x, x_{n-1}) = R,\ f(x) > f(x_{n-1}) \right\}$$

Then the probability of the success is equal to

$$\frac{\text{Vol } \Omega(x_{n-1}, R)}{\text{Vol } S_m(x_{n-1}, R)} \tag{1}$$

We shall approximate the probability (1) from above as follows. Let

$$S_{m,\beta,l}(x_{n-1}, R) = \left\{ x : \varrho(x, x_{n-1}) = R,\ \varrho(x,l) \leqslant R\sin\beta \right\}$$

and

$$\alpha = \min \left\{ \beta : S_{m,\beta,l}(x_{n-1}, R) \supset \Omega(x_{n-1}, R),\ 0 \leqslant \beta \leqslant \pi,\ l \in L_{x_{n-1}} \right\}$$

Let λ be the $l \in L_{x_{n-1}}$ at which the above minimum is achieved. Then $S_{m,\alpha,\lambda}(x_{n-1}, R)$ is the minimal "circle" on the sphere $S_m(x_{n-1}, R)$ containing the set $\Omega(x_{n-1}, R)$; 2α is the vertex angle of the spherical cone which cuts off this circle on the sphere; λ is the axis of the above cone (see fig. 1).

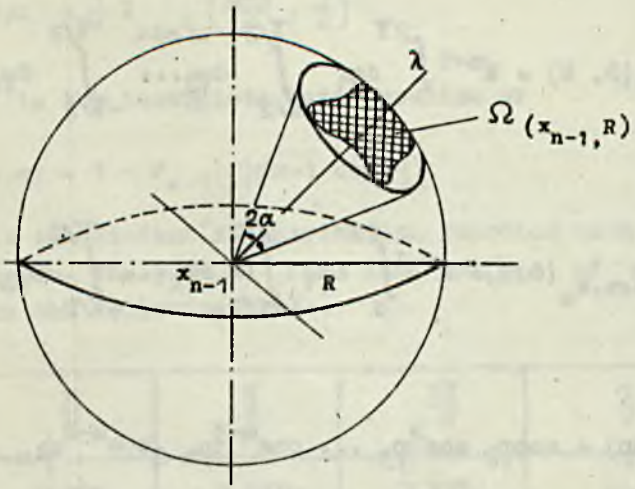


Fig. 1

The probability (1) may be approximated from above by

$$\frac{\text{Vol } S_{m,\alpha,\lambda}\ (x_{n-1},\ R)}{\text{Vol } S_m\ (x_{n-1},\ R)} \qquad (2)$$

Transfer the point x_{n-1} into the origin and let λ be the m-th coordinate-axis e_m. Denoting (2) by $p(m,\alpha)$ we have

$$p(m,\alpha) = \frac{\text{Vol } S_{m,\alpha,e_m}\ (0,\ R)}{\text{Vol } S_m(0,\ R)} \qquad (3)$$

Let $x^1, x^2, \ldots, x^m$ are the coordinates of the point x. Using the diffeomorphism

$$\begin{aligned}
x^1 &= R\cos\varphi_1\cos\varphi_2 \cdots \cos\varphi_{m-2}\cos\varphi_{m-1}\\
x^2 &= R\sin\varphi_1\cos\varphi_2 \cdots \cos\varphi_{m-2}\cos\varphi_{m-1}\\
x^3 &= R\sin\varphi_2\cos\varphi_3 \cdots \cos\varphi_{m-2}\cos\varphi_{m-1}\\
&\cdots\\
x^m &= R\sin\varphi_{m-1}
\end{aligned}$$

we obtain

$$\text{Vol } S_m(0, R) = R^{m-1} \int_0^{2\pi} d\varphi_1 \int_{-\pi/2}^{\pi/2} d\varphi_2 \cdots \int_{-\pi/2}^{\pi/2} d\varphi_{m-2} \int_{-\pi/2}^{\pi/2} C(\varphi) d\varphi_{m-1}$$

$$\text{Vol } S_{m,\alpha,e_m}(0,R) = R^{m-1} \int_0^{2\pi} d\varphi_1 \int_{-\pi/2}^{\pi/2} d\varphi_2 \cdots \int_{-\pi/2}^{\pi/2} d\varphi_{m-2} \int_{-\pi/2}^{-\frac{\pi}{2}+\alpha} C(\varphi) d\varphi_{m-1}$$

where

$$C(\varphi) = \cos\varphi_2 \cos^2\varphi_3 \cdots \cos^{m-3}\varphi_{m-2} \cos^{m-2}\varphi_{m-1}$$

For $p(m,\alpha)$ we have then

$$p(m,\alpha) = \frac{\int_{-\frac{\pi}{2}}^{-\frac{\pi}{2}+\alpha} \cos^{m-2} t\,dt}{\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{m-2} t\,dt}$$

If the function f is convex then $\alpha \in \left[0, \frac{\pi}{2}\right]$ and

$$p(m,\alpha) = \frac{\int_0^{\alpha} \sin^{m-2} t\,dt}{2\int_0^{\pi/2} \sin^{m-2} t\,dt} \qquad (4)$$

The probability $p(m,\alpha)$ may be calculated directly from the above formula (4) or by suitable tables with regard to

$$p(m,\alpha) = \frac{1}{2} I_{\sin^2\alpha}\left(\frac{m-1}{2}, \frac{1}{2}\right)$$

where $I_x(a,b)$ is the incomplete beta function or

$$p(m,\alpha) = 1 - F_{m-1}\left(\left|\sqrt{m-1}\ \mathrm{ctg}\alpha\right|\right)$$

where $F_k(x)$ is the Student's distribution function with k degrees of freedom. The $p(m,\alpha)$ for some values of m and α are given in the following table:

m \ α	0	$\frac{\pi}{8}$	$\frac{\pi}{4}$	$\frac{3\pi}{8}$	$\frac{\pi}{2}$
2	0	0.125	0.250	0.375	0.5
3	0	0.038	0.146	0.313	0.5
4	0	0.012	0,091	0.269	0.5
5	0	0.0043	0.058	0.227	0.5
10	0	<0.0001	0.0075	0.123	0.5
20	0	very small	0.0002	0.048	0.5

References

[1] MUCENIEKS V.A., RASTRIGIN L.N.: Prodviženie k celi metodom slučajnogo poiska, Izv. AN Latv. SSR, Ser. fiz. i techn. Nauk, 1964, 2, 93-104.

[2] RASTRIGIN L.N.: Statističeskie metody poiska, Izd. "Nauka", Moskwa 1968.

[3] ZIELIŃSKI R.: On Convergence of a Randomized Optimization Procedure, Algorytmy, VII, 12, 1970, 29-32

UWAGI O EFEKTYWNOŚCI PEWNEJ METODY LOSOWEGO SZUKANIA MAKSIMUM

Streszczenie

W pracy rozważa się następujący sposób budowy ciągu kolejnych przybliżeń do maksimum funkcji f: $x_{n+1} = x_n + \xi$ jeżeli $f(x_n + \xi) > f(x_n)$ oraz $x_{n+1} = x_n$ w przeciwnym przypadku, przy czym ξ jest losowym wektorem. Podano oszacowanie z góry prawdopodobieństwa zdarzenia $\{f(x_n + \xi) > f(x_n)\}$ dla funkcji wypukłych.

ОЦЕНКА ЭФФЕКТИВНОСТИ ОДНОГО МЕТОДА СЛУЧАЙНОГО ПОИСКА

Резюме

В статье рассматривается следующий алгоритм случайного поиска максимума функции $f(x)$: $x_{n+1} = x_n + \xi$ когда $f(x_n + \xi) > f(x_n)$ и $x_{n+1} = x_n$ в противном случае, ξ -случайный вектор. Дана оценка сверху вероятности случайного события $\{f(x_n + \xi) > f(x_n)\}$ для выпуклых функции.

Warszawa, ul. Krzywickiego 34
A.M.

ALGORYTMY, XI, N° 19, 1974, 29-47 519.821.2:519.25

GENEROWANIE n-WYMIAROWYCH SYMPLEKSÓW SKIEROWANYCH LOSOWO

Sylwestra DUTKIEWICZ
Przemysłowy Instytut Telekomunikacji
Warszawa, ul. Poligonowa 30

Pracę złożono 7.09.1973

Opisano metodę konstrukcji n-wymiarowych sympleksów losowo skierowanych, znajdujących zastosowanie w pewnych zagadnieniach Monte Carlo. Podany algorytm oparty jest na podstawowych pojęciach geometrii i przedstawiony w sposób pozwalający na zastosowanie go przy obliczeniach na elektronicznych maszynach cyfrowych. Praca zawiera również program napisany w języku Gier Algol 4 na podstawie podanej metody konstrukcji oraz zilustrowana jest przykładowymi wynikami dla kilku wymiarów przestrzeni.

1. WSTĘP

Rozważamy zagadnienie generowania sympleksu losowo skierowanego.

Przez sympleks w przestrzeni E^n rozumiemy układ (n+1) punktów $X_1, \ldots, X_{n+1}$ takich, że dla pewnego punktu A (nazwanego środkiem sympleksu)

$$\| X_i - A \| = \text{const} \qquad i = 1, \ldots, n+1$$

oraz

$$\sphericalangle(\overrightarrow{X_iA}, \overrightarrow{X_jA}) = \text{const} \qquad \text{dla } i \neq j$$

Wszędzie dalej przyjmować będziemy, że punkt A jest początkiem układu współrzędnych w E^n oraz, że

$$\| X_i - A \| = 1 \qquad i = 1, \ldots, n+1$$

Niech X_1 będzie wyróżnionym wierzchołkiem sympleksu. Przez losowo skierowany sympleks rozumiemy taki sympleks, w którym X_1 jest zmienną losową o rozkładzie jednostajnym na sferze jednostkowej o środku w początku układu współrzędnych.

Losowo skierowane sympleksy znajdują zastosowanie w pewnych zagadnieniach Monte Carlo np. przy konstrukcji zrandomizowanych skończenie-różnicowych oszacowań gradientu (por. [1]) oraz w zagadnieniach estymacji współczynników funkcji regresji (por. [2], [3]).

Losowo skierowany sympleks będziemy generowali w następujący sposób:

1. Ustalimy pewien sympleks jako wyjściowy. Za ten sympleks przyjmujemy taki, którego wierzchołki dane są kolejnymi kolumnami następującej macierzy

$$S = [S_1, \ldots, S_{n+1}] = \begin{bmatrix} u_1 & -r_1 & 0 & \ldots & 0 & 0 \\ u_2 & u_2 & -r_2 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ u_{n-1} & u_{n-1} & u_{n-1} & \ldots & -r_{n-1} & 0 \\ u_n & u_n & u_n & \ldots & u_n & -r_n \end{bmatrix}$$

gdzie

$$u_j = \sqrt{\frac{n+1}{j(j+1)n}} \qquad r_j = \sqrt{\frac{j(n+1)}{(j+1)n}}$$

2. Wylosujemy punkt $w = (w_1, \ldots, w_n)$ według rozkładu równomiernego na sferze jednostkowej.

3. Skonstruujemy układ współrzędnych, w którym wektory S_1 i w będą leżały w jednej płaszczyźnie.

4. a) Przedstawimy punkty sympleksu w skonstruowanym układzie współrzędnych.

b) Obrócimy płaszczyznę wektorów S_1, w tak, że punkt S_1 przejdzie na w ((n - 2)-wymiarowa podprzestrzeń ortogonalna do S_1 i w jest niezmiennikiem tego obrotu).

c) Punkty nowo otrzymanego sympleksu przedstawimy w wyjściowym układzie współrzędnych.

2. KONSTRUKCJA UKŁADU WSPÓŁRZĘDNYCH, W KTÓRYM WEKTORY S_1 i w BĘDĄ LEŻAŁY W JEDNEJ PŁASZCZYŹNIE

Konstrukcję tę można przeprowadzić w następujący sposób: przyjmijmy jako wektor kierunkowy pierwszej osi szukanego układu wektor w i oznaczmy go przez X_1 oraz oznaczmy wektor S_1 przez X_2 (pamiętając, że nie musi być $S_1 \perp w$).

W pierwszym kroku znajdujemy wektor $X_3 = (x_{31}, \ldots, x_{3n})$ prostopadły do X_1 i X_2, co jest równoważne z rozwiązaniem następującego układu równań

$$x_{31}s_{11} + x_{32}s_{12} + \ldots + x_{3n}s_{n1} = 0$$
$$x_{31}w_1 + x_{32}w_2 + \ldots + x_{3n}w_n = 0$$

Obliczymy x_{31}, x_{32} w zależności od $x_{33}, x_{34}, \ldots, x_{3n}$ przyjmując np.: $x_{33} = 1$ $x_{34} = \ldots = x_{3n} = 0$. Jako rozwiązanie naszego układu otrzymamy wektor $X_3 = (x_{31}, x_{32}, 1, 0, \ldots, 0)$ prostopadły do X_1 i X_2. Wektor ten przyjmiemy jako wektor kierunkowy trzeciej osi konstruowanego układu współrzędnych.

Następnie w ten sam sposób obliczamy pozostałe n-3 wektory, żądając aby każdy kolejno obliczany wektor X_i $i = 4, \ldots, n$ był prostopadły do obliczonych poprzednio X_j $j = 1, \ldots, i - 1$.

W kolejnych krokach wyrażać będziemy niewiadome $x_{i,1}, \ldots$ $\ldots, x_{i,i-1}$ za pomocą $n - i + 1$ pozostałych, przyjmowanych zawsze w postaci $x_{i,i} = 1$, $x_{i,i+1} = \ldots = x_{i,n} = 0$. Wektor X_i będzie miał więc postać $(x_{i,1}, \ldots, x_{i,i-1}, 1, 0, \ldots, 0)$ i przyjmować go będziemy za wektor kierunkowy i-tej osi współ-

rzędnych konstruowanego układu. Dostaniemy w ten sposób układ n wektorów $X_1, \ldots, X_n$ taki, że

$$X_1 X_i = 0 \qquad i = 3, 4, \ldots, n$$
$$X_1 X_2 = 0 \qquad i = 3, 4, \ldots, n$$

przy czym $X_1 X_2$ nie musi zniknąć.

Geometrycznie oznacza to, że wektory X_1, X_2 leżą w jednej płaszczyźnie, do której prostopadłe są wszystkie pozostałe wektory X_i, $i = 3, 4, \ldots, n$.

Ponieważ X_1 przyjęliśmy jako wektor kierunkowy pierwszej osi współrzędnych, aby otrzymać układ, w którym X_1, X_2 leżą w jednej płaszczyźnie wystarczy znaleźć wektor X_2' prostopadły do X_1, X_3, $X_4, \ldots, X_n$. Wektora tego szukamy podaną wyżej metodą i otrzymamy w postaci $X_2' = (x'_{2,1}, x'_{2,2}, \ldots, x'_{2,n})$. Przyjmujemy go jako wektor kierunkowy drugiej osi układu.

W ten sposób skonstruowany został układ n wektorów parami prostopadłych taki, że wektor X_2 (czyli S_1) leży w płaszczyźnie wektorów X_1 (czyli w) i X_2'.

Normując te wektory otrzymamy układ n wersorów prostopadłych $X''_1, \ldots, X''_n$, które przyjmujemy za wersory osi szukanego układu współrzędnych.

Kolejne wersory tego układu zapisać można w kolejnych wierszach macierzy ortonormalnej

$$X'' = \begin{bmatrix} x''_{11} & x''_{12} & \cdots & x''_{1n} \\ x''_{21} & x''_{22} & \cdots & x''_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x''_{n1} & x''_{n2} & \cdots & x''_{nn} \end{bmatrix}$$

3. OBRÓT SYMPLEKSU, PRZY KTÓRYM PUNKT S_1 PRZEJDZIE NA PUNKT w WYBRANY LOSOWO

Jeśli obliczymy iloczyn $S' = X''S$, to kolejne kolumny macierzy S' przedstawiać będą wierzchołki sympleksu w nowych współrzędnych.

Zgodnie z konstrukcją bazy wektor $S_1' = (s_{11}', s_{12}', 0, \ldots, 0)$ leżeć będzie w płaszczyźnie wersorów X_1'', X_2''. Jeśli więc dokonamy w tej płaszczyźnie obrotu względem środka układu współrzędnych, to n - 2 wymiarowa podprzestrzeń ortogonalna do X_1'', X_2'' będzie jego niezmiennikiem.

Obróćmy więc tę płaszczyznę o kąt $\sphericalangle(w, S_1)$ taki, że punkt S' przejdzie na punkt w.

W tym celu obliczamy $\cos \sphericalangle(w, S_1) = wS_1$ ($\|w\| = \|S_1\| = 1$) oraz $|\sin \sphericalangle(w, S_1)| = \sqrt{1 - \cos^2 \sphericalangle(w, S_1)}$, znak sinusa ustalamy badając, w której ćwiartce płaszczyzny leży wektor S_1'.

Znając $\sin \sphericalangle(w, S_1)$ i $\cos \sphericalangle(w, S_1)$ dokonujemy obrotu

$$s_{1i}'' = s_{1i}' \cos \sphericalangle(w, S_1) + s_{2i}' \sin \sphericalangle(w, S_1)$$

$$s_{2i}'' = -s_{1i}' \sin \sphericalangle(w, S_1) + s_{2i}' \cos \sphericalangle(w, S_1)$$

$i = 1, \ldots, n+1$. (Pozostałe współrzędne punktów sympleksu pozostaną bez zmian).

Macierz

$$S'' = \begin{bmatrix} s_{11}'' & s_{12}'' & \cdots & s_{1n}'' \\ s_{21}'' & s_{22}'' & \cdots & s_{2n}'' \\ s_{31}' & s_{32}' & \cdots & s_{3n}' \\ \cdots & \cdots & \cdots & \cdots \\ s_{n1}' & s_{n2}' & \cdots & s_{nn}' \end{bmatrix}$$

przedstawia kolejne wierzchołki sympleksu powstałego po obrocie, zapisane w skonstruowanym układzie współrzędnych.

Pozostaje już tylko wyrazić S'' w wyjściowym układzie współrzędnych $e_1, \ldots, e_n$, tzn. po przejściu

$$\hat{S} = X^T S''$$

Kolumny tak obliczonej macierzy $\hat{S}$ przedstawiać będą wierzchołki sympleksu wyjściowego S po dokonaniu przekształcenia izometrycznego przestrzeni E^n takiego, że punkt S_1 sympleksu przejdzie na wybrany losowo punkt w.

4. KRÓTKI OPIS PROGRAMU DLA EMC GIER

I. Program korzystający z podanego w rozdziałach 2 i 3 algorytmu, zapisany w języku Gier Algol 4, wymaga zadeklarowania następujących wielkości:

- tablica X typu real, wymiaru nxn, z której korzystamy przy konstrukcji nowej bazy i która po wykonaniu obliczeń jest macierzą przejścia do nowych współrzędnych (a po transpozycji macierzą przejścia do współrzędnych wyjściowych),
- tablica S typu real, wymiaru $n \times (n+1)$, będąca macierzą sympleksu S opisanego we wstępie, a po wykonaniu obliczeń zawierająca przekształcony sympleks,
- zmienna prosta s typu real, której przypisujemy wartość $\sin \sphericalangle (S_1, w)$,
- zmienna prosta c typu real, której przypisujemy wartość $\cos \sphericalangle (S_1, w)$,
- zmienna prosta n typu integer, wskazująca wymiar przestrzeni.

Pozostałe występujące w programie oraz schematic zmienne i tablice traktowane będą jako pomocnicze.

II. a) Przy losowaniu punktu program korzysta z procedury generowania liczb pseudolosowych o rozkładzie normalnym z wartością przeciętną równą zero i odchyleniem standardowym równym 1 - patrz [4].

b) Przy rozwiązywaniu układu równań opisanego w rozdz. 2 stosowana jest procedura U(A, n, R) rozwiązywania układu n równań z n niewiadomymi o współczynnikach zapisanych w macierzy A, metodą eliminacji Gaussa bez wyboru elementu głównego.
Wynik podstawiony jest na wektor R.
Procedura niszczy macierz A będącą jej argumentem.
Taka metoda rozwiązywania układu równań wymaga osobnego rozpatrywania przypadku gdy $w_1 = 0$ oraz gdy $S_i = w_i \quad i = 1, 2, \ldots, n$.
Oba te przypadki uwzględnione są w załączonym programie.
Dla uproszczenia obliczeń jako płaszczyzna obrotu przyjęta została płaszczyzna wektorów X_n, w, co nie ma oczywiście wpływu na wyniki obliczeń.

c) Procedura IL(A, B, C, n)) wykonuje mnożenie macierzy A $(n \times n)$ i B $(n \times (n + 1)$.
C $(n \times (n + 1))$ jest macierzą zawierającą iloczyn.
Procedura nie niszczy macierzy A i B.

d) Można zauważyć, że po wywołaniu procedury IL (X, S, Y, n) element Y [1, 1] przedstawiać będzie wartość $\cos \sphericalangle (S_1, w)$, a element Y [n, 1] określi znak $\sin \sphericalangle (S_1, w)$, następnie stosowane są standardowe wzory na obrót płaszczyzny.

e) Po kolejnym wywołaniu procedury IL(X^T, Y, S, n) w tablicy S znajdują się współrzędne sympleksu S po dokonaniu przekształcenia.

III. Danymi wejściowymi dla programu są:

a) liczba całkowita dodatnia będąca wymiarem przestrzeni

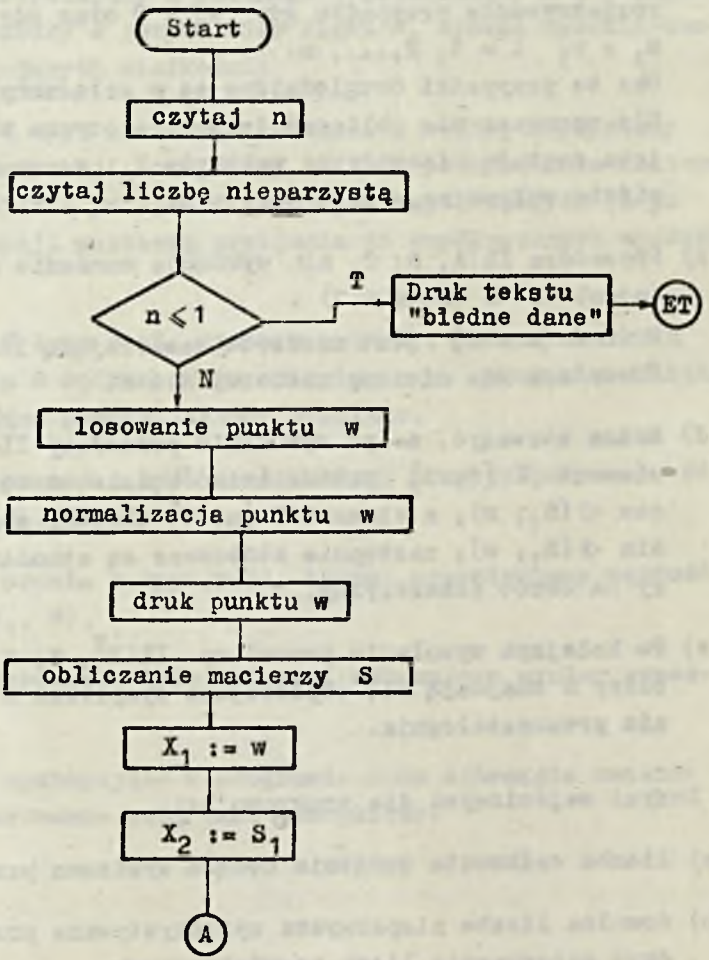
b) dowolna liczba nieparzysta wykorzystywana przez procedurę generowania liczb pseudolosowych.

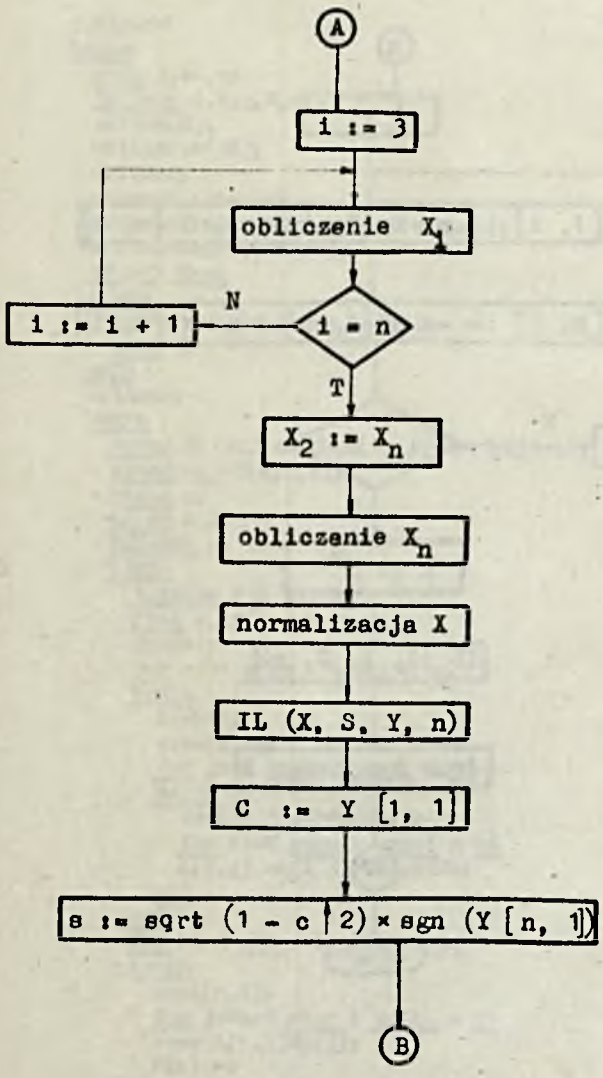
W wyniku działania programu zostają wyprowadzone na drukarkę wierszową współrzędne wylosowanego wektora w oraz współrzędne sympleksu S po dokonaniu obrotu. W przypadku $n \leqslant 1$ zostaje wydrukowany tekst "bledne dane".

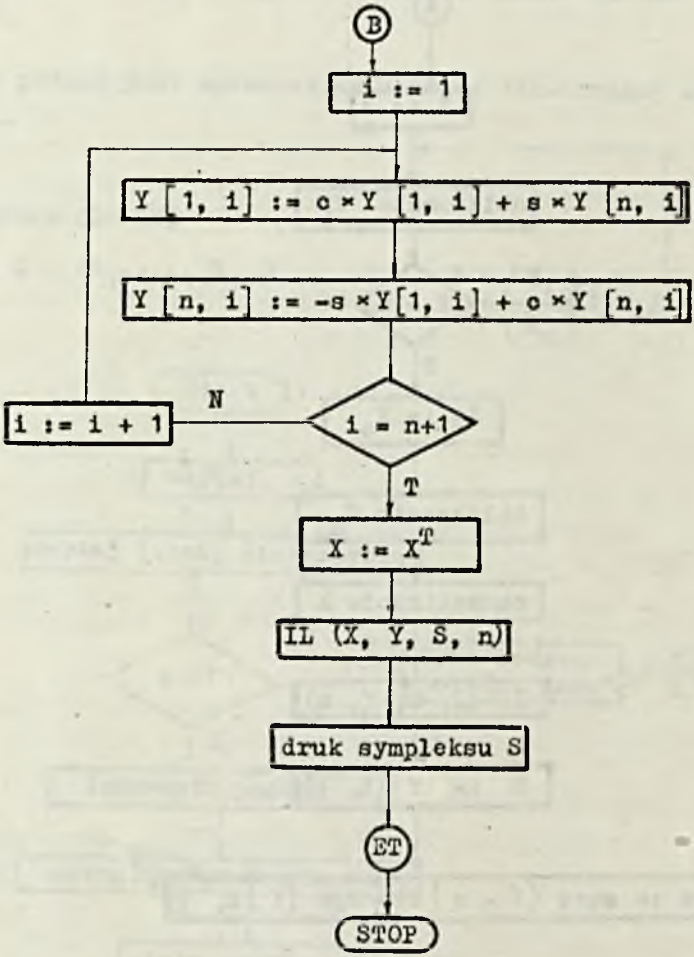
W rozdz. 5 podany jest uproszczony schemat ilustrujący stosowaną metodę.

5. OGÓLNY SCHEMAT BLOKOWY

Oznaczenia $S = (S_1, \ldots, S_{n+1})$ $\quad X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$




A
i := 3
obliczenie X_i
N
i := i + 1
i = n
T
$X_2 := X_n$
obliczenie X_n
normalizacja X
IL (X, S, Y, n)
C := Y [1, 1]
s := sqrt (1 - c ↑ 2) × sgn (Y [n, 1])
B


B
i := 1
Y [1, i] := c × Y [1, i] + s × Y [n, i]
Y [n, i] := -s × Y [1, i] + c × Y [n, i]
i = n+1
N
i := i + 1
T
X := X^T
IL (X, Y, S, n)
druk sympleksu S
ET
STOP

6. PROGRAM NAPISANY W JĘZYKU GIER ALGOL 4

```
1,algol<
begin
  real s,s1,c;
  integer n,i,j,k,m;
  select(8);
  writechar(58);
  writecr;
  n:=read integer;
  writetext(‡<wymiar przestrzeni=‡);
  writeinteger(‡dd‡,n);
  if n<1 then
  begin
    writetext(‡<bledne dane‡);
    goto ET
  end;
  writecr;
  begin
    array X[1:n,1:n],Y,S[1:n,1:n+1],w[1:n];
    procedure U(A,n,R);
    value n;
    array A,R;
    integer n;
    begin
      integer H,h,i,j,N;
      real s,t;
      N:=n+1;
      for h:=1 step 1 until n do
      begin
        H:=h+1;
        s:=A[h,h];
        for j:=H step 1 until N do
        begin
          t:=A[h,j]:=A[h,j]/s;
          for i:=H step 1 until n do
          A[i,j]:=A[i,j]-A[i,h]×t;
        end
      end;
      for h:=n step -1 until 1 do
      begin
        s:=A[h,N];
        for j:=h+1 step 1 until n do
        s:=s-A[h,j]×R[j];
        R[h]:=s
      end
    end;
    procedure IL(A,B,C,n);
    value n;
    integer n;
    array A,B,C;
```

```
    begin
      integer i,j,k,N;
      real s;
      N:=n+1;
      for k:=1 step 1 until N do
      for i:=1 step 1 until n do
      begin
        s:=0;
        for j:=1 step 1 until n do
        s:=A[i,j]×B[j,k]+s;
        C[i,k]:=s
      end
    end;
    integer random;
    boolean gln;
    real rrandom;
    core code gln,random,rrandom;
    3,46
    1,44
    1,45
    grn re2
    pa re3 t11
    is(b2), pms a2
    mln re1 X
    is(b2), gr sa2
    t1-4, acre2
e3:
    bt11t-1
    hvr-5
    arn re2,srre4
    gr re2
    nkf4
    is(b3), grf sa3
    arn re2, hrs1
e2:
    0
e1:
    152587890625
e4:
    qq192
    e
    random:=read integer;
    s:=0;
    writecr;
    writetext(‡<wylosowany punkt w‡);
    writecr;
    for i:=1 step 1 until n do
    begin
      gier(gln);
      X[1,i]:=rrandom;
      s:=s+X[1,i]↑2
    end;
    s:=sqrt(s);
```

```
    for i:=1 step 1 until n do
    begin
      X[1,i]:=X[1,i]/s;
      write('-p.dddd_10-dd',X[1,i]);
      writecr;
      for j:=1 step 1 until i do
      S[i,j]:=sqrt((n+1)/(i×(i+1)));
      for j:=i+1 step 1 until n+1 do
      S[i,j]:=0;
      S[i,i+1]:=-sqrt((i×(n+1))/((i+1)×n))
    end;
    for i:=2 step 1 until n do
    for j:=1 step 1 until n do
    X[i,j]:=if i=2 then S[j,1] else if i=j then 1 else 0;
    for i:=1 step 1 until n do
    if abs(X[1,i])<10-8 ∨ abs(X[1,i]-X[2,i])<10-8 then
    m:=i else if i+1 then
    begin
      Y[1,1]:=X[1,1];
      Y[2,1]:=X[2,1];
      X[1,1]:=X[1,i];
      X[2,1]:=X[2,i];
      X[1,i]:=Y[1,1];
      X[2,i]:=Y[2,1];
      m:=m+1;
      goto F
    end else
    begin
      m:=1;
      goto F
    end;        if m=n then go to ES ;
F:  for i:=2 step 1 until n-1 do
    begin
      for j:=1 step 1 until i do
      for k:=1 step 1 until i+1 do
      if k=i+1 then Y[j,k]:=-X[j,k] else Y[j,k]:=X[j,k];
      U(Y,i,w);
      for j:=1 step 1 until i do X[i+1,j]:=w[j];
    end;
    for j:=1 step 1 until n do X[2,j]:=X[n,j];
    X[n,n]:=1;
    for i:=1 step 1 until n-1 do
    for j:=1 step 1 until n do
    if j=n then Y[i,j]:=-X[i,j] else Y[i,j]:=X[i,j];
    U(Y,n-1,w);
    for j:=1 step 1 until n-1 do X[n,j]:=w[j];
    for i:=1 step 1 until n do
    begin
      s1:=0;
      for j:=1 step 1 until n do
      s1:=s1+X[i,j]↑2;
      s1:=sqrt(s1);
      for j:=1 step 1 until n do X[i,j]:=X[i,j]/s1;
    end;
```

```
      if m=1 then
      for i:=1 step 1 until n do
      begin
        Y[i,1]:=X[i,1];
        X[i,1]:=X[i,m];
        X[i,m]:=Y[i,1]
      end;
      IL(X,S,Y,n);
      c:=Y[1,1];
      s:=sqrt(1-c↑2);
      if Y[n,1]<0 then s:=-s;
      for i:=1 step 1 until n+1 do
      begin
        s1:=Y[1,i];
        Y[1,i]:=cXY[1,i]+sXY[n,i];
        Y[n,i]:=-sXs1+cXY[n,i]
      end;
      for i:=1 step 1 until n do
      for j:=i+1 step 1 until n do
      begin
        s1:=X[i,j];
        X[i,j]:=X[j,i];
        X[j,i]:=s1
      end;
      IL(X,Y,S,n);
ES:   writecr;
      writetext(‡<Sympleks po dokonaniu obrotu‡);
      writecr;
      for i:=1 step 1 until n do
      begin
        for j:=1 step 1 until n+1 do
        write(‡-p.dddd␣-dd‡,S[i,j]);
        writecr
      end
    end;
ET:
end
run<
ø
res,pr<
reprint,pr<
pr<
```

7. PRZYKŁADOWE WYNIKI I CZASY OBLICZEŃ

```
WYMIAR PRZESTRZENI=    5

WYLOSOWANY PUNKT W
-1.8006 E-1
 4.6093 E-1
-5.1354 E-1
-4.2417 E-1
 5.5805 E-1

SYMPLEKS PO DOKONANIU OBROTU
-1.8006 E-1-6.9667 E-1 5.7204 E-1-1.2021 E-1-1.4127 E-1 5.6620 E-1
 4.6093 E-1 2.7182 E-1-3.1500 E-1-4.7807 E-1-5.1024 E-1 5.7051 E-1
-5.1354 E-1 4.1969 E-1 3.5570 E-1-6.7946 E-1 2.7621 E-1 4.1406 E-2
-4.2417 E-1 4.1284 E-1 2.6389 E-1 4.7257 E-1-7.4583 E-1 2.0699 E-2
 5.5805 E-1-1.5706 E-2 6.1432 E-1-2.6635 E-1-2.9520 E-1-5.9312 E-1


WYMIAR PRZESTRZENI=    8

WYLOSOWANY PUNKT W
 3.2670 E-1
-2.7403 E-1
-2.1926 E-1
-3.1742 E-1
 4.4006 E-1
 4.8864 E-1
-2.5553 E-1
-6.8266 E-2

SYMPLEKS PO DOKONANIU OBROTU
 3.2670 E-1-2.6315 E-1-1.2538 E-1-1.5718 E-1-5.7707 E-1
-2.7403 E-1 5.8547 E-1-8.0327 E-1 5.7358 E-2-4.6408 E-3
-2.1926 E-1 4.0690 E-1 3.5747 E-1-8.6984 E-1 1.4821 E-2
-3.1742 E-1 2.1855 E-1 3.7790 E-1 3.8618 E-1-6.9038 E-1
 4.4006 E-1 4.1825 E-1 5.7912 E-2 3.9193 E-2-2.0794 E-1
 4.8864 E-1 3.8594 E-1 1.5026 E-2-4.2429 E-3-2.5862 E-1
-2.5553 E-1 1.4553 E-1 2.1024 E-1 2.1360 E-1 2.5797 E-1
-6.8266 E-2 1.7202 E-1 1.3976 E-1 1.3809 E-1 1.1596 E-1


 4.4382 E-1 5.8005 E-1-2.0391 E-1-2.3881 E-2
-1.4610 E-1 1.6622 E-1 5.0459 E-2 7.7041 E-2
-9.7240 E-2 1.0824 E-1 4.4947 E-2 5.9481 E-2
-7.4253 E-3-4.2882 E-2 1.6117 E-1 1.1431 E-1
-7.6897 E-1 2.7945 E-1-1.8196 E-1-7.5999 E-2
 3.5987 E-1-7.0325 E-1-1.9621 E-1-8.7146 E-2
-1.5008 E-1 1.3568 E-1-9.1536 E-1 5.7774 E-2
-1.6976 E-1 1.7693 E-1 1.3563 E-1-9.7989 E-1

WYMIAR PRZESTRZENI=    7

WYLOSOWANY PUNKT W
-5.2922 E-1
-2.5415 E-1
-2.1493 E-1
-3.6399 E-1
-4.2508 E-1
-2.5476 E-2
-1.9727 E-1

SYMPLEKS PO DOKONANIU OBROTU
-5.2922 E-1-2.6400 E-1-1.1735 E-1 8.4001 E-1-2.7594 E-1
-2.5415 E-1 4.8501 E-1-6.5623 E-1-4.7443 E-2 2.6039 E-1
-2.1493 E-1 7.3534 E-1-1.2182 E-2-7.2027 E-2-5.1536 E-1
-3.6399 E-1 9.7206 E-2 5.6118 E-1-1.6786 E-1-5.1328 E-1
-4.2508 E-1 1.0227 E-1 4.3053 E-1-8.5259 E-2 5.1597 E-1
-2.5476 E-2 3.3524 E-1 6.3330 E-2 4.9058 E-1-7.4453 E-3
-1.9727 E-1 1.4849 E-1 2.2618 E-1 1.0410 E-1 2.4640 E-1
```

```
-2.2292 *-1 3.4491 *-1 2.4513 *-2
 2.4577 *-1 8.9131 *-2 1.7751 *-1
-4.5031 *-1 2.4635 *-1-1.4674 *-1
 4.0253 *-1-2.9880 *-2 2.1410 *-1
-6.8367 *-1-1.3695 *-2 1.5892 *-1
 1.6216 *-2-8.8508 *-1-3.8317 *-2
 2.3964 *-1 1.6723 *-1-9.3477 *-1

WYMIAR PRZESTRZENI=    9

WYLOSOWANY PUNKT W
 1.8030 *-1
-3.3685 *-1
 4.3583 *-1
-9.9947 *-2
 2.4286 *-1
 8.4062 *-2
 4.4589 *-2
 5.1916 *-1
 5.6261 *-1

SYMPLEKS PO DOKONANIU OBROTU
 1.8030 *-1-7.2260 *-1-3.8503 *-1 2.0265 *-1-1.6673 *-1
-3.3685 *-1 5.1153 *-1-8.2065 *-1 9.9368 *-2 6.2062 *-2
 4.3583 *-1 2.5767 *-1-6.8377 *-2-8.1566 *-1-1.9812 *-1
-9.9947 *-2 2.6713 *-1 2.0732 *-1 2.9351 *-1-9.3917 *-1
 2.4286 *-1 1.6802 *-1-2.3745 *-2 2.5262 *-1 7.8915 *-2
 8.4062 *-2 1.6071 *-1 5.2028 *-2 2.0866 *-1 1.1021 *-1
 4.4589 *-2 1.4354 *-1 6.1840 *-2 1.7958 *-1 1.0557 *-1
 5.1916 *-1 5.2522 *-2-2.3091 *-1 1.7756 *-1-7.9181 *-2
 5.6261 *-1 3.1810 *-2-2.6499 *-1 1.6274 *-1-1.0610 *-1


 1.0166 *-1 2.8148 *-3-1.9958 *-2 3.6434 *-1 4.4255 *-1
 8.9168 *-2 7.9185 *-2 7.6885 *-2 1.1570 *-1 1.2360 *-1
 1.6468 *-2-6.2564 *-2-8.0772 *-2 2.2650 *-1 2.8903 *-1
 4.2996 *-2 2.8499 *-2 2.5160 *-2 8.1520 *-2 9.2990 *-2
-9.4957 *-1-3.3803 *-2-4.4513 *-2 1.3621 *-1 1.7299 *-1
 1.8174 *-1-9.8315 *-1-1.3321 *-2 8.9105 *-2 1.0995 *-1
 1.5934 *-1 1.3954 *-1-9.9189 *-1 7.1111 *-2 8.6780 *-2
 1.0737 *-1 3.8663 *-2 2.2834 *-2-8.2809 *-1 2.2008 *-1
 8.9239 *-2 1.7297 *-2 7.2236 *-4 2.8043 *-1-7.7376 *-1
```

Wymiar przestrzeni	Przybliżony czas obliczeń w sekundach
3	5
5	11
6	15
9	23
10	26
11	33
12	41

Literatura

[1] ZIELIŃSKI R.: A Randomized Finite-Differential Estimator of the Gradient. Algorytmy, Vol. X, Nr 18, 1973, p. 21-30.

[2] ZIELIŃSKI R.: Pewna metoda planowania doświadczeń dla estymacji gradientu regresji drugiego stopnia. Matematyka Stosowana (w druku).

[3] ZIELIŃSKI R.: A Simplex Design for Gradient Estimation in Quadratic Regression, Zastosowania Matematyki (w druku).

[4] KACZMARSKA E., SALWICKI A.: Generowanie liczb pseudolosowych, Warszawa, Zakład Obliczeń Numerycznych i Katedra Metod Numerycznych. Sprawozdania, z. 3, Wyd. U.W. 1967.

ГЕНЕРИРОВАНИЕ n - МЕРНЫХ СИМПЛЕКСОВ НАПРАВЛЕННЫХ СЛУЧАЙНО

Резюме

Работа содержит описание конструкции n - мерных симплексов направленных случайно, имеющих применение в некоторых проблемах Монте-Карло, и представляет метод их генерирования.

Через n - мерный симплекс направленный случайно разумеется такая система точек $X_1, \ldots, X_{n+1}$, что справедливы следующие равенства:

$$\|X_i\| = 1 \qquad i = 1, \ldots, n+1$$

$$\sphericalangle(X_i, X_j) = \text{конст} \qquad i \neq j$$

а вершина X_1 является случайной величиной равномерно распределенной на единичной сфере.

Вопрос конструкции такого симплекса решен следующим образом:

I. Некоторый симплекс установляется как начальный
2. Точка w выбирается случайно согласно с равномерном распределением на единичной сфере
3. Начальный симплекс поворачивается мак, чтобы вершина X_1 нашлась в точке w .

Алгоритм генерации основан на основных понятиях геометрии, а его форма разрешает применение ЭВМ.

В работе представляется тоже программа вычислений на языке Cier Algol 4 и примеры вычислений для разных размерностей n .

GENERATION OF THE n-DIMENSIONAL RANDOM ORIENTED SIMPLEXES

Summary

The paper contains a description of a method of construction of n-dimensional random oriented simplexes used in Monte Carlo methods.

The n-dimensional random oriented simplex is defined as such system of points $X_1, \ldots, X_{n+1}$, that the following formulaś hold:

$$\|X_i\| = 1 \qquad i=1,\ldots, n+1$$

$$\sphericalangle(X_i, X_j) = \text{const} \qquad i \neq j$$

X_1 being a random variable with the uniform distribution on the elementary sphere.

The problem of construction of such a simplex is solved as follows:

1. A certain simplex is fixed as initial.
2. The point w according to the uniform distribution on the elementary sphere is sampled.
3. The rotation of the initial simplex is made in such a way that the vertex X_1 is transformed onto w.

The algorithm of computations is based on elementary geometrical conceptions and can be employed in computer aided calculations.

A Gier Algol 4 program was written according to the presented method. The program is included in the paper, as well as the illustrating examples for different dimensions.

Warszawa, ul. Krzywickiego 34
R.Z.

ALGORYTMY, XI, N° 19, 1974, 69-70

681.3.001:517

PEWIEN ALGORYTM IDENTYFIKACJI MINIMALNEGO I SPÒJNEGO AUTOMATU O OGRANICZONEJ LICZBIE STANÒW

Lesław KRZYWDA

Studium Doktoranckie
Politechniki Warszawskiej

Pracę złożono 22.01.1973

W artykule omówiono problem określania struktury logicznej dowolnego spójnego i minimalnego automatu typu Mealy o liczbie stanów mniejszej niż n (n jest pewną ustaloną liczbą) oraz znanym alfabecie wejściowym i wyjściowym. Po wprowadzeniu pojęcia tablicy regularnej automatu omówiono podstawy teoretyczne oraz podano ogólny opis algorytmu identyfikacji automatu.

WPROWADZENIE

Oznaczmy badany automat przez B. Automat ten opisany jest uporządkowaną piątką $\{Q, X, Y, \delta, \lambda,\}$ gdzie

Q - zbiór stanów automatu B, $Q = \{\alpha_1, \alpha_2, \alpha_3, \ldots \alpha_k\}$
X - alfabet wejściowy
Y - alfabet wyjściowy
δ - funkcja przejść $\quad \delta : Q \times X^* \longrightarrow Q$
λ - funkcja wyjść $\quad \lambda : Q \times X^* \longrightarrow Y^*$

Dla uproszczenia opisu algorytmu przyjmujemy $X = Y = \{0,1\}$, chociaż może on być łatwo rozszerzony na dowolne skończone X i Y. W ogólności zadanie identyfikacji dowolnego automatu nie jest rozwiązalne, dlatego też konieczne jest wprowadzenie pewnych dodatkowych założeń odnośnie automatu B, niezbędnych do

tego, aby zadanie to miało jednoznaczne rozwiązanie (z dokładnością do przemianowania stanów).

Niech $x = x^1x^2x^3 \ldots\ldots x^m$ oznacza słowo wejściowe automatu, gdzie $x^i \in X = \{0,1\}$

1. Liczba stanów automatu B jest mniejsza od pewnej ustalonej liczby n, $k < n$

2. Automat B jest spójny i minimalny
   Oznacza to, że

$$\bigwedge_{\alpha_i \in Q} \bigwedge_{\alpha_j \in Q} \bigvee_{x} \delta(\alpha_i, x) = \alpha_j$$

oraz

$$\bigwedge_{\alpha_i \in Q} \bigwedge_{\alpha_j \in Q} \bigvee_{x} \lambda(\alpha_i, x) \neq \lambda(\alpha_j, x)$$

OZNACZENIA I DEFINICJE

Wprowadzimy teraz kilka pojęć i oznaczeń koniecznych do przejrzystego opisu algorytmu.

Wartością słowa x o długości m nazywamy

$$w_x^m = \sum_{i=1}^{m} x^i 2^{m-i}, \quad 0 \leq w_x^m \leq 2^m - 1$$

Dla danego m w_x^m w sposób jednoznaczny określa postać słowa x.

Niech $y(x) = y^1y^2y^3 \ldots\ldots y^m$, gdzie $y^i \in Y = \{0,1\}$, oznacza słowo wyjściowe będące odpowiedzią automatu na słowo x.

Jeżeli α_r jest stanem początkowym automatu B, to

$$y^i = \lambda(\delta(\alpha_r, x^1x^2 \ldots\ldots x^{i-1}), x^i)$$

Przez $l(x)$ oznaczać będziemy długość słowa x.

Określimy teraz na zbiorze $Q = \{\alpha_1, \alpha_2, \ldots\ldots \alpha_k\}$ stanów automatu B następującą relację równoważności R_s

$\alpha_i \; \alpha_j \; \alpha_i \; R_s \; \alpha_j \Longleftrightarrow \alpha_i$ oraz α_j są s-równoważne $s \geq 1$

Dwa stany α_i oraz α_j automatu nazywamy s-równoważnymi, jeżeli dla każdego ciągu x o długości 1

$$\lambda(\alpha_i, x) = \lambda(\alpha_j, x)$$

jeżeli $1 \leq s$

Stany, które nie są s-równoważne nazywamy s-rozróżnialne.

Relacja R_s dzieli zbiór Q na klasy abstrakcji. Oznaczmy ten podział przez P_s, zaś elementy tego podziału, klasy abstrakcji, przez P_{s_i}. Zbiory P_{s_i} są niepuste i rozłączne.

Dla $i \neq j \quad P_{s_i} \cap P_{s_j} \neq \emptyset$

Zgodnie z definicją relacji R_s każdy zbiór P_{s_i} zawiera wszystkie te stany automatu, które są s-równoważne między sobą. Przez m_s oznaczać będziemy liczbę zbiorów P_{s_i} w podziale P_s. Zgodnie z założeniem automat B posiada k stanów, więc dla dowolnego podziału P_s spełniona jest nierówność

$$m_s \leq k$$

Poniżej zostaną podane twierdzenia związane z własnościami automatu B, które będą miały istotne znaczenie w dalszych rozważaniach.

Twierdzenie 1

Jeżeli podziały $P_s \neq P_{s-1}$ to $m_s \geq s + 1$.

Twierdzenie 2

Jeżeli automat B posiada k stanów oraz $P_s \neq P_{(s-1)}$, to liczba stanów w każdej klasie P_{s_i} podziału P_s jest nie większa niż $(k-s)$.

Twierdzenie 3

Jeżeli automat B posiada k stanów, to $P_k = P_{k-1}$.

Dowody powyższych twierdzeń można znaleźć np. w [2].

Twierdzenie 4

Jeżeli automat B posiada k stanów, to dowolny jego podzbiór r stanowy, gdzie $2 \leqslant r \leqslant k$, zawiera co najmniej dwa stany, które są (k-r+1)-rozróżnialne.

Dowód:

Z twierdzenia 2 wynika, że jeżeli weźmiemy pod uwagę podział P_s dla automatu B o k stanach, to liczba stanów w każdej klasie P_{s_i} jest nie większa niż (k-s).

Wobec powyższego liczba stanów w każdej klasie podziału $P_{(k-r+1)}$ jest nie większa niż $k-(k-r+1) = r-1$.

Zatem jeżeli mamy zbiór r stanów automatu, to co najmniej dwa stany z tego zbioru powinny się znajdować w dwóch różnych klasach podziału $P_{(k-r+1)}$ i w związku z tym te dwa stany będą (k-r+1) - rozróżnialne. Stąd jako wniosek otrzymujemy następujące twierdzenie.

Twierdzenie 5

Jeżeli automat B posiada k stanów, to dla każdej pary α_i, α_j jego stanów istnieje słowo wejściowe x o długości $l \leqslant k-1$, które rozróżnia te stany.

Inaczej mówiąc dowolne dwa stany automatu B są (k-1) - rozróżnialne.

Twierdzenie 6

Jeżeli B jest automatem minimalnym i spójnym, to istnieje takie słowo x_*, że $\bigwedge_{\alpha \in Q} y(x_*)$ jednoznacznie określa jego stan końcowy $\delta(\alpha, x_*)$.

Dowód:

Słowa x_* będziemy poszukiwać w postaci złożenia słów $a_1, a_2, \ldots\ldots$

$$x_* = a_1 a_2 a_3 \ldots\ldots\ldots a_s$$

Weźmy pod uwagę słowo $a = a_1 a_2 \ldots\ldots a_r$ złożone z niepustych słów $a_1, a_2, \ldots\ldots a_r$.

Oznaczmy przez D_r podział zbioru Q wszystkich stanów na klasy abstrakcji względem następującej relacji równoważności R_a

$$\bigwedge_{\alpha_i \in Q} \bigwedge_{\alpha_j \in Q} \alpha_i R_a \alpha_j \Leftrightarrow \lambda(\alpha_i, a_1 a_2 \ldots\ldots a_r) = \lambda(\alpha_j, a_1 a_2 \ldots\ldots a_r)$$

Oznaczmy elementy (klasy abstr.) podziału D_r przez D_{r_i}, zaś przez p_r liczbę niepustych klas podziału D_r.

$$p_r \leqslant k$$

Niech r = 1. Z twierdzenia 5 wynika, że dla każdych dwóch stanów automatu B, α_i oraz α_j istnieje słowo a_1 o długości $l(a_1) \leqslant k-1$ takie, że $\lambda(\alpha_i, a_1) \neq \lambda(\alpha_j, a_1)$

$$p_1 \geqslant 2$$

Niech K_{r_i} będzie obrazem zbioru D_{r_i} przy przekształceniu $\delta(\alpha, a_1 a_2 \ldots\ldots a_r)$.

Jeżeli przy podziale D_r dla każdego $i \leqslant p_r$ zbiór K_{r_i} jest zbiorem jednoelementowym, to oczywiście $x_* = a_1 a_2 \ldots a_r$ oraz $l(x_*) = l(a_1) + l(a_2) + \ldots\ldots + l(a_r)$.

W przeciwnym wypadku istnieje K_{r_i} zawierający przynajmniej dwa różne stany α_g i α_h.

Z twierdzenia 5 wynika, że istnieje słowo a_{r+1} takie, że

$$\lambda(\alpha_g, a_{r+1}) \neq \lambda(\alpha_h, a_{r+1}) \quad \text{oraz} \quad l(a_{r+1}) \leqslant k-1$$

Wobec powyższego relacja $R_{aa_{r+1}}$ następująca

$$\bigwedge_{\alpha_i \in Q} \bigwedge_{\alpha_j \in Q} \alpha_i R_{aa_{r+1}} \alpha_j \Leftrightarrow \lambda(\alpha_i, a_1 a_2 \ldots a_r a_{r+1}) =$$

$$= \lambda(\alpha_j, a_1 a_2 \ldots a_r a_{r+1})$$

tworzy podział D_{r+1} zbioru Q na klasy abstrakcji taki, że

$$p_{r+1} > p_r$$

Z przeprowadzonej analizy wynika, że istnieje pewna liczba s taka, że $a = a_1 a_2 \ldots\ldots a_s$ i podział D_s względem relacji R_a zawiera tylko takie klasy D_{s_i}, których odpowiednie obrazy K_{s_i} są jednoelementowe.

To zaś oznacza, że $x_* = a_1 a_2 a_3 \ldots\ldots a_s$.

Zgodnie z przyjętymi oznaczeniami ciąg x_* posiada następujące właściwości:

a. $\lambda(\alpha_i, x_*) \neq \lambda(\alpha_j, x_*) \Rightarrow \alpha_i \neq \alpha_j$

b. $\bigwedge_{\alpha_g} \bigwedge_{\alpha_h} \delta(\alpha_g, x_*) \neq \delta(\alpha_h, x_*) \Rightarrow \lambda(\alpha_g, x_*) \neq \lambda(\alpha_h, x_*)$

Maksymalna długość słowa x_* otrzymanego dzięki procedurze opisanej w dowodzie twierdzenia 6 spełnia warunek

$$l(x_*) \leq (k-1)^2$$

Można udowodnić, że $l(x_*) \leq k(k-1)/2$.

Dowód w [1], ale znalezienie słowa x_* spełniającego ten warunek jest bardzo kłopotliwe. Dlatego też w algorytmie opisanym w niniejszym opracowaniu zrezygnowano z minimalizacji długości ciągu x_*, zaś dowód twierdzenia 6 stanowi podstawę opisanego dalej algorytmu identyfikacji automatu.

Oznaczmy przez M tablicę przejść automatu B określającą funkcję przejść

$$M[i, x] = j \Leftrightarrow \delta(\alpha_i, x) = \alpha_j$$
$$x \in X = \{0,1\}$$

Podobnie przez W oznaczymy tablicę wyjść automatu B

$$W[i, x] = y \Leftrightarrow \lambda(\alpha_i, x) = y$$

$$x \in X = \{0,1\}$$
$$y \in Y = \{0,1\}$$

M i W mają wymiar (k×2)

Zauważmy, że wyznaczenie tablic M i W jest równoznaczne z całkowitym określeniem struktury badanego automatu B.

Definicja funkcji δ_i^n automatu B

$$\delta_i^n : P \rightarrow Q$$

gdzie P jest zbiorem wszystkich słów wejściowych o długości l

$$l < n$$

$$\delta_i^n(x) = \alpha_j \Longleftrightarrow \delta(\alpha_i, x) = \alpha_j$$

Funkcja δ_i^n przyporządkowuje jednoznacznie każdemu słowu ze zbioru P jeden stan α_j automatu B.

Twierdzenie 7

Dowolna funkcja δ_i^n automatu B jednoznacznie określa jego tablicę przejść M.

Dowód:

Niech α_g będzie dowolnym stanem automatu B

Jeżeli $\alpha_g = \alpha_i$

to
$$M[i, 0] = \delta(\alpha_i, 0) = \delta_i^n(0)$$
$$M[i, 1] = \delta(\alpha_i, 1) = \delta_i^n(1)$$

Jeżeli $\alpha_g \neq \alpha_i$, to ze względu na spójność i minimalność automatu B istnieje słowo wejściowe x o długości $l(x) \leqslant n-2$ takie, że

$$\delta(\alpha_i, x) = \alpha_g$$

Weźmy teraz pod uwagę słowa x0 oraz x1

$$x0 \in P \quad i \quad x1 \in P$$

$$M\,[g, 0] = \delta(\alpha_g, 0) = \delta(\alpha_i, x0) = \delta_i^n\,(x0)$$

$$M\,[g, 1] = \delta(\alpha_g, 1) = \delta(\alpha_i, x1) = \delta_i^n\,(x1)$$

Zatem dla dowolnego g wyznaczyliśmy g-ty wiersz tablicy M.

Definicja funkcji μ_i^n automatu B

$$\mu_i^n : P \longrightarrow Y$$

gdzie P jest zbiorem wszystkich słów wejściowych o długości l

$$l < n$$

$$\mu_i^n\,(x) = y \Longleftrightarrow \mu\,(\alpha_i, x) = y$$

Funkcja μ_i^n przyporządkowuje jednoznacznie każdemu słowu ze zbioru P jeden symbol $y \in Y = \{0,1\}$.

Funkcja $\mu\,(\alpha, x) : Q \times X\,X^* \longrightarrow Y$ określa ostatni symbol słowa $\lambda(\alpha, x)$.

Twierdzenie 8

Dowolna funkcja μ_i^n automatu B jednoznacznie określa jego tablicę wyjść W.

Dowód:

Niech α_g będzie dowolnym stanem automatu B
Jeżeli $\alpha_g = \alpha_i$

to $$W\,[i, 0] = \lambda(\alpha_i, 0) = \mu_i^n\,(0)$$
$$W\,[i, 1] = \lambda(\alpha_i, 1) = \mu_i^n\,(1)$$

Jeżeli $\alpha_g \neq \alpha_i$, to ze względu na spójność i minimalność automatu B istnieje słowo x o długości $l(x) \leqslant n-2$ takie, że $\delta(\alpha_i, x) = \alpha_g$.

Weźmy teraz pod uwagę słowa x0 oraz x1

$$x0 \in P \quad i \quad x1 \in P$$

$$W\,[g, 0] = \lambda(\alpha_g, 0) = \mu(\alpha_i, x0) = \mu_i^n(x0)$$
$$W\,[g, 1] = \lambda(\alpha_g, 1) = \mu(\alpha_i, x1) = \mu_i^n(x1)$$

Wobec powyższego dla dowolnego g wyznaczyliśmy g-ty wiersz tablicy W.

Definicja tablicy A_i

Przez A_i oznaczać będziemy tablicę o $(n-2)$ kolumnach oraz 2^{n-2} wierszach, której j-ty wiersz zawiera słowo wyjściowe $y = y^1y^2y^3\ldots\ldots y^{n-2}$ o długości $(n-2)$ takie, że

$$y = \lambda(\alpha_i, x^1x^2x^3\ldots\ldots x^{n-2})$$

gdzie słowo $x = x^1x^2x^3\ldots\ldots x^{n-2}$ ma wartość $W_x^{n-2} = (j-1)$

Definicja tablicy B_i

Przez B_i oznaczać będziemy tablicę o $(2n-3)$ kolumnach oraz 2^{2n-3} wierszach, której j-ty wiersz zawiera słowo wyjściowe y $y = y^1y^2y^3\ldots\ldots y^{2n-3}$ będące odpowiedzią automatu B na ciąg $x = x^1x^2x^3\ldots\ldots x^{2n-3}$ o wartości $W_x^{2n-3} = (j-1)$, czyli $y = \lambda(\alpha_i, x^1x^2\ldots\ldots x^{2n-3})$

Oznaczmy przez $B_i[j]$ j-ty wiersz tablicy B_i i analogicznie przez $A_i[j]$ j-ty wiersz tablicy A_i.

Dowolną tablicę K o $2n-3$ kolumnach i 2^{2n-3} wierszach nazywać będziemy regularną tablicą automatu B wtedy i tylko wtedy, jeżeli

$$\bigvee_{\alpha_i \in Q} \quad \bigwedge_{1 \le j \le 2^{2n-3}} \quad K[j] = \lambda(\alpha_i, x)$$

gdzie x ma długość $(2n-3)$ i wartość $W_x^{2n-3} = (j-1)$, $K[j]$ oznacza j-ty wiersz tablicy K.

Zgodnie z podaną definicją $K = B_i$.

Twierdzenie 9

Jeżeli B jest minimalnym automatem o k stanach, gdzie $k < n$, to

$$\alpha_i \neq \alpha_j \Longleftrightarrow A_i \neq A_j; \quad \alpha_i, \alpha_j \in Q$$

Dowód

Z twierdzenia 5 wynika, że dla każdych dwóch stanów α_i i α_j automatu B istnieje słowo wejściowe x o długości $l = (n-2)$, które rozróżnia te stany.

Zatem $\alpha_i \neq \alpha_j \Longleftrightarrow \lambda(\alpha_i, x) \neq \lambda(\alpha_j, x)$
Słowo x ma długość (n-2) więc $\lambda(\alpha_i, x)$ jest $(W_x^{n-2} + 1)$ wierszem tablicy A_i, zaś $\lambda(\alpha_j, x)$ jest $(W_x^{n-2} + 1)$ wierszem tablicy A_j.

Wobec powyższego

$$\lambda(\alpha_i, x) \neq \lambda(\alpha_j, x) \Longleftrightarrow A_i \neq A_j$$

stąd

$$\alpha_i \neq \alpha_j \Longleftrightarrow A_i \neq A_j$$

Zauważmy, że jeżeli k jest liczbą stanów automatu, to istnieje k różnych tablic A_i.

Analogicznie można udowodnić twierdzenie następujące:

Twierdzenie 10

Jeżeli B jest automatem minimalnym o k stanach, gdzie $k < n$, to

$$\alpha_i \neq \alpha_j \Longleftrightarrow B_i \neq B_j$$

Z powyższych twierdzeń wynika, że każdemu stanowi α_i automatu B jest jednoznacznie przyporządkowana odpowiednia tablica A_i oraz B_i.

Niech $\circledast$ oznacza następujące dwuargumentowe działanie w zbiorze słów o długości (2n-3)

Jeżeli

$$z_1 = z_1^1 z_1^2 \ldots\ldots\ldots z_1^{2n-3}$$

$$z_2 = z_2^1 z_2^2 \ldots\ldots\ldots z_2^{2n-3}$$

to $z_3 = z_1 \oplus z_2 = z_3^1 z_3^2 \ldots\ldots\ldots z_3^{2n-3}$

gdzie $z_3^i = 0 \Longleftrightarrow z_1^i = z_2^i$

$z_3^i = 1 \Longleftrightarrow z_1^i \neq z_2^i$

Słowo $z_1 \oplus z_2$ posiada symbol "0" na tych pozycjach, na których słowa z_1 i z_2 mają identyczne symbole, zaś symbol "1" na tych pozycjach, na których słowa z_1 i z_2 mają symbole różne.

Oznaczmy przez $p(z)$ liczbę kolejnych symboli "0" w słowie z licząc od lewej strony do pierwszego symbolu "1", np. dla $z = 0000110100$ $p(z) = 4$

Wykorzystując wprowadzone wyżej oznaczenia można sformułować następujące twierdzenie

Twierdzenie 11

Jeżeli x_1 oraz x_2 są dowolnymi słowami wejściowymi takimi, że $l(x_1) = l(x_2) = (2n-3)$, to dla dowolnego stanu α_i automatu B spełniona jest nierówność

$$p(x_1 \oplus x_2) \leqslant p(\lambda(\alpha_i, x_1) \oplus \lambda(\alpha_j, x_2))$$

Twierdzenie 12

Dowolna regularna tablica B_i automatu B jednoznacznie określa:

a. wszystkie tablice A_j
b. funkcję δ_i^n
c. funkcję μ_i^n

Dowód

Dana jest regularna tablica B_i automatu B.

a. Niech α_g będzie dowolnym stanem automatu B.

Z założenia spójności B wynika, że istnieje słowo x_o o długości $l(x_o) \leqslant n-2$ przeprowadzające automat ze stanu α_i do α_g

$$\delta(\alpha_i, x_o) = \alpha_g$$
$$\lambda(\alpha_i, x_o) = y_i(x_o)$$

Wykażemy, że na podstawie B_i można wyznaczyć dowolny wiersz tablicy A_g.

Niech s będzie dowolną liczbą taką, że $1 \leqslant s \leqslant 2^{n-2}$.
Weźmy pod uwagę słowo $x = x_o x_1 x_2$
gdzie $l(x) = (2n-3)$, $l(x_1) = (n-2)$
oraz $w_{x_1}^{n-2} = (s-1)$
x_2 jest dowolnym słowem długości $l(x_2) = (n-1) - l(x_o)$

$$y_1(x) = \lambda(\alpha_i, x) = \lambda(\alpha_i, x_o x_1 x_2) = \lambda(\alpha_i, x_o)\lambda(\delta(\alpha_i, x_o), x_1 x_2) =$$

$$= \lambda(\alpha_i, x_o)\,\lambda(\alpha_g, x_1 x_2) = \lambda(\alpha_i, x_o)\,\lambda(\alpha_g, x_1)\,\lambda(\delta(\alpha_g, x_1), x_2)$$

Z definicji tablicy A wynika, że $\lambda(\alpha_g, x_1)$ jest s-tym wierszem tablicy A_g.

Postępując w ten sposób dla każdego g i s wyznaczymy wszystkie tablice A_i automatu B.

b. Weźmy pod uwagę dowolne słowo wejściowe x_o o długości $l(x_o) \leqslant n-1$

Słowo x_o przeprowadza automat B ze stanu α_i w pewien stan α_h, dla którego możemy zbudować tablicę A_h w sposób opisany w części a. dowodu.

Tablica A_h jednoznacznie określa stan α_h
zatem $$\delta_i^n(x_o) = \alpha_h$$

c. Weźmy pod uwagę dowolne słowo wejściowe x_0 o długości $l(x_0) \leqslant n-1$

Istnieje słowo wejściowe $x = x_0x_1$ o długości $l(x) = (2n-3)$ i wartości w_x^{2n-3}.

W $(w_x^{2n-3} + 1)$ wierszu tablicy B_i znajduje się słowo wyjściowe $\lambda(\alpha_i, x) = \lambda(\alpha_i, x_0)\,\lambda(\delta(\alpha_i, x_0), x_1)$

Niech $\mu(\alpha, x)$ oznacza ostatni symbol słowa $\lambda(\alpha, x)$, wtedy $\mu_i^n(x_0) = \mu(\alpha_i, x_0)$

Z twierdzeń 9, 10, 12 wynika, że dla jednoznacznego określenia struktury badanego automatu B wystarczy wyznaczyć jedną dowolną regularną tablicę B_i. Dowody powyższych twierdzeń zawierają również efektywną procedurę postępowania w tym wypadku, chociaż jest to procedura bardzo pracochłonna.

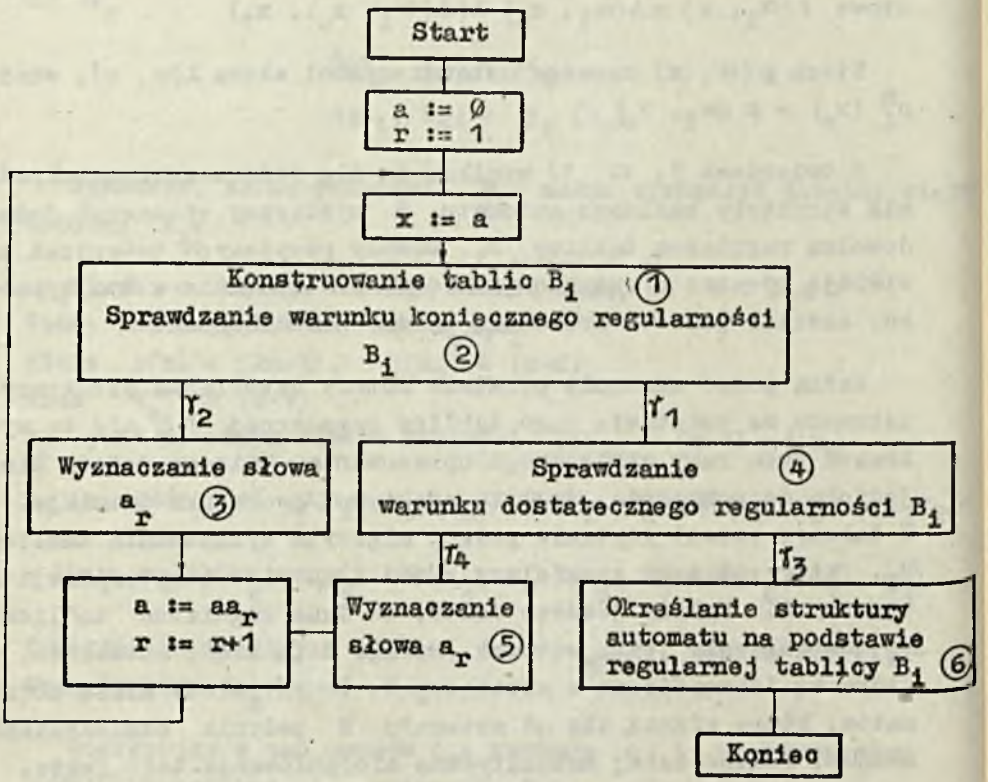
Można podać znacznie prostsze metody określania struktury automatu na podstawie jego tablicy regularnej B_i, ale to wykracza poza ramy niniejszego opracowania. Opisana metoda służy jedynie do pokazania, że taka efektywna procedura istnieje. W dalszej części zostanie podany algorytm wyznaczania tablicy B_i, który stanowi zasadniczą część algorytmu identyfikacji automatu B. Należy jeszcze dodać, że dana regularna tablica B_i określa nie jeden automat B, ale całą klasę automatów, które są izomorficzne z automatem B. Innymi słowy klasę automatów, które różnią się od automatu B jedynie oznaczeniami stanów. Pominę tutaj matematyczne sformułowanie tego faktu.

OPIS ALGORYTMU IDENTYFIKACJI AUTOMATU B

Przedstawiony tutaj algorytm składa się z trzech zasadniczych części:

A. Poszukiwanie ciągu x_* oraz konstruowanie tablic B_i.
B. Sprawdzanie czy utworzona tablica B_i jest regularna.
C. Określanie struktury badanego automatu na podstawie wyznaczonej tablicy regularnej B_i.

Ogólny schemat blokowy algorytmu pokazany został na rys. 1. Zgodnie z twierdzeniem 12 do określenia struktury automatu konieczna jest znajomość przynajmniej jednej regularnej tablicy B_i automatu B. Tablicę taką możemy jednak zbudować tylko wtedy, gdy posiadamy możliwość doprowadzenia automatu B do ustalonego stanu początkowego α_i. Taką możliwość daje nam



Rys. 1. Ogólny schemat blokowy algorytmu identyfikacji automatu B

γ_1 - Zakończony proces tworzenia jednej z tablic B_i, która spełnia warunek konieczny regularności,
γ_2 - Pewna tablica B_i nie spełnia warunku koniecznego regularności
γ_3 - Utworzona w części ① tablica B_i jest regularna
γ_4 - Utworzona w części ① tablica B_i nie jest regularna

zdefiniowany wyżej ciąg x_*, który jednak musimy dopiero wyznaczyć na podstawie pewnych eksperymentów przeprowadzonych z automatem B. Metoda poszukiwania takiego ciągu x_* podana została w dowodzie twierdzenia 6. Tutaj zostanie ona omówiona z praktycznego punktu widzenia w zastosowaniu do konstrukcji odpowiedniego eksperymentu z automatem B.

Stanowi ona adaptacyjną procedurę modyfikowania postaci ciągu a na podstawie wyników przeprowadzonych eksperymentów przy założeniu, że $a = x_*$.

Otóż na początku zakłada się, że $a = \emptyset$. Oznacza to, że po wprowadzeniu na wejście automatu B dowolnego słowa automat znajduje się w tym samym stanie α_i. Przy tak określonym słowie x_* konstruujemy tablicę B_i. Konstrukcja tablicy B_i opisana jest w części ① algorytmu.

Następnie dokonuje się sprawdzenia czy zbudowana tablica B_i jest regularna (Część ④ algorytmu).

Jeżeli wynik sprawdzania jest pozytywny oznacza to, że założenie $x_* = a$ jest słuszne, a utworzona tablica B_i jest tablicą regularną badanego automatu.

Jeżeli jednak wynik sprawdzania jest negatywny oznacza to, że $x_* \neq a$ i na podstawie przeprowadzonego eksperymentu ze słowem a wyznaczone jest słowo a_1.

Metoda określania ciągu a_1 (ogólnie a_r) opisana została w części ③ i ⑤ algorytmu identyfikacji.

Następnie bierzemy $a = a_1$ i przy założeniu $x_* = a$ przystępujemy do konstrukcji tablic B_i i sprawdzania ich regularności, jak to zostało opisane dla ciągu $a = \emptyset$.

Postępując dalej w ten sposób określamy pewien ciąg $a = a_1 a_2 \ldots a_r$. Zakładając, że $x_* = a$ przystępujemy do konstruowania tablic B_i, a następnie do sprawdzania ich regularności. Jeżeli utworzona tablica B_i jest regularna, to oznacza, że założenie $x_* = a$ jest słuszne i w związku z tym prze-

chodzimy do części ⑥ algorytmu, w której następuje określenie struktury badanego automatu B na podstawie utworzonej uprzednio tablicy B_i.

Jeżeli jednak utworzona tablica B_i nie jest regularna, to przechodzimy do części ③ lub ⑤ algorytmu w celu wyznaczenia ciągu a_{r+1}.

Następnie modyfikujemy ciąg a (a: = $a_1a_2a_3\ldots\ldots a_ra_{r+1}$) i zakładając $x_* = a$ powtarzamy opisaną wyżej procedurę.

Z twierdzenia 6 wynika, że proces poszukiwania ciągu x_* zakończy się dla $r \leqslant k-1$.

① Tworzenie tablic B_i automatu B.

Przystępując do konstruowania tablic B_i mamy ustalony ciąg a, o którym zakładamy, że jest ciągiem x_*.

Wobec tego y (a) określa nam jednoznacznie stan w jakim znajduje się automat B po wprowadzeniu na jego wejście ciągu a.

Niech p będzie liczbą różnych możliwych ciągów y (a). Oznaczmy je $Y_a[1]$, $Y_a[2]$, ... $Y_a[p]$.

Każdy z nich określa pewien stan końcowy automatu B.

Stany te oznaczmy odpowiednio:

$$\alpha'_1,\ \alpha'_2,\ \ldots\ldots\ \alpha'_p;\quad \alpha'_i \in Q$$

(stany α'_i nie muszą być różne).

Niech B_1^a, B_2^a, ... B_p^a oznaczają odpowiednie tablice B_i, które będą konstruowane dla tych stanów, przy założeniu $x_* = a$. Oznaczmy jeszcze przez k_i pewną zmienną, której wartość będzie liczbą wierszy tablicy B_i^a, jakie zostały już wyznaczone w trakcie eksperymentu konstruowania tablicy B_i^a.

Na początku eksperymentu $k_1 = k_2 \ldots\ldots = k_p = 0$. Eksperyment składa się z ciągu słów wejściowych postaci następującej:

$$ax_1ax_2ax_3 \ldots\ldots ax_t$$

gdzie $l(x_t) = (2n-3)$
Wartość ciągu x_t zależy od odpowiedzi automatu B na poprzedzający ciąg a.

Otóż jeżeli $y(a) = Y_a[j]$, to oznacza, że automat B znajduje się w stanie α_j' i podając na jego wejście odpowiedni ciąg x_t możemy wyznaczyć kolejny wiersz tablicy B_j^a. Numer tego wiersza określa aktualna wartość wyrażenia $(k_j + 1)$. Wobec tego zgodnie z definicją tablicy B_j wartość ciągu x_t

$$w_{x_t}^{2n-3} = k_j$$

Proces konstruowania tablic B_i^a zostaje zakończony z chwilą, gdy dla pewnego j zostały wyznaczone wszystkie wiersze tablicy B_j^a (przechodzimy wtedy do części ④) lub też gdy stwierdzimy, że któraś z konstruowanych tablic B_i^a nie spełnia warunku koniecznego regularności (przechodzimy wtedy do części ③).

② Sprawdzanie czy tworzona tablica B_j^a spełnia warunek konieczny regularności.

Podstawą przy określaniu warunku koniecznego regularności jest twierdzenie 11.

Weźmy pod uwagę dwa słowa wejściowe x i x' takie, że

$$l(x) = l(x') = (2n-3)$$

$$w_x^{2n-3} = (k_j - 1) \qquad w_{x'}^{2n-3} = w_x^{2n-3} - 1 = (k_j - 2)$$

Z twierdzenie 11 wynika, że dla słów x i x' spełniona jest nierówność

$$p(x \oplus x') \leqslant p(\lambda(\alpha_j, x) \oplus \lambda(\alpha_j, x'))$$

$\lambda(\alpha_j, x)$ oraz $\lambda(\alpha_j, x')$ są odpowiednio $(k_j - 1)$ oraz (k_j) wierszem tablicy B_j^a.

Zatem powyższa nierówność przyjmuje postać

$$p(x \oplus x') \leqslant p(B_j^a[k_j] \oplus B_j^a[k_j - 1])$$

Nierówność ta stanowi warunek konieczny regularności tablicy B_j^a. Oznacza on, że jeżeli dwa kolejne słowa wejściowe x i x' nie różnią się na s pierwszych pozycjach, to odpowiednie słowa wyjściowe $\lambda(\alpha_j, x)$ oraz $\lambda(\alpha_j, x')$ nie różnią się na co najmniej s pierwszych pozycjach.

③ Wyznaczanie słowa a_r

Jeżeli przy konstruowaniu tablic B_i^a dla pewnego słowa x_o o wartości $w_{x_o}^{2n-3} = (k_j - 1)$ nie jest spełniony warunek konieczny regularności tablicy B_j^a, oznacza to, że założenie $x_* = a$ było błędne.

Wobec powyższego istnieją dwa stany automatu B, α_g i α_h takie, że

$$\delta(\alpha_g, a) = \alpha_j$$
$$\delta(\alpha_h, a) = \alpha_w$$
$$\lambda(\alpha_g, a) = \lambda(\alpha_h, a)$$
$$\alpha_j \neq \alpha_w$$
$$\lambda(\alpha_j, x_o) \neq \lambda(\alpha_w, x_o)$$

Słowo x_o jest więc ciągiem rozróżniającym stany α_j i α_w. Wobec powyższego zgodnie z definicją ciągu a_r w dowodzie twierdzenia 6 możemy przyjąć

$$a_r = x_o$$

Jeżeli jednak

$$\lambda(\alpha_j, x_o) \neq \lambda(\alpha_w, x_o) \quad \text{oraz} \quad d = p(\lambda(\alpha_j, x_o) \oplus (\alpha_w, x_o))$$

to

$$\lambda(\alpha_j, b) \neq \lambda(\alpha_w, b)$$

gdzie b jest ciągiem pierwszych d+1 symboli ciągu x_0. Zatem możemy przyjąć $a_r = b$.

④ Sprawdzanie warunku dostatecznego regularności utworzonej tablicy B_i^a

Warunek dostateczny regularności tablicy B_i^a ma postać

$$\bigwedge_{\alpha_g \in Q} [\lambda(\alpha_g, a) = Y_a[i] \Rightarrow \delta(\alpha_g, a) = \alpha'_i]$$

Spełnienie tego warunku oznacza, że podczas konstruowania tablicy B_i^a stanem początkowym był zawsze ten sam stan α_i, co jest równoważne definicji regularnej tablicy B_i^a.

Przypuśćmy, że B_i^a nie jest regularna. Wtedy istnieje taki stan α_h, że $\lambda(\alpha_h, a) = Y_a[i]$

zaś $\delta(\alpha_h, a) \neq \alpha'_i$

Niech np. $\delta(\alpha_h, a) = \alpha_s$

Z twierdzenia 5 wynika, że istnieje słowo wejściowe b o długości (n-2), dla którego

$$\lambda(\alpha'_i, b) \neq \lambda(\alpha_s, b)$$

Niech x_1 oznacza dowolne słowo wejściowe o długości (n-1). Podczas konstruowania tablicy B_i^a na automat podawane były wszystkie możliwe słowa postaci $x = bx_1$. Gdyby wtedy stanem początkowym był zarówno stan α'_i jak też α_s, to B_i^a nie spełniałaby warunku koniecznego regularności. Stąd wniosek, że stanem początkowym mógł być tylko jeden z tych stanów, np. stan α'_i. Chcąc ustawić automat w stan początkowy α_s - aby odróżnić ten stan od stanu α'_i podając na wejście słowo b - należy podać na automat wszystkie możliwe słowa postaci $x = bx_2$, gdzie $l(x_2) < n-1$. Istnieje bowiem co najmniej jedno słowo x_2 takie, że $l(x_2) < n-1$ oraz $\delta(\delta(\alpha'_i, bx_2), a) = \alpha_s$.

Niech A_i^a oznacza tablicę typu A_i dla stanu α'_i jaką określa utworzona tablica B_i^a.

Jeżeli $w_b^{n-2} = j$ to $A_i^a[j+1] = \lambda(\alpha'_i, b)$

Wobec powyższego, aby sprawdzić czy spełniony jest warunek dostateczny regularności tablicy B_i^a należy dla każdego słowa b o długości $l(b) = n-2$ przeprowadzić z automatem B eksperyment, polegający na podawaniu na jego wejście kolejno wszystkich możliwych słów postaci bx_2, gdy odpowiedź automatu na poprzedzające słowo a jest $y(a) = Y_a[i]$ oraz sprawdzeniu czy $y(b) = A_i^a[j+1]$. Jeżeli dla każdego słowa b oraz każdego x_2 określonych wyżej mamy $y(b) = A_i^a[j+1]$, to oznacza, że nasze założenie o istnieniu stanu α_s było błędne, a więc utworzona tablica B_i^a jest regularna. Wobec powyższego B_i^a określa wtedy jednoznacznie strukturę badanego automatu B, oczywiście z dokładnością do przemianowania stanów.

Jeżeli jednak dla pewnego słowa b oraz x_2

$y(ab\,x_2) = Y_a[i]\; y(b)\; y(x_2)$ oraz $y(b) = A_i^a[j+1]$

to oznacza, że warunek dostateczny regularności B_i nie jest spełniony i należy przejść do części ⑤ algorytmu.

Należy jeszcze dodać, że opisany warunek dostateczny regularności jest pomijany w czasie eksperymentu, jeżeli ciąg $a = a_1 a_2 \ldots\ldots a_r$ był $r = (n-1)$ razy modyfikowany podczas konstruowania tablic B_i, ponieważ wtedy zgodnie z dowodem Tw 6 ciąg a jest ciągiem doprowadzającym x_*.

⑤ Określenie ciągu a_r, gdy utworzona tablica B_i^a nie spełnia warunku dostatecznego regularności.

Zgodnie z wyżej opisaną procedurą niespełnienie warunku dostatecznego regularności tablicy B_i^a oznacza, że istnieją takie dwa stany α_g i α_h, że dla danych słów a oraz b spełnione są następujące zależności:

$$\delta(\alpha_g, a) = \alpha'_i$$
$$\delta(\alpha_h, a) = \alpha_s$$
$$\lambda(\alpha_g, a) = \lambda(\alpha_h, a)$$

$$\alpha'_i \neq \alpha_s$$

$$\lambda(\alpha'_i, b) \neq \lambda(\alpha_s, b) \qquad \text{gdzie} \quad l(b) = n-2$$

Zatem w kolejnej modyfikacji ciągu a należy przyjąć $a_r = b$, co wynika z definicji ciągu a_r w dowodzie Tw 6.

Opisany algorytm identyfikacji dowolnego spójnego i minimalnego automatu typu Mealy o ograniczonej liczbie stanów nie jest optymalny i wiele jego ozęści można znacznie uprościć jednak kosztem większej komplikacji tak samego algorytmu jak też opisu. Właśnie ze względu na większą przejrzystość opisu zrezygnowano tutaj ze szczegółowego omówienia niektórych części algorytmu, podając tylko zasadniczą koncepcję eksperymentu identyfikacji automatu.

Literatura

[1] AJZERMAN M.A., GUSEV L.A., ROZONOER L.I.: Logika Avtomaty Algoritmy, F.M. Moskwa 1963

[2] GILL A.: Introduction to the Theory of Finite-State Machines, Mc Grow-Hill, New York 1962

[3] KIME C.R.: An Organization for Checking Experiments on Sequential Circuits, IEEE Trans. on El. Comp., February 1966, 113-115

[4] TRACHTENBROT B.A.: Konečnyje avtomaty - povedenie i sintez, Moskwa 1970

НЕКОТОРЫЙ АЛГОРИТМ ИДЕНТИФИКАЦИИ ПРОИЗВОЛЬНОГО СИЛЬНО СВЯЗАННОГО И МИНИМАЛЬНОГО АВТОМАТА MEALY С ОГРАНИЧЕННОМ ЧИСЛОМ СОСТОЯНИЙ

Резюме

Статья занимается проблеммой определения логической структуры произвольного сильно связанного и минимального автомата типа Mealy с числом состояний меньше чем n (n - установленное число) и с известным входным и выходным алфавитом.

После введения понятия регулярной таблицы автомата, разобрано теоретические основы и представлено общее описание алгоритма идентификации автомата.

AN ALGORITHM IDENTIFICATION OF THE STRUCTURE OF THE STRONGLY CONNECTED AND REDUCED MACHINE WITH LIMITED NUMBER OF STATES

Summary

This paper is devoted to the problem of deriving logic structure of the strongly connected and reduced Mealy machine, where number of states is less than n (n is fixed). The alphabets input and output are known.

Beginning with an explanation of the regular matrix of machine, basis of the theory are discussed and general description of identification algorithm of machine is presented.

Warszawa, ul. Krzywickiego 34
A.M.

ALGORYTMY, XI, No 19, 1974, 71-87 681.3.001:681.3.04:519.14

O PEWNEJ METODZIE KODOWANIA STANÓW AUTOMATÓW SKOŃCZONYCH

Tomasz ADAMSKI

Studium Doktoranckie
Politechniki Warszawskiej

Pracę złożono 14.02.1973

W pracy opisano warunki konieczne i dostateczne istnienia pewnego sposobu kodowania stanów automatu skończonego. Podano algorytm metody w postaci schematu blokowego. Opisane wyniki mogą znaleźć zastosowanie w kodowaniu stanów automatów asynchronicznych.

1. SFORMUŁOWANIE PROBLEMU

Przyjmujemy jako obowiązującą w pracy następującą definicję automatu.

Definicja 1

Automatem nazywamy uporządkowaną piątkę $\langle Q, X, Y, \delta, \lambda \rangle$, gdzie Q, X, Y są niepustymi zbiorami zaś δ i λ odwzorowaniami: $\delta : Q \times X \to Q$; $\lambda : Q \times Y \to Y$. Zbiory Q, X, Y nazywamy odpowiednio: zbiorem (lub alfabetem) stanów, alfabetem wejściowym i alfabetem wyjściowym. Funkcje δ i λ nazywamy odpowiednio funkcją przejść i funkcją wyjść. Elementy zbioru Q nazywamy stanami wewnętrznymi lub krótko stanami. Elementy zbiorów X, Y nazywamy odpowiednio stanami wejść i stanami wyjść.

Automatem skończonym nazywamy taki automat $A = \langle Q, X, Y, \delta, \lambda \rangle$, dla którego zbiory Q, X, Y są skończone. W dalszym ciągu będą rozważane tylko automaty skończone, więc słowo automat będzie

zawsze oznaczało automat skończony. Przez N został oznaczony zbiór liczb naturalnych.

Definicja 2

Dla danego automatu $A = \langle Q, X, Y, \delta, \lambda \rangle$ każdą różnowartościową funkcję $\Phi : Q \longrightarrow \{0,1\}^k$, gdzie $k \in N$, nazywamy kodowaniem stanów automatu A, a ciąg $\Phi(q) \in \{0,1\}^k$, gdzie $q \in Q$ nazywamy kodowaniem stanów automatu A, a ciąg $\Phi(q) \in \{0,1\}^k$, gdzie $q \in Q$ nazywamy ciągiem kodującym lub kodowaniem stanu q.

Niech będzie dany automat $A = \langle Q, X, Y, \delta, \lambda \rangle$, gdzie $Q = \{q_1, \ldots, q_{n+1}\}$ jest zbiorem stanów. Chodzi o znalezienie dla funkcji $\delta : Q \times X \longrightarrow Q$ takiego algorytmu kodowania stanów q_i automatu A, by zerojedynkowe ciągi kodujące każdych dwu stanów, pomiędzy którymi istnieje przejście pod wpływem pojedynczej litery z alfabetu wejściowego X, różniły się tylko na jednej pozycji. W dalszym ciągu powyższa własność kodowania stanów będzie nazywana krótko własnością (π).

2. WPROWADZENIE I WARUNKI ISTNIENIA KODOWANIA (π)

Każdemu automatowi n-stanowemu A odpowiada pewien graf skierowany Γ (diagram stanów automatu), w którym wierzchołkom zostały przyporządkowane w sposób jednoznaczny stany, a skierowane krawędzie odpowiadają przejściu od stanu do stanu pod wpływem określonej litery z alfabetu wejściowego. W dalszych rozważaniach nie będzie ważny kierunek przejścia od stanu do stanu ani też krawędzie grafu Γ odpowiadające pozostaniu automatu A w tym samym stanie. Graf Γ przyporządkowany automatowi A będzie więc traktowany jako graf nieskierowany bez krawędzi odpowiadających pozostaniu w tym samym stanie. Zamiennie będą używane dalej określenia wierzchołek grafu i stan automatu.

Definicja 3

Drogą w grafie Γ łączącą dwa wierzchołki tego grafu X_0 i X_k nazywamy każdy ciąg $(X_0, k_{01}, X_1, k_{12}, X_2, \ldots, X_{k-1}, k_{k-1,k}, X_k)$,

gdzie X_i dla $i = 0,1,\ldots, k$ jest wierzchołkiem a k_{ij} krawędzią łączącą wierzchołek X_i z X_j. Drogę w grafie Γ nazywamy drogą prostą jeżeli $X_i \neq X_j$ dla każdego $i \neq j$. Długością drogi $(X_0, k_{01}, X_1, k_{12}, X_2, \ldots, X_{k-1}, k_{k-1,k}, X_k)$ nazywamy liczbę wchodzących w jej skład krawędzi. Drogą zamkniętą nazywamy drogę, dla której $X_0 = X_k$. Cyklem nazywamy drogę zamkniętą, dla której $X_i \neq X_j$ dla $i \neq j$, $i,j = 1,2,\ldots k$.

Rozważany poniżej automat A ma ustaloną liczbę n+1 stanów, gdzie $n \in N$. Zakładamy, że graf Γ odpowiadający automatowi A jest grafem spójnym, tzn. takim grafem, w którym dla każdej pary wierzchołków istnieje droga łącząca te wierzchołki. Nie zmniejsza to ogólności rozważań, ponieważ w przypadku niespójności Γ można rozbić go na pewien zbiór n' rozłącznych podgrafów spójnych Γ_k, gdzie $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \ldots \cup \Gamma_{n'}$ oraz zastosować opisaną niżej metodę kodowania (*) oddzielnie dla każdego podgrafu.

Lemat 1

Jeżeli $\bar{Q} = n+1$ oraz kodowanie stanów Φ automatu A ma własność (*), to wszystkie ciągi kodujące $\Phi(q_i)$ (gdzie $q_i \in Q$ dla $i = 1,2,\ldots, n+1$) różnią się najwyżej na n ustalonych pozycjach.

Szkic dowodu: Niech automatowi A odpowiada niezorientowany graf spójny Γ. Istnieje w grafie Γ najdłuższa droga prosta. Oznaczmy ją przez d_1. Droga d_1 nie może być dłuższa od n. Ma zatem długość n_1; $n_1 \leqslant n$. Oznaczmy zbiór wszystkich wierzchołków grafu Γ przez Z, a zbiór wierzchołków pokrywanych przez d_1 jako Z_1. Jak wynika z założenia liczba pozycji w ciągu kodującym, które ulegają zmianie przy przejściu drogi d_1 równa jest najwyżej n_1, ponieważ w każdym kroku, tzn. przy każdym przejściu krawędzi, może ulec zmianie inna pozycja. Jeżeli $Z_1 = Z$, to teza twierdzenia jest udowodniona. Jeżeli $Z_1 \subsetneq Z$, to istnieje najdłuższa droga prosta d_2 taka, że wszystkie wierzchołki będące elementami d_2 oprócz jednego, końcowego wierzchołka należą do zbioru $Z-Z_1$. Oznaczmy zbiór wierzchołków pokrywanych przez d_2 przez Z_2. Droga d_2 ma długość $n_2 = \bar{Z}_2 \geq 1$

zatem na najwyżej n_2 pozycjach ciągów kodujących mogą nastąpić zmiany przy przejściu tej drogi. Zbiory Z_1 i Z_2 mają jeden element wspólny, a więc:

$$n+1 \geqslant \overline{\overline{Z_1 \cup Z_2}} = \overline{\overline{Z}}_1 + \overline{\overline{Z}}_2 - 1 = (n_1 + 1) + (n_2 + 1) - 1 =$$
$$= n_1 + n_2 + 1$$

Skąd $n \geqslant n_1 + n_2$, ale $n_1 + n_2$ to maksymalna liczba zmian pozycji w ciągach kodujących wierzchołki pokrywane przez drogi d_1 i d_2 (czyli wierzchołki $Z_1 \cup Z_2$). Jeżeli więc $Z_1 \cup Z_2 = Z$, to w ciągach kodujących stany automatu A nie zmieni się więcej niż $n_1 + n_2$ pozycji (gdzie $n_1 + n_2 \leqslant n$), a więc teza twierdzenia jest spełniona.

Jeżeli $Z_1 \cup Z_2 \subsetneq Z$, to podobnie jak to czyniliśmy dla drogi d_2, poszukujemy najdłuższej drogi prostej d_3, której wierzchołek końcowy należy do $Z_1 \cup Z_2$, natomiast pozostałe wierzchołki należą do $Z - (Z_1 \cup Z_2)$. Rozumując jak wyżej dochodzimy do wniosku, że w ciągach kodujących stany (wierzchołki) należące do $Z_1 \cup Z_2 \cup Z_3$ (Z_3 jest zbiorem wierzchołków pokrywających drogę prostą d_3) może ulec zmianie najwyżej $n_1 + n_2 + n_3 \leqslant n$ pozycji, gdzie n_i; $i = 1,2,3$ jest długością i-tej drogi prostej d_i. W przypadku gdy $Z_1 \cup Z_2 \cup Z_3 = Z$ teza twierdzenia jest spełniona. Jeśli $Z_1 \cup Z_2 \cup Z_3 \subsetneq Z$, to rozumując analogicznie jak wyżej znajdujemy najdłuższą drogę prostą d_4, której wierzchołek końcowy należy do $Z_1 \cup Z_2 \cup Z_3$, natomiast pozostałe wierzchołki należą do $Z - (Z_1 \cup Z_2 \cup Z_3)$. Kontynuując rozumowanie stwierdzamy, że $n_1 + n_2 + n_3 + n_4 \leqslant n$.

Ponieważ graf Γ zawiera skończoną liczbę wierzchołków dla pewnego $r \in N$ dostaniemy: $Z = Z_1 \cup Z_2 \cup \ldots \cup Z_r$ oraz $n_1 + n_2 + \ldots + n_r \leqslant n$, gdzie Z_i jest zbiorem wierzchołków pokrywanych przez drogę d_i, n_i długością drogi d_i, $i = 1,2,\ldots,r$ a $n_1 + n_2 + \ldots + n_r$ maksymalną liczbą mogących ulec zmianie pozycji ciągów kodujących. Teza twierdzenia jest zatem spełniona.

Z lematu 1 wynika, że niecelowe jest używanie do kodowania według reguły (ℵ) ciągów zerojedynkowych o długości n', gdzie

$n' > n$ ponieważ i tak $n' - n$ ustalonych pozycji pozostałoby niezmienionych dla każdego ciągu kodującego. Jednak długość poszukiwanych ciągów kodujących nie może być mniejsza od n, ponieważ dla pewnych automatów (zakładamy $\bar{\bar{Q}} = n+1$) dopiero ciągi o długości n umożliwiają kodowanie o własności (x). Dalej z uwagi na powyższy fakt zakłada się, że poszukiwanymi ciągami kodującymi są ciągi o długości n.

Lemat 2

Jeżeli stany automatu A są zakodowane wg reguły (x) i $q_i \in Q$ jest dowolnym ustalonym stanem to:

1. istnieje takie kodowanie o własności (x), że q_i odpowiada ciąg $\underbrace{(0, 0, \ldots, 0)}_{n}$
2. jeżeli przestawimy pozycję k i i w ciągach kodujących wszystkich stanów, to uzyskane w ten sposób nowe kodowanie stanów posiada własność (x).

Dowód: Niech pierwotnie stanowi q_i odpowiada ciąg $\{a_i\}$. Weźmy pod uwagę pozycję k w tym ciągu, dla której $a_k = 1$. Jeżeli takiej pozycji nie ma to znaczy, że istniejące kodowanie czyni zadość punktowi 1 tezy. Zmieniając we wszystkich ciągach kodujących na k-tej pozycji 0 na 1 i 1 na 0 zachowujemy własność (x), a w ciągu $\{a_i\}$ pojawi się 0 na pozycji k. W ten sam sposób możemy postąpić dla następnej pozycji k, dla której $a_k = 1$ aż do wyczerpania wszystkich jedynek w ciągu $\{a_i\}$. Przypisujemy zatem stanowi q_i ciąg $(0, \ldots, 0)$ nie zmieniając własności (x). Druga część tezy jest oczywista.

Twierdzenie 1

Jeżeli kodowanie stanów automatu A posiada własność (x) i automatowi A odpowiada graf Γ (zgodnie z założeniem uczynionym na początku rozdziału jest to graf nieskierowany bez krawędzi odpowiadających pozostaniu w tym samym stanie), to w grafie Γ nie istnieje cykl (lub ogólniej droga zamknięta) o nieparzystej długości.

Dowód: Przyporządkujmy każdemu ciągowi $\{a_i^j\}_{i=1}^n$ kodującemu stan q_j liczbą naturalną c_j tak, że:

$$c_j = \sum_{i=1}^{n} a_i^j \cdot 2^i \qquad (1)$$

Jeżeli A ma kodowanie o własności (*), to przejściu ze stanu q_i do q_j pod wpływem pojedynczej litery odpowiada przyrost $c_j - c_i = \pm 2^k$, gdzie $k \in N$. Ponieważ w cyklu suma tych przyrostów jest równa zeru, długość cyklu musi być liczbą parzystą. Zatem nie istnieje w Γ cykl ani droga zamknięta o nieparzystej długości.

W powyższym twierdzeniu zbędne jest założenie spójności grafu Γ (postulowane na początku rozdziału i obowiązujące w całej pracy). Jeżeli jednak graf Γ jest spójny, to twierdzenie 1 można sformułować w sposób równoważny używając pojęcia relacji równowartości $\sim_{q_i}$ określonej w zbiorze Q wzorem:

$$q' \sim_{q_i} q'' \overset{df}{\Longleftrightarrow} \varrho(q', q_i) = \varrho(q'', q_i) \qquad (2)$$

gdzie: q', q'', $q_i \in Q$ oraz $\varrho : Q \times Q \to N$ jest funkcją przyporządkowującą każdej parze stanów liczbę naturalną będącą najmniejszą długością drogi łączącej te stany w grafie nieskierowanym Γ. Dla każdego stanu q_i istnieje odpowiednia relacja $\sim_{q_i}$.

Twierdzenie 1'

Jeżeli spełnione są założenia twierdzenia 1, graf Γ jest grafem spójnym oraz $\bigwedge_{d \in X} \bigwedge_{q \in Q} \sim (\delta(q, d) = q)$ to

$$\bigwedge_{q \in Q} \bigwedge_{K \in Q/\sim_q} \bigwedge_{q', q'' \in K} \bigwedge_{d \in X} \sim (\delta(q', d) = q'') \qquad (3)$$

Warunek (3) oznacza, że dla dowolnego $q \in Q$ w żadnej z klas abstrakcji relacji równowartości $\sim_q$ nie istnieją przejścia pod wpływem pojedynczej litery z alfabetu wejściowego. Łatwo

można wykazać, że przy spełnieniu założenia twierdzenia 1 warunek (3) równoważny jest warunkowi (3'), w którym pierwszy kwantyfikator ogólny z (3) zastąpiony jest szczegółowym. Fakt ten pozwala na uproszczenie procedury sprawdzania warunku koniecznego istnienia kodowania (x).

$$\bigvee_{q\in Q}\ \bigwedge_{K\subseteq Q/\sim_q}\ \bigwedge_{q',q''\in K}\ \bigwedge_{d\in X} \sim(\delta(q', d) = q'') \qquad (3')$$

W dalszych rozważaniach zakładamy, że graf Γ odpowiadający automatowi A jest nietrywialny, tzn. liczba wierzchołków $\overline{\overline{Q}} = n+1 > 1$. Trójka uporządkowana $K = \langle Z_2, \oplus, \odot \rangle$, gdzie $Z_2 = \{0,1\}$; $0,1 \in N$, a "$\oplus$" i "$\odot$" są działaniami zadanymi tabelkami:

x \ y	0	1
0	0	1
1	1	0

$x \oplus y$

x \ y	0	1
0	0	0
1	0	1

$x \odot y$

jest ciałem. Działanie "$\oplus$" jest dodawaniem modulo 2. Działanie "$\odot$" jest mnożeniem modulo 2 (działanie to można również uważać za działanie mnożenia z dwuelementowej algebry Boole'a). W przypadku, gdy $n = 1$ mamy $\overline{\overline{Q}} = 2$, $Q = \{q_1, q_2\}$ i kodowanie stanów automatu A mające własność (x) zawsze istnieje (takim kodowaniem jest np. funkcja Φ zadana na Q w sposób następujący: $q_1 \longmapsto \Phi(q_1) = (0)$; $q_2 \longmapsto \Phi(q_2) = (1)$.

Rozważmy teraz przypadek gdy $n > 1$. Produkt n-krotny L ciała $K = \langle Z_2, \oplus, \odot \rangle$, gdzie $n = \overline{\overline{Q}} - 1 > 1$;

$$L = \underbrace{K \times K \times \ldots \times K}_{n} \qquad (4)$$

nie jest już ciałem ponieważ posiada dzielniki zera. Zgodnie z definicją produktu algebr, algebra L jest trójką uporządkowaną $\langle B, +, . \rangle$, gdzie zbiór $B = \underbrace{\{0,1\} \times \{0,1\} \times \ldots \times \{0,1\}}_{n}$, a działania dodawania "+" i mnożenia "." elementów B zdefiniowane są następująco:

$$a+b=(a_1, a_2,\ldots, a_n) + (b_1, b_2,\ldots, b_n) \overset{df}{=} (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n) \quad \text{dla każdego } a, b, \in B \tag{5}$$

$$a \cdot b = (a_1, a_2,\ldots, a_n) \cdot (b_1, b_2,\ldots, b_n) \overset{df}{=} (a_1 \odot b_1, a_2 \odot b_2, \ldots, a_n \odot b_n) \quad \text{dla każdego } a, b \in B \tag{6}$$

Algebra $L = \langle B,+,\cdot\rangle$ jest pierścieniem przemiennym Boole'a z jednością. Skorzystamy z tego faktu w dalszych rozważaniach.

Zerem pierścienia L jest $\underline{0} = \underbrace{(0, 0,\ldots, 0)}_{n}$, jedynką $\underline{1} = \underbrace{(1, 1,\ldots, 1)}_{n}$. Niech $x_i \in B$, $i = 1,\ldots, n+1$ będzie ciągiem kodującym stan q_i oraz I i E zbiorami zdefiniowanymi następująco:

$$I = \left\{(i,j);\ \bigvee_{d \in X} ((\delta(q_i, d) = q_j \vee \delta(q_j, d) = q_i) \wedge i < j)\right\}$$

$$E = \left\{(a_1, a_2,\ldots, a_n) \in B \ \bigvee_{i}^{\cdot}\ a_i = 1\right\} \tag{7}$$

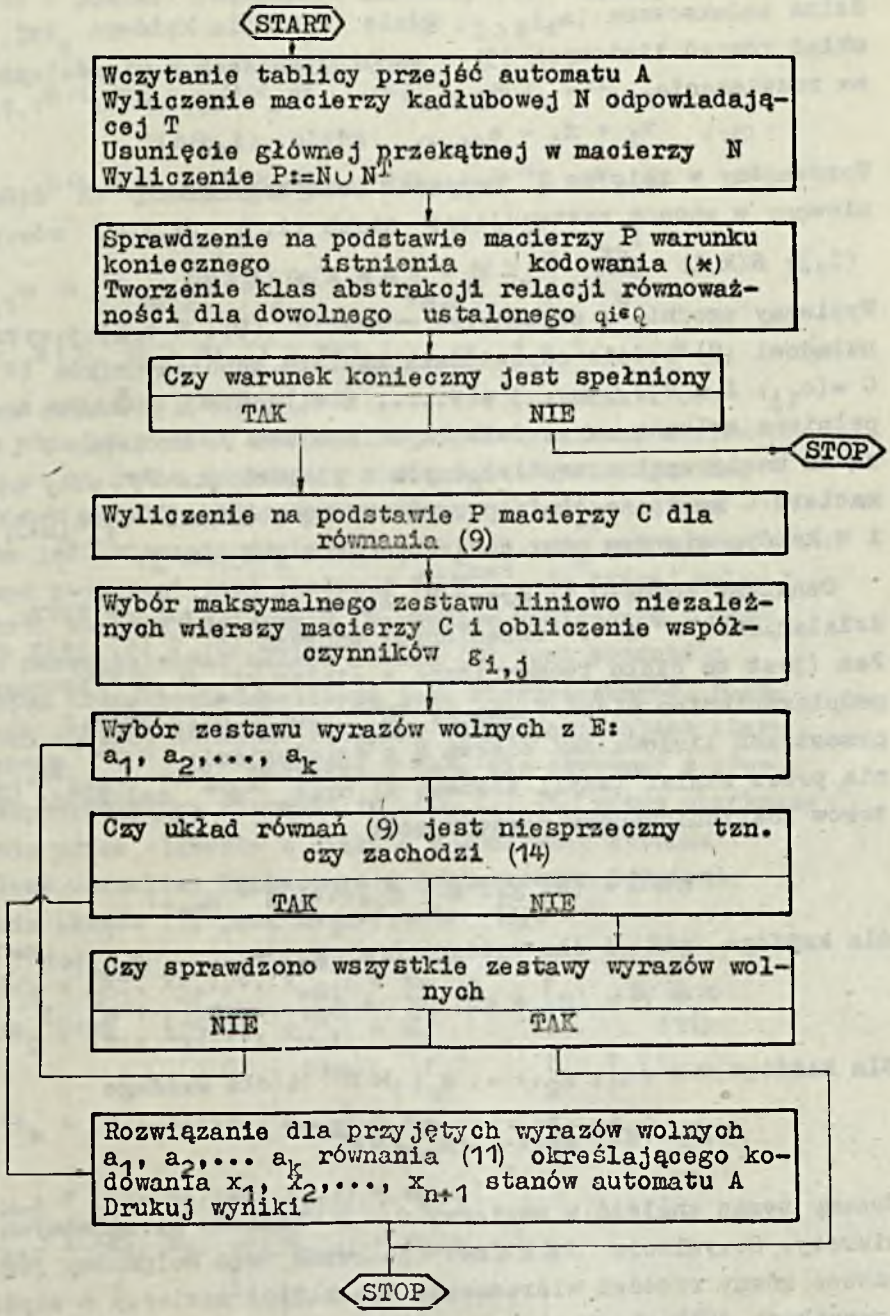
Następujący oczywisty lemat zawiera równoważne a bardziej przydatne do dalszych rozważań sformułowanie wyjściowej własności (ж).

Lemat 3

Kodowanie $\Phi : Q \ni q_i \to x_i \in B$ stanów automatu $A = \langle Q,X,Y,\delta,\lambda\rangle$ posiada własność (ж) wtedy i tylko wtedy gdy:

$$\bigwedge_{(i,j) \in I} x_i + x_j \in E \tag{8}$$

W celu znalezienia kodowania o własności (ж) dla automatu A musimy więc znaleźć takie $x_1, x_2,\ldots, x_{n+1} \in B$ by prawdziwy był związek (8). Rozważane zagadnienie zostało zatem sprowadzone do następującego problemu algebraicznego: czy istnieje taka ro-



Rys. 1. Schemat blokowy algorytmu kodowania stanów automatu skończonego $A = \langle Q, X, Y, \delta, \lambda \rangle$ według reguły (*)

dzina indeksowana $(a_t)_{t \in I}$, gdzie $a_t \in E$ dla każdego $t \in I$, że układ równań liniowych (9) o współczynnikach z pierścienia L ma rozwiązanie.

$$x_i + x_j = a_{(i,j)} \quad \text{gdzie} \quad (i,j) \in I \tag{9}$$

Wprowadźmy w zbiorze I porządek leksykograficzny R zdefiniowany w sposób następujący: niech $(i,j), (k,l) \in I$ wówczas

$$(i,j)\ R(k,l) \iff^{df} [(i < k) \vee ((i = k) \wedge (i \leqslant k))] .$$

Wypiszmy zgodnie z porządkiem równania (9). W takiej sytuacji układowi (9) będzie odpowiadała macierz współczynników $C = (c_{ij})$ $i = 1,\ldots,m$; $j = 1,\ldots, n+1$, gdzie $m = \overline{\overline{I}}$ oraz uzupełniona kolumną wyrazów wolnych macierz D. Macierze C i D są tu macierzami o współrzędnych z pierścienia B, przy czym macierz C ma szczególnie prostą budowę ponieważ $c_{ij} \in \{\underline{0},\underline{1}\}$ B i w każdym wierszu mamy dokładnie dwie jedynki $\underline{1}$.

Oznaczmy zbiór $\{\underline{0},\underline{1}\}$ przez M. Zbiór $M = \{\underline{0},\underline{1}\} \subset B$ wraz z działaniami "+" i "·" określonymi wzorami (5) i (6) jest ciałem (jest to ciało izomorficzne z ciałem $\langle Z_2, \oplus, \odot \rangle$ będącym podpierścieniem pierścienia B. M^{n+1} można więc uważać za przestrzeń liniową nad ciałem M z działaniami: "$\otimes$" mnożenia przez skalar (czyli element M) oraz "$\#$" dodawania wektorów zdefioniowanymi następująco:

$$\xi \otimes \alpha = (\xi, \xi_1, \xi \cdot \xi_2, \ldots, \xi \cdot \xi_{n+1})$$

dla każdego $\xi \in M$ i dla każdego $\alpha = (\xi_1, \xi_2, \ldots, \xi_{n+1}) \in M^{n+1}$

$$\alpha \# \beta = (\xi_1^1 + \xi_1^2, \xi_2^1 + \xi_2^2, \ldots, \xi_{n+1}^1 + \xi_{n+1}^2)$$

dla każdego $\alpha = (\xi_1^1, \xi_2^1, \ldots, \xi_{n+1}^1) \in M^{n+1}$ i dla każdego

$$\beta = (\xi_1^2, \xi_2^2, \ldots, \xi_{n+1}^2) \in M^{n+1}$$

Możemy teraz znaleźć w macierzy C, k liniowo niezależnych wierszy. Oczywiście $1 \leqslant k \leqslant n+1$ ponieważ rząd kolumnowy jest zawsze równy rzędowi wierszowemu dla każdej macierzy o współczynnikach w dowolnym ciele, a kolumn mamy jedynie n+1. Niech $\alpha_1, \alpha_2, \ldots, \alpha_k \in M^{n+1}$ będą liniowo niezależnymi wierszami z C, m-k pozostałych wierszy $\alpha_{k+1}, \alpha_{k+2}, \ldots, \alpha_n \in M^{n+1}$ daje się

przedstawić w postaci kombinacji liniowej (10) wierszy $\alpha_1, \alpha_2, \ldots, \alpha_k$

$$\begin{aligned}
&(k+1)\quad g_{k+1,1} \otimes \alpha_1 \# g_{k+1,2} \otimes \alpha_2 \# \ldots \# g_{k+1,k} \otimes \alpha_k = \alpha_{k+1} \\
&(k+2)\quad g_{k+2,1} \otimes \alpha_1 \# g_{k+2,2} \otimes \alpha_2 \# \ldots \# g_{k+2,k} \otimes \alpha_k = \alpha_{k+2} \\
&\qquad\vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\quad \vdots \\
&(m)\quad g_{m,1} \otimes \alpha_1 \# g_{m,2} \otimes \alpha_2 \# \ldots \# g_{m,k} \otimes \alpha_k = \alpha_m
\end{aligned} \tag{10}$$

gdzie: $g_{ij} \in \{0, 1\} = M$; $i = k+1, \ldots, m$; $j = 1, \ldots, k$.

Każdy ze związków (k+1), (k+2),..., (m) jest wektorowo zapisanym układem równań liniowych o współczynnikach w ciele M (lub co na jedno wychodzi w ciele Z_2). Rozwiązując każdy z tych m-k układów obliczamy współczynniki g_{ij}; są one wyznaczone jednoznacznie, ponieważ $\alpha_1, \alpha_2, \ldots, \alpha_k$ można uważać za bazę podprzestrzeni liniowej $V \subset M^{n+1}$, do której należą wszystkie wiersze macierzy C. Warunkiem koniecznym i dostatecznym rozwiązalności układu równań (9) jest zgodnie z twierdzeniem Kroneckera-Capelliego jego niesprzeczność, rozumiana w ten sposób, że każde równanie (12) odpowiadające wierszom $\alpha_{k+1}, \alpha_{k+2}, \ldots, \alpha_m$ macierzy C daje się otrzymać z równań (11) odpowiadającym wierszom $\alpha_1, \alpha_2, \ldots, \alpha_k$ przez obustronne mnożenie przez elementy z ciała M i dodawanie stronami. Innymi słowy warunkiem koniecznym i dostatecznym istnienia rozwiązania układu (9) jest zachodzenie (13).

$$\begin{cases}
\alpha_1 \times (x_1, x_2, \ldots, x_{n+1}) = a_1 \\
\alpha_2 \times (x_1, x_2, \ldots, x_{n+1}) = a_2 \\
\qquad\qquad\vdots \\
\alpha_k \times (x_1, x_2, \ldots, x_{n+1}) = a_k
\end{cases} \tag{11}$$

$$\begin{cases}
\alpha_{k+1} \times (x_1, x_2, \ldots, x_{n+1}) = a_{k+1} \\
\alpha_{k+2} \times (x_1, x_2, \ldots, x_{n+1}) = a_{k+2} \\
\qquad\qquad\vdots \\
\alpha_m \times (x_1, x_2, \ldots, x_{n+1}) = a_m
\end{cases} \tag{12}$$

gdzie: $a_1, \ldots, a_m \in E$ są wyrazami wolnymi układu (9), a

$\alpha \times (x_1, \ldots, x_{n+1})$ oznacza $\sum_{i=1}^{n+1} g_i x_i$; $\alpha = (g_1, \ldots, g_{n+1}) \in M^{n+1}$

$$\begin{cases} g_{k+1,1}a_1 + g_{k+2,2}a_2 + \ldots + g_{k+1,k}a_k = a_{k+1} \\ g_{k+2,1}a_1 + g_{k+1,2}a_2 + \ldots + g_{k+2,k}a_k = a_{k+2} \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ g_{m,1}\ a_1 + g_{m,2}\ a_2 + \ldots + g_{m,k}\ a_k = a_m \end{cases} \tag{13}$$

Uzyskaliśmy w ten sposób twierdzenie następujące:

Twierdzenie 2

Dla danego automatu $A = \langle Q, X, Y, \delta, \lambda \rangle$ istnieje kodowanie Φ jego stanów o własności (*) wtedy i tylko wtedy, gdy istnieją takie $a_1, a_2, \ldots, a_k \in E$, że:

$$\bigwedge_{i=k+1,\ldots,m} g_{i,1}a_1 + g_{i,2}a_2 + \ldots + g_{i,k}a_k \in E \tag{14}$$

gdzie k jest rzędem macierzy C. Kodowania $x_i = \Phi(q_i)$ stanów $q_i \in Q$; są rozwiązaniami układu równań (11).

Twierdzenie powyższe nie podaje efektywnej metody wyszukiwania wyrazów wolnych $a_i \in E$; $i = 1, 2, \ldots, k$ spełniających (14). Z lematów 1 i 2 wynika jednak, że wystarczy sprawdzić (14) tylko dla pewnych układów wyrazów wolnych $(a_1, \ldots, a_k)$ by móc stwierdzić, czy istnieje jakikolwiek układ $(a_1, \ldots, a_k)$ spełniający (14). Szerzej zasygnalizowany tu problem zostanie omówiony w opisie algorytmu.

3. OPIS ALGORYTMU KODOWANIA

Niech jak poprzednio graf nieskierowany Γ odpowiadający automatowi $A = \langle Q, X, Y, \delta, \lambda \rangle$ będzie grafem spójnym, a funkcja δ zadana tablicą przejść T.

Definicja 4

Macierzą przejść automatu $A = \langle Q, X, Y, \delta, \lambda \rangle$, gdzie $\bar{Q} = n+1$ nazywamy macierz:

(R_{ij}) $i,j = 1,2,\ldots, n+1$ gdzie $R_{ij} = \{x \in X : \delta(q_i, x) = q_j\}$

Definicja 5

Kadłubową (szkieletową) macierzą przejść odpowiadającą automatowi A nazywamy macierz uzyskaną z macierzy przejść automatu A przez zastąpienie niepustych zdarzeń regularnych będących współczynnikami macierzy przejść przez 1 dwuelementowej algebry Boole'a $\langle \{0,1\}, \cup, \cap, - \rangle$, a zdarzeń pustych przez 0 tej algebry.

Mając tablicę przejść T automatu można znaleźć kadłubową macierz przejść N. Metoda przejścia z tablicy przejść T do kadłubowej macierzy przejść N jest prosta i nie wymaga szerszego omówienia. Ze znalezionej kadłubowej macierzy przejść N usuwamy główną przekątną (ponieważ nie interesuje nas pozostawanie automatu w tym samym stanie po podaniu na wejście automatu pojedynczej litery), a następnie sumujemy macierz N z macierzą transponowaną N^T. Uzyskujemy w ten sposób kadłubową macierz przejść $P = N \cup N^T$ opisującą graf nieskierowany Γ (bez krawędzi odpowiadających pozostaniu w tym samym stanie) odpowiadający automatowi A. Sumowanie dwu macierzy kadłubowych odbywa się tak samo jak sumowanie dwu macierzy o współczynnikach w dowolnym ciele z tym, że zamiast działania sumowania elementów ciała wykorzystywane jest działanie sumowania "$\cup$" z dwuelementowej algebry Boole'a $\langle \{0,1\}, \cup, \cap, - \rangle$.

Badanie warunku koniecznego istnienia kodowania o własności (*) opiera się na twierdzeniu 1'. Dla dowolnie wybranego stanu np. q_1 sprawdzamy zachodzenie warunku (3'). Podział zbioru stanów Q na klasy abstrakcji względem relacji równoważności $\sim_{q_1}$ uzyskujemy wykorzystując fakt, że macierz P^k (macierze kadłubowe mnożymy w zwykły sposób wykorzystując działania "$\cup$" i "$\cap$" odpowiednio zamiast dodawania i mnożenia) podaje wszystkie pary stanów, pomiędzy którymi istnieje przejście

pod wpływem dokładnie k liter. Jeżeli warunek konieczny jest spełniony, to na podstawie obliczonej macierzy P określamy macierz C. Macierz C ma n+1 kolumn i $\bar{\bar{I}}$ wierszy. Liczba $\bar{\bar{I}}$ jest równa liczbie współrzędnych macierzy P równych 1. Jeżeli w macierzy P współrzędna $P_{i,j} = 1$ oraz $(i,j) \in I$, to odpowiada jej wiersz macierzy C postaci:

$$(0,\ldots,0,\underbrace{1}_{i},0,\ldots,0,\underbrace{1}_{j},0,\ldots,0) \quad (n+1 \text{ składników})$$

W przypadku, gdy graf Γ odpowiadający automatowi ma szczególnie prostą strukturę (jest pojedynczym cyklem lub daje się pokryć drogą prostą) procedura kodowania może być znacznie uproszczona. Wykorzystać można w tym celu kod Graya i tablice Karnaugha. Ten szczególny przypadek nie będzie w dalszym ciągu odrębnie rozpatrywany.

W sytuacji gdy Γ nie ma prostej struktury stosujemy metodę ogólną kodowania. Po określeniu macierzy C wybieramy z niej zbiór liniowo niezależnych wierszy obliczając jednocześnie współczynniki $g_{i,j} \in M = \{0, 1\}$, przez które musimy mnożyć wiersze liniowo niezależne dla otrzymania danego wiersza z C. Następnie dokonujemy sprawdzenia czy istnieje realizacja kodowania (*) dla wybranego zestawu wyrazów wolnych. Sposób wyboru zestawów wyrazów wolnych jest następujący. Wybieramy w grafie Γ wierzchołek o największej liczbie połączeń; oznaczmy go q_{max}. Jeżeli graf Γ nie ma prostej struktury, to do q_{max} dochodzi $p \geq 3$ krawędzi, którym odpowiada p równań z (11). Załóżmy, że jest to p pierwszych równań. Z lematów 1 i 2 wynika, że w badanych zestawach wyrazów wolnych $(a_1, a_2, \ldots, a_k)$ układu równań (11) jako wyrazy $a_1, a_2, \ldots, a_p$ można przyjąć następujące ustalone ciągi:

$$a_1 = (\underbrace{0, 0, \ldots, 0, 1}_{n})$$

$$a_2 = (0, 0, \ldots, 0, 1, 0) \qquad \text{gdzie } n = \bar{\bar{Q}} - 1$$

$$a_3 = (0, 0, \ldots, 0, 1, 0, 0)$$

$$\vdots$$

$$a_p = (0, 0, \ldots, 0, \underbrace{1, 0, \ldots, 0}_{p})$$

a jako $a_{p+1}, a_{p+2}, \ldots, a_k$ wybierać tylko takie ciągi, w których "1" znajduje się na jednej z pozycji ujętych w klamry

$$a_{p+1} = (0, 0, \ldots, 0, \overbrace{0, 1, \ldots, 0}^{p+1}, \overbrace{\quad}^{n})$$

$$a_{p+2} = (0, 0, \ldots, 0, \overbrace{1, 0, \ldots, 0}^{p+2})$$

$$\vdots$$

$$a_k = (0, 0, \ldots, \overbrace{0, \ldots, 0, 1, 0, \ldots, 0}^{k}, \overbrace{\quad}^{n})$$

W przypadku najmniej korzystnym będziemy mieli tylko 3 ustalone wyrazy, w najkorzystniejszym - wszystkie, co pozwoliłoby na rozstrzygnięcie istnienia kodowania (*) w jednym sprawdzeniu. Jak wynika z powyższych uwag nie musimy dokonywać sprawdzenia niesprzeczności (9) dla wszystkich zestawów $(a_1, \ldots, a_k)$ ponieważ wystarczy to zrobić tylko dla niektórych.

Jeżeli dla jakiegoś zestawu $(a_1, \ldots, a_k)$ układ (9) jest niesprzeczny, to kodowanie (*) dla automatu A istnieje. W celu znalezienia konkretnych ciągów kodujących rozwiązujemy dla tego zestawu układ równań (11) np. metodą eliminacji Gausa. Ponieważ rozwiązujemy układ równań nie w ciele liczb rzeczywistych, lecz w pewnym skończonym pierścieniu $L = \overbrace{K \times K \times \ldots \times K}^{n}$ metoda eliminacji musi ulec z konieczności modyfikacji wynikającej z innego sensu działań mnożenia, dodawania, odejmowania. Z uwagi na postać macierzy C nie zachodzi potrzeba dzielenia przez elementy pierścienia. Jest to istotne ponieważ dzielenie w pierścieniu L nie zawsze jest określone. Jeśli nie wszystkie pozycje w uzyskanych ciągach kodujących są w sposób istotny wykorzystywane, to jako ciągi kodujące można przyjąć ciągi krótsze eliminując zbędne nie ulegające zmianie pozycje. Inną drogą prowadzącą do minimalizacji długości ciągów kodujących (przy zachowaniu własności (*) kodowania) jest odpowiednia kolejność w doborze zestawów wyrazów wolnych $(a_{p+1}, a_{p+2}, \ldots, a_k)$, dla których badamy niesprzeczność układu równań (9). Rozpoczynamy mianowicie sprawdzanie niesprzeczności (9) od takich $(a_{p+1}, a_{p+2}, \ldots, a_k)$, by wykorzystać w sumie jak najmniejszą liczbę pozycji. Pierwszy znaleziony układ $(a_{p+1}, \ldots, a_k)$, dla

którego układ równań (9) jest nieosprzeczny daje po rozwiązaniu (11) ciągi kodujące o minimalnej liczbie pozycji w sposób istotny wykorzystywanych.

Sporządzony opis algorytmu w języku ALGOL pozwolił na ocenę liczby niezbędnych obliczeń przy realizacji algorytmu na maszynie cyfrowej. Momentem krytycznym jest duża liczba sprawdzeń niesprzeczności układu(9) dla dobieranych zestawów wyrazów wolnych $(a_1, a_2, \ldots, a_k)$. Nie gra to jednak istotniejszej roli dla automatów, dla których $\bar{Q} \leq 10$ i na średnio szybkiej maszynie cyfrowej czas wykonania programu nie powinien przekroczyć 30 min. Ponieważ liczba obliczeń przy kodowaniu metodą ogólną automatu n-stanowego przy $n \geq 4$ jest w przybliżeniu proporcjonalna do liczby $4 \cdot 5 \cdot 6 \ldots (n-1) \cdot (n-1)$ może się zdarzyć, że kodowany automat, dla którego $\bar{Q} > 10$ nie da się zakodować w dostatecznie krótkim czasie.

Wśród znanych metod kodowania stanów automatu asynchronicznego (por. [5]), sprowadzających problem kodowania stanów do problemu uniknięcia wyścigów krytycznych, korzystnie wyróżnia opisywaną metodę kodowania fakt niezależności algorytmu kodowania od funkcji wyjść λ automatu.

Literatura

[1] HARTMANIS J.: On the State Assignment Problem for Sequential Machines, IEEE Trans. EC - 2/61.

[2] OPIAL Z.: Algebra wyższa, PWN, Warszawa 1972.

[3] ORE O.: Wstęp do teorii grafów, PWN, Warszawa 1966.

[4] TRACZYK T.: Wstęp do teorii algebr Boole'a, PWN, Warszawa 1970.

[5] TRACZYK W.: Synteza automatów asynchronicznych, Wydawnictwa Politechniki Warszawskiej, Warszawa 1969

[6] WALIGÓRSKI S: Algebraiczna teoria automatów, Algorytmy, vol. VI, Nr 11/1969

О НЕКОТОРОМ МЕТОДЕ КОДИРОВАНИЯ СОСТОЯНИЙ ОКОНЧЕННЫХ АВТОМАТОВ

Резюме

В работе описаны обязательные и достаточные условия позволяющие применить некоторый способ кодирования состояния оконченного автомата. Указан алгоритм метода в виде блоксхемы. Описанные результаты могут быть применены в кодировании состояний асинхронных автоматов.

ON A CERTAIN METHOD OF CODING FINITE AUTOMATA STATES

Summary

Described necessary and sufficient conditions permitting to apply a certain way of coding finite automata states. Given the algorithm of the method in the form of a block scheme. The described results may be applied to code states of asynchronic automata.

ALGORYTMY, XI, N° 19, 1974, 89-110 517.5:519.281.2

APROKSYMACJA ZA POMOCĄ FUNKCJI LINIOWEJ METODĄ NAJMNIEJSZYCH KWADRATÓW ODLEGŁOŚCI PUNKTÓW DANYCH OD PUNKTÓW FUNKCJI APROKSYMUJĄCEJ

Zdzisław DĘBICKI

Ośrodek Badawczo-Rozwojowy Podstaw Technologii i Konstrukcji Maszyn
Warszawa - Anin

Pracę złożono 23.06.1973

Stosowanie metody najmniejszych kwadratów do aproksymacji funkcją liniową w powszechnie stosowany standardowy sposób prowadzi do określenia położenia prostej, które w stosunku do danego zbioru punktów aproksymowanych jest zależne od doboru układu współrzędnych prostokątnych. Położenie prostej nie jest niezmiennikiem izometrii.

Dokonując obrotu układu punktów w stosunku do przyjętego układu współrzędnych prosta aproksymująca odchyla się coraz bardziej od początkowego położenia, co wskazuje na powstawanie znacznych błędów przy stosowaniu tej metody.

Natomiast jeżeli prostą poprowadzić tak, aby suma kwadratów odległości punktów od prostej była minimalna, to położenie tej prostej w stosunku do układu aproksymowanego jest niezmiennicze.

Dla takiej zasady aproksymacji zostały wyprowadzone odpowiednie wzory oraz podany jest program w Algolu 60.

1. WSTĘP

Dana jest funkcja f określona przez zbiór punktów o współrzędnych (x_1, y_1), (x_2, y_2),... (x_n, y_n).

W celu przybliżonego przedstawienia tej funkcji można poszukiwać funkcję aproksymującą stosując standardową metodę najmniejszych kwadratów, przyjmując jako zmienną niezależną zmienną x i funkcję aproksymującą φ_x lub przyjmując jako zmienną niezależną zmienną y i funkcję aproksymującą φ_y.

W pierwszym wypadku parametry funkcji aproksymującej wyznaczamy z zależności

$$S_y = \sum_{i=1}^{n} (y_i - \varphi(x_i))^2 = \min \qquad (1)$$

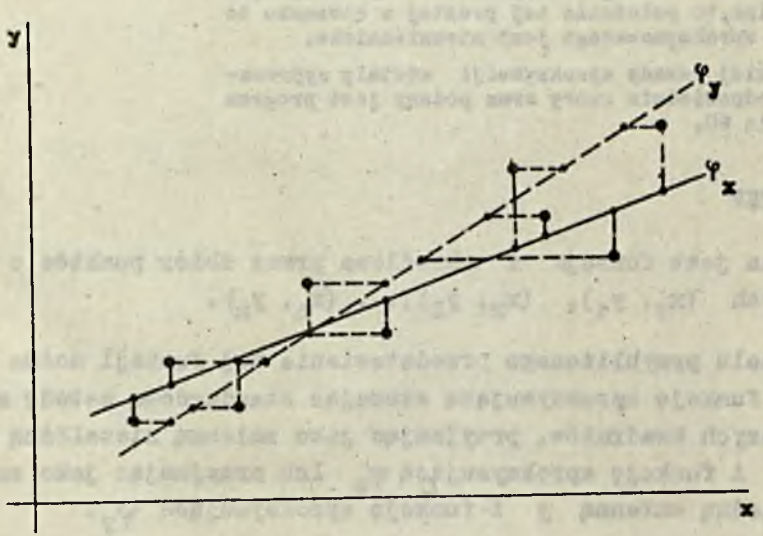
a w drugim przypadku

$$S_x = \sum_{i=1}^{n} (x_i - \psi(y_i))^2 = \min \qquad (2)$$

Funkcje wyznaczone na podstawie obu podanych warunków są różne, co przykładowo pokazano na rys. 1 dla aproksymacji przeprowadzonej za pomocą funkcji liniowej.

W przypadku, gdy w rozpatrywanym zagadnieniu nie ma przyczyn do wybrania zmiennej niezależnej, powstaje wątpliwość, które przybliżenie jest właściwe.

Analizę tego zagadnienia można przeprowadzić rozpatrując zmienność położenia linii prostej aproksymującej w stosunku do danego układu punktów, która występuje w wyniku zmiany położenia tego układu punktów w stosunku do układu współrzędnych na skutek obrotu układu współrzędnych.



Rys. 1

2. ZMIANA POŁOŻENIA PROSTEJ APROKSYMUJĄCEJ W STOSUNKU DO DANEGO UKŁADU PUNKTÓW PRZY OBROCIE UKŁADU WSPÓŁRZĘDNYCH

Każdy układ punktów można sprowadzić przez dokonanie odpowiedniego obrotu i przesunięcia układu współrzędnych do położenia takiego, że prosta aproksymująca, wyznaczona za pomocą warunku określonego zależnością (1), pokryje się z osią X-ów, tzn. równanie tej prostej określi zależność

$$y = 0;$$

Takie położenie punktów przyjmujemy jako wyjściowe. W ogólnym przypadku współczynniki prostej aproksymującej

$$y = ax + b; \tag{3}$$

wyznaczają zależności:

$$a = \frac{n.\Sigma xy - \Sigma x.\Sigma y}{n\Sigma x^2 - (\Sigma x)^2} \tag{4}$$

$$b = \frac{\Sigma y.\Sigma x^2 - \Sigma x.\Sigma xy}{n\Sigma x^2 - (\Sigma x)^2}; \tag{5}$$

Dokonując obrotu układu o kąt - φ otrzymamy nowe współrzędne punktów

$$x' = x\cos\varphi - y\sin\varphi;$$

$$y' = x\sin\varphi + y\cos\varphi;$$

i nową wartość współczynnika kątowego prostej aproksymującej

$$a' = \frac{n.\Sigma x'y' - \Sigma x'.\Sigma y'}{n.\Sigma x'^2 - (\Sigma x')^2} \tag{6}$$

Po podstawieniu w zależności (6) niżej podanych wartości

$$\Sigma x' = \cos\varphi.\Sigma x - \sin\varphi.\Sigma y;$$

$$\Sigma y' = \sin\varphi.\Sigma x + \cos\varphi.\Sigma y;$$

$$(\Sigma x')^2 = \cos^2\varphi(\Sigma x)^2 + \sin^2\varphi(\Sigma y)^2 - 2\sin\varphi.\cos\varphi.\Sigma x.\Sigma y;$$

$$\Sigma x'^2 = \cos^2\varphi\Sigma x^2 + \sin^2\varphi\Sigma y^2 - 2\sin\varphi.\cos\varphi.\Sigma xy;$$

$$\Sigma x'y' = \cos\varphi.\sin\varphi.\Sigma x^2 + \cos^2\varphi\Sigma xy - \sin^2\varphi\Sigma xy - \sin\varphi\cos\varphi\Sigma y^2;$$

otrzymamy

$$a' = \frac{n(\cos\varphi \cdot \sin\varphi \Sigma x^2 + \cos^2\varphi \Sigma xy - \sin^2\varphi \Sigma xy - \sin\varphi \cos\varphi \Sigma y^2) - }{n(\cos^2\varphi \cdot \Sigma x^2 + \sin^2\varphi \Sigma y^2 - 2\sin\varphi \cdot \cos\varphi \Sigma xy) - \cos^2\varphi (\Sigma x)^2 -} \quad (7)$$

$$\frac{- \cos\varphi \sin\varphi (\Sigma x)^2 - \cos^2\varphi \Sigma x \cdot \Sigma y + \sin^2\varphi \Sigma x \Sigma y + \sin\varphi \cos\varphi (\Sigma y)^2}{- \sin^2\varphi (\Sigma y)^2 + 2 \sin\varphi \cos\varphi \cdot \Sigma x \Sigma y};$$

Ponieważ jako położenie wyjściowe układu punktów przyjęliśmy takie ich położenie, że prosta aproksymująca określona jest dla tego położenia równaniem

$$y = 0;$$

tzn. $a = 0$; i $b = 0$;

więc z zależności (4) i (5) mamy

$$n\Sigma xy - \Sigma x \cdot \Sigma y = 0;$$

$$\Sigma y \cdot \Sigma x^2 - \Sigma x \cdot \Sigma xy = 0;$$

Z rozwiązania tych równań otrzymujemy

$$\Sigma y = 0;$$

$$\Sigma xy = 0;$$

Podstawiając te wartości do zależności (7) otrzymujemy

$$a' = \frac{n \cdot \sin\varphi \cdot \cos\varphi (\Sigma x^2 - \Sigma y^2) - \sin\varphi \cdot \cos\varphi (\Sigma x)^2}{n (\cos^2\varphi \Sigma x^2 + \sin^2\varphi \Sigma y^2) - \cos^2\varphi (\Sigma x)^2};$$

$$a' = \frac{(n \Sigma x^2 - (\Sigma x)^2) - n \Sigma y^2}{(n \cdot \Sigma x^2 - (\Sigma x)^2) \cdot \operatorname{ctg}\varphi + n \cdot \Sigma y^2 \cdot \operatorname{tg}\varphi};$$

i dalej

$$a' = \frac{1 - \dfrac{n\Sigma y^2}{n\Sigma x^2 - (\Sigma x)^2}}{\operatorname{otg}\varphi + \dfrac{n\Sigma y^2}{n\Sigma x^2 - (\Sigma x)^2} \cdot \operatorname{tg}\varphi}; \qquad (8)$$

Wyrażenie

$$\varkappa = \frac{n\Sigma y^2}{n\Sigma x^2 - (\Sigma x)^2}; \qquad (9)$$

może służyć jako ocena stopnia skupienia punktów pod względem możliwości uzyskania prawidłowych wyników przy stosowaniu standardowej metody najmniejszych kwadratów.

Mamy bowiem:

1) dla $\varkappa = 0$; mamy $a' = \operatorname{tg}\varphi$;

tzn., że prosta aproksymująca zmienia swe położenie, tak samo jak i układ punktów przy obrocie całego układu współrzędnych, a zatem zachowuje ona stałe położenie w stosunku do danego układu punktów. Takie skupienie jest najlepsze. Zachodzi ono gdy

$$\varkappa = 0; \quad \text{tzn.} \quad \Sigma y^2 = 0;$$

Ma to miejsce wówczas, gdy wszystkie dane punkty leżą na jednej prostej.

2) dla $\varkappa = 1$ mamy $a' = 0$;
tzn., że prosta aproksymująca nie zmienia swego kątowego położenia w stosunku do początkowego położenia układu współrzędnych, a zatem zmienia swe położenie w stosunku do układu punktów danych zmieniających swe położenie razem z obrotem układu współrzędnych. Oznacza to, że prosta aproksymująca może być nachylona do układu punktów pod dowolnym kątem (zależnym tylko od kątowego położenia współrzędnych).

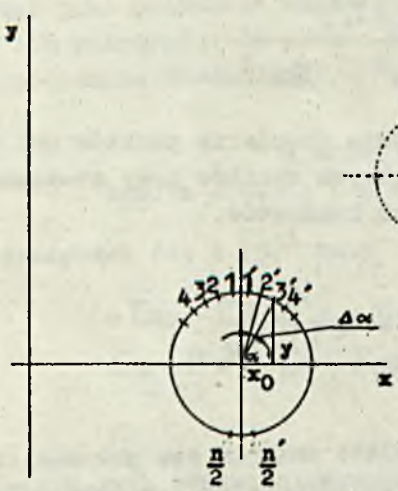
Przypadek taki zachodzi np. dla punktów rozmieszczonych na okręgu (rys. 2). Można to udowodnić wykazując, że dla punktów okręgu określonego równaniem

$$(x - x_o)^2 = R^2 - y^2; \tag{10}$$

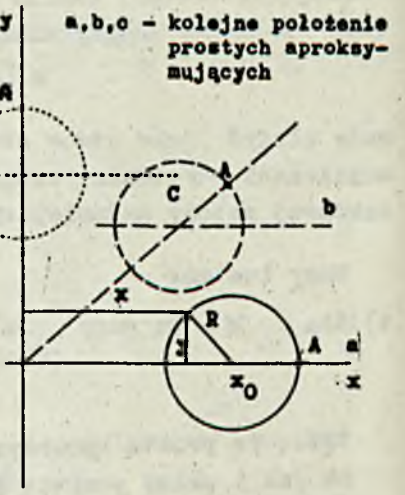
spełniony jest warunek

$$\varkappa = 1;$$

tzn. $$n\Sigma y^2 = n \cdot \Sigma x^2 - (\Sigma x)^2; \tag{11}$$



Rys. 2a



Rys. 2b

Dowód:

Do rozważań przyjmujemy parzystą liczbę punktów rozmieszczonych na okręgu symetrycznie do prostej

$$X = X_o;$$

wg rys. 2a.

Z przekształcenia równania (10) mamy

$$x = x_o \pm \sqrt{R^2 - y^2};$$

Po podstawieniu tej wartości do równania (11) otrzymujemy

$$n\Sigma y^2 = n^2x_o^2 + n^2R^2 - n\Sigma y^2 \pm 2x_o\Sigma\sqrt{R^2 - y^2} - \left(nx_o \pm \Sigma\sqrt{R^2 - y^2}\right)^2; \tag{12}$$

Ponieważ dla przyjętego układu punktów mamy:

$$\Sigma\sqrt{R^2 - y^2} = \Sigma(x_o - x) = 0;$$

więc z zależności (12) otrzymujemy

$$2\Sigma y^2 = n R^2;$$

i następnie

$$\Sigma\left(\frac{y}{R}\right)^2 = \frac{n}{2}; \qquad (13)$$

Wprowadzając oznaczenie (rys. 2a)

$$\frac{2\pi}{n} = \Delta\alpha;$$

mamy

$$\frac{n}{2} = \frac{\pi}{\Delta\alpha};$$

Mamy również

$$\frac{y}{R} = \sin\alpha;$$

Podstawiając dwie ostatnie wartości do zależności (13) otrzymujemy

$$\Sigma \sin^2\alpha = \frac{\pi}{\Delta\alpha};$$

skąd

$$\pi = (\Sigma \sin^2\alpha) \cdot \Delta\alpha;$$

Przechodząc do ciągłego rozmieszczenia punktów na okręgu mamy $n \longrightarrow \infty$; $\Delta\alpha \longrightarrow d\alpha$
i w związku z tym

$$\pi = \int_0^{2\pi} \sin^2\alpha \, d\alpha = \pi;$$

a więc spełniony jest warunek (11).

3) Dla ogólnego przypadku, tj. dla $0 < \varkappa < 1$ zmienność współczynnika kątowego a prostej aproksymującej w zależności od obrotu układu o kąt φ podana jest na rys. 3a i 3b.

Maksymalna wartość a wyznaczona z warunku $\frac{da'}{d\varphi} = 0$; wynosi

$$a'_{max} = \frac{n\Sigma x^2 - (\Sigma x)^2 - n\Sigma y^2}{2\sqrt{n.\Sigma y^2}\ .\ \sqrt{n\ .\Sigma x^2 - (\Sigma x)^2}}; \tag{14}$$

dla

$$\varphi = \operatorname{arctg} \sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n\Sigma y^2}}; \tag{15}$$

Otrzymana wartość a'_{max} związana jest z odpowiadającym jej kątem φ prostą zależnością.

Mamy bowiem

$$\operatorname{tg}\varphi = \sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n\Sigma y^2}};$$

$$a' = \frac{n\Sigma x^2 - (\Sigma x)^2}{2\sqrt{n\Sigma y^2}\ .\sqrt{n\Sigma x^2 - (\Sigma x)^2}} - \frac{n\Sigma y^2}{2\sqrt{n\Sigma y^2}\ .\sqrt{n\Sigma x^2 - (\Sigma x)^2}} =$$

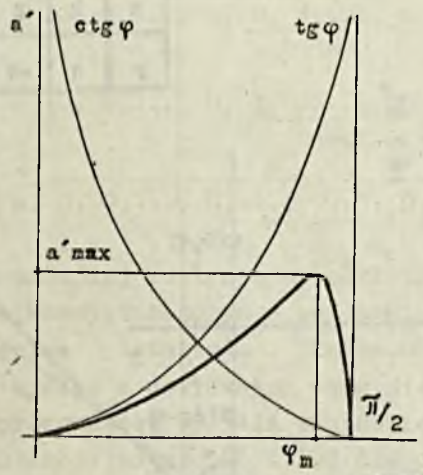
$$= \frac{1}{2}\sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n\ .\Sigma y^2}} - \frac{1}{2}\frac{1}{\sqrt{\frac{n\Sigma x^2 - (\Sigma x)^2}{n\Sigma y^2}}} =$$

$$= \frac{1}{2}\ (\operatorname{tg}\varphi - \operatorname{ctg}\varphi);$$

i ostatecznie

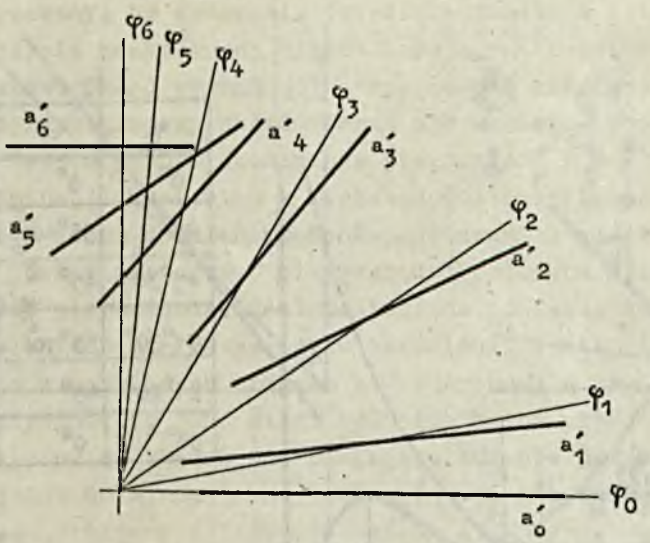
$$a'_{max} = \operatorname{tg}\alpha_{max} = -\operatorname{tg} 2\varphi; \tag{16}$$

Wyniki te wskazują na niebezpieczeństwo stosowania standardowej metody najmniejszych kwadratów dla pewnych układów punktów.



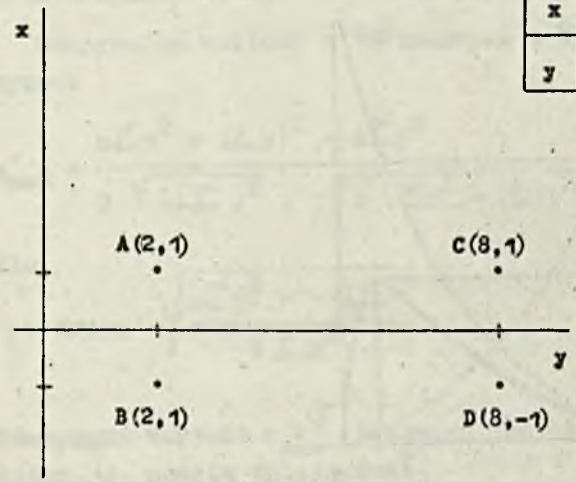
$$\varphi_m = \arctan \frac{n\Sigma y^2 - (\Sigma y)^2}{n\Sigma x^2}$$

Rys. 3a



Rys. 3b

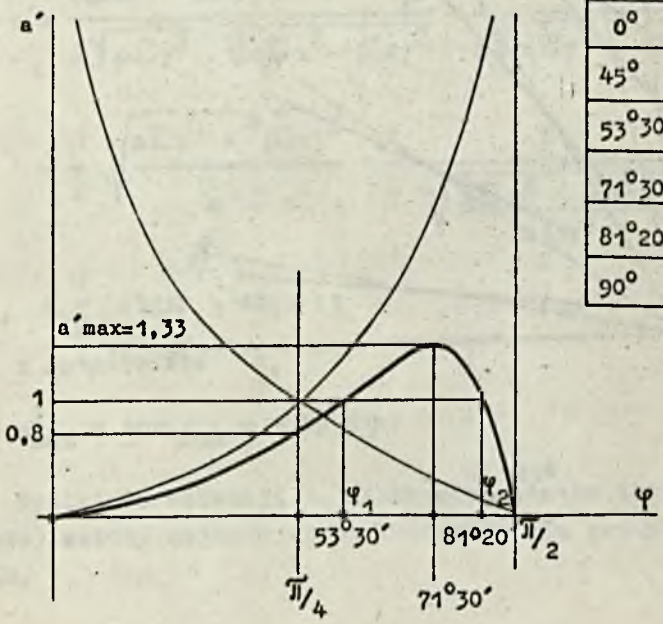
	A	B	C	D
x	2	2	8	8
y	1	-1	1	-1



$\Sigma_x = 20$
$\Sigma_y = 0$
$\Sigma_{xy} = 0$
$\Sigma_x^2 = 136$
$\Sigma_y^2 = 4$

Rys. 4a

φ	α	$a' = tg\alpha$
0°	0°	0,0
45°	$38^\circ40'$	0,8
$53^\circ30'$	45°	1,0
$71^\circ30'$	$53^\circ10'$	1,33
$81^\circ20'$	45°	1,0
90°	0°	0,0



Rys. 4b

φ= 0; a'= 0; α = 0
2,1
8,1
2,-1
8,-1
φ=45 a'=0,8 α=38°40'
φ-α =6°20'
φ=53°30' a'= 1
α=45°
φ-α =8°30'
φ = 71°30'
a max = 1,333
α max = 53°10'
φ- α = 18°20'
φ= 81°20'
a = 1
α = 45°
φ-α= 36°20'
φ = 90°
a = 0
α = 0
φ -α = 90°

Rys. 4c

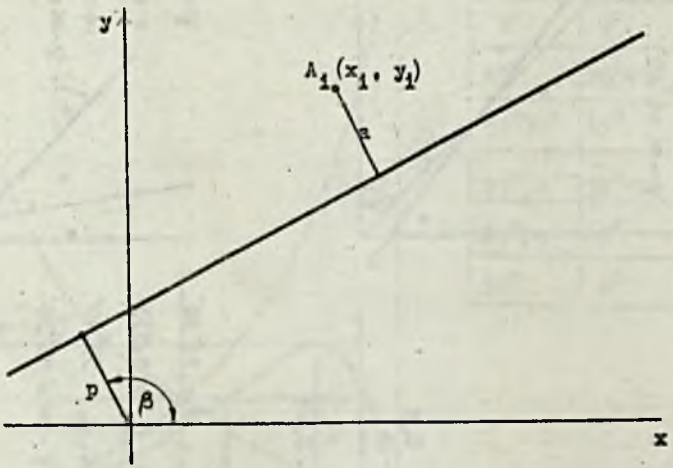
Dla konkretnego przykładu układu punktów podanego na rys. 4a pokazany jest na rys. 4b przebieg funkcji $a' = f(\varphi)$ oraz na rys. 4c sześć wybranych położeń tego układu punktów, otrzymanych w wyniku obrotu układu współrzędnych o kąt φ i odpowiadające im położenia prostej aproksymującej.

Stałe położenie prostej aproksymującej w stosunku do układu punktów niezależnie od położenia tego układu punktów w układzie współrzędnych otrzymujemy przy przeprowadzeniu aproksymacji, opierając się na sumowaniu kwadratów odległości punktów danych od funkcji aproksymującej. W takim przypadku krzywa aproksymująca będzie niezmiennikiem układu współrzędnych. Jej położenie względem punktów nie zależy od układu współrzędnych.

3. APROKSYMACJA ZA POMOCĄ FUNKCJI LINIOWEJ NA PODSTAWIE SUMOWANIA KWADRATÓW ODLEGŁOŚCI PUNKTÓW DANYCH OD FUNKCJI APROKSYMUJĄCEJ

Do rozważań przyjmujemy równanie normalne prostej (rys. 5).

$$x \cdot \cos\beta + y \sin\beta - p = 0 \qquad (17)$$



Rys. 5

Odległość punktu Ai (xi, yi) od prostej określonej równaniem (17) jest

$$a = x_i \cos\beta + y_i \sin\beta - p; \tag{18}$$

Prosta (17) będzie najlepiej przybliżoną do funkcji określonej przez zbiór punktów o współrzędnych (x_1, y_1), (x_2, y_2) ... ... (x_n, y_n), jeśli parametry jej będą określone z warunku (19).

$$S_p = \sum_{i=1}^{n} (x_i \cos\beta + y_i \sin\beta - p)^2 = \min; \tag{19}$$

Warunkiem koniecznym istnienia minimum jest

$$\frac{\partial S_p}{\partial p} = -2 \sum_{i=1}^{n} (x_i \cos\beta + y_i \sin\beta - p) = 0; \tag{20}$$

$$\frac{\partial S_p}{\partial \beta} = 2 \sum_{i=1}^{n} (x_i \cos\beta + y_i \sin\beta - p)(-x_i \sin\beta + y_i \cos\beta) = 0 \tag{21}$$

skąd otrzymujemy

$$\cos\beta \cdot \Sigma x + \sin\beta \Sigma y = n \cdot p;$$

$$-\sin\beta \cos\beta \Sigma x^2 + \cos^2\beta \Sigma xy - \sin^2\beta \Sigma xy + \sin\beta \cos\beta \Sigma y^2 + p \sin\beta \Sigma x - p \cos\beta \Sigma y = 0;$$

Z rozwiązania tego układu równań otrzymujemy

$$\operatorname{tg} 2\beta = \frac{2(\Sigma x \cdot \Sigma y - n \cdot \Sigma xy)}{n(\Sigma y^2 - \Sigma x^2) + (\Sigma x)^2 - (\Sigma y)^2}; \tag{22}$$

po wyznaczeniu kąta β możemy wyznaczyć

$$p = \frac{1}{n}(\cos\beta \Sigma x + \sin\beta \cdot \Sigma y); \tag{23}$$

Dla wyznaczenia parametrów prostej w postaci

$$y = ax + b;$$

mamy z równania (17)

dla $\beta \neq 0 \pm \pi$;

$$a = -\operatorname{ctg}\beta \; ; \tag{24}$$

$$b = \frac{1}{n}\,(\operatorname{ctg}\beta \cdot \Sigma x + \Sigma y); \tag{25}$$

a dla $\beta = 0 \pm \pi$;

otrzymamy równanie prostej

$$x = p = \frac{1}{n}\,(\cos\beta \cdot \Sigma x + \sin\beta \cdot \Sigma y) = \frac{\Sigma x}{n} \; ; \tag{26}$$

Prosta aproksymująca wyznaczona w ten sposób nie zmienia swego położenia w stosunku do danego układu punktów. Suma kwadratów dla prostej aproksymującej wyznaczonej w ten sposób będzie

$$S_p = \Sigma(x_i \cos\beta + y_i \sin\beta - \frac{1}{n}\,(\cos\beta \Sigma x_i + \sin\beta \Sigma y_i))^2 \tag{27}$$

4. WYZNACZANIE PARAMETRÓW PROSTEJ APROKSYMUJACEJ WEDŁUG METODY PODANEJ W POZ. 3 ZA POMOCĄ MASZYNY CYFROWEJ

Wyznaczenie parametrów a i b prostej aproksymującej na podstawie zależności (22), (24), (25) i (26) oraz odpowiadającej jej sumie kwadratów Sp można dokonać korzystając z niżej podanej procedury napisanej w języku Algol 60[x].

```
procedure AKWAP (n, i, a, b, c, d, w, ETYKIETA);
  value n;
  integer n, i; real a, b, c, d, w; label ETYKIETA;.
comment Procedura oblicza parametry a i b równania prostej
        aproksymującej y = ax + b oraz sumę kwadratów od-
        ległości punktów od prostej aproksymującej - w.
        Na parametry procedury należy podstawić
        n - liczbę punktów
        c - współrzędne punktów Xi
        d - współrzędne punktów Yi
        Dla spełnionego warunku
        a = 0∧b = 0 - brak rozwiązania;
```

[x] procedura ta została opracowana przez autora w czasie ćwiczeń na XXVI Kursie Zastosowań Matematyki w Warszawie

```
begin  real  e, f, g, h;
       a:=b:=e:=f:=g:=0;
       for i:=1 step 1 until n do
       begin
       h: = c ; w : = d;
       e : = e + h;
       f : = f+w;
       a: = a+h*h;
       b : = b+w*w;
       g : = g+h*w;
   end i ;
a : = n*(b-a) + e*e - f*f ;
b : = 2*(e*f - n*g) ;
if  a = 0∧b = 0  then  goto  ETYKIETA  else
   if abs (a) ⩽ abs  (b/6.703  10 153) then
                        g : = 2.356194
else
   g:=arctan (b/a) * 0,5 + 1.570796;
   a : = sin (g);
   b : = cos (g);
   h : = 0;
   for i: = 1 step 1 until n do
      begin
   h : = c*b + d*a - (b*e + a*f) /n
   w : = w + h*h ;
   end i ;
   a : = - b/a;
   b : = (- a*e + f) /n;
   end AKWAP
```

Za pomocą tej procedury zostały wyznaczone parametry prostej aproksymującej dla funkcji określonej następującym zbiorem punktów:

Xi	10.12	18.20	22.01	24.95	30.20	34.35	37.80	39.80	41.60	46.60
yi	11.28	12.12	14.30	13.11	15.62	17.65	16.23	19.21	26.20	24.00

Otrzymano równanie

$$y = 0.393\ x + 4.084;$$

i sumę kwadratów odległości

$$Sp = 0.67705588 \cdot 10^2;$$

Dla porównania wyznaczono równanie prostych aproksymujących przy liczeniu kwadratów w kierunku osi X i w kierunku osi Y.

Otrzymane wyniki podane są w tabeli

Odcinki odmierzane w kierunku	Równanie prostej	Suma kwadratów
prostopadłym	$y_p = 0.393\ x + 4.084$	$S_p = 0.67705588 \cdot 10^2$
osi X	$y_y = 0.379\ x + 5.425$	$S_y = 0.35523751 \cdot 10^3$
osi Y	$y_x = 0.490\ x + 2.019$	$S_x = 0.44028431 \cdot 10^3$

Wszystkie proste przecinają się w punkcie

$$X_o = 30.503;$$
$$y_o = 16.972;$$

Proste te pokazane są na rys. 6.

Podana procedura pozwala na wyliczenie parametrów prostej aproksymującej dla kąta α pochylenia tej prostej w przedziale

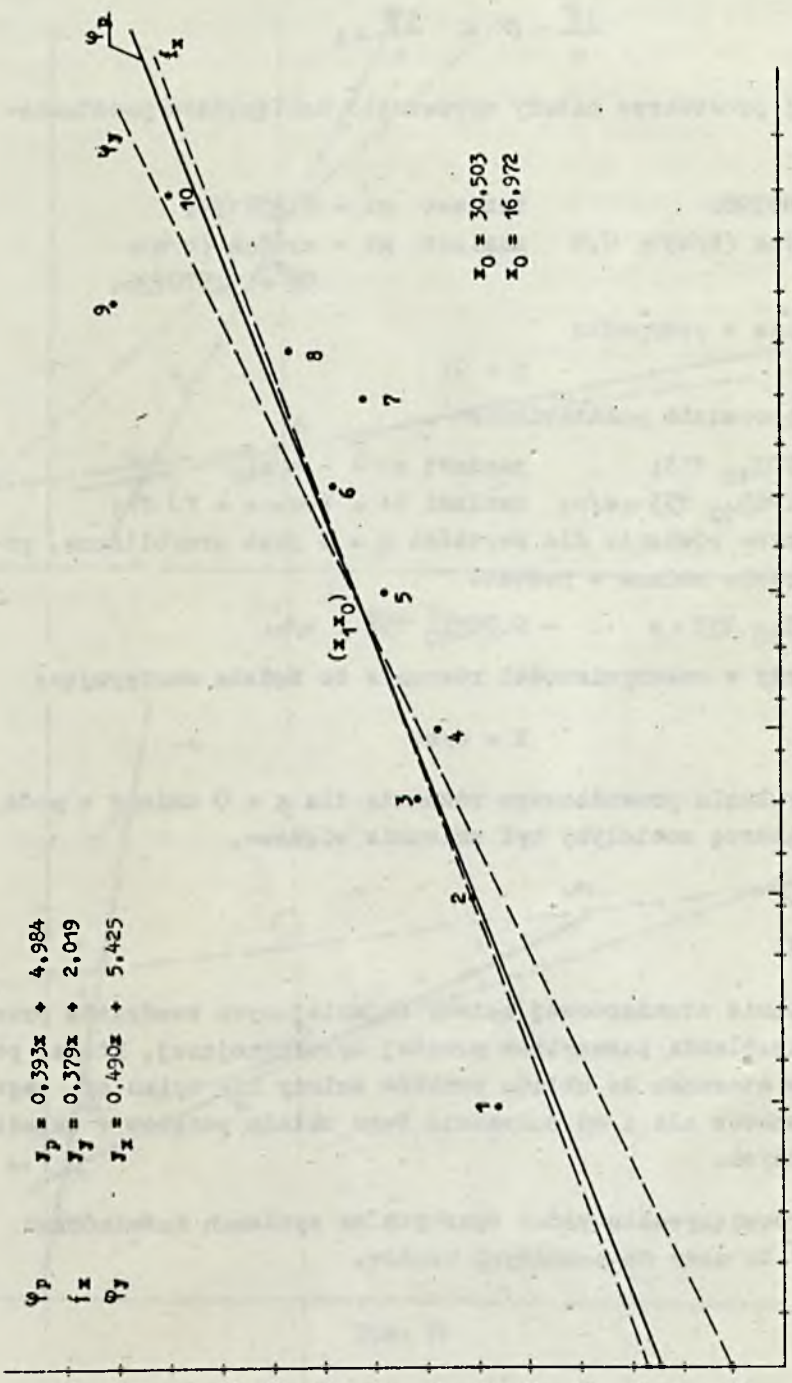
$$-\frac{\pi}{4} < \alpha \leqslant \frac{\pi}{4};$$

tzn. dla kąta β w przedziale

$$\frac{\pi}{4} < \beta \leqslant \frac{3\pi}{4};$$

Natomiast dla przedziału dla kąta α

$$\frac{\pi}{4} < \alpha \leqslant \frac{3\pi}{4};$$


φ_p $y_p = 0{,}393x + 4{,}984$
f_x $y_y = 0{,}379x + 2{,}019$
φ_y $y_x = 0{,}490x + 5{,}425$
$x_0 = 30{,}503$
$x_0 = 16{,}972$
$(x_1 x_0)$
1
2
3
4
5
6
7
8
9
10
φ_p
f_x
φ_y
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48
2 4 6 8 10 12 14 16 18 20 22 24 26 28

Rys. 6

tzn. dla przedziału dla kąta

$$\frac{3\pi}{4} < \beta \leq \frac{5\pi}{4};$$

w podanej procedurze należy wprowadzić następujące podstawienia:

g: = 0.785398; zamiast g: = 2.356194;
g: = arctan (b/a) * 0,5 zamiast g: = arctan (b/a)*
05 + 1.570796;

a następnie w przypadku

g = 0;

należy wprowadzić podstawienie

a : = 6.703_{10} 153; zamiast a: = - b/a;
b : = -6.703_{10} 153 *e/n; zamiast b: = (-a*e + f) /n;

Otrzymane równanie dla wartości g = 0 jest przybliżone, ponieważ będzie podane w postaci

y = 6.703_{10} 153 · x - 6.703_{10} 153 . e/n;

podczas gdy w rzeczywistości równanie to będzie następujące

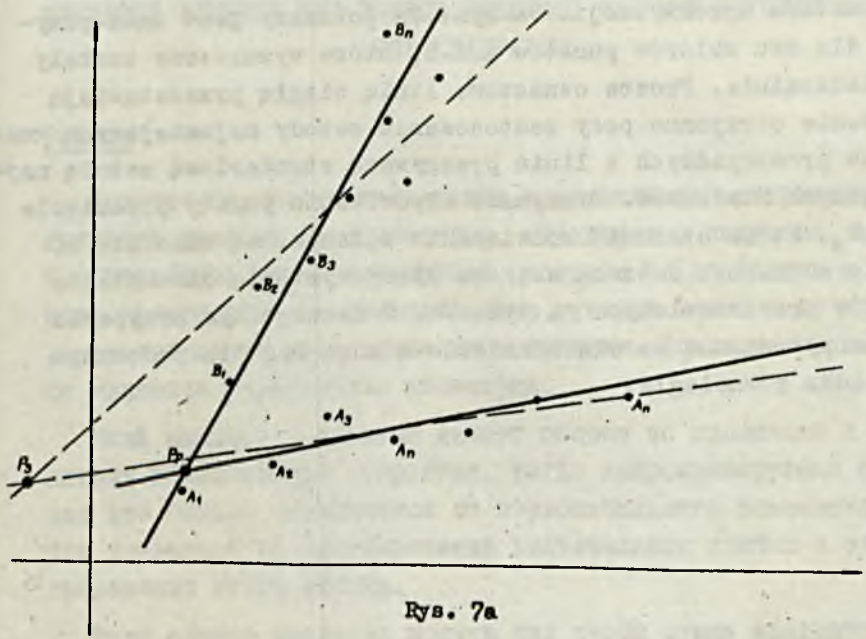
X = e/n;

Do uzyskania prawidłowego równania dla g = 0 zmiany w podanej procedurze musiałyby być znacznie większe.

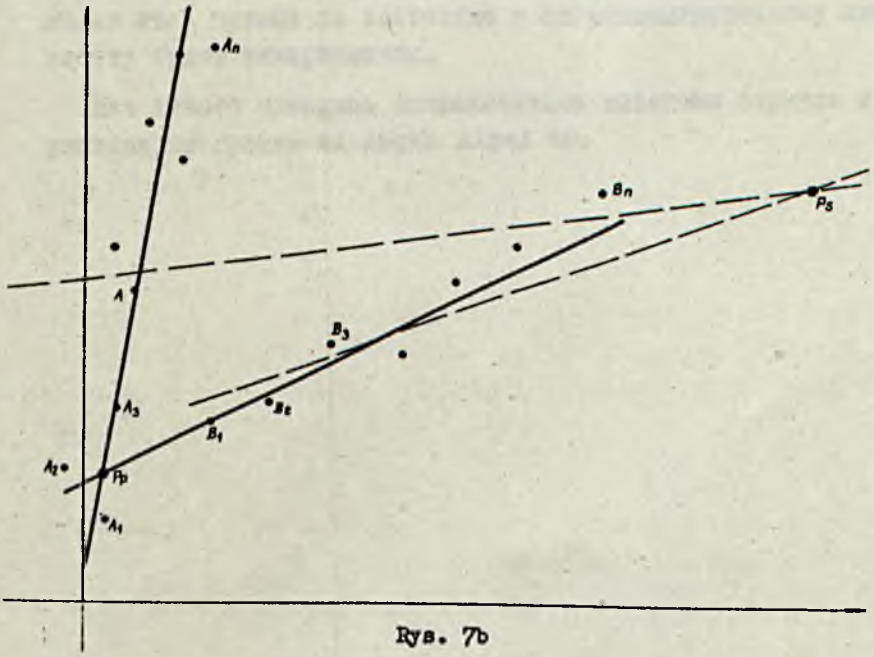
5. WNIOSKI

Stosowanie standardowej metody najmniejszych kwadratów prowadzi do określenia parametrów prostej aproksymującej, której położenie w stosunku do układu punktów zależy nie tylko od tego układu punktów ale i od położenia tego układu punktów w układzie współrzędnych.

Przy rozwiązywaniu zadań opartych na wynikach doświadczeń prowadzić to może do poważnych błędów.



Rys. 7a



Rys. 7b

Jako przykład przytoczyć można zadanie, którego rozwiązaniem będzie wyznaczenie punktu przecięcia się dwu prostych wyznaczonych metodą aproksymacji. Na rys. 7a pokazany jest taki przykład dla dwu zbiorów punktów A i B, które wyznaczone zostały doświadczalnie. Proste oznaczone linią ciągłą przedstawiają położenie otrzymane przy zastosowaniu metody najmniejszych kwadratów prostopadłych a linią przerywaną standardową metodą najmniejszych kwadratów. Otrzymane odpowiednio punkty przecięcia P_p i P_s, które stanowią rozwiązanie zadania, są znacznie od siebie oddalone. Jeszcze większe błędy wystąpią dla układów punktów przedstawionych na rys. 7b. W szczególnym przypadku proste wyznaczone metodą standardową mogą być dla podobnego przypadku równoległe.

АППРОКСИМАЦИЯ РАССТОЯНИЯ ПУНКТОВ ДАННЫХ ОТ ПУНКТОВ АППРОКСИМИРУЮЩЕЙ ФУНКЦИИ ПРИ ПОМОЩИ ЛИНЕЙНОЙ ФУНКЦИИ МЕТОДОМ НАЙМЕНЬШИХ КВАДРАТОВ

Резюме

Применение метода наименьших квадратов для аппроксимации линейной функции общеизвестным, стандартным способом, ведёт к определению положения прямой, которое по отношению к данному множеству аппроксимированных пунктов зависит от подбора расположения прямоугольных координат. Положение прямой не является инвариантом изометрии.

Если множество пунктов делает оборот по отношению к принятому расположению координат, тогда аппроксимирующая прямая всё больше отклоняется от первоначального положения, что указывает на возникновение значительных ошибок в случае применения этого метода.

Если однако провести прямую так чтобы сумма квадратов расстояния пунктов от прямой была минимальная, тогда положение этой прямой по отношению к аппроксимированному множеству будет инвариантном.

Для такого принципа аппроксимации выведены формулы и указана программа на языке Algol 60.

APPROXIMATION BY MEANS OF A LINEAR FUNCTION USING THE LEAST SQUARE METHOD OF THE DISTANCE OF POINTS FROM THE APPROXIMATING LINE

Summary

A commonly applied standard way of using the method of least squares for approximation by means of a linear function leads to the determination of the position of the line, which with respect to the given set of approximated points depends on the choice of rectangular coordinate arrangement. The position of the line is not an invariant of isometry.

Turning the set of points in relation to the accepted coordinate arrangement the approximating line deviates still more from its primary position which indicates the appearance of significant errors when using this method.

But, if the line is led in a way which ensures that the sum of squares of the distance of points from the line is minimal, then the position of this line with respect to the approximated arrangement is invariant.

For such an approximation principle adequate formulas have been written and a program is given in ALGOL 60 language.

Warszawa, ul. Krzywickiego 34
K.M.